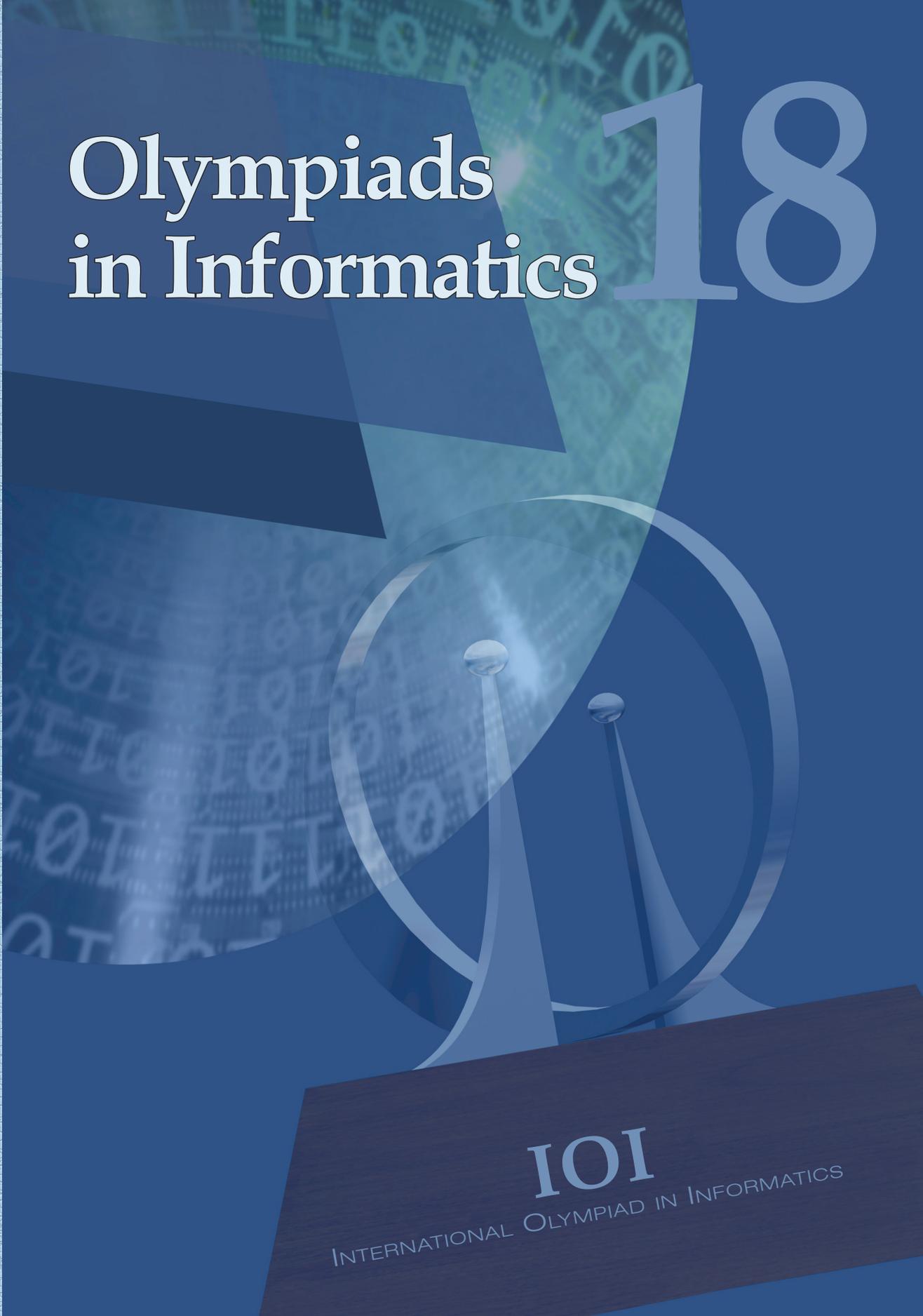


Olympiads in Informatics

18



IOI

INTERNATIONAL OLYMPIAD IN INFORMATICS

ISSN 1822-7732

**INTERNATIONAL OLYMPIAD IN INFORMATICS
VILNIUS UNIVERSITY**

OLYMPIADS IN INFORMATICS

Volume 18 2024

Selected papers of
the International Conference joint with
the XXXVI International Olympiad in Informatics
Alexandria, Egypt, 1–8 September, 2024



OLYMPIADS IN INFORMATICS

Editor-in-Chief

Valentina Dagiene
Vilnius University, Lithuania, valentina.dagiene@mif.vu.lt

Executive Editors

Mile Jovanov
Sts. Cyril and Methodius University, North Macedonia, mile.jovanov@finki.ukim.mk

Technical Editor

Tatjana Golubovskaja
Vilnius University, Lithuania, tatjana.golubovskaja@mif.vu.lt

International Editorial Board

Benjamin Burton, University of Queensland, Australia, bab@maths.uq.edu.au
Michal Forišek, Comenius University, Bratislava, Slovakia, misof@ksp.sk
Gerald Futschek, Vienna University of Technology, Austria, futschek@ifs.tuwien.ac.at
Marcin Kubica, Warsaw University, Poland, kubica@mimuw.edu.pl
Luigi Laura, Uninettuno University, Rome, Italy, luigi.laura@uninettunouniversity.net
Ville Leppänen, University of Turku, Finland, villelep@cs.utu.fi
Krassimir Manev, New Bulgarian University, Bulgaria, kmanev@nbu.bg
Seiichi Tani, Nihon University, Japan, tani.seiichi@nihon-u.ac.jp
Willem van der Vegt, Windesheim University for Applied Sciences, The Netherlands,
w.van.der.vegt@windesheim.nl

The journal Olympiads in Informatics is an international open access journal devoted to publishing original research of the highest quality in all aspects of learning and teaching informatics through olympiads and other competitions.

<https://ioinformatics.org/page/ioi-journal>

ISSN 1822-7732 (Print)
2335-8955 (Online)

© International Olympiad in Informatics, 2024
Vilnius University, 2024
All rights reserved

Foreword

The International Olympiad in Informatics (IOI) is an annual global competition in informatics for individual contestants from over 80 invited countries. The event also includes a one-day scientific conference for delegation leaders, organizers, and guests, providing an excellent opportunity for the IOI community to communicate and exchange ideas. Many countries have a variety of topics to present and discuss.

The IOI Journal focuses on the research and practice of computing professionals who teach informatics to talented secondary and high school students. The journal is closely connected to the scientific conference held annually during the IOI. The 18th volume of the journal has two tracks: the first section focuses on research, and the second section is devoted to sharing national experiences. This volume features contributions from regular contributors as well as new authors.

Researchers from Italian universities, Giorgio Audrito, Sara Capecchi, Madalina G. Ciobanu, and Luigi Laura, have analyzed *Giochi di Fibonacci* (Fibonacci’s games), a programming contest for upper primary and lower secondary school students. The contest is organized in three phases: the first phase is based solely on logical and algorithmic quizzes, while the other two phases involve coding using a Blockly environment integrated into the contest platform.

Mirvari Mammadli, Nihad Mammadli, and Jamaladdin Hasanov presented a model for analyzing contestant progress in real-time coding contests, emphasizing the critical need for effective measures in assessing code similarity and plagiarism. Current coding contest platforms often lack robust procedures to identify and address these issues, compromising the integrity of the evaluation process. To tackle these challenges, the authors propose a novel system that leverages advanced techniques to analyze code and collective behavior, providing a holistic evaluation of submissions. This system enhances the accuracy of performance assessment and maintains fairness and credibility in real-time coding contests. The findings from this study highlight the importance of integrating sophisticated mechanisms to ensure the authenticity of code submissions and uphold the competitive nature of coding competitions.

Tom Verhoeff, in his paper with the intriguing title “Staying DRY with OO and FP”, discusses the coding principle of not repeating code and compares various tactics to achieve DRY code in both object-oriented and functional programming contexts. He encourages IOI team leaders to study the examples in this article with their contestants. Verhoeff believes that higher-order functions, which use functions as parameters and return functions, are powerful tools. While it takes some practice to get used to the functional style, he is convinced that clever contestants will enjoy it.

Representatives from this year’s IOI hosting country, Egypt, Eslam M. Wageed, Yousry S. Elgamal, Ossama M. Ismail, and Mohamed H. Abdrabou, have shared their

experiences on non-formal education as a solution to the requirements of the era. They suggest implementing non-formal education through seminars, training camps, and workshops, including additional activities like competitions or extracurricular learning. Their study investigates the impact of non-formal educational approaches on engineering and computer science students' academic performance and their chances of obtaining a job after graduation.

Felix Steinert, Julia Kummer, Martina Landman, and Lukas Lehner reported on a two-day informatics workshop for Austrian pupils aged eleven to thirteen. The workshop included unplugged activities about algorithms, AI, robotics, and block-based programming using Scratch and Sphero BOLT. Feedback from 110 participants showed a significant gain in knowledge and high interest in computer science. The report details the favorite activities and experiences of the ten workshop leaders.

Several papers in this volume focus on contests and Olympiads: popularizing science competitions by M. Kaykobad; statistical analyses of IOI 2011 to 2023 performance data, emphasizing returning contestants and identifying geographic trends, by E. Lee, T. Reizin, and F. E. Wu; preparing the youngest students for programming contests by K. Manev; Olympiads without words by P. S. Pankov and E. J. Bayalieva; and advancements in algorithmic problem-solving by A. Taneja and A. Kothari.

In the second part of the volume, authors from Belarus, Japan, Palestine, and Latin American countries share their experiences, news, and approaches. K. Mirjalali and A. Behjati presented an IOI project report on enhancing the Task Preparation System, while A. Yusubov discussed updates to the official IOI website.

We extend our deepest gratitude to everyone who contributed to this volume, especially the authors and reviewers. Their dedication and hard work in writing, reviewing, and refining the papers have been crucial in creating this exceptional collection. We also warmly thank all the participants, speakers at the conference, and members of the IOI community. We hope this has been a memorable and enriching experience for all involved.

Editors

Giochi di Fibonacci Year II: Competitive Blocks Programming for Young Students

Giorgio AUDRITO¹, Sara CAPECCHI¹, Madalina G. CIOBANU²,
Luigi LAURA³

¹*Department of Computer Science, University of Torino, Italy*

²*University of Salerno, Italy*

³*Uninettuno University, Rome, Italy*

e-mail: {giorgio.audrito,sara.capecchi}@unito.it, mciobanu@unisa.it,

luigi.laura@uninettunouniversity.net

Abstract. We organized the second edition of *Giochi di Fibonacci* (Fibonacci’s games), a programming contest for upper primary and lower secondary schools students; contestants compete in their own age division. The contest is organized in three phases, where the first one is based only on logical and algorithmical quizzes, whilst the other two deal with coding using a Blockly environment integrated in our contest’s platform. In this paper we report our experience and analyze the feedback collected from both students and teachers.

Keywords: programming contest, Olympiads in Informatics, peer education, programming training.

1. Introduction

The scientific-cultural side of computer science, also referred to as computational thinking, helps to develop logical skills and the ability to solve problems creatively and efficiently, qualities that are important for all future citizens.

The significance of incorporating computational thinking and programming into the curriculum of primary and lower secondary education cannot be overstated, a trend underscored by the notable success of initiatives like Bebras¹ (Dagienė, 2008). Dagienė *et al.* (Dagienė *et al.*, 2022) provide an insightful overview of how computational thinking is being adopted globally in primary education; the challenges and considerations surrounding the integration of informatics into primary education, from curriculum design to teachers’ perspectives, are thoroughly examined in (Dagienė *et al.*, 2019).

¹ <https://www.bebras.org/>

A wide array of methodologies has been explored to facilitate this integration, including unplugged education (Pluhár, 2021; der Vegt, 2016), employing platforms like Scratch (Fagerlund et al., 2020), and introducing robot programming (Kanemune et al., 2017) and LEGO robotics (Souza et al., 2018). Additionally, gamification strategies have been effectively utilized to engage students (Combéfis et al., 2016), as highlighted by Dolinsky (Dolinsky, 2022).

In Italy, according to the current national curricular recommendations, computer science related topics are included in broad areas of cross-disciplinary key citizenship digital competence area or general technology subject area and “*whenever possible, students can be introduced to simple and flexible programming languages in order to develop a taste for creation and for the accomplishment of projects [...] and in order to understand the relationships between source code and resulting behavior.*”² As a consequence, the implementation of the curricular recommendations is delegated to self-motivated teachers who propose valuable initiatives also in informatics education. In this context, initiatives bringing students and teachers closer to programming play a very important role.

The “Giochi di Fibonacci” (Fibonacci’s games) (Audrito et al., 2023) are a programming contest in upper primary and lower secondary education, with each age division competing separately, aimed at enhancing students’ computational thinking and programming skills. In the first edition (last year) “Giochi di Fibonacci” were divided in three distinct stages, where the initial stage is solely based on logical and algorithmic assessments, similar to Bebras, while the other two phases involve the use of coding, either via Scratch or a specially designed simplified pseudo-code programming environment catered to this competition. The lessons learned in the first year edition (Audrito et al., 2023) have been the base for the design of this year’s edition, that we discuss in this paper.

On the bright side, as we decided, we improved the competition by focusing only on a Block programming language (based on Blockly³); on the dark side, we experienced a huge bug in the newly designed but not enough tested new platform for the second phase. It would be nice to say that we corrected the bug immediately and it not affected the competition but, to be fair, it had definitely a non neglectable impact.

Thus, in this paper we report our experience in running (a buggy version of) an improved Giochi di Fibonacci competition. In Section 2, we recall the structure of the competition and report the emergence of the bug and the countermeasures taken. Then, in Section 3 we discuss the results of the feedback obtained, using questionnaires, from students and teachers. Finally, Section 4 addresses final remarks and conclusions.

² https://www.miur.gov.it/documents/20182/51310/DM+254_2012.pdf

³ The choice of relying on Blockly instead of the more well-known similar platform Scratch is due to its easier integration with our website. Blockly can be tried online at: <https://blockly.games>

2. Giochi di Fibonacci

A detailed description of the (first edition of the) *Giochi di Fibonacci* is discussed in (Audrito *et al.*, 2023). In this section, we briefly present the three phases, that mimic the structure of the Italian Olympiads in Informatics (Audrito *et al.*, 2021). The first phase does not involve coding, and problems proposed are similar to the ones of Bebras, thus aiming to involve students from that competition to participate. The first edition phases were organised in the following way:

- **First phase:** logical, algorithmic and program comprehension quizzes, similar to Bebras but with more weight on “program comprehension” skills.
- **Second phase:** simple programming tasks (in Scratch or conventional languages).
- **Third phase:** more difficult programming tasks.

In the following sections we discuss each of the phases of the second edition, describing mainly the changes from the first edition, according to the *lessons learned* from (Audrito *et al.*, 2023).

2.1. First Phase

The first phase aimed to address logical and algorithmic thinking. Since the first phase last year was successful and well-received, we maintained the same structure for the test, with only a minor difference: presenting program comprehension tasks as block-based code, instead of as flowcharts.

We designed the phase acknowledging the diversity in school resources, that is, not all educational institutions are equipped with either fixed or mobile computer labs. Consequently, we offered schools the flexibility to select their preferred competition method: either via traditional pen-and-paper format or through an online platform. This approach ensured equal opportunity for all participants, regardless of their access to technolo-

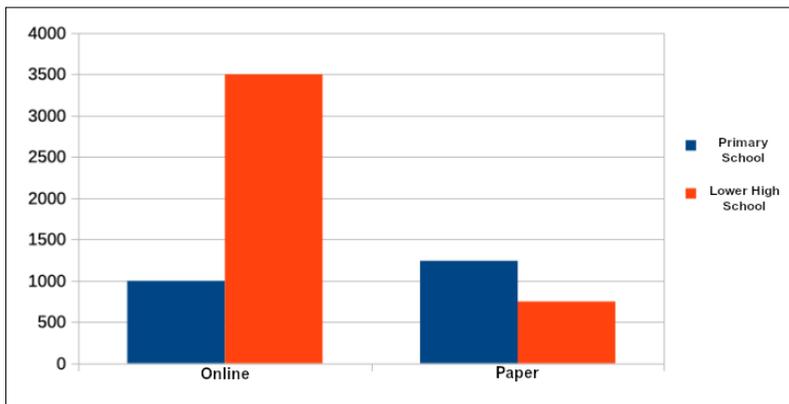


Fig. 1. Participating students by type of test.

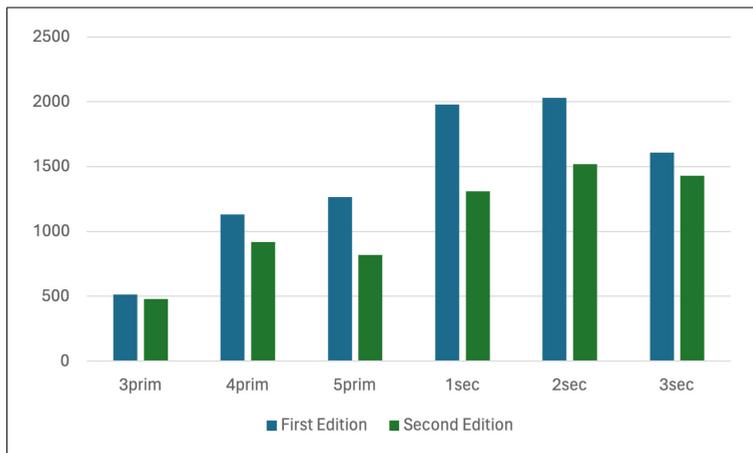


Fig. 2. Participating students by years, comparing the two editions.

gy. By analyzing the choice of testing methods across different educational levels (see Fig. 1), we see that in lower high schools there was a greater preference for the online test over the traditional pen-and-paper format. Conversely, in primary schools, the opposite trend was observed, with a higher prevalence of the pen-and-paper method over online testing. This may suggest a more widespread availability of computer labs in lower high schools w.r.t. primary schools.

6616 students participated in the first phase: 2352 primary school students and 4264 lower secondary school students. 2216 took the test on paper, while 4400 took it online. A total of 18 lower secondary school students obtained a full score of 50/50, while only one primary school student obtained a full score of 45/45. The scores were overall a bit lower than what we aimed: the average score was 14.7 for primary school and 17.8 for secondary school, while the median score was 15 for both primary and secondary school.

Fig. 2 shows the number of participating students divided by year of study, for both editions: 3prim, 4prim and 5prim are the last 3 years of primary school, while 1sec, 2sec and 3sec represent the 3 years of lower high school. We can see that the participation has been similarly distributed in the two editions, but lower overall for the second edition. This may have been a consequence of the exercises being too difficult last year, especially in the second phase.

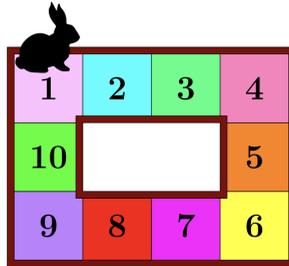
2.1.1. Primary School

The test for primary school contained 9 questions to be solved in 50 minutes, divided into three parts as follows:

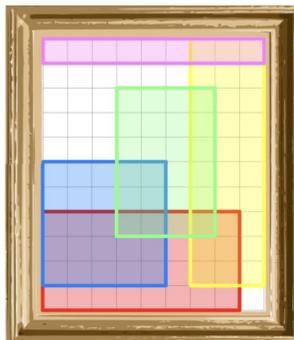
- Logical thinking questions (4 multiple-choice questions).
- Algorithmic thinking questions (2 open-ended numeric questions).
- Questions on interpretation of block programs. (3 multiple-choice questions).

In all three parts, the questions were ordered by increasing difficulty. Few sample questions follow.

Question 2. The rabbit game board contains 10 squares, numbered from 1 to 10, with square 10 adjacent to square 1. Tip-Tap starts from square 1 and, in 4 subsequent turns, advances by 9, by 4, by 8 and finally by 7 boxes. In the end, which square is it on? Multiple Choice Answers: A) 4, B) 9, C) 1, D) 5, E) 3

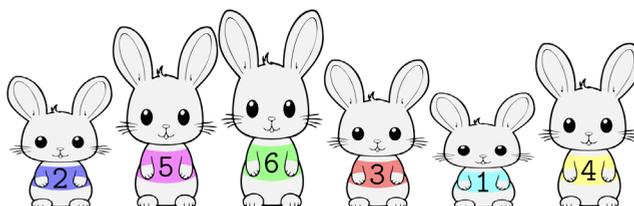


Question 3. Tip-Tap received 5 rectangular postcards, measuring (in cm) 8×4 , 5×5 , 3×10 , 9×1 and 4×6 . Now he wants to buy a rectangular bulletin board in which to put the postcards, possibly overlapping but not rotated. For example, this is a possible bulletin board of area $9 \times 11 = 99$ that contains postcards.



To save money, Tip-Tap would like to purchase the smallest bulletin board possible: how much is the smallest area (in cm^2) of a bulletin board that can contain all postcards? Multiple Choice Answers: A) 80, B) 90, C) 0, D) 10, E) 85

Question 5.1. Tip-Tap friends all lined up for the count! Each of them has a different height, written on their t-shirt.



In one move, *Tip-Tap* can choose two consecutive friends and remove the shorter of the two from the line. What is the minimum height of a friend who can remain in line after 5 moves?

Question 5.2. What is the minimum height of a friend who can remain in line after 4 moves?

Question 6. In what order should these instructions be placed to obtain the number 6 in the variable x ? [We show the english translation on the right side of the original image of the contest.]

- | | |
|---|---|
| 1: il contenuto di x diventa 2 | 1: the content of x becomes 2 |
| 2: il contenuto di x viene moltiplicato per 2 | 2: the content of x gets doubled |
| 3: il contenuto di x aumenta di 1 | 3: the content of x gets increased by one |

Multiple Choice Answers: A) 3,2,1; B) 2,3,1; C) 2,1,3; D) 1,2,3; E) 1,3,2

Fig. 3 reports the response distribution for the different questions. The second question had the highest number of correct answers. Conversely, question 5.2 recorded the highest count of incorrect responses. Moreover, question 3 and question 6 stood out as the ones with the highest frequency of unanswered responses, possibly suggesting that their statement was hard to understand.

2.1.2. Lower High School

The test for lower high school students consisted of 10 questions to be solved in 50 minutes. The questions were multiple choice or numerical open response, and were divided into three parts:

- Logical thinking questions (3 multiple-choice questions).

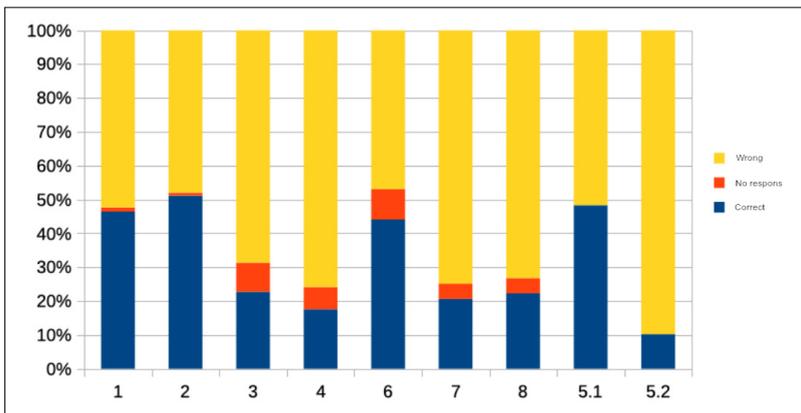
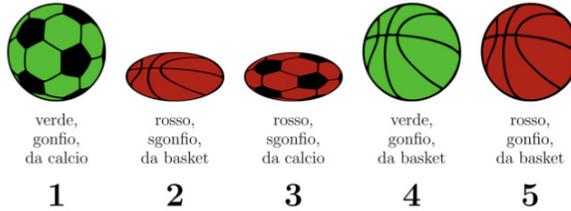


Fig. 3. Response distribution in primary school.

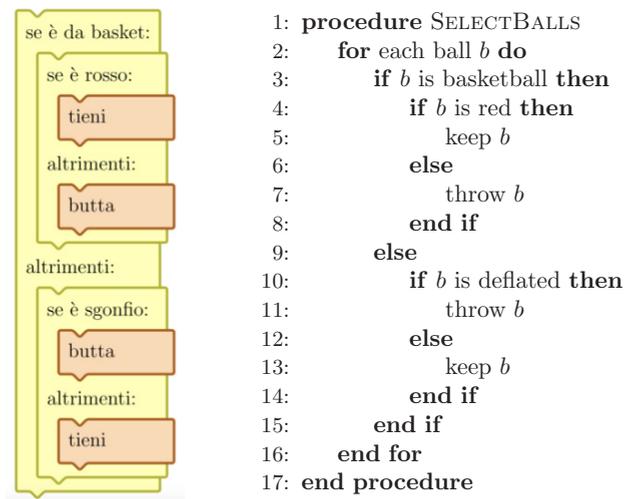
- Algorithmic thinking questions (4 open-ended numeric questions).
- Questions on interpretation of block programs. (3 multiple-choice questions).

In all three parts, the questions were ordered by increasing difficulty. Few sample questions follow.

Question 7. *Tip-Tap has to decide whether to throw away some of these balloons:*



To do this, he follows this procedure: [Pseudocode in english is shown to the right side of the original image of the contest]



Which balls does Tip-Tap throw? Multiple Choice Answers: A) 2,3,4; B) 1,4; C) 3,4; D) 1,5; E) 1,2,5

Question 4.1 was the question with the greatest number of correct answers, and question 4.2 was the question with the greatest number of incorrect answers. Those questions were identical to questions 5.1 and 5.2 reported previously for primary schools. Question 7 was the question left blank by the greatest number of students. Question 7 might have been left blank because it was a question that required some competence on block programming. However, question 8 is also a question on block programming and was the question with the second highest number of correct answers, so other factors might be into play.

2.2. Second Phase

In this second edition, we significantly changed the second phase, leading to what we evaluated as a considerable improvement. This change stems directly from the valuable feedback received from teachers last year. In particular, in the first edition, the second phase was perceived too difficult and the web system hard to use. Many students lacked even rudimentary computer programming skills, including familiarity with block-based programming concepts. This deficiency is largely attributed to the minimal exposure to computer science in the standard education curriculum in Italy, particularly up to the age of 14. Addressing this disparity is one of our primary objectives: to introduce students to the world of information technology at an earlier stage, fostering a deeper and more meaningful engagement with the subject matter.

In the previous edition, we allowed the use of traditional languages or Scratch. For each supported language (including Scratch), they were provided with a starter file containing the code handling input and output, and the students had to implement only the main procedure. It was up to the students to edit and run the program with external software and tools. The obstacle in solving the exercises was not only in the difficulty of the exercises but, as emerged from the feedback, also in the method provided to perform the exercises. In particular with Scratch, the devised system required to interact back-and-forth with two separate websites (the scratch website and the contest website), and this increased the barrier of accessibility to the contest.

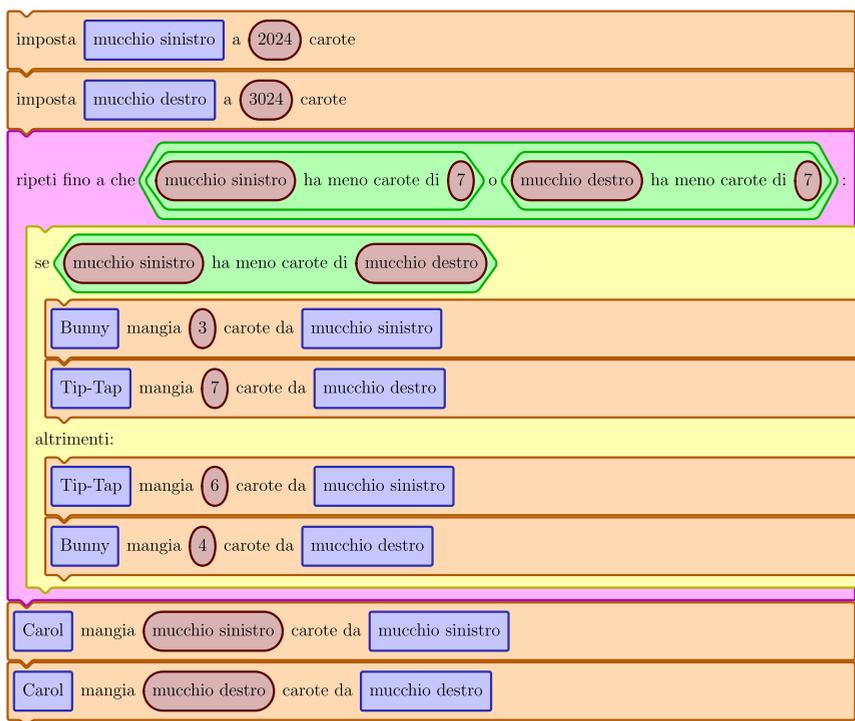
This year we added an extension to our platform in order to integrate Blockly and avoid the above mentioned issues. The test had the same structure for both primary and secondary schools, and consisted of 6 questions to be solved in 100 minutes. The questions were divided into two parts:

- 3 questions on interpretation of block programs.
- 3 block programming interactive questions.

Each block programming question required to write a single blockly program, which was then evaluated on three levels of increasing difficulty. All questions were accessible from a same page in the contest website, which was based on the same QuizMS platform used on the first phase, but with an extension providing Blockly support. In both parts, the questions were ordered by increasing difficulty. The set of questions was different for primary and secondary schools, as detailed in the following.

2.2.1. Primary School

Question 3. *The rabbits at the Fibonacci farm have prepared two huge piles of carrots. At the beginning the left pile contains 2024 carrots, while the right pile contains 3024. Bunny, Tip-Tap and Carol eat them following this procedure: [Pseudocode in english is shown below the original image of the contest]*



- 1: Set *left pile* to 2024 carrots
- 2: Set *right pile* to 3024 carrots
- 3: **repeat**
- 4: **if** *left pile* has fewer than 7 carrots **or** *right pile* has fewer than 7 carrots **then**
- 5: **if** *left pile* has fewer carrots than *right pile* **then**
- 6: *Bunny* eats 3 carrots from *left pile*
- 7: *Tip-Tap* eats 7 carrots from *right pile*
- 8: **else**
- 9: *Tip-Tap* eats 6 carrots from *left pile*
- 10: *Bunny* eats 4 carrots from *right pile*
- 11: **end if**
- 12: **end if**
- 13: **until** *left pile* has fewer than 7 carrots **or** *right pile* has fewer than 7 carrots
- 14: *Carol* eats *left pile* carrots from *left pile*
- 15: *Carol* eats *right pile* carrots from *right pile*

How many carrots does Carol eat? Multiple Choice Answers: A) 0, B) 4, C) 1, D) 2, E) 8

Question 4.1. *Tip-Tap* needs to sort out his old collection of N footballs. Since he doesn't have room for all of them, he decided to keep all the inflated soccer balls and basketballs, while throwing away the deflated basketballs. To do this, *Tip-Tap* can perform the following actions:

- *Keep*: put the next ball on the shelf.

- *Throw*: throw away the next ball in the bin.
- *Soccer ball*: true if the next ball is a soccer ball.
- *Inflated balloon*: true if the next balloon is inflated.
- *Finish*: finish putting the balloons away.

Write a program that allows Tip-Tap to sort all his balloons! (see Fig. 4)

Fig. 5 reports the response distribution for the different questions. Question 4.1 had the highest number of correct answers, while question 3 had the highest number of blank answers. On the other hand, questions 5.3, 6.2 and 6.3 were not solved by any student.

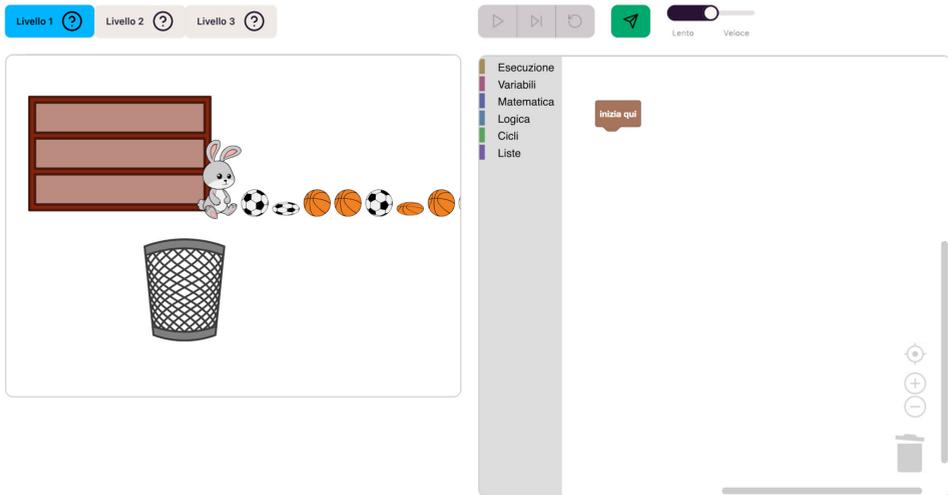


Fig. 4. Second Phase – Primary School – Question 4.1.

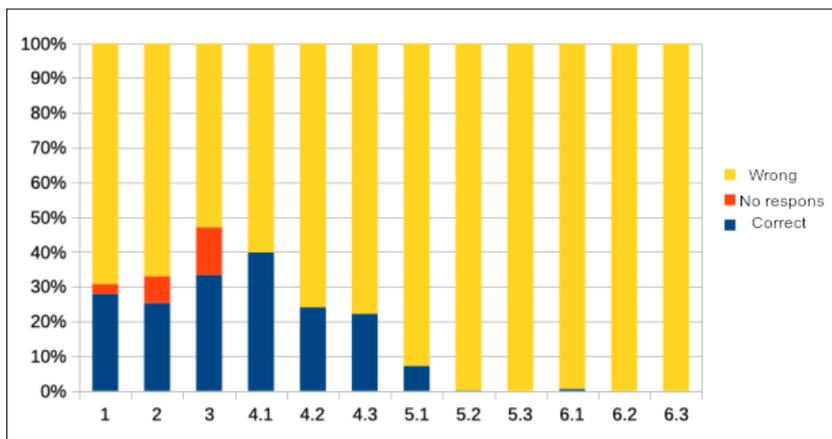
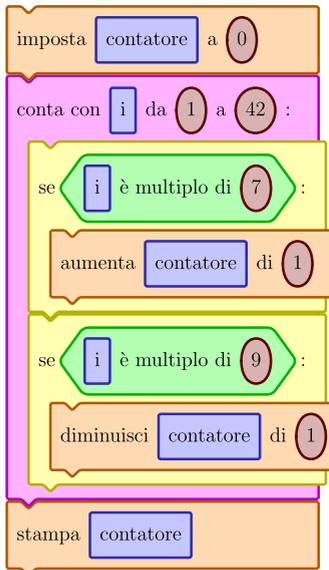


Fig. 5. Second Phase – Response distribution in primary school.

2.2.2. Lower High School

Question 3 While fixing up his attic, Tip-Tap came across a very old programming book. On the first page he found the following procedure: [Pseudocode in english is shown to the right side of the original image of the contest]



```

1: Set counter to 0
2: for i from 1 to 42 do
3:   if i is a multiple of 7 then
4:     Increase counter by 1
5:   end if
6:   if i is a multiple of 9 then
7:     Decrease counter by 1
8:   end if
9: end for
10: Print counter
    
```

Unfortunately the next page is ruined so Tip-Tap can't understand which number will be printed at the end... help him! What number is printed from the last block? Multiple Choice Answers A) 2, B) 6, C) 10, D) 4, E) 0

Question 6.3 Tip-Tap wants to build a new shed for his farm! First, he needs to build the two supporting columns: one on the left S centimeters high, and one on the right D centimeters high. To do this he plans to stack some blocks taken from a construction set, composed of a single block for every possible height between a minimum of 1 centimeter and a maximum of M centimeters, and which in total reach exactly the total height of the two columns. Now you can do these operations:

- right column height: the current height of the right column.
- left column height: the current height of the left column.
- stack block i on the right: adds the block i centimeters high to the right column, if it has not already been used.
- stack block i on the left: adds the block i centimeters high to the left column, if it has not already been used.
- finish: complete the columns and build the canopy.

Help Tip-Tap complete the shed as planned! (see Fig. 6)

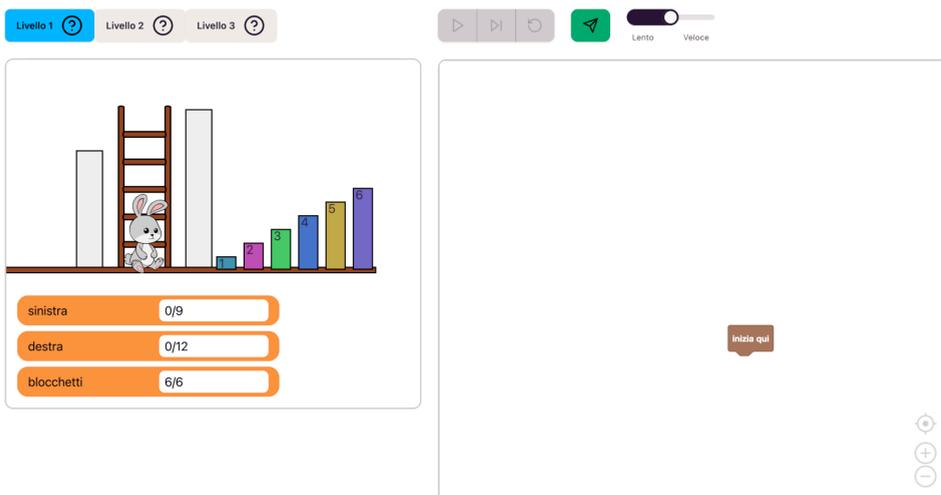


Fig. 6. Second Phase – Lower High School -Question 6.3.

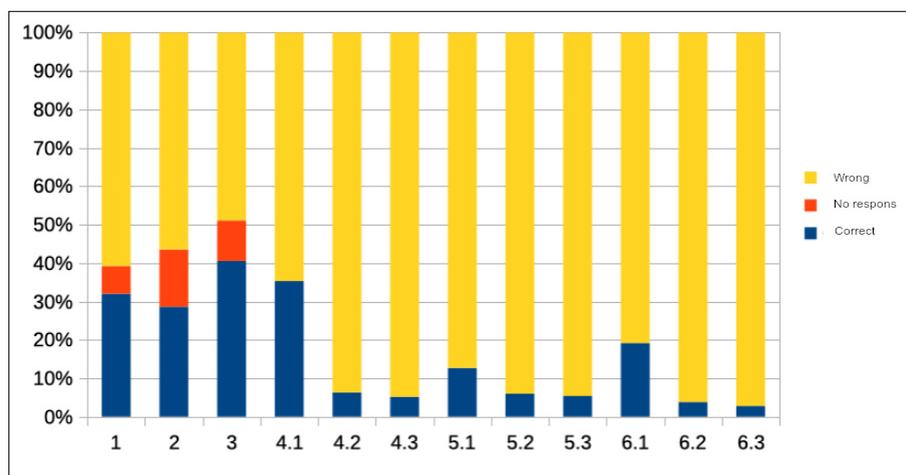


Fig. 7. Second Phase – Response distribution in lower high school.

Fig. 7 reports the response distribution for the different questions. Question 3 had the highest number of correct answers. Conversely, question 2 (which was the same as question 3 for primary school) recorded the highest count of wrong answers. Question 6.3 was the level solved by the lowest number of students.

2.2.3. The Hunt for the Bug

The second phase was scheduled to be held during March 13th, 2024. Each school could choose the two-hour window for the test at any time during the day, to accommodate

their needs. Few schools asked to have the test on the 12th or 14th due to logistic reasons and were granted this possibility. The online platform used for the test allowed participants to directly register for the test, using a passphrase given by their teacher. Their answers in the test were locally saved in the browser's cache, with Blockly questions being visualized and scored directly in the browser, so that unreliable connection with the back-end could not impair the students' experience. The website would still try to regularly synchronize the students' answers with a back-end, leveraging the Google Firebase development platform. The teacher had also access to a managing dashboard, where he could see the results of all of its students in real-time, as they were saved in the Firebase back-end.

The contest operations seemed to proceed smoothly both on the 12th and on the 13th morning. Around noon, a teacher wrote us to report that the results she was seeing in the teachers' dashboard did not match the results the participants were seeing in their webpage, and were actually consistently lower. We first thought that could be due to errors from the teachers' side, or network problems from the school: but with some interaction with the teacher, thanks to her cooperation, we realised that indeed the result synchronisation was not working properly. Quickly we realised that this was due to a misconfiguration in Firebase, that rejected updates above a certain maximum size. That maximum size was not a problem for classic questions, that can be encoded with few bytes, but it was relevant for Blockly tasks, for which the whole Blockly program was saved and easily reached the maximum allowed size.

By 1 PM the allowed size was raised, preventing the problem from happening for the 15% of schools that still had to start the test. However, many schools were affected severely, as the students' answers on Blockly tasks were mostly not saved. We immediately gave instruction to the teachers on how they might still recover results through the browsers' cache: 30% of schools managed to reconstruct the students' scores by this means, and 13% of schools declared that their students were not affected by the bug (probably because of low results on Blockly tasks). The remaining 42% of schools were affected by the bug and unable to reconstruct the scores.

This bug was a very unfortunate experience, making it hard to properly select the best students for the final phase. In order to mitigate its consequences, we opted for a selection criteria heavily grounded on a per-school basis, since students in a same school were identically affected by the bug. We admitted the first student of each school, regardless of his score, and the second student provided he reached a given minimum score. Given the very good results we obtained in the final phase, we believe that this mitigation strategy worked reasonably well.

Of course, the bug had an impact on the impression of the contest to the affected parties. However, the bug did not affect the test experience per se, during which the student would obtain the correct feedback from the system and could enjoy the challenge regardless. From the feedback gathered, we believe that only few of the schools were significantly unhappy of the experience, with most of them being instead supportive and understanding.

2.3. Third Phase

Since last year the third phase seemed too difficult for primary school students, only lower high school students participated this year in the third phase. The test consisted of 4 block programming questions to be solved in 3 hours. Questions were sorted by increasing difficulty. The problems were the following:

Question 1. *Tip-Tap loves chocolate, so he bought himself a chocolate bar made of $N \times M$ squares. His K farm-mates would also like to eat chocolate, and Tip-Tap is too good to not give them some! Then, for K times he breaks the tablet into two rectangular parts, not necessarily equal, and gives one of the two to one of his K companions. At the end of the process, he will keep the remaining last piece for himself.*

The tablet can only be broken along the edges of the squares, horizontally or vertically, so as not to divide any square in two. Furthermore, once a part is broken, it is immediately taken and eaten by a friend without giving him the opportunity to break it further. Tip-Tap would like to know how to break the bar K times so that he can keep the most possible pieces at the end.

You can use these blocks:

- width: the current width of the tablet.
- height: the current height of the tablet.
- companions: the number of companions who still ask for chocolate.
- break x squares horizontally: break the tablet horizontally, leaving x rows for a partner:
- break x squares vertically: break the tablet vertically, leaving x columns for a partner:
- finish: eat the remaining chocolate.

Help him break the bar K times while keeping as many squares as possible! (see Fig. 8)

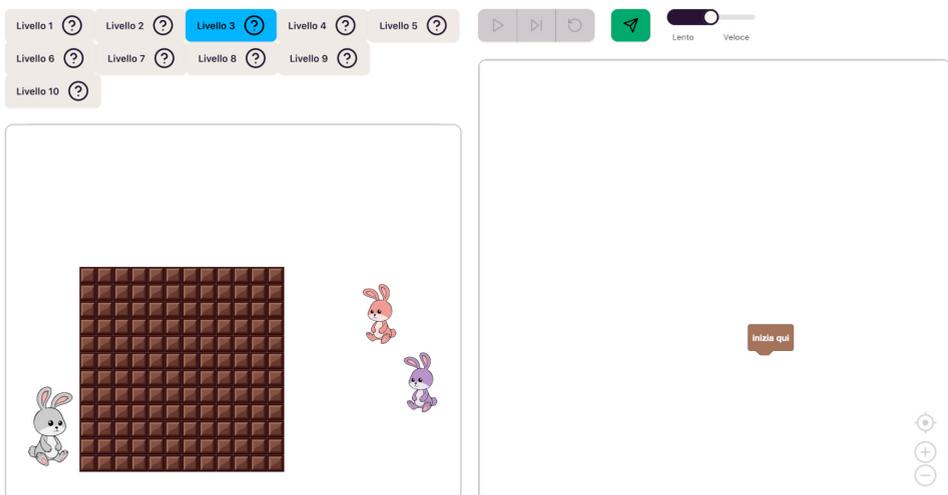


Fig. 8. Third Phase – Question 1.

Question 2. *The rabbits of the Fibonacci farm have just bought a new very efficient electric car! They can't wait to try it, so they organize a test trip to the nearby mountains. The route they want to take is made up of uphill and downhill sections. Along the route there are N charging stations where you can stop, at different heights. The car uses one unit of energy to climb 1 meter in altitude, while it gains one unit of energy by descending 1 meter in altitude, and it does not need energy to advance on flat terrain. Unfortunately, the machine starts without energy, and to recharge it can wait a minute at one of the charging stations for each unit of energy it wants to obtain at that point. You can use these blocks:*

- *N : The length N of the path.*
- *energy: the current amount of energy.*
- *altitude of charging station i : the altitude of the i -th charging station on the route.*
- *advance: continue your journey to the next column, if you have enough energy.*
- *recharge for x minutes: wait x minutes at a charging station to recharge x units of energy.*
- *finish: turn off the machine.*

The rabbits start from charger 1, and must arrive at charger N . Plan the trip to the mountains, ensuring that the car does not stop before arriving! (see Fig. 9)

Question 3. *Carol dropped her calculator, and now it doesn't work as it should! The only working keys are $-$, \times , 1 and 2. To use the calculator she is forced to start from either number 1 or number 2 (by pressing the corresponding key), and apply one of the 4 possible operations that still work, zero or more times:*

- *subtract 1;*
- *subtract 2;*
- *multiply by 1;*
- *multiply by 2.*

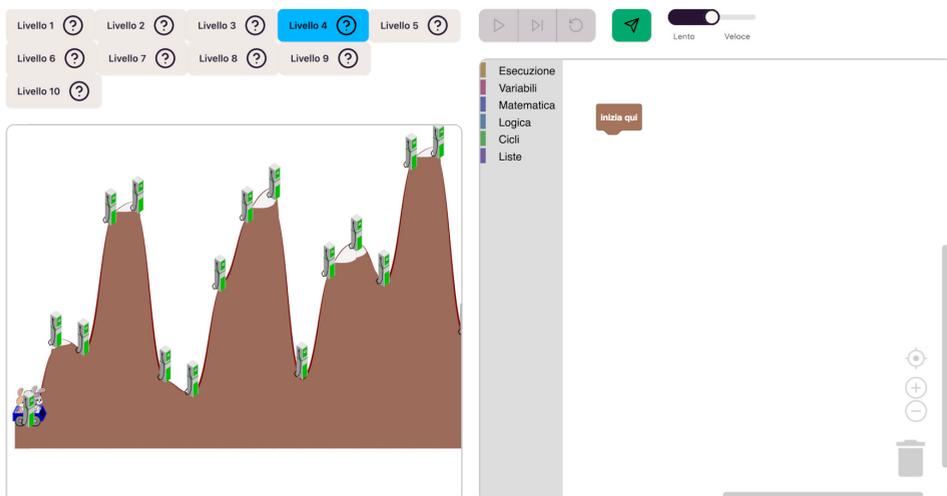


Fig. 9. Third Phase – Question 2.

To prank his friends, she would like to reach number N on the calculator. How many operations must be performed at least to achieve this? You can use these blocks:

- N : the number N it wants to reach.
- ends in x operations: reports that it is possible to reach the number N in x operations.

Note that you are not asked to reconstruct the operations to be performed: just calculate the necessary number of operations! (see Fig. 10)

Question 4. The brand new SuperBunny video game is finally on the market! Bunny, the protagonist of the video game, in each level must overcome N obstacles numbered from 1 to N . On each obstacle there are two platforms (at different heights) on which Bunny can jump: the obstacle number i is made up of a higher platform which is at a height of A_i meters, and of a lower platform at a height of B_i meters.

Bunny starts from the ground at height 0 and must first jump onto obstacle number 1 by choosing one of the two platforms. Once he reaches obstacle 1, he will choose one of the two platforms of the next obstacle, 2, and jump onto it. The objective of the game is to overcome all the obstacles in order up to obstacle number N . Even though Bunny can choose which obstacle platform to jump onto each time, not all jumps are the same! In fact, the bigger the jump, the longer it takes to do it. To jump from the platform at height h to a platform on the next obstacle at height k , Bunny will take an amount of seconds equal to the absolute difference between h and k . The total time taken to complete a level is the sum of the times taken in each jump. How many seconds does it take for Bunny to complete the level? You can use these blocks:

- N : the number N of obstacles.
- high platform i : the height A_i of the highest i -th platform.
- low platform i : the height B_i of the lowest i -th platform.

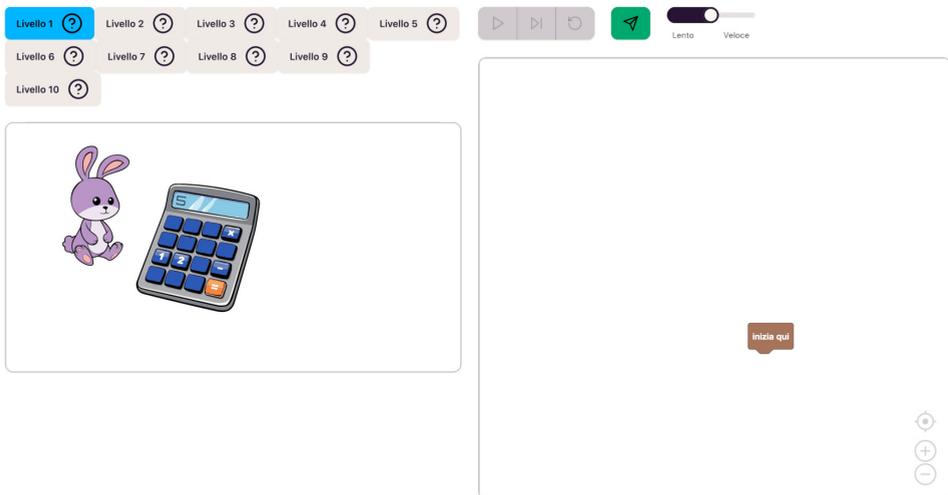


Fig. 10. Third Phase – Question 3.

- absolute difference between x and y : the absolute difference $|x - y|$ between x and y .
- minimum between x and y : the minimum value between two numbers x and y .
- finish in x time: reports that it is possible to reach the N -th obstacle in x time.

Furthermore, if you need it, you will have the possibility to write down a value of your choice on each platform (see Fig. 11) with these blocks:

- high platform i value: the value written on the i -th high platform.
- low platform i value: the value written on the i -th low platform.
- set value of high platform i to x : write the value x on the i -th high platform.
- set value of lower platform i to x : write the value x on the i -th lower platform.

79 students from 55 lower high schools participated in the third (and final) phase, of which 75 obtained a non-zero score. 39 students were awarded: 20 bronze, 11 silver and 8 gold medals. Fig. 12 reports the response distribution for the different questions. The

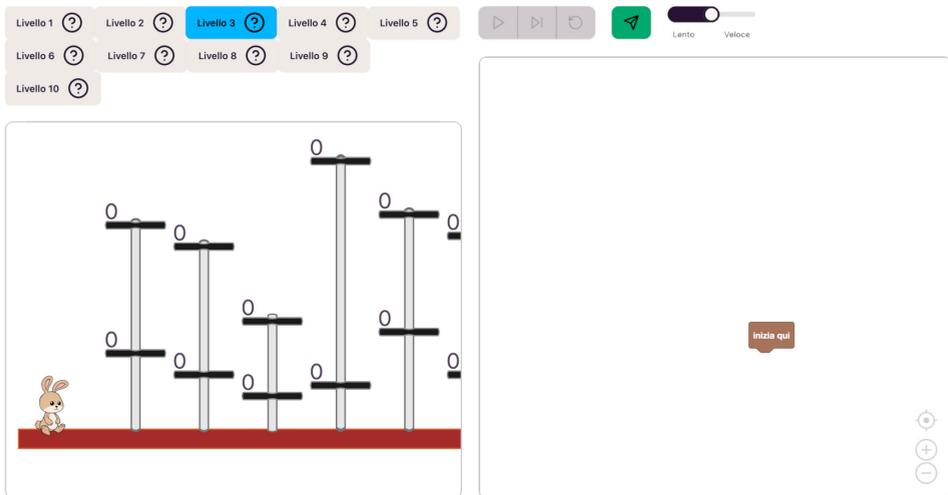


Fig. 11. Third Phase – Question 4.

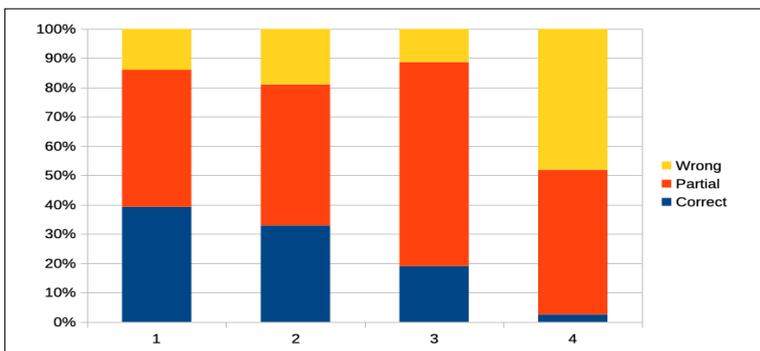


Fig. 12. Third Phase – Response distribution.

question ordering reflected the number of fully correct answers (10/10 levels done). Questions 1-2-3 all had a similarly low number of wrong answers, with most students completing at least some levels. Question 4 was much more difficult, as it required dynamic programming for a full solution, which was achieved only by two students.

3. Students and Teachers' Feedback

After the first phase of this years' edition, we collected feedback from teachers without involving students to maximize participation: asking teachers to make a high number of students fill out questionnaires could have been a barrier to participation. For the second and third phases of this year's edition, we instead performed a more systematic evaluation of the students' and teachers' feedback. For the second phase, the test was split in the two parts of program *comprehension* and coding, and we followed this split during feedback collection from students, thus asking the questions separately for the two parts. We adapted some of the question proposed by MEEGA+KIDS (Gresse von Wangenheim et al., 2020; Petri et al., 2019), a model to evaluate the quality of educational games used as instructional strategy for computing education. Participants (both teachers and students) were asked to assess their level of agreement with each of the following sentences using a 5-point Likert scale, ranging from "Strongly disagree" to "Strongly agree":

- S1 Questions were sufficiently clear.
- S2 The graphical presentation of the test was enticing.
- S3 Using the web platform to answer questions was not cumbersome.
- S4 The topics covered were interesting.
- S5 Questions were adequately challenging.
- S6 Questions were not repetitive and boring.
- S7 Students had fun working on the questions.
- S8 Solving the test made students feel satisfied.
- S9 The things learned from questions were satisfactory.
- S10 Students' results depended on their personal commitment and skill.
- S11 I'd recommend others to participate in the future.

Context information was also asked to teachers, to assess the backgrounds and initiatives the students had before the tests, including:

- C1 List the preparation activities the students had before the second phase test.
- C2 Detail whether the second phase bug impacted the scores of your students.

The information above described was collected in order to measure:

- RQ1 The impact of various preparatory activities on students' results.
- RQ2 The correlation between the teachers' impressions on their students and direct students' feedback.
- RQ3 The impact of the bug on students' satisfaction metrics.
- RQ4 The difference between the program comprehension and coding sections in satisfaction metrics.
- RQ5 Which factors are more decisive on the students' satisfaction.
- RQ6 Which factors are more decisive on the intention to recommend participation.

We focus our research on the second phase in lower high school, for which we have an almost complete feedback. Many students and teachers in primary schools did not give feedback, and that could sway the distribution of results. For questions considering students' results, we restrict only to the 58% of schools that were not affected by the bug or could reconstruct the scores, excluding the 42% of schools with unreliable scores.

3.1. Overall Results

Table 1 summarises the overall feedback gathered. We received feedback from 920 students out of the 1300 participating (71%), and 56 teachers out of the 66 participating (85%). Out of them, 445 students and 32 teachers come from schools able to handle the bug. For all questions, the standard deviation of responses was low, as most respondents gave intermediate agreement to most questions (neither agree nor disagree). The overall feeling was slightly more positive than negative for students (with an average score of about 3.2), and more decidedly positive for teachers (with an average score of about 3.6). Still, the results suggest that there is margin for improvement in future editions. It is worth mentioning that the best scores were seen in questions S6, S3, S10 for students, S2 and S4 for teachers. This suggests that the platform was effective, and that the novelty of tasks and their appropriateness to test students' skills were widely recognised. Indeed, also by looking at individual suggestions from participants, the main source for discontent was task difficulty, perceived too high by a good fraction of participants.

On the 14th of May we streamed, using YouTube, the Awards Cerimony⁴ that included also some video contributions by participants.

Table 1

Summary of the feedback collected, by scoring agreement from 1 (strongly disagree) to 5 (strongly agree). Students' feedback is split for test part A (program comprehension) and B (coding)

	students (A)		students (B)		teachers	
	<i>mean</i>	<i>st.dev</i>	<i>mean</i>	<i>st.dev</i>	<i>mean</i>	<i>st.dev</i>
S1	2.76	1.01	2.61	1.12	3.14	0.86
S2	3.31	1.15	3.31	1.18	3.91	0.80
S3	3.41	1.27	3.41	1.27	3.92	0.88
S4	2.83	1.13	2.87	1.15	3.96	0.90
S5	3.22	1.08	3.15	1.12	3.00	1.22
S6	3.42	1.15	3.25	1.17	4.16	0.88
S7	3.04	1.19	3.01	1.20	3.27	1.14
S8	3.39	1.25	3.37	1.23	3.23	1.09
S9	3.28	1.15	3.23	1.13	3.78	0.98
S10	3.73	1.09	3.59	1.17	3.23	1.09
S11	3.17	1.22	3.17	1.22	3.71	1.04
TOT	3.23	0.72	3.18	0.79	3.57	0.73

⁴ Available at: <https://www.youtube.com/watch?v=sNhQy4MW5zk>

3.2. Research Questions

We analysed the correlation matrix between all gathered questions, in order to answer RQ1–RQ6. A detailed report follows for each item.

3.2.1. *Impact of Preparatory Activities*

To assess RQ1, we measured the correlation between the main preparatory activities gathered in C1 and students' scores, for the 445 students with scores unaffected by the bug. The four main preparatory activities performed were:

1. Blockly-based training classes.
2. Scratch-based training classes.
3. Students' trying to solve the demo contest we prepared.
4. Teachers' explaining the solution of the demo contest we prepared.

Unfortunately, the correlation of scores with the preparatory activities turned out to be not statistically significant (ranging from -0.02 to 0.11). We believe, however, that this might be a consequence of the feedback gathering method used, and possibly of the bug swaying results. For the future, we plan to investigate further details about preparation activities (to estimate how long they prepared for and whether they had done other activities, courses, etc.).

3.2.2. *Correlation between Students and Teachers Impressions*

To assess RQ2, we considered all 920 feedbacks and looked at the correlation between questions S1–S11 asked to students and to teachers. These correlations were very low, ranging from -0.03 to 0.11 : this suggests that teachers' feedback is not a good predictor of students' feedback.

3.2.3. *Impact of the Bug on Student Satisfaction*

The impact of the bug on students' satisfaction (RQ3) was also not particularly relevant: correlation with questions S1–S11 ranged from -0.02 to 0.11 . This is probably due to the fact that feedback from students was gathered at the end of the test, and the students' experience during the contest was not affected by the bug. Most students learned about the bug much later, and that might have had an effect on feedback only if feedback was collected later in time.

3.2.4. *Difference between Program Comprehension and Coding*

The correlation between answers from students on test part A and B (RQ4) was, instead, very strong. It ranged from a minimum of 0.56 (for question S5) to a maximum of 0.69 (for questions S4, S8, S9, S10). Apparently, the students were not able to differentiate their experience on the two parts of the test, suggesting that they were balanced enough in their main qualities.

3.2.5. Main Factors for Students’ Satisfaction

To assess RQ5, we considered as main satisfaction metrics the answers to questions S6, S7, S8, S9. As main candidate factors for affecting students’ satisfaction, we considered their results, personal interest (S4) or presentation (S1, S2, S3). The correlation with results turned out to be unexpectedly low, ranging from 0.06 (S6 for part B) to 0.13 (S7-S8 for part A). The strongest correlation was with students’ interest (S4), ranging from 0.27 to 0.54 for the various pairs for questions, with an average correlation of 0.43. A lower but still meaningful correlation was registered with presentation, ranging from 0.21 to 0.48 for the various pairs of questions, with an average correlation of 0.36.

3.2.6. Main Factors for Recommending Participation

To assess RQ6, we first excluded students’ scores and having experienced the bug as main factors, since those had very low correlations (from -0.06 to 0.09) with the intention of recommending participation (S11). We then looked at the correlations with every other question asked, for both students’ and teachers’ (see Table 2). The correlations were all quite strong, but personal interest (S4) stands out as the best predictor for S11, with satisfaction (S7, S8, S9) following closely. Given that personal interest also has a very high correlation with satisfaction, it seems likely that it could be the main cause for S11 as well. It is worth mentioning that even though S6 (boringness) is also a measure of satisfaction, it had much lower correlation with S11. This may be caused by the fact that S6 had the highest scores overall, getting good agreement also for students which didn’t engage with the test. We also remark that the correlation was lowest with S3 (probably also because of its higher scores overall), and with S1 and S2 for teachers.

3.3. Lessons Learned

The feedback gathered mostly confirmed the choice of the new test structure and web platform design, which we plan to keep for the upcoming year. The overall satisfaction scores could still be improved, with one of the main causes being the perceived difficulty of tasks. However, given that this is a second phase for a selected audience, we do not plan to lower the difficulty of the tasks: instead, we plan to instruct teachers to

Table 2
Correlation of other questions with recommending participation (S11)

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
students (part A)	0.42	0.44	0.29	0.52	0.28	0.35	0.47	0.38	0.45	0.32
students (part B)	0.40	0.50	0.29	0.46	0.35	0.40	0.52	0.45	0.53	0.37
teachers	0.20	0.09	0.29	0.57	0.53	0.40	0.57	0.53	0.42	0.40

administer the test to fewer students, trying to restrict to only those truly interested and with the necessary skills to enjoy the test. In fact, this year the percentage of participants passed from phase 1 to phase 2 was particularly high (32%), with respect to the percentage from phase 2 to phase 3 (4%) and similar percentages last year (11% and 5%). Since participating schools handle the first round of selection autonomously, we can only suggest a course of action to teachers, but how to implement it is ultimately up to them. We reckoned that there is a widespread will from teachers to include as much people as possible in all phases of the competition. Next year, we will try to manage this impulse by clarifying that the second phase is only intended for the very best students of a school, and suggesting to use this phase as a guided class activity for the other students, with the teacher helping them in solving the tasks. We will also require a higher level of commitment for participation in the second phase, asking each student to register to our website in order to participate (this year, registration was required only for the third phase).

4. Conclusions

The *Giochi di Fibonacci* is a programming contest for upper primary and lower secondary students, structured to foster computational thinking and programming skills across separate age groups. In its inaugural edition last year, the contest was organized into three distinct stages. The first stage focused exclusively on logical and algorithmic quizzes akin to those found in Bebras, while the subsequent stages required participants to code, either in Scratch or in a simplified pseudo-code environment specifically developed for the competition.

This year, in the second edition, we improved the structure of the contest mainly by swapping Scratch for an integrated Blockly module, smoothing the user experience. Unfortunately, as detailed in Section 2.2.3, this also added a significant bug in the second stage.

Overall, the mostly positive feedback received largely supported our decision to adopt the new test format and web platform design, which we intend to maintain for the next year. While there is room to improve overall satisfaction ratings, the primary concern was the tasks' perceived difficulty. However, since this was a second phase intended for a selected group, we do not anticipate reducing the task difficulty. Instead, we aim to guide teachers to limit the number of students taking the test, focusing on those who are genuinely interested and possess the requisite skills to appreciate the challenge.

References

- Audrito, G., Ciobanu, M., Laura, L. (2023). *Giocchi di Fibonacci: Competitive programming for young students. Olympiads in Informatics*, 17, 19–31.
- Audrito, G., Di Luigi, W., Laura, L., Morassutto, E., Ostuni, D., *et al.* (2021). The Italian Job: Moving (Massively) Online a National Olympiad”. *Olympiads in Informatics*, 15, 3–12.
- Combéfis, S., Gytautas Beresnevičius, G., Dagienė, V. (2016). Learning programming through games and contests: Overview, characterisation and discussion. *Olympiads in Informatics*, 10.1, 39–60.
- Dagienė, V. (2008). The BEBRAS contest on informatics and computer literacy—students drive to science education. In: *Joint Open and Working IFIP Conference. ICT and Learning for the Net Generation, Kuala Lumpur*, pp. 214–223.
- Dagienė, V., Jevsikova, T., Stupurienė, G. (2019). Introducing informatics in primary education: curriculum and teachers’ perspectives. In: *Informatics in Schools. New Ideas in School Informatics: 12th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2019, Larnaca, Cyprus, November 18–20, 2019, Proceedings 12*. Springer, pp. 83–94.
- Dagienė, V., Jevsikova, T., Stupurienė, G., Juškevičienė, A. (2022). Teaching computational thinking in primary schools: Worldwide trends and teachers’ attitudes”. In: *Computer Science and Information Systems* 19.1, pp. 1–24.
- Dolinsky, M.S. (2022). Primary School Programming Olympiads in Gomel Region (Belarus). *Olympiads of Informatics*, 16, 107–123.
- Fagerlund, J., Hakkinen, P., Vesisenaho, M., Viiri, J. (2020). Assessing 4th grade students’ computational thinking through Scratch programming projects. *Informatics in Education*, 19(4).
- Kanemune, S., Shirai, S., Tani, T. (2017). Informatics and programming education at primary and secondary schools in Japan. *Olympiads in Informatics*, 11, 143–150.
- Petri, G., Gresse von Wangenheim, C., Ferreti Borgatto, A. (2019). MEEGA+, Systematic Model to Evaluate Educational Games. In: *Encyclopedia of Computer Graphics and Games*. Ed. by Newton Lee. Springer.
- Pluhár, Z. (2021). Extending computational thinking activities. *Olympiads in Informatics*, 15, 83–89.
- Souza, I.M.L., Andrade, W.L., Sampaio, L.M.R., Souto O Araujo, A.L. (2018). A Systematic Review on the use of LEGO® Robotics in Education. In: *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, pp. 1–9.
- der Vegt, W.V. (2016). Bridging the Gap between Bebras and Olympiad: Experiences from the Netherlands. *Olympiads in Informatics*, 10, 223–230.
- Gresse von Wangenheim, C., Petri, G., Ferreti Borgatto, A. (2020). MEEGA+KIDS: A Model for the Evaluation of Games for Computing Education in Secondary School. In: *RENOTE* (2020). <https://api.semanticscholar.org/CorpusID:225504503>



G. Audrito is involved in the training of the Italian team for the IOI since 2006, and since 2013 is the team leader of the Italian team. Since 2014, he has been coordinating the scientific preparation of the OIS and of the first edition of the IIOT. He got a Ph.D. in Mathematics in the University of Turin, and currently works as a Junior Lecturer in the University of Turin.



S. Capecchi received the M.Sc. degree in Computer Science at the University of Florence in 2002 and a Ph.D. in Computer Science (2003-2006) at the Department of Computer Science, University of Florence. She currently works as a assistant professor in the University of Turin. Her main research interests include computer science education.



M.G. Ciobanu is involved in Italian Olympiads in Informatics since 2009. She got a Ph.D in Computer Science in the University of Salerno, and in addition to being a high school teacher and also a research fellow in the University of Salerno.



L. Laura is currently the president of the organizing committee of the Italian Olympiads in Informatics that he joined in 2012; previously, since 2007, he was involved in the training of the Italian team for the IOI. He is Associate Professor of Theoretical Computer Science in Uninettuno university.

Popularizing Science and Science Competitions

Mohammad KAYKOBAD

*Department of Computer Science and Engineering
BRAC University, 66 Mohakhali
Dhaka-1212, Bangladesh
Fellow, Bangladesh Academy of Sciences
e-mail: kaykobad@bracu.ac.bd*

Abstract. This positional paper addresses the fact that while the present civilization is the outcome of hard labor of scientists and engineers, our society appears to fall behind in ensuring continuity of scientific endeavor through providing knowledge workers sufficient incentives for their sleepless nights and unselfish commitment to science and technology and to mankind. In this article we emphasize necessity of science and popularizing it to science students, scientific workers and common mass through innovative competitions and dissemination through mass media. This will keep young people with aptitude in science involved in the education of science and technology and thus help sustainable development of civilization through science and technology.

Keywords: science education, competition.

1. Introduction

This is not a research paper and it expresses personal opinions of the authors gained from experiences in science work, science education, science competitions and other events. Scientists and engineers have been playing significant role in creating the civilization of the present day from those of ancient times, when human beings could not establish their superiority over other species. Nevertheless, it does not appear that our society is yet ready to recognize duly the hard work knowledge workers put together to create a better world for all of us. Ease of life and societal recognition do not appear to be for these dedicated souls that work so hard to discover the truth of Nature and utilize it to the benefit of mankind. Even then each branch of science and technology is fathomed, new branches are created by the scientists and engineers although they are not as much recognized as workers of other fields of work. It is now becoming more and more difficult for new comers to extend the horizon of any branch of science and technology.

In the early 2000s Lyman and Varian (Lyman and Varian, 2003) of UC Berkeley estimated that total amount of new information stored annually on different media double approximately every three years, whereas the IDC's (International Data Corporation) (Gantz and Reinsel, 2012) regular updates of "Digital Universe" suggests that digital universe is doubling in size every two years. Moreover, the data generated worldwide was projected to grow from 33 zettabytes in 2018 to an estimated 175 zettabytes by 2025. IDC projects (IDC Study, 2020) that by 2020 the digital universe will reach 40 zettabytes (ZB), which is 40 trillion GB of data, or 5,200 GB of data for every person on Earth. This amount exceeds previous forecasts by 5 ZBs, resulting in a 50-fold growth from the beginning of 2010.

So it is clear that for processing so much of information we need many dedicated brilliant souls to work in many different fields and in their still greater number of intersections. Never ever human civilization faced a situation where it needed more knowledge workers than now. But unfortunately leaders of civilization do not seem to be appreciative of the need of development of science and technology for the civilization to sustain and prosper. This era has become a lot too much market oriented with commitment and dedication being replaced by market economy. Chat-GPT suggests that globalization and competition, funding cuts and financial pressures and increased emphasis on employability are the main factors driving education to market orientation. While there are potential benefits like responsiveness to labor market, innovation and efficiency and diversity of options there are concerns as well like narrowing of educational goals, inequality and commercialization of education. This has resulted in our bright students with praiseworthy aptitude in physics, chemistry, mathematics and other branches of science and technology opting for education in other branches that they did not pursue their education in only because now opportunities in those branches have been too much for them to take their eyes away and stick to the field they have proven capability and aptitude in. The society seems to have lost its grip to ensure its own survival and enrichment. However, responsibility of this self-destruction of the society should not only be borne by the overall leaders of the society, it will also lie on the shoulders of knowledge workers who are failing to convince the society of their usefulness for the survival of the civilization.

No field of human endeavors can be neglected even the least recognized, least appreciated, sufficiently ignored field like science and technology at least since without it no other field of human endeavors can survive not to speak of excel.

We know sports and entertainment are very popular in our society of *homo sapiens*, be it very brutal event like wrestling, boxing or Spanish style bull fighting (corrida de toros) although we are supposed to excel over other species especially in our brain power and not necessarily on our brutal power or strength. A sportsman of the highest order of popularity, say in football, is appreciated and awarded by our society with a sum of some 100 million Euro at the age of thirty (Rudling, 2024). Similar figures (Forbes, 2023; Wilson, 2024; Medium, 2024) can be found in tennis, golf, boxing and other sports. There are hundreds of the highest achievers in science and technology who are recognized for their achievement by Nobel Prizes only at the flag end of their life, when these earthly awards could hardly add to any enjoyment. Commitment of science

workers/educationists can hardly be exaggerated and the society seems to be indifferent to it. Can this difference in recognition be logically established? Dr Haim Ginott (Ginott, 1972) once rightly phrased, “Teachers are expected to reach unattainable goals with inadequate tools. The miracle is that at times they accomplish this impossible task.” And recognition of these committed efforts by the society has been well spelled out by Evan Esar (Esar, n.d.) in the following statement “America believes in education: the average professor earns more money in a year than a professional athlete earns in a whole week.” While ordinary people may fail to appreciate the necessity of science and technology for the progress of mankind should the leaders follow the footsteps of the popular trends or they should come forward to save humanity by appreciating the feat and encouraging knowledge workers?

2. What is Wrong?

In no case should popularity be the only consideration for degree of appreciation or recognition. The need of civilization, its priority should also be addressed for our survival. However, at the same time academicians have immeasurably failed to play their role in creating due appreciation of the society for knowledge works. In fact, can we confidently say that science has become popular among science students and science workers? Could we be confident that the best student in physics would prefer association of a renowned scientist than that of a well-known entertainer? If the words physics and scientist are replaced by corresponding entities of entertainment or sports would there be any doubt in the answer to the question? So we have to address the problem of making science events popular among science students at least as popular as entertainment or sports are popular to them. So far we have failed to do so. Let me give an example. Before 2010 in IOI (IOI, 2010) we the leaders and deputy leaders did not have clue as to the performance of our teams. Possibly even contestants did not know much about performance of other competitors. For the first time, in IOI’2010 held at the University of Waterloo there was something for the observers waiting outside for long 5 hours about performance of their team members. Whereas in sports performance of contestants are not only known to contestants but also to spectators for their enjoyment. Our inability of making the science-based contests more entertaining to spectators has resulted in making it difficult to organize science events due to absence of sponsorship even from technology-driven companies who would prefer spending their money to areas other than science. Science exhibitions and science week events appear much more thrilling for students with the presence of an entertainer or sportsman and not with the presence of a scientist. Students and young people, who will be earning their livelihood in the name of science, find sports people and entertainers more attractive than people who excel in their own professions. This has resulted not because these young people have wrong attitude rather because the whole society has been ignoring science workers to a level that it is extremely difficult to imagine that knowledge workers could be role models for young people. Scientists are often recognized at the national level with medals that possibly may not have any monetary value, although we do not fail

to recognize best cooks or even pickle makers with sizable cash prizes, as we do to accomplishment in other non-academic fields. It seems society feels scientists are priests, should be happy with fragrance less flowers and do not have any earthly needs to fulfill whereas achievers of any other field should be worshipped with flowers of fragrance and excessive money.

Popularity of sports is drawn from the fact that good performance is profusely rewarded. Performance is recorded, analyzed and grouped in many ways for the consumption of common people, and internet is overburdened with statistics of excessive orders. The information is readily available to people of all strata. Moreover, quiz competitions are organized that force these statistics and facts to be memorized by young people. Good performances in sports is so adequately rewarded that in a flash of a second the achiever becomes a hero and popular among common mass. The same is not true in case of academic competitions. We have not been able to popularize academic competitions even in the academia. I cannot be sure whether in any country we have recorded feats of meritorious students excelling in public examinations. Say in Bangladesh we used to evaluate exam papers on physics, mathematics or any other subject of our post grade 10 public examinations for some forty years. Nobody knows even the maximum marks obtained in a subject where some 200,000 to nearly 2 million students sat for the exam not to speak of other records. The students, who obtained highest marks during the 40 year period of exams in each year, are definitely gifted with praiseworthy aptitude in the relevant subject. Unfortunately examination authorities failed to appreciate the feat, and failed to inspire other students with this feat and possibly failed to create challenge in young people to beat the records. This feat in academic competitions is a lot more reliable than that in any other field where degree of uncertainty is much more than that in any conceivable academic competitions. For example, in a game like cricket a good batsman can score a double century in the first innings followed by a duck in the second one. Such variation is inconceivable in education. In sports like cricket we talk about records in the second innings or that of the 4th wicket. Can we imagine the corresponding statistics in the academic field? How many universities and departments having world class reputation can claim to have records of performance of students in various disciplines and in different combinations? Recently in the age of IT, ACM ICPC (baylor.edu, n.d.) is finding it extremely difficult to find names of teams that excelled or became world champions in early days of this competition – that too not of distant old days but as recent as 1978. We have immeasurably failed in recording the list of world champions. Can we name a form of sports in which we have failed to record the names? We have invented information technology and failed immeasurably to use this technology to the benefit and flourishing of our field, whereas people of other fields are using it to their advantage.

3. What is to be Done?

Sports organizers and people in entertainment are highly successful in popularizing their events amongst common mass. However, involvement of large sums of money contrib-

utes to its popularity. If football and cricket players would have received 100 times less money than they are getting now these events would have lost its glamour, and would not have received this much popularity. If winning Wimbledon title is a feat that can be recognized by giving a prize money of over two million pounds (Wimbledon, 2023), how much should the winning team of ACM ICPC or the champion of IOI team receive? Is the later a lesser feat than the former one? In cricket even cracking a board placed out of the field by a flying ball is rewarded with monetary prizes. Organizers of games and entertainment programs are very successful in attracting CEOs of large enterprises to perform their corporate social responsibility through promoting and sponsoring their events. Academic administrators should also be able to inspire and convince knowledgeable CEOs to invest their resources towards academic events like Olympiads, programming contests and other events that will sharpen and enrich skill of young people that will move the civilization forward. Personalities like Alfred Nobel (Nobel Prize) (nobelprize.org, n.d.) and John Charles Fields (Field's medal) (Wikipedia, 2024) have done it. Recently Clay Mathematics Institute initiated Millennium award, and the result was immediately visible. The Russian mathematician Grigori Perelman (Clay Mathematics Institute, 2024) solved the long standing Poincare Conjecture, although opted out of taking million dollar prize money possibly expressing sheer unhappiness against the indifferent attitude of the society to academic feats. In the past different scientific societies used to inspire academic and research excellence, if properly motivated, leadership of industries will come forward to help promote academic events. In order to arouse interest of the common mass events around these academic activities should be publicized over all possible media both electronic and printing. Moreover, interesting statistics related to these events should be made easily available to concerned people as sporting statistics are.

4. Events around Academic Activities

So if we want academic events to gain popularity we must create events around it. For example, International Collegiate Programming Contest was first televised at Stockholm creating a lot of thrill as to which teams are getting winning positions. First solution of a particular problem was awarded. In this way fastest solution time can also be awarded. Once upon a time it was difficult to believe that common people will be watching as boring a mental sport as chess is. Fortunately, even this sport with insignificant body movement could be made popular by televising it. Academic competitions should be opened for public enjoyment without first taking it for granted that there will be no interest among common mass to these mentally seriously involved games. Olympiads and programming contests should be publicized in mass media to attract attention of common mass so that they can appreciate commendable aptitudes of contestants in these events. International Olympiad in Informatics, International Physics Olympiad, International Mathematics Olympiad should be shown on TVs to inspire young people and find their heroes in the winners of these prestigious events. Achievements in this

area should be duly recognized and rewarded to inspire young people's interest in aptitude and skill building. Leaders and enthusiasts of programming contests had to survive the ignorance of the results of International Olympiad in Informatics until IOI2010 organizers decided in favor of spectators and made a scoreboard available for them in the same spirit as it is being done in ICPC World Finals. I am sure it was enjoyable to well-wishers of programming contests. How can a game be played with both spectators and players kept in complete darkness as to the results? We should begin to think how excellence in all our academic activities can be enjoyed not only by people of the field but by people at large.

5. Conclusion

Human beings are supreme among all creatures not because they have superior organs like legs, arms, body, ears or eyes. In fact we do not have any superiority in these limbs. Usain Bolt will find himself in hopeless condition running a race against a feverish leopard, nor the strongest of human beings will be comparable to an elephant in strength. Human beings are superior in their brain power to any other animal on earth. For development of each limb we need to do exercise. This is also true in case of development of brain for which exercise is thinking for solving problems. While exercise can only improve capacity of each of our organs only by a finite times say 2, 3 or 4, capacity of brain power can be improved a thousand fold. For a country like Bangladesh the only surplus is population which can be developed into human resources through exercise of their brains. This can be cost effectively done with the introduction of healthy competitions with lucrative prize money to attract the young people of the country.

We must find ways and means to bring academic competitions to common mass, arouse their interest in these competitions and possibly make all sorts of statistics available especially to young people in order to create avenues for them to excel in their knowledge. We should look for support of mass media so that information of academic events gain popularity in our society, and achievers get a better visibility in our society and the future generation does not opt for other areas of activities whence they have the required aptitude and due recognition in the society. Leaders of our nations should be convinced of the usefulness of duly recognizing scientific feats and adequately rewarding them so that science and technology are not thought of as neglected areas of human endeavors.

This paper is attempting to initiate a discussion on the matter that there must be very lucrative incentives for science workers to pursue science. Knowledge in every field of science is growing exponentially. Possibly 2000 years back the whole knowledge of physics could have been packed in a single volume. Now it will require a thousand volume, and a science worker with a limited longevity will have to surf in a vast ocean of knowledge to find avenues for its progress, if he is at all lucky to find. Should not we make science working full of incentives for young people to pursue?

References

- Lyman P., Varian, H.R. (2003). *How Much Information, 2003*. Technical report, UC Berkeley. Retrieved at Retrieved at 28 May, 2024. <https://www.ischool.berkeley.edu/research/publications/2003/how-much-information-2003>
- Gantz, J., Reinsel, D. (2012). *The Digital Universe in 2020: Big Data, Bigger Digital Shadows and Biggest Growth in the Far East*. Retrieved at 10 April 2024. <https://www.cs.princeton.edu/courses/archive/spring13/cos598C/idc-the-digital-universe-in-2020.pdf>
- IDC Study (2020). *IDC Study: Digital Universe in 2020*. Retrieved at 10 April 2024. <https://www.kdnuggets.com/2012/12/idc-digital-universe-2020.html>
- Rudling, M. (2024). *15 Highest-paid Footballers in the World in 2024*. Retrieved at Retrieved at 28 May, 2024. <https://squaremile.com/sport/highest-paid-football-players/>
- Forbes (2023). *The World's Highest-Paid Tennis Players 2023*. Retrieved at Retrieved at 28 May, 2024. <https://www.forbes.com/sites/brettknight/2023/08/25/the-worlds-highest-paid-tennis-players-2023/?sh=1fd8aecc6956>
- Wilson, J. (2024). *Highest Paid Golf Players of all Time: PGA Tour Money list*. Retrieved at Retrieved at 28 May, 2024. <https://www.radiotimes.com/tv/sport/golf/highest-paid-golf-players-all-time/>
- Medium (2024). *The Top 10 Highest Paid and Richest Boxers in the World*. Retrieved at 10 April, 2024. <https://medium.com/@filipinoonlinesportsbook/the-top-10-highest-paid-and-richest-boxers-in-the-world-f54284e5d903>
- Ginott, H.G. (1972). *Teacher and Child: A Book for Parents and Teachers*. New York, Macmillan, p. 15. Retrieved at 29 May, 2024. <https://wist.info/ginott-haim/67256/>
- Esar, E. (n.d.). *Quotations by Author: Evan Esar (1899–1995)*. Quotations Page. Retrieved at 28 May, 2024. http://www.quotationspage.com/quotes/Evan_Esar
- IOI (2010). *IOI 2010, Waterloo, Ontario, Canada, The 22nd International Olympiad in Informatics, August 14–21*. University of Waterloo. Retrieved at 28 May, 2024. <http://www.ioi2010.org/>
- baylor.edu (n.d.). Retrieved at 10 April, 2024. <http://cm.baylor.edu/ICPCWiki/Wiki.jsp?page=History>
- Wimbledon (2023). *The Championships, Wimbledon, 2023 Prize Money*. Retrieved at 29 May, 2024. https://www.wimbledon.com/pdf/The_Championships_2023_Prize%20Money.pdf
- nobelprize.org (n.d.). *The Nobel Prize*. Retrieved at 29 May, 2024. <http://www.nobelprize.org/>
- Wikipedia (2024). *Fields Medal*. Retrieved at 29 May, 2024. http://en.wikipedia.org/wiki/Fields_Medal
- Clay Mathematics Institute (2024). *The Millennium Prize Problems: Solved problems: Poincaré Conjecture*. Retrieved at 29 May, 2024. <https://www.claymath.org/millennium-problems/>



M. Kaykobad – is one of the pioneers of introducing Mathematics Olympiad, Informatics Olympiad and Science Olympiad in Bangladesh. He is also one of the pioneers of ACM ICPC having brought his team to the world finals for the 20 years. He is a distinguished professor of CSE Department of BRAC University. He is a Fellow of Bangladesh Academy of Sciences (BAS). He regularly writes science and education popularizing columns in popular daily newspapers of the country. He was awarded the BAS Gold medal for his excellence in research by the Prime Minister of the country and a Gold medal by the President of the country for his contribution to the culture of programming contest in the country. He also received Outstanding coach award at Honolulu, Hawaii in 2002, senior coach award at Warsaw, Poland in 2013 and ICPC Foundation life-time coach award in 2019 at Porto, Portugal. Dr Kaykobad has been able to inspire many of his undergraduate students to publish their research findings in international journals of repute. M. Kaykobad completed his PhD in 1986 in the fields of computational complexity from The Flinders University of South Australia. He has two Master's degree – one from Odessa State Maritime University, Ukraine, and the other from Asian Institute of Technology, Bangkok, Thailand.

Trends on Returning Contestants and Geography at the International Olympiad in Informatics

Ethan LEE¹, Tymofii REIZIN², Farrell Eldrian WU³, Filbert Ephraim WU⁴

¹*Stanford University, Stanford, California, United States of America*

²*Charles University, Prague, Czechia*

³*Massachusetts Institute of Technology, Cambridge, Massachusetts, United States of America*

⁴*Massachusetts Institute of Technology, Cambridge, Massachusetts, United States of America*

e-mail: eth127@stanford.edu, timreizin@gmail.com, farrellw@mit.edu, filbertw@mit.edu

Abstract. In this paper, we conduct several statistical analyses of IOI 2011 to 2023 performance data, with a focus on tracking returning contestants and identifying geographic trends. This paper identifies several properties of IOI performance data, such that it has strong internal validity while still being subject to random noise. Visualizations are presented throughout to aid the IOI community’s understanding of students’ competitive programming progress. Afterwards, the geographical analysis shows that countries’ IOI performance is correlated to demographic indicators such as population and the Human Development Index. It is, however, more strongly related to competitive programming interest in the country.

Keywords: global talent development, computer science education, academic competitions.

1. Introduction and Background

A. Background on the International Olympiad in Informatics

The International Olympiad in Informatics (IOI) is an annual algorithmic programming competition over eighty countries involved around the world. Based on a local selection process (often a series of competitions and training camps), each country sends a team of up to four pre-collegiate students to the IOI. At the competition, contestants are to attempt six algorithmic tasks, presented in the format of two competition days featuring three tasks each day, typically with one excursion day in between, and with each competition day lasting five hours. The current format, consistent from IOI 2011 until now, weighs each problem equally with a full score of 100 points, for a total of 600 points. Medals are then awarded to approximately one-half of all contestants, with gold, silver, and bronze medals awarded in an approximate ratio of 1:2:3. (IOI Regulations)

As a long-running international event, the IOI has substantial recognition as a venue to identify and develop pre-collegiate students' skills in computer science, with a focus on programming skills and algorithmic problem-solving. (IOI Syllabus) In many countries, the local competitions and training programs leading up to the IOI is seen as a major, and sometimes, the primary opportunity for high school students to explore computer science and related subjects. Numerous universities around the world also consider IOI results in their admissions, scholarship, and course placement processes. Therefore, it is of substantial interest to improve the IOI community's collective understanding of the dynamics of IOI results on a global scale.

The "International Olympiad in Informatics—Statistics" website stores IOI competition data ever since the contest's inception in 1989. The available data is extensive: for competitions starting 2010, the website provides the score for each contestant for each competition task, with contestants' results over multiple IOIs are linked through a personal profile page and a time-indexed "Past Participations" graphic showing progress over the past IOIs of all the contestants at a particular IOI. Other performance-related data such as a contestant's total score, medal, rank, and percentile are also provided, together with the gold/silver/bronze medal cutoffs at each year. Altogether, the historical IOI results provide a starting point for statistical analysis and data visualizations, towards yielding insights on contestants' IOI performance trajectories and countries' aggregate IOI performance.

B. Research Questions

This paper focuses on exploratory data analysis, particularly on returning contestants' performance over time and aggregate per-country performance.

For the returning contestant analysis, the objective is to broadly seek an understanding of the overall "IOI experience" of a multiple-time contestant, which in turn uncovers salient insights on the educational role of the IOI in developing talent in computer science. The primary statistical instrument in this analysis is the difference between a single contestant's performance in two consecutive IOIs, aggregated in different ways. This statistic is then analyzed using statistical techniques such as regression analysis to investigate and visualize different trends.

For the country comparison analysis, the objective is to showcase geographic trends in IOI results by identifying a range of indicators that correlate to a country's IOI performance, with the goal of better understanding the global landscape of advanced computer science education. In this analysis, we focus on the aggregate performance of a country over a decade-long period spanning 2011 to 2023 which is then regressed on a specific set of indicators and categorized by geographic region.

C. Data Processing Procedures

In this section, we describe the data processing produces used in the paper's analysis, which are aimed to reduce statistical noise caused by inherent variations in the IOI while

maintaining having sufficiently relevant data for a meaningful analysis. There are two aspects to this: first, subsetting the data, and second, calculating a standardized measure of contestant performance.

Our analysis uses IOI result data starting from 2011, for data reliability purposes. This is shortly after the IOI adapted its current “subtask” partial credit structure for most tasks. Before then, there were concerns regarding the reliability of the IOI’s partial credit system of the IOI (Verhoeff, 2006); the change in partial credit format can have effects on the overall distribution of IOI results due to different contestant approaches. The 2011 competition is also right after the IOI started publishing complete scores for all contestants rather than only the medalists. (The 2009 and 2010 IOIs also had an experimental 8-problem rather than 6-problem format.) For the country analysis, specifically, we only analyze with at least half participation over the IOIs from 2011 to 2023, as average performance metrics can be skewed by incomplete IOI teams.

As for contestant performance, we develop a scaled metric designed for consistency in interpretation across years, due to limitations in the comparability of the numerical IOI score (out of 600) across years. Specifically, using the numerical IOI score is extremely susceptible to changes in the contest’s difficulty and subtask structure. While the percentile ranking properly adjusts for difficulty, this metric is still sensitive to changes in the shape of the distribution of scores and does not distinguish between performance at the highest levels. (In 2023, for example, the entire gold medalist range, spanning over 200 points, spans fewer than ten percentile points.)

To alleviate the above concerns, we calculate a “scaled score” that is a piecewise linear interpolation. In our scale, we use a piecewise linear mapping where the lowest score is mapped to 0, the highest score is mapped to 6, and the bronze, silver, and gold cutoffs are mapped to 2, 3, and 4, respectively. This mapping can be interpreted as a reference to an “ideal IOI” where the bronze, silver, and gold cutoffs are at 200, 300 and 400 points, respectively, which is, after rounding to the nearest hundreds, the average cutoffs from 2011–2023. In addition, we assign the 25th-percentile (middle of the “no medal” group) score to a scaled score of 1, and the 96.67th-percentile (middle of the “gold medal” group) score to a scaled score of 5. This calculated statistic will hereafter be called the “scaled score,” which is different from either the percentile (out of 100%) or the raw score (out of 600).

2. Returning Contestant Analysis

In this section, we present the analysis of returning contestants at the IOI, along two aspects. First, we visualize the joint distribution of a returning contestant’s performance and improvement across multiple IOIs; from here, we draw inferences that characterize the nature of improvement in competitive programming at different skill levels. Second, we describe progress of contestants with multiple IOIs to better understand the nature of long-term learning in competitive programming. In doing so, we also demonstrate the “reversion to the mean” phenomenon common in statistics (Barnett et. al, 2005) as seen in IOI contestant data.

In total, there are 1038 participants with more than one IOI from 2011 to 2023, which provides ample data for the analysis in this section. The trends shown are mostly rather strong and unlikely to be affected by random noise in the data. The selection of this set of recurring contestants indeed may be biased due to different team selection policies across countries and other confounding factors, though we believe that statistical properties depicted are still meaningful. The analysis presented in Figures 4–6 also quantify and contextualize such bias.

A. Cross-distribution of Performance Across Multiple IOIs

As an initial analysis, we visualize the scatterplot (Fig. 1) of a multiple-IOI contestant’s performance over consecutive IOIs. For this analysis, we use the “scaled score” from 0 to 6, as described previously. So that the datapoints within each scatterplot are comparable to each other, we show separate plots depending on whether we are comparing the first/second or second/third attempts, and whether the contestant had two or 3+ IOIs. We do not analyze IOIs after the third due to small sample sizes in this group. These data divisions were chosen to balance each plot having sufficient data and a useful interpretation.

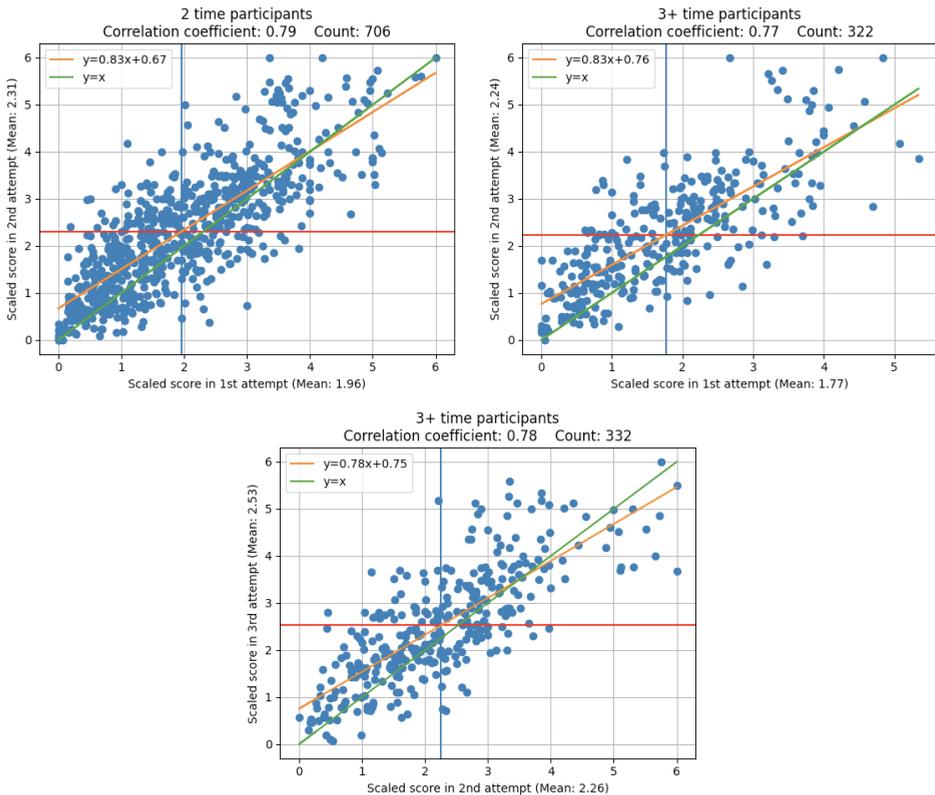


Fig. 1. Scatterplot of returning contestants’ performance in consecutive IOIs.

As seen in Fig. 1, a strong correlation is seen across the three scatterplots. The correlation coefficients, a measure of statistical association between the two variables, are calculated to range from 0.77 to 0.79. It is worth noting, however, that the data for three-or-more-IOI contestants have a greater proportion of outliers at the upper-right corner of the scatterplot, indicating less predictability for high-performers in this group.

The bivariate regression line (a least-squares-optimal prediction of the later attempt's scaled score based on the previous attempt's scaled score) in each scatterplot is plotted in yellow, while the 45-degree $y=x$ line is plotted in green. The regression coefficients are shown to range from 0.78 to 0.83, again indicating strong dependence across IOIs participated. Two features of the scatterplots indicate a general trend of improvement over consecutive IOIs. First, the regression line, for most of the corresponding datapoint x -values, is above the 45-degree line, and second, most of the datapoints in the scatterplot are above the 45-degree line.

The strong correlations in Fig. 1 imply that a contestant's performance at one IOI has substantial predictive power towards performance at the next IOI. It is unclear, however, from the scatterplot whether a single linear trend is appropriate for all IOI performance levels, given the vastly different nature of crucial tasks encountered at different levels. Therefore, we plot a linear spline regression (Marsh and Cormier, 2001) with knots at the integer scaled scores. This method allows flexibility in the regression plot while ensuring continuity of the predicted value and avoiding overfitting from higher order regressors. The results are shown in Fig. 2, where the centroids of the datapoints in each interval are also corresponded to both axes.

From Fig. 2, a positive slope within each scaled score interval is consistently seen, showing that variations of performance, even those within a medal category, yield predictive power towards future IOI performance. The slopes, however, vary substantially. For example, the slope in the $[0, 1)$ scaled score interval (corresponding to a previous IOI performance lower than the 25th percentile of that year) tends to be substantially larger than all other slopes in the same spline. This observation may be interpreted as that having a baseline level of programming and algorithmic proficiency, up to the point of solving the easiest set of IOI subtasks, is crucial towards improving further towards the IOI medal level.

Further examining the scaled score range of $[0, 2)$ in the previous IOI, corresponding to participants who fell short of winning a medal, we note that those with a scaled score in $[0, 1)$, corresponding to a performance below the 25th percentile, are very unlikely (under 10% chance) to win a medal at the following IOI. Meanwhile, those with a scaled score $[1, 2)$, corresponding to a performance at least the 25th percentile, have roughly even odds to do so. Specifically, the regression spline consistently predicts a scaled score of 1.50 (typically between the 35th to 40th percentiles) to correspond to a roughly a bronze medal cutoff performance at the following IOI.

The improvement in the 25th to 50th percentile bucket of the previous IOI, however, is not typical among medalists. Around 40% of bronze medalists achieve a silver medal or better at the next IOI, with a regression spline predicting a middling bronze medalist

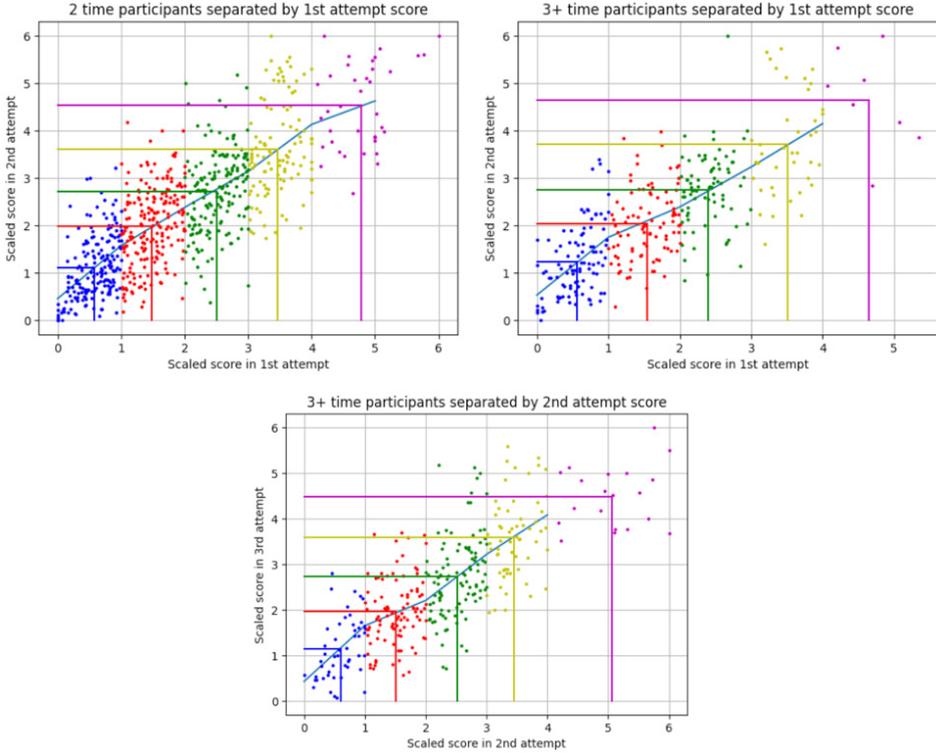


Fig. 2: Regression splines of scaled scores in consecutive IOI attempts

(with a scaled score of 2.50) to achieve an expected scaled score of around 2.80 at the next IOI, short of the silver cutoff of 3.00. Among silver medalists, around 35% manage to improve to a gold medal at the following IOI, while around 20% drop to bronze; meanwhile, very few bronze medalists (under 5%) improve to a gold medal, indicating a large skill gap between bronze and gold medalists.

Next, we consider the distribution of improvement across consecutive IOIs. In this analysis, we divide the dataset in scaled score intervals of size 0.50 for all contestants up to the silver medal level (scaled score of 4.00), plus a single interval for the gold medalists. This data division is chosen to balance having an ample number of datapoints in each bucket while having sufficiently many buckets to demonstrate overall trends. For each bucket, we produce a quantile plot of the scaled score improvement, featuring endpoints indicating the 10th and 90th percentiles, a box spanning the 25th to 75th percentiles, and a red line at the median. The plot is shown in Fig. 3, and a summary of the percentiles is provided in Table 1.

The quantile plot and Table 1 show a general trend of the average improvement decreasing as the initial IOI performance increases, which may be due to the “regression to the mean” phenomenon that is elaborated in the next subsection. A notable exception, however, is the lowest performance interval $[0, 0.5)$, corresponding to contestants

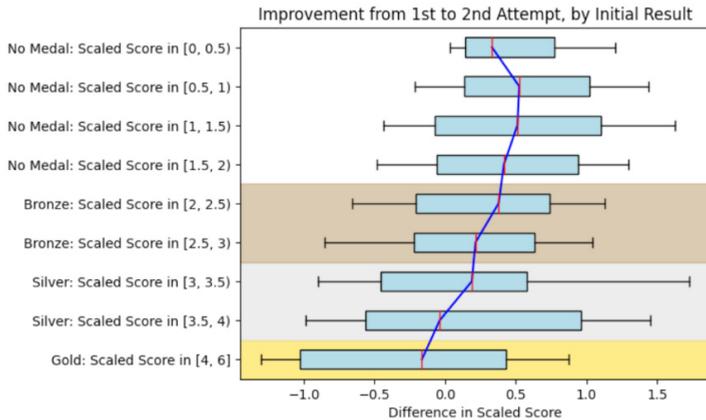


Fig. 3. Quantile plot of improvement across consecutive IOIs, separated by initial result bucket.

Table 1
Summary statistics of scaled score improvement across consecutive IOIs

1st Attempt Result	count	mean	std	10%	25%	50%	75%	90%
No Medal: Scaled Score in [0, 0.5)	112	0.52	0.53	0.04	0.14	0.33	0.77	1.21
No Medal: Scaled Score in [0.5, 1)	171	0.61	0.67	-0.22	0.13	0.52	1.02	1.44
No Medal: Scaled Score in [1, 1.5)	151	0.57	0.8	-0.44	-0.07	0.51	1.1	1.62
No Medal: Scaled Score in [1.5, 2)	145	0.45	0.71	-0.49	-0.06	0.42	0.94	1.31
Bronze: Scaled Score in [2, 2.5)	143	0.3	0.79	-0.68	-0.21	0.38	0.74	1.13
Bronze: Scaled Score in [2.5, 3)	112	0.21	0.82	-0.9	-0.22	0.22	0.63	1.05
Silver: Scaled Score in [3, 3.5)	78	0.24	1.01	-0.9	-0.46	0.19	0.58	1.76
Silver: Scaled Score in [3.5, 4)	61	0.12	0.95	-0.99	-0.56	-0.04	0.96	1.45
Gold: Scaled Score in [4, 6]	54	-0.22	0.92	-1.3	-1.03	-0.16	0.43	0.89

with less than half the score of the 25th percentile contestant. Contestants in this bucket have a lower median scaled score improvement (0.33) than all other no-medal buckets (ranging from 0.42 to 0.51), again indicating the difficulty of improvement among the lowest-performing IOI contestants which may be due to lacking rudimentary programming skills to fully engage in IOI preparation.

Besides median performance, quantile plots allow analysis of variance and skew. Overall, the variance in improvement tends to be large, with the 10th and 90th percentiles spanning more than one, and often two, scaled score points, which correspond to a medal range.) We also notice that the amount of improvement tends to be more variable among medalists than non-medalists at the previous IOI, with silver and gold medalists having the largest variance. Among non-medalists, those above the 25th percentile have a higher variance than those below the 25th percentile due to being more equipped to achieve higher results after a year of preparation. As for skew, there is no discernible trend that is not attributable to noise in the data or the scaled score having a zero lower bound, such as the lowest performance bucket being skewed to the right.

B. Progress of Multiple-IOI Contestants Over Time

In this subsection, our analysis concerns understanding the progress of contestants over multiple (two or more) IOIs. As a first step, we visualize the starting profile of IOI contestants with at least two IOIs. Fig. 4 plots the distribution of first-IOI results of these contestants, separated by the number of IOIs attended and capped at four. The distribution is shown as a cumulative histogram on proportion of contestants in the bottom quartile, second quartile, bronze medal, silver medal, and gold medal ranges. These cumulative proportions are shown relative to the actual proportions over all contestants in a typical IOI.

The visualization in Fig. 4 shows that the starting result of a two-time IOI contestant has roughly similar, though slightly worse, performance distribution as the typical IOI contestant. For contestants with more than two IOIs attended, however, the starting results become increasingly skewed towards lower result, with the proportion in the silver and gold ranges combined being under two-thirds the usual proportion. This observation is likely due to the country-based qualification process of the IOI, as it's easier for a contestant to qualify when faced with a weaker pool of contestants in higher cohorts. Note, however, that the set of 4+-time participants is too small to draw statistically-valid comparisons.

Next, we consider tracking the distribution of a contestant's progress over time. To do so, we consider two plots. In Fig. 5, the contestants are grouped by total number of attempts (capped again at four), then the mean scaled score of the contestant over multiple attempts are tracked on a path with arrows to compare the magnitude of progress over time. Fig. 6 then shows quantile plots separated by IOI order (first, second, or IOI order), with a similar 10th/25th/50th/75th/90th-percentile scheme used as in Fig. 3, allowing for comparison relative to the uniform distribution of percentiles when considering all IOI contestants. The raw data in Fig. 5 is also presented in Table 2 for ease of reference.

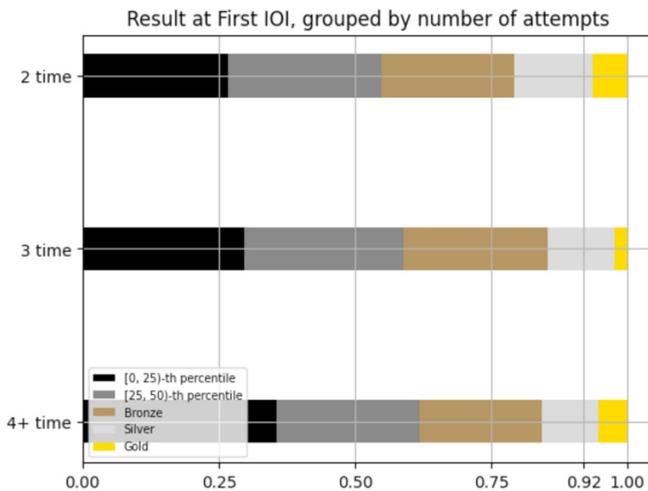


Fig. 4. Histogram of first-IOI result for multiple-IOI contestants, grouped by number of IOIs.

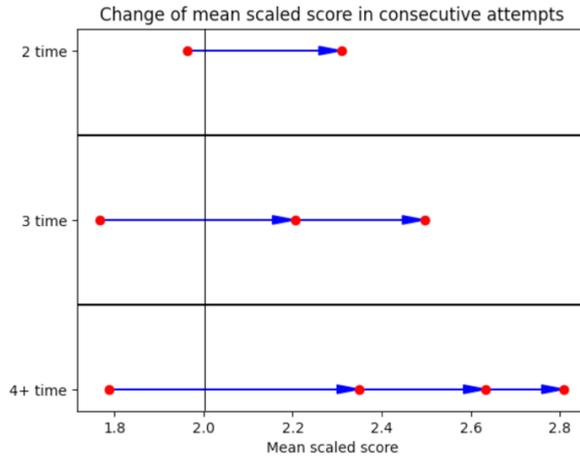


Fig. 5. Progression of mean scaled score in consecutive IOIs, separated by number of IOIs.

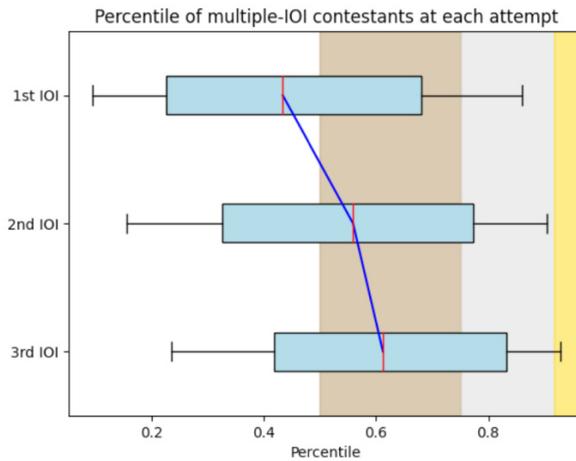


Fig. 6. Quantile plot of percentile distribution separated by ordering of IOI attempt.

Table 2
Summary statistics of percentile distribution separated by IOI order

Percentile in:	count	mean	std	10%	25%	50%	75%	90%
1st IOI	1028	0.46	0.27	0.1	0.23	0.43	0.68	0.86
2nd IOI	1038	0.54	0.27	0.16	0.33	0.56	0.77	0.9
3rd IOI	334	0.6	0.25	0.23	0.42	0.61	0.83	0.93

Fig. 4 shows that on average, contestants tend to improve over multiple IOIs, though the improvement is typically small and amounts to less than a single medal category over consecutive IOIs. Within a contestant’s trajectory, improvements tend to decrease over time; for example, for a contestant with four or more IOIs, the improvement from the

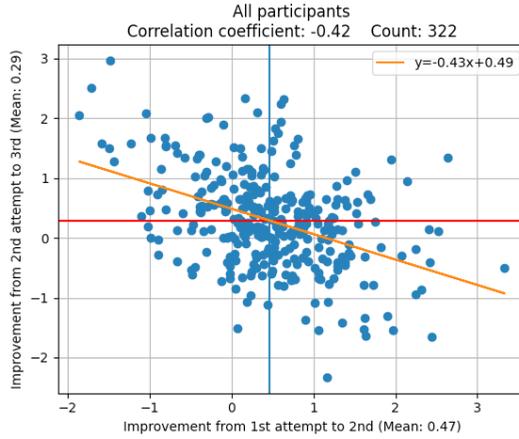


Fig. 7. Scatterplot of performance changes in consecutive IOIs, aggregated data.

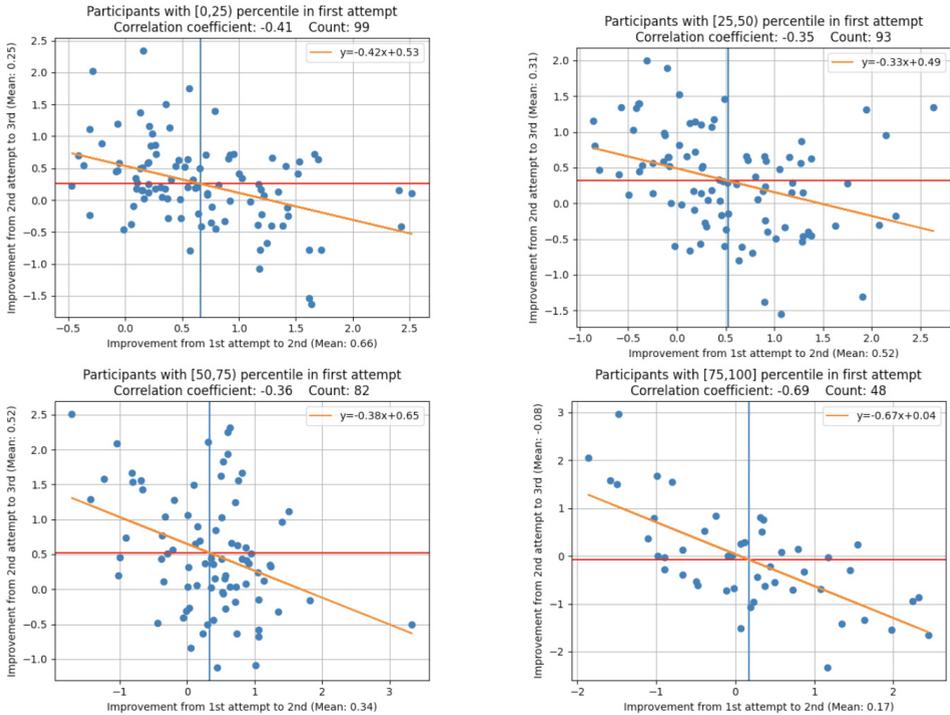


Fig. 8. Scatterplot of performance changes in consecutive IOIs, separated by initial performance.

first to the second IOI is approximately the same as the combined improvements from the second to third and the third to the fourth IOs. This phenomenon shows diminishing marginal returns of more IOI experience, as the easier areas for improvement are likely acted upon by a contestant's second IOI. Contestants with more total IOIs, while starting

out at a lower performance level, on average also tend to improve faster and reach higher levels of final IOI performance.

Fig. 5 and Table 2 show that while multiple-IOI contestants start out slightly worse than the average contestant, they eventually improve to be stronger. This observation is true across all quantiles and all performance levels. For example, the median at a multiple-IOI contestant's first IOI is at the 43rd percentile, which improves to the 56th percentile in the second IOI then to the 61st percentile in the third IOI. Diminishing improvements are seen at the higher quantiles (50th and above) but not at the lower quantiles (10th and 25th), which suggests different learning and improvement dynamics at lower as compared to higher performance levels.

Finally, for the set of contestants with at least three IOIs, we compare, under the scaled score metric, the change in performance from the first to second IOI with the change in performance from the second to third IOI. Fig. 7 shows a single scatterplot with a negative correlation, while Fig. 8 separates the data based on the performance at the first IOI. Both plots consistently show the "regression to the mean" phenomenon, indicating that there is a substantial amount of idiosyncratic variance in IOI performance at all levels of competition.

3. Country Classification and Analysis

In this section, we present a cross-sectional analysis of the aggregate performance of countries participating at the IOI. For this analysis, we compare the average percentile of the contestants from a country from IOI 2011 through 2023; to avoid including data with too much noise, we subset our analysis to countries with at least 26 contestants (half of the maximum 52) in this time. We first compare this aggregate performance metric against two predictors, population and Human Development Index (HDI), then afterwards consider differences between geographic regions as well as measures of interest in competitive programming.

While the IOI is officially an individual competition, preparation and selection for this competition is typically done on the country level, so it is still of interest to consider the aggregate performance for a country; such data is also subject to less variance than individual participant or anecdotal data. We also use the participant-to-country correspondence provided at the time of IOI registration and consistent with IOI regulations, with no attempt to distinguish country of origin in the case of immigration or foreign diasporas. While these directions may be interesting for future research, we believe these are less necessary for an initial analysis.

For the country classification into regions, we use the United Nations Geoscheme, which divides the countries in the world into sub-divisions of continents. However, we make the following changes to balance the number of IOI-participating countries in each category:

- Only one category is provided for Africa, as well as for Latin America and the Caribbean
- Central Asia and Southern Asia are combined into a single category

- The United States, Canada, Australia, New Zealand, and the United Kingdom are assigned the “Anglosphere” category due to their cultural similarities and to avoid having multiple small categories (such as Northern America and Oceania)

Thus, our geographical analysis involves eleven regions, with seven to thirteen IOI-participating countries in each:

- 1–4: Asia: Southeastern Asia, Eastern Asia, Central and Southern Asia, Western Asia
- 5–8: Europe: Eastern Europe, Southern Europe, Northern Europe, Western Europe
- 9: Africa, 10: Latin America, 11: Anglosphere

A. Comparison of Average IOI Percentile to Population and Human Development Index

As a first analysis, we generate scatterplots to compare the average IOI percentile of each country’s contestants with the population and HDI. Our prior is that having a large population means having a larger pool of talent to draw from, irrespective as to how well this talent is nurtured in the country. Meanwhile, the HDI is a widely used metric for development which correlates to the availability of educational resources and strength of institutions. Therefore, these two metrics are a good start towards uncovering the causes that explain the variation between different countries’ average IOI performance.

For population, we use a log scale as is standard in economic studies, typically done to avoid distortion from the roughly log-normal distribution of country populations. For HDI, we use a linear scale due to the approximately linear distribution among IOI-participating countries. We also plot a trendline on each scatterplot to evaluate the strength of the correlation with these two indicators. The points in the scatterplot are also colored by region for ease of reference and for a preliminary display of regional trends, for example, as to which regions are mostly above or below each trendline. The results are shown in Fig. 9.

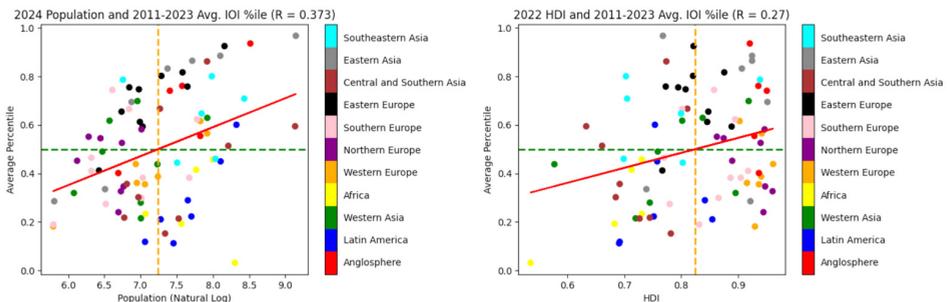


Fig. 9. Scatterplot of average IOI percentile by country, with log Population and HDI.

In Fig. 9, a moderate correlation of 0.373 is seen for the log population variable, while a slightly weaker correlation of 0.270 is seen for the HDI variable. The correlations, and particularly the population correlation, also appear to be a trend for the dataset as a whole and are not driven by a small number of outliers in the data. These correlation coefficients show that both explanatory variables have some correlation with a country's IOI performance, but a lot of the variation is unexplained by these two indicators.

B. Regression Analysis and Residual Grouping by Region

Next, we consider a regression analysis using the same data as the previous subsection, the results of which are shown in Table 3. The regression results show that both the log population and the HDI metric are significant in their predictive power towards a country's IOI performance, even when the other is considered. The overall R-squared, however, is only 0.312, meaning that most (around 70%, and likely higher due to possible overfitting) of the variance is unexplained by these two predictors.

The regression analysis yields a fitted model, so we can take calculate the residual for each country datapoint, representing the variance in country average IOI percentiles unexplained by the predictors. We then calculate the mean and standard deviation of the residual by geographical region, a summary of which is presented in Table 4. The residual mean represents each region's overall strength at the IOI beyond what is predicted by population size and HDI, while the residual standard deviation is a measure of heterogeneity of country strengths.

Due to the small bucket sizes, testing for statistical significance will be difficult. The residual means column in Table 4, however, still provides a broad picture as to the strengths of countries in each geographical region after controlling for population and HDI. Three regions, Eastern Europe, Eastern Asia, and Southeastern Asia, have high residual means (at least ten percentile points), with Eastern Europe having by far the

Table 3
Regression output of average IOI percentile on HDI and log population

OLS Regression Results						
=====						
Dep. Variable:	Average Percentile		R-squared:			0.312
Model:		OLS	Adj. R-squared:			0.294
Method:		Least Squares	F-statistic:			17.88
No. Observations:		82	AIC:			-35.68
Df Residuals:		79	BIC:			-28.46
Df Model:		2				
=====						
	coef	std err	t	P> t	[0.025	0.975]

constant	-1.5287	0.340	-4.495	0.000	-2.206	-0.852
2022 HDI	1.0051	0.226	4.451	0.000	0.556	1.455
2024 Population (ln)	0.1657	0.032	5.237	0.000	0.103	0.229

Table 4
Residual summary statistics by geographical region

Region	Residual Mean	Residual Stdev.	Count
Eastern Europe	0.220	0.094	10
Eastern Asia	0.122	0.106	7
Southeastern Asia	0.105	0.153	6
Western Asia	0.059	0.151	8
Anglosphere	0.007	0.131	5
Southern Europe	-0.009	0.195	10
Central & Southern Asia	-0.018	0.193	9
Northern Europe	-0.044	0.173	8
Africa	-0.170	0.120	5
Western Europe	-0.171	0.057	7
Latin America	-0.221	0.115	7

highest, consistent with common knowledge on these countries' scientific education traditions. (Lovheim, 2021) Meanwhile, the three regions with the lowest residual means are Africa, Western Europe, and Latin America, likely related to the lack of prominence of the scientific Olympiads in these regions, and in the case of Western Europe, relative to other educational opportunities.

C. Incorporating Codeforces Registration Data

In this final analysis, we consider a country-level metric of student interest in competitive programming, the number of active participants on the CodeForces (CF) competitive programming platform. Fig. 10 shows a scatterplot, where we note that the CF participant count measures both a country's population and level of interest in competitive programming.

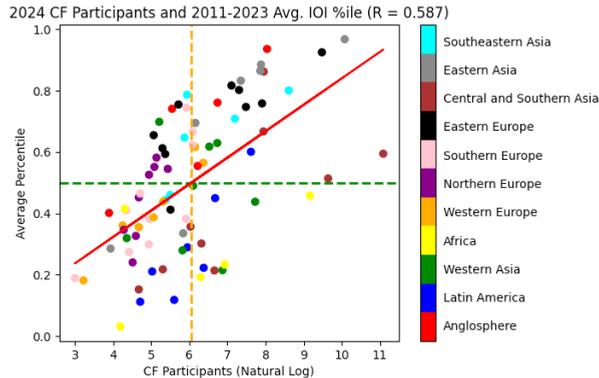


Fig. 10. Scatterplot of CodeForces participant count and country average percentile.

Table 5
Regression output of average IOI percentile on HDI, log population, and log CF density

OLS Regression Results						
Dep. Variable:	Average Percentile	R-squared:	0.602			
Model:	OLS	Adj. R-squared:	0.587			
Method:	Least Squares	F-statistic:	39.32			
No. Observations:	82	AIC:	-78.59			
Df Residuals:	78	BIC:	-68.96			
Df Model:	3					
	coef	stderr	t	P> t	[0.025	0.975]
constant	-1.1855	0.264	-4.487	0.000	-1.711	-0.660
2022 HDI	1.2334	0.175	7.030	0.000	0.884	1.583
2024 Population (ln)	0.2608	0.027	9.553	0.000	0.206	0.315
2024 CF Density (ln)	0.2645	0.035	7.542	0.000	0.195	0.334

The correlation in Fig. 10 is noticeably stronger than that in either scatterplot of Fig. 9, showing the drastic importance of the level of student interest and engagement in a practice platform towards achieving strong IOI results. To place this correlation in context, we run a second regression, similar to the regression in the previous section, where in addition to HDI and population we include a “CF Density” metric which is the ratio of CF participants in a country to its total population. The regression results in Table 4 show a much higher R-squared metric of 0.602 compared to 0.312 the previous regression, meaning that the CF density explains nearly half of the residual variance not predicted by population size and HDI.

4. Conclusion and Further Work

This paper appears to be the first large-scale analysis of IOI data, particularly with the focus on returning contestants and geography. Overall, the analyses provide statistical justification for several trends that are likely to be known in an approximate sense within the IOI community, while providing convincing evidence for interested parties outside the IOI community.

A. Summary of Findings

From the analysis on returning contestants’ IOI performance trajectories, the overarching observation is that IOI performance, as a metric, has good internal validity, given the large correlation between consecutive IOI results. (Figures 1 and 2) There is, however, still substantial variance in skill acquisition over time and unpredictability in the results of a single IOI. These are seen by the large spreads in improvement over consecutive IOIs (Fig. 3) and the reversion to the mean phenomenon (Fig. 7). We have also seen diminishing marginal improvements over multiple consecutive IOIs. (Figures 4 and 5)

Meanwhile, from the geographical analysis, we demonstrated that while broad demographic statistics such as population and Human Development Index (HDI) are significantly correlated with a country's aggregate IOI performance. (Fig. 8 and Table 3) A more important factor, however, is the level of competitive programming interest in the country. (Tables 5) We have also identified regions that are substantially stronger or weaker than what would have been predicted from population and HDI alone, while classifying some regions as more heterogeneous or homogeneous than others in terms of their countries' IOI performance.

B. Recommendations for Further Research

There are many directions in which the analyses in this paper can be extended to yield an even deeper understanding of the educational nature of the IOI in its global setting.

As a first step, it may be interesting to combine the two directions considered in this paper and analyze their interactions. This paper considers the trajectory of returning contestants and the aggregate performance of countries separately, though there can be some insight in analyzing the distribution of improvement in returning contestants from different categories of countries. Analysis, however, might be more difficult due to having few datapoints at this proposed level of granularity.

The concern of having few datapoints made conducting statistical tests difficult particularly for the returning contestant analysis, as the sheer variance in differences in performances causes standard errors to be large. Therefore, only the less surprising and insightful trends could be tested under this framework. Eventually, as the IOI continues to run over the years and generates more high-quality data, it might be feasible to revisit this approach. For example, after ten more IOIs, the amount of usable returning contestant data will nearly double.

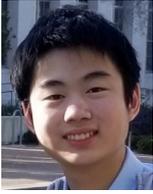
Finally, the IOI statistics can be combined with a qualitative analysis and substantive interpretation of the nature of the IOI. There are many interesting directions, such as considering which tasks may be favorable to different populations; for example, identifying tasks that are more approachable by first-time, less experienced contestants or contestants from weaker countries. This direction can also be applied understand the role of the IOI in global talent development and identification, such as for example, in what ways are strong contestants from weaker countries tend to be different from a typical strong contestant.

Acknowledgements

We thank Jay Leeds, Gerard Francis Ortega, and Payton Yao for discussion and insights.

References

- Barnett, A.G., van der Pols, J.C., Dobson, A.J. (2004). Regression to the mean: What it is and how to deal with it. *International Journal of Epidemiology*, 34(1), 215–220. <https://doi.org/10.1093/ije/dyh299>
- Codeforces Country Ratings. Retrieved from <https://codeforces.com/ratings/countries>
- International Olympiad in Informatics International Committee (2023). IOI Regulations. Retrieved from <https://ioinformatics.org/files/regulations23.pdf>
- International Olympiad in Informatics International Scientific Committee (2024). IOI Syllabus. Retrieved from <https://ioinformatics.org/files/ioi-syllabus-2024.pdf>
- International Olympiad in Informatics – Statistics. Retrieved from <https://stats.ioinformatics.org/>
- Lövheim, D. (2021). Cold War fostering of scientific elites: International youth olympiads in chemistry and physics 1967–1984. *History of Education*, 50(5), 685–703. <https://doi.org/10.1080/0046760x.2021.1890239>
- Marsh, L.C., Cormier, D.R. (2005). *Spline regression models*. Sage Publications.
- Verhoeff, T. (2006). The IOI is (not) a science olympiad. *Informatics in Education*, 5(1), 147–159. <https://doi.org/10.15388/infedu.2006.25>
- United Nations Department of Economic and Social Affairs. (n.d.). United Nations geoscheme. Retrieved from <https://esa.un.org/MigFlows/Definition%20of%20regions.pdf>
- United Nations Development Programme. (n.d.). Human Development Index (HDI) Dataset. Retrieved from <https://hdr.undp.org/data-center/human-development-index>



E. Lee – is an incoming first-year undergraduate at Stanford University. He plans to pursue a major in the area of applied mathematics. He is an avid enjoyer of mathematics and algorithmic competitions, and his accomplishments in these areas include earning a USA Math Olympiad bronze medal as well as qualifying for the Platinum Division of the USA Computing Olympiad. He shares his interests by educating others about problem-solving and organizing his own contests as well.



T. Reizin – is studying his Bachelors in Computer Science at Charles University in Prague. During his high school years he actively participated in the Ukrainian Olympiads in Informatics. After graduating he continued his involvement with the Ukrainian olympiads, by teaching in preparatory camps and helping with organization of team selection. He also managed to advance the ICPC European Championship. His research interests lie in combinatorics, in particular extremal combinatorics.



F.E. Wu – is a PhD student in Operations Research at the Massachusetts Institute of Technology, studying information-theoretic algorithms for reinforcement learning and decision making. In high school, he won a gold medal at the IMO and a bronze medal at the IOI. Prior to starting the PhD, he worked in a quantitative research position in industry and as an undergraduate was extensively involved in teaching probability and statistics. He also coaches the Philippines' IOI 2024 team.



F.E. Wu is an incoming first-year undergraduate at the Massachusetts Institute of Technology. He represented the Philippines at the International Mathematical Olympiad (IMO) and the IOI, winning a silver medal and bronze medal, respectively, in 2023. Within competitive programming, besides competing he is interested in building learning communities and making the learning process more enjoyable. He plans to continue studying mathematics and computer science in college.

Analysis and Evaluation of the Contestant's Progress in Real-time Coding Contests

Mirvari MAMMADLI, Nihad MAMMADLI, Jamaladdin HASANOV

ADA University, School of IT and Engineering

Ahmadbey Aghaoglu str. 61, 1008 Baku, Azerbaijan

e-mail: mmammadli13254@ada.edu.az; nmammadli14004@ada.edu.az; jhasanov@ada.edu.az

Abstract. This paper presents a model for analyzing contestant progress in real-time coding contests, emphasizing the critical need for effective measures in assessing code similarity and plagiarism cases. Current coding contest platforms often need more robust procedures to identify and address these issues, compromising the integrity of the evaluation process. To tackle these challenges, we propose a novel system that leverages advanced techniques to analyze code and collective behavior, providing a holistic evaluation of submissions. The system enhances performance assessment accuracy and maintains fairness and credibility in real-time coding contests. The findings and insights from this study shed light on the importance of integrating sophisticated mechanisms to ensure the authenticity of code submissions and uphold the competitive nature of coding competitions.

Keywords: plagiarism, contest, CMS, real-time, C++.

1. Introduction

The rise of online platforms has shifted competitive programming and coding contests into the digital world, bringing forth a pressing issue – plagiarism. As the number of online coding platforms increases and resources for cheating become more available, some contestants' temptation to engage in unethical practices has grown exponentially. Ensuring fairness in coding contests is crucial and a central concern for researchers and instructors. However, most coding contest platforms lack solid systems to identify plagiarism in real time, jeopardizing the validity of the evaluation process.

Moreover, the analysis of contest data is pivotal regarding the quality and fairness of coding competitions. As mentioned in (Hasanov *et al.*, 2021), based on the study of the International Olympiad in Informatics (IOI), statistical analysis of the competition data can provide valuable insights into the strategies used, competition dynamics, and contestant performance. Contest data analysis in competitions enables the organizers, coaches, and participants to identify the shortcomings, strengths, and areas for improvement. Therefore, we ensure real-time data visualization to facilitate further analysis in

the proposed system. Using data-driven approaches, coding contests can be optimized to promote fair competition.

This review explores plagiarism detection in programming, specifically in real-time coding contests, emphasizing the need for customized solutions to address their unique challenges.

1.1. *What is Code Similarity, and How Does it Work*

Code similarity involves assessing the resemblance between different code pieces to identify potential plagiarism instances. Understanding the concept of code similarity in the context of programming and software development is essential. A comparison of syntactic and semantic structures of code pieces usually takes place when assessing code similarity. One of the most common approaches includes token-based comparison (Yetthapu, 2023). The source code is divided into essential components known as tokens, which could consist of parts such as literals, identifiers, and keywords. Subsequently, the overlap or similarity of these tokens across various code segments is examined. This particular method does not necessarily consider structure or semantics. Thus, this method captures syntactic similarities between code segments (Prechelt *et al.*, 2000). To maintain the integrity of coding contests, our project introduces checker.js, which includes three methods: comparator, cppChecker, and remover. Traditional methods face challenges adapting to source code nuances, as noted by Prechelt *et al.* (2002). Also, there has been widespread use of string matching algorithms like the Longest Common Subsequence (LCS) algorithm (Myers, 1986). However, our approach presents a remover method that goes beyond conventional string matching, using regular expressions to improve accuracy in code similarity detection.

1.2. *Plagiarism in Contests*

One of the main reasons for jeopardizing the evaluation process's integrity in programming contests is plagiarism. As mentioned in (Wu *et al.*, 2022), cheating has become challenging. Real-time detection of plagiarism, crucial for contest integrity, is a key focus. The comparator method from our project, which is aligned with timely interventions during competitions, uses async for concurrent processing. This strategic approach ensures real-time plagiarism evaluations, emphasized by Jeske *et al.* (2018). The cppChecker method, managing multilingual code comparison, is vital given the array of programming languages used in coding contests. As participants engage in diverse coding languages, the capability to manage this variability becomes crucial in maintaining fair and unbiased contest conditions. Academic institutions and contest organizers prioritize solid measures to restrain plagiarism, fostering an atmosphere where skills and imagination can be seen. Checking and comparing each piece of code could be too time-consuming for the instructors. Therefore, before introducing our system, an analysis of

previous systems and their advantages and disadvantages is considered in Section 2. The approach used in the project is mentioned in Section 4.

2. Previous Work

Some of the prominent and extensively cited state-of-the-art plagiarism detection systems, according to Burrows *et al.* (2007), are JPlag (Prechelt *et al.*, 2002) and MOSS (Schleimer *et al.*, 2003). Additionally, there exist several other noteworthy systems, such as Sim (Gitchell & Tran, 1999), Plague (Clough, 2000), YAP (Wise, 1996), Plaggie (Hage *et al.*, 2011), and FPDS (Mozgovoy *et al.*, 2005). Each system employs diverse methodologies and algorithms to address the intricate challenges of detecting plagiarism in programming code (Đurić & Gasevic, 2013).

In 1996, during his student project at Karlsruhe University, Guido Malpohl created a plagiarism detection tool, JPlag, which evolved into the first online system. Subsequently, with the effort of Emeric Kwemou and Moritz Kroll, the online system transformed into a web service (Prechelt *et al.*, 2000). This tool specializes in plagiarism detection across C, C++, Java, and Scheme programming languages. The underlying architecture of JPlag relies on the Greedy String Tiling comparison algorithm (Prechelt *et al.*, 2002). Compared to other tools, JPlag demonstrates superior plagiarism detection performance (Yetthapu, 2023). As stated in (Yetthapu, 2023), JPlag processes submitted source code assignments and then generates HTML pages as output, with code similarity values ranging from 0% to 5% indicative of no plagiarism and 100% representing blatant plagiarism. Intermediate values, such as 40%, warrant further manual investigation for a conclusive judgment (Prechelt *et al.*, 2000).

Code comparison tools like **MOSS** (Measure of Software Similarity) are frequently used in software development projects to find instances of code plagiarism. MOSS is a free online service created in 1994 by Aiken *et al.* at Stanford University. It functions as a web service. It offers an easy-to-use web interface with Windows and UNIX operating systems. MOSS splits code into continuous substrings called K-grams using the Winnowing algorithm and hashes each K-gram (Luke *et al.*, 2014). MOSS uses several algorithms to increase efficiency, such as control flow analysis, code structure analysis, and string matching. About twenty-five programming languages, including Java, C, C++, Python, JavaScript, Matlab, VHDL, and Verilog, are supported by it (Hage *et al.*, 2010). Results from MOSS are presented in HTML through a graphical interface that highlights suspicious code fragments, the percentage of similarity, tokens, and matched lines. Each matched pair is clickable, facilitating manual inspection (Yetthapu, 2023).

As mentioned in (Schleimer *et al.*, 2003), which connects to our topic, the authors investigate the concept of local document fingerprinting algorithms to identify copying across large datasets accurately. They describe the efficient winnowing algorithm and demonstrate its performance within 33% of the lower bound. The paper discusses the difficulties in detecting partial copies and suggests using the k-grams mentioned above and hashes as fingerprints. The article emphasizes the importance of selecting the appropriate “k” value to eliminate coincidental matches.

In the context of document fingerprinting, the article explains the mechanics of the winnowing algorithm, as well as querying and optimal hash selection. It examines the correctness of local algorithms for finding matching substrings and provides a lower bound on their density. The experiments with web data demonstrate the algorithm's performance.

The paper emphasizes the importance of handling low-entropy strings in fingerprinting, as demonstrated in experiments with non-uniformly random data. It introduces robust winnowing, which reduces density, emphasizing its usefulness in specific applications.

The study examines the copying structure of 20,000 web pages, identifying a non-uniform data distribution and discussing the power law relationship between k-gram frequency and rank. MOSS, a plagiarism detection service that uses robust winnowing, efficiently detects document similarities (Schleimer *et al.*, 2003).

The paper introduces and evaluates the winnowing algorithm for document fingerprinting, addressing challenges while emphasizing practical applications such as plagiarism detection (Schleimer *et al.*, 2003).

(Sharma *et al.*, 2021) also mentioned "code clone detection". In the article, various types of machine-learning techniques for source code analysis were researched. A section about code clone detection was similar to our project. However, in the article, the emphasized approach is ML (machine learning), which differs from ours. Apart from investigating the methods for automatically detecting plagiarism, the study mentioned approaches to validate the accuracy of clones reported by existing tools. The methodology in the article entails creating a dataset of source code samples classified as non-clones or clones. Following that, feature extraction techniques are used to identify relevant features. Then, they are then input into machine-learning models for training and evaluation. The models can detect clones within sample pairs (Sharma *et al.*, 2021, p.19). It is worth noting that while our project focuses on code clone detection, our approach differs from the ML-centric methodology discussed in this study.

As mentioned (Đurić & Gasevic, 2013), contestants can modify the source code in several ways, including lexical and structural forms. Most common examples of lexical modifications could include modification of source code formatting, addition, modification, or deletion of comments, language translations, reformatting or modification of program output, etc. Lexical modifications often do not require advanced programming skills, unlike structural modification, which involves changing the order of variables in statements, changing the order of statements within code blocks, reordering code blocks, adding redundant statements or variables, modifying control structures, changing data types and modification of data structures, method inlining and method refactoring, redundancy and so on. To counteract these changes, plagiarism detection tools examine the program's structure and its lexical elements, using techniques such as tokenization, abstract syntax tree comparison, and semantic analysis to identify suspicious patterns and similarities. The system proposed in the article showed more promising results than JPlag when considering the structural and lexical modifications. While considering these structural modifications, it is essential to know the programming language that will be used so that modifications do not result in compilation errors or run-time exceptions

(Đurić & Gasevic, 2013). Since the IOI contest we are preparing for will be held in C/C++, our system was adopted to consider the use of language.

The source code similarity detection tools can be categorized into two operational modes: online and offline (Đurić and Gasevic, 2013, p. 7). Based on this categorization, our project follows the online paradigm. Every process happens in real time and with the use of a socket.

According to (Đurić and Gasevic, 2013), there are several types of plagiarism tools, such as text-based plagiarism detection, attribute-oriented plagiarism detection, and so on. Since the text-based plagiarism tools ignore the code syntax and just compare English words, it is not the most helpful approach to take when it comes to coding contests. A better approach would be using attribute-oriented similarity detection, where fundamental properties are used, such as the number of unique operands, operators, and so on. However, there are more suitable approaches than this since the same number of variables, loops, or conditional statements might be considered plagiarism. The approach best suited for these conditions would be structure-oriented similarity detection. It includes tokenization and string-matching algorithms to determine plagiarism (Đurić and Gasevic, 2013). While reviewing the existing approaches, it became evident that the approach we will use for our project is a structure-oriented similarity detection tool.

2.1. Challenges of Similarity Detection

Detecting plagiarism in code is challenging, and similarity rates might be excessively high. This happens due to the concise nature of simple tasks which require fewer lines of code. Hence, the contestants unintentionally produce remarkably similar code pieces. For instance, a very primitive example could be printing “Hello World” in a console in a few lines of code, which is possible, thus contributing to the prevalence of similar code structures. One potential resolution strategy for this issue involves exempting code segments with fewer than 10 to 15 lines from plagiarism detection or adjusting the plagiarism assessment based on the code’s line count.

Moreover, a challenge arises in cases where contestants are required to implement functions within pre-provided code. In this particular situation, the tool will count the provided code as part of the similarity, thus artificially inflating the similarity rate. A potential mitigation strategy involves providing the plagiarism detector with a predetermined code sample in such scenarios. Subsequently, the detector tool would be configured to disregard the designated code piece when evaluating similarity, further enhancing the assessment’s accuracy.

3. Design of an Alarm and Monitoring Dashboard

One of the goals of our project is a visual dashboard that provides a far friendlier user experience and more analytical possibilities. The seats will be displayed in the monitoring

User ID	Full Name	Username	Time	Event Type
11	Vusal Rzayev	VioletVulture	Mar 28, 2024, 6:27:59 PM	After-event
11	Vusal Rzayev	VioletVulture	Mar 27, 2024, 4:07:53 PM	After-event
12	Narmin Mustafayeva	NavyNightingale	Mar 27, 2024, 4:07:23 PM	Plagiarism
12	Narmin Mustafayeva	NavyNightingale	Mar 27, 2024, 4:06:33 PM	Plagiarism
9	Emin Quliyev	YellowYak	Mar 27, 2024, 4:06:23 PM	Plagiarism
12	Narmin Mustafayeva	NavyNightingale	Mar 27, 2024, 3:16:59 PM	Plagiarism
9	Emin Quliyev	YellowYak	Mar 27, 2024, 3:18:29 PM	Plagiarism
9	Emin Quliyev	YellowYak	Mar 27, 2024, 3:05:57 PM	Plagiarism
9	Emin Quliyev	YellowYak	Mar 27, 2024, 3:05:52 PM	Plagiarism
39	Rufat Huseynov	SaffronSeal	Mar 27, 2024, 3:05:47 PM	Plagiarism
35	Aysel Mammadova	CoralCrocodile	Mar 27, 2024, 3:02:07 PM	Plagiarism

Fig. 3. Alarms Log.

For the submission, we have several kinds of alarms:

- **Similarity alarm** – When the contestant submits their work, it goes through the process of checking for plagiarism. When the similarity rate is more than or equal to 70%, a notification on top of the page will pop up, and the seat color will blink red.
- **Scoring preceding another event** – If any kind of event (such as going to WC, print, and so on) happened within 45 minutes before the submission and the similarity rate is more than or equal to 50%, the seat will blink yellow, and the notification with similarity rates between two submissions will be seen.

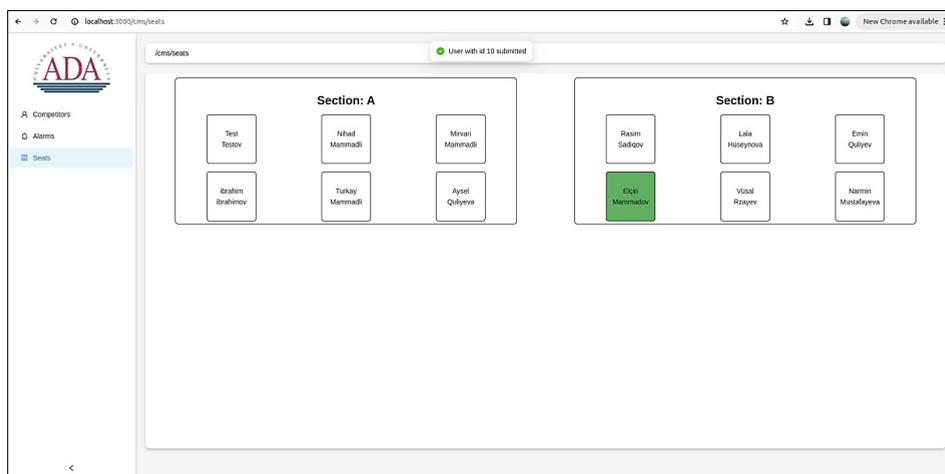


Fig. 4. Submission without any kind of possible plagiarism.

- **Sudden increase alarm** – After the submission, the contestant’s similarity rate increases by 30% or more. The notification about the sudden increase will appear, and the similarity rate before and now will be shown in the message above. The color representing this alarm will be orange.

When none of the conditions are met for alarm during the submission process, the seat will blink in green, and a notification about the submission will be seen (Fig. 4):

Every single process described works with the help of a socket, which gives us real-time data.

4. Experiments

4.1. How our Code Operates

Our project’s innovative system presents a novel approach to plagiarism detection. The remover method pre-processes code submissions by filtering out common structures and removing comments. The cppChecker method handles C++ code, demonstrating adaptability to different syntaxes. The comparer method manages real-time comparisons using async for concurrent processing. This systematic process ensures a comprehensive evaluation of multiple C++ files.

As mentioned above, the plagiarism checker employs the Longest Common Sequence algorithm. No external libraries were implemented when writing the plagiarism checker tool. Moreover, when creating the system, Node.js was chosen because of its advantages, versatility, and benefits. Also, the Express library was selected to facilitate API development. React, and AntDesign were used on the front end to create a clear and visually appealing UI (user interface).

The project is organized in a repository¹, containing the backend and the front end.

The backend includes the plagiarism checker mentioned above and the APIs.

At the front end of the web application, there are three pages: Users, Alarms, and Seats. Corresponding to those pages, there are three primary APIs: the “Sections”, “Alarms”, and the “Users” API. The corresponding API is called on each page, displaying the data on the screen. The Users page has been designed to showcase the relevant information retrieved from the “Users” API. The information (Names, Surnames, Usernames, and emails) is in a tabular format. All the alarms related to the submissions are displayed on the alarms page. There are three main types of alarms: plagiarism, which is shown in red; scoring preceding after-event alarm, which is shown in yellow; and sudden increase alarm, shown in orange. The alarms page derives the needed data from the “Alarms” API, which takes the data from alarms, users, and alarm_types tables. Finally, there is a Seats page that displays the seats and the configuration of the contestants. This page requires data from the “Sections” API. The “Sections” API takes data from submissions.json and the sections, users, and submissions tables.

¹ <https://github.com/NihadMammadli/SDP>

On the seat page, upon user selection, a collapsed form displays the contestant's most recent submission, revealing the code and common line similarity rate. This structured approach ensures that cases of plagiarism are clearly seen at the end of the process.

4.2. CMS Structure

The Content Management System (CMS) is a foundational element, providing a structured framework for managing and organizing code submissions. Its architecture facilitates efficient real-time comparisons and assessments, contributing to the overall success of the plagiarism detection tool. CMS has been used in contests such as IOI since 2017 and has become a strong nominee, becoming the established standard for the IOI (Hasanov *et al.*, 2021). Our implementation of the CMS is crucial for conducting efficient real-time comparisons in our project. It dramatically enhances the effectiveness of our plagiarism detection tool. Incorporating CMS into our project involves following a specific procedure. This included setting up and running the CMS, creating separate admin and contestant user accounts, and defining commands designed for the contestant users. As part of our workflow, we generated content tests, established test cases, and executed the code submission process using the contestant user's credentials. The data generated during this submission process is redirected to a dedicated database integrated with our Node.js code. This integration allows us to perform real-time plagiarism checks as data flows through our system.

4.3. Our Architecture

The architecture goes as follows (see Fig. 5):

1. Competitor makes their submission to CMS.
2. The newly submitted code gets stored in the "Submissions" table in DB (SQL Postgres).
3. Socket.js continuously checks for new submissions in real time. DB returns the submission ID to socket.js when a new submission has been made.
4. Socket.js requests the newly submitted code from Python Downloader.py.
5. Python Downloader.py requests the newly submitted code from CMS and returns the requested code to Python Downloader.py.
6. Python Downloader.py sends the code it got from CMS to the Socket.js.
7. Socket.js sends the code to Checker.js to check for plagiarism in new submissions.
8. After checking for plagiarism, it sends the alarm to the "Alarms" table and the similarity score to the "Similarities" table, and it gets stored in DB. At the same time, for quicker access, the Checker inserts submissions in JSON format to Submissions JSON.
9. The Admin requests the data (Alarms/Seats/Competitors) through React CMS, which takes it from Data.js. Data.js requests the submissions from Submissions

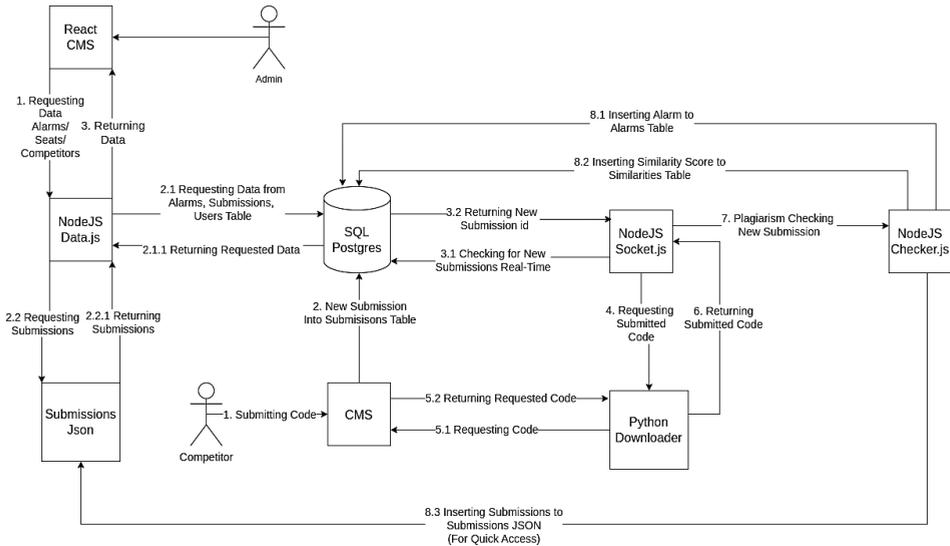


Fig. 5. Architecture Diagram.

JSON and returns them to Node.js. Later, Node.js returns the data to React CMS, and the Admin sees the needed data.

4.4. Our Setup

Our team uses the most recent version of CMS in the system. We have integrated the plagiarism checker tool, visual boards to monitor submissions, and alarms to detect instances of cheating. The alarm mechanism mentioned above provides real-time alerts to identify potential cheating events of the contestants. Moreover, the logs with the alarm type and submissions are kept for further investigation of cheating cases when needed.

The upcoming IOI contest of 2024 will be held in Egypt in September. We intend to refine our system even more to ensure it is ready for use in coding competitions such as IOI. Our system for “Analysis and evaluation of the contestant’s progress in real-time Coding Contests” has been tested by simulating all possible scenarios. It included identifying different kinds of alarms and automating the users’ submission process. The testing aimed to replicate the potential occurrences at IOI24, demonstrating the effectiveness of our system in maintaining fairness and integrity in coding competitions.

5. Conclusion

The literature review provides a comprehensive background on plagiarism detection in programming. The increase in the usage of online platforms for contests has brought a considerable challenge regarding the integrity of the evaluation process. Our team pro-

posed a novel system that uses sophisticated methods when analyzing the structure of codes submitted by the contestants and their behavior throughout the process to address the issue arising. It is essential to ensure fairness and authenticity in such competitions. Our system aims to provide a broad solution to plagiarism detection in competitions by using tools like the LCS (Longest Common Subsequence) algorithm and Node.js, as well as complex and sophisticated methodologies like structure-oriented similarity detection.

Through experiments and simulations, the system was validated for the accuracy of the approach used. It demonstrated the ability to detect instances of plagiarism accurately in real time. Thanks to alarms and monitoring dashboards, the system offers a framework for contest organizers to detect and analyze potential cheating incidents in real time.

We are committed to refining and enhancing our system since Egypt has an upcoming IOI 2024 competition. We hope to contribute to the coding competition society and offer a fair environment for competitive programming contestants.

The studies highlight the importance of integrity during competitions and mechanisms and technologies that ensure the quality of the competitive nature of the competitions. A robust framework will be provided through the system proposed by our team to maintain a fair and competitive environment.

References

- Burrows, S., Tahaghoghi, S.M.M., Zobel, J. (2007). Efficient and effective plagiarism detection for large code repositories. *Software-Practice & Experience*, 37(2), 151–175
- Durić, Z., Gasevic, D. (2013). A Source Code Similarity System for Plagiarism Detection. *The Computer Journal*, 56(1), 70–86. <https://doi.org/10.1093/comjnl/bxs018>
- Gitchell, D., Tran, N. (1999). Sim: a utility for detecting similarity in computer programs. *Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education, New Orleans, Louisiana, USA, 24–28 March*, pp. 266–270. ACM New York, NY, USA.
- Hage, J., Rademaker, P., van Vugt, N. (2010). A comparison of plagiarism detection tools. ISSN: 0924-3275. Retrieved from <http://www.cs.uu.nl/research/techreps/repo/CS2010/2010-015.pdf>
- Hage, J., Rademaker, P., Van Vugt, N. (2011). Plagiarism detection for Java: a tool comparison. In: *Proceedings of the 1st Computer Science Education Research Conference, CSERC '11, Heerlen, The Netherlands, 7–8 April*, pp. 33–46, ACM New York, NY, USA.
- Hasanov, Jamaladdin, Gadirli, Habil, Bagiyev, Aydin. (2021). On Using Real-Time and Post-Contest Data to Improve the Contest Organization, Technical/Scientific Procedures and Build an Efficient Contestant Preparation Strategy. *Olympiads in Informatics*. 23–36. DOI: 10.15388/oi.2021.03.
- Jeske, H.J., Lall, M., Kogeda, O.P. (2018). A real-time plagiarism detection tool for computer-based assessments. *Journal of Information Technology Education. Innovations in Practice*, 17, 23. <https://jite.org/documents/Vol17/JITEv17IIPp023-035Jeske3991.pdf>
- Luke, D., P.S., D., Johnson, S.L., Sreepabha, Varghese, E.B. (2014). Software Plagiarism Detection Techniques: A Comparative Study. ISSN: 0975–9646. Retrieved from <https://ijcsit.com/docs/Volume%205/vol5issue04/ijcsit2014050441.pdf>
- Mammadli, N. (2024). SDP. GitHub. <https://github.com/NihadMammadli/SDP>
- Mozgovoy, M., Frederiksson, K., White, D.R., Joy, M.S., Sutinen, E. (2005). Fast plagiarism detection system. *Lecture Notes in Computer Science*, 3772/2005, 267–270.
- Myers, E.W. (1986). An O(ND) Difference Algorithm and Its Variations. *Algorithmica*, 2(1–4), pp. 251–266. <http://www.xmailserver.org/diff2.pdf>
- Prechelt, L., Malpohl, G., Philippsen, M. (2002). Finding Plagiarisms among a Set of Programs with JPlag. *Journal of Universal Computer Science*, 8(11), 1016–1038. <https://www.researchgate.net/>

- publication/2832828_Finding_Plagiarisms_among_a_Set_of_Programs_with_JPlag
 Prechelt, L., Malpohl, G., Philippsen, M. (2000, March 28). JPlag: Finding Plagiarisms among a Set of Programs. DOI: 10.3217/jucs-008-11-1016. URL:
https://www.jucs.org/jucs_8_11/finding_plagiarisms_among_a/Prechelt_L.html
- Prechelt, L., Malpohl, G., Philippsen, M. (2002). Finding Plagiarisms among a Set of Programs with JPlag. *Journal of Universal Computer Science*, 8(11), 1016–1038.
- Schleimer, S., Wilkerson, D.S., Aiken, A. (2003). Winnowing: Local Algorithms for Document Fingerprinting. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, 9–12 June*, pp. 76–85. ACM New York, NY, USA.
- Sharma, T., Kechagia, M., Georgiou, S., Tiwari, R., Sarro, F. (2021). A Survey on Machine Learning Techniques for Source Code Analysis. *ACM Transactions on Software Engineering and Methodology*, 0(0), 0–0. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>
- Všianský, R., Dlabolová, D., Foltýnek, T. (2017). Source Code Plagiarism Detection for PHP Language. *European Journal of Business Science and Technology*, 3(2), 106–117. DOI: 10.11118/ejobsat.v3i2.100
- Wise, M.J. (1996). YAP3: Improved Detection of Similarities in Computer Programs and Other Texts. *ACM SIGCSE Bulletin*, 28(1), 130–134.
- WU, Runfan & LV, Aohui & Zhao, Qiyang. (2022). Detecting Plagiarism as Out-of-distribution Samples for Large-scale Programming Contests. *Olympiads in Informatics*. 89–106. DOI: 10.15388/oi.2022.08.
- Yang, H., Lian, W., Wang, S., Cai, H. (2023, May). Demystifying Issues, Challenges, and Solutions for Multilingual Software Development. In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)* (pp. 1840–1852). IEEE. <https://chapering.github.io/pubs/icse23haoran.pdf>
- Yetthapu, Sudheer (2023). Source Code Plagiarism Detection Using JPlag & Stack Overflow Data. *Masters Theses & Specialist Projects*. Paper 3620. <https://digitalcommons.wku.edu/theses/3620>



M. Mammadli is currently a last year student at ADA University pursuing her bachelor’s degree in Information Technology. She holds the position of an IT business analyst at ERP-INTEL LLC.



N. Mammadli is a last year student in the bachelor’s program of “Computer Science” offered by ADA University. Concurrently, he works as a software developer at ERP-INTEL LLC.



J. Hasanov is an Associate Professor of Computer and Information Sciences in the School of IT and Engineering at ADA University. Dr. Hasanov is mainly focused on computer vision problems medical imaging and video captioning domains. Additional to the research field, Dr. Hasanov teaches the management aspects of the IT in production and operation. Dr. Hasanov has been an ITC member for the period of 2017–2020 and led HTC during IOI 2019.

Preparing of Youngest Students for Participation in Programming Contests

Krassimir MANEV

zh.k. Yavorov, bl. 12A, entr. A., Sofia, Bulgaria
e-mail: krmanev@gmail.com

Abstract. The goal of this paper is to present in brief the analysis of large set of tasks from Bulgarian national and regional programming contests for age group E (4th–5th grades), published in (Manev, 2023). As a result of the analysis, the identified in more than 350 tasks topics are arranged in order of smoothly increasing difficulty and could be used as training curriculum for preparation of programming contestants of early age.

Keywords: programming contests, training curriculum, beginners.

1. Introduction

Programming contests in Bulgaria are organized in five age groups (Manev *et al.*, 2007). In the youngest group E students of 4th–5th grades which just started learning programming take part. That is why choosing tasks for their first programming contests must be in accordance with the stage of learning a programming language (C/C++) they reached.

Recently we published a book aimed to help the process of training the youngest students for participation in programming contest, as well as to identify the types of tasks, which are appropriate for the distinct stages of the preparation (Manev, 2023). In the process of creating the book all tasks for group E from the national programming contest since 2007 was analyzed in the order they appeared in contests' calendar:

- Autumn tournament (AT, held usually in November, 51 tasks);
- Winter tournament (WT, that was held in December since 2009 to 2017, 27 tasks);
- First round of the National Olympiad in Informatics (NOI1, held usually in January, 45 tasks);
- Second round of the National Olympiad in informatics (NOI2, held usually in February, 45 tasks);
- Third round of the National Olympiad in informatics (NOI3, held usually in Marc, 87 tasks);

- Spring tournament (SpT, held usually in April, 48 tasks);
- Summer tournament (SuT, held in June from 2018 till now, 15 tasks).

Recently two regional contests are also organized, tasks of which were also analyzed:

- Sofia autumn tournament (SAT, held in October, 18 tasks);
- Sofia spring tournament (SST, held in beginning of April, 15 tasks).

A total of 351 tasks we analyzed, part of them solved as illustrations of the corresponding theoretical material, and for each of the others a guidance for solving was given – shorter or vaster.

In this paper we shortly present the chapters of the book. The chapters are ordered in the way we are teaching them when training young contestants. So, the content (and the order of topics) could be used by the coaches as one-year curriculum for training beginners.

2. Preliminary Knowledge and Skills

We suppose that before beginning of their preparation for programming contests the pupils passed the course Intro to programming in C/C++, which is why we are not including in the book text about the basic programming skills. The first chapter, called **Programming Contests**, is dedicated to such knowledge and skills about the programming competitions that introductory course in programming does not contain.

The name of the first section in the chapter is **The Contest Tasks**. Because the goal of introduction to programming is to familiarize learners with the syntax and the semantics of the programming language, the tasks that are solved in such course are, in some sense, “artificial” – they are strongly oriented to some specific syntactical or semantical rules. Traditionally the contest tasks are different (Verhoeff, 2008). They look like taken from the real life and they are real life problems indeed, in most cases. Beside the inevitable problem formulation, the competitive task must contain strict description of input and output formats, the constraints on input data, sample test data and correct output for the sample test data – sections that are not usual for tasks solved in introductory courses but are so important for the programming contests.

Second and third sections of the chapter – **The contest systems** and **Evaluation of a contest tasks** – are introducing future contestants in the process of grading their solutions. Nowadays using programming contest grading systems is inevitable. In Bulgarian programming contests we recently are using own grading system BOS, which is conform with the systems used in International Olympiads, and that all contestants, including the youngest must know (Petrov & Kelevedjiev, 2022). Good knowledge of grading system and the process of grading leads contestant to a systematic approach they have to apply during the work on a contest task, in order to escape rejections for: Unsuccessful compilation (CE), Memory limit exceeding (MLE), Canceling the execution because an exception (RTE), and especially the Time limit exceeding (TLE) and Generating of a wrong answer (WA).

Fourth, fifth and sixth sections of the chapter are dedicated to some important for competitive programming elements of the programming language and the integrated development environment which are not a subject of the introductory course. In section **Including Standard Functions Libraries** we only stress on the modern form of including libraries with preprocessor instruction `#include <bits/stdc++.h>`, which liberate contestants of necessity to memorize where a standard function is defined. Section **Accelerating Input and Output** is particularly important. Beginners prefer to use stream input/output (`cin` and `cout`) instead the functions for formatted input/output. That is why they must be able to desynchronize stream input/output from the formatted one. Section **Preprocessing** in C/C++ is introducing helpful instructions that the preprocessor offer.

In seventh section – **Memory usage** – we briefly describe the organization of the computer’s memory. Main issue for the beginners there is avoiding declarations of arrays inside the functions body, to not overfull the stack of the process. Not easy for understanding concepts standard input and standard output are discussed in the eight sections of the chapter. This discussion is particularly important for avoiding the bad practice of typing input data from keyboard when testing the code. This bad practice is unacceptable for programming contests because it is time consuming and risky. Last section of the chapter is brief introduction to possibilities of the command interpreter. The focus is placed on using command procedures with redirection of the standard input/output to text files. Without using this approach testing of the code, especially with large test cases, is impossible.

3. Early Stage of the Preparation

We call the early stage of the preparation the period when usage of loops and arrays are avoided. In this stage tasks in Bulgarian programming contest for beginners are created on the base of knowledge and skills of the contestants gained in math classes.

3.1. Quotient and Remainder

Second chapter of the book is dedicated to topic **Quotient and Remainder** – the results of the integer division n/m of two natural numbers n and m , $m \neq 0$. The theory in this topic is elementary and well known from math classes. One typical task in this group, for solving of which only ability to compose correct arithmetic expressions, is the following:

KLETKI (AT 2015, Group E)*. K pigeons landed in a line of N cages, one pigeon in a cage, $K \leq N$. The number of the empty cages between two neighbor pigeons is called

* Full statements of the tasks mentioned in the article could be found (in Bulgarian), trough the link Tasks (in Bulgarian **Задачи**) and then trough the links E for different national contests, on the training site of the Bulgarian Olympiads in Informatics <https://arena.olimpiici.com>

distance between them. When landing pigeons are trying to maximize the distances between each two neighbors. Having in mind that it is not always possible to make distances equal, write a program to find the number of minimal distances of the best landing.

For example, if $N = 8$ and $K = 4$ then the best landing will be with two 1-distance couples and one 2-distance couple, so the asked number is 2.

Solving tasks for finding quotient and/or remainder is an appropriate moment to introduce the standard functions `floor` and `ceil`, which are not studied neither in school nor in an introductory course.

3.2. Ordering of Numbers and Characters

Next topic that we teach is **Ordering of numbers and characters**. Without using arrays and loops, the tasks of this kind require ordering (we do not even mention the term sorting) of only 3–4 numbers or characters. Nevertheless, the moment is appropriate to attract attention of the students to the big topic **Sorting**. This is the moment to attract the attention of the students on the fact that values of the variables of type `char` are numerical and the order is the natural – *letters* are ordered as in the alphabet – all capitals before all small letters, and *digits* by their numerical values. So, there is no difference in the procedures of ordering numbers and characters.

The natural technique bubble sort, which is easy understandable by the beginners, is used. We start with demonstrating the operation exchanging of two values – first by using an intermediate variable and then we introduce the standard function `swap`. As a side effect of ordering, we get the minimum element of the set, which is the first element in the arrangement, and the maximum, which is the last element.

Another side effect is that we carefully introduce students in the topic **Time complexity of the algorithms** (in the worst case) which is of enormous importance for the future contestants and must be systematically taught. Time complexity in this case is eased to introduce as the number of executed by the program operations. It is important for students to understand that the complexity of these tasks is a constant function and to perceive the asymptotic notation $O(1)$. The following task is typical for this topic:

GUESSN (NOI2, 2023). *Three of the numbers $a_1, a_2 = a_1 + d, a_3 = a_2 + d$ and $a_4 = a_3 + d$ are given in random order. Write a program to find the fourth number.*

For example, if the numbers 4, 8 and 6 are given, after ordering them we could conclude that fourth number is 2 or 10. If the given numbers are 10, 1 and 4, then after the ordering the gap between 1 and 4 is 3, the gap between 4 and 10 is 6 so the answer is 7.

Then in the chapter we discuss some more difficult ordering tasks. First, we consider tasks for ordering of objects with two parameters by two criteria – first by one of the criteria, and when we have two objects with equal parameters by this criteria values – then by the other. One such task is the following:

BOOKS (NOII, 2015). *Four books of given height and thickness must be arranged so that of two books of different heights, the one with the greater height is on the left, and if they are of equal height, on the left to be the thicker one. Write a program to find such an arrangement.*

For example, if the heights/thickness of the four books are 2/23, 70/150, 70/100 and 22/37, then the asked order is 70/150, 70/100, 22/37 and 2/23. This task is appropriate also to demonstrate that there is not much difference between ordering in decreasing and increasing order.

A version of ordering by two criteria is when values of the second parameter, for example the ordering number in the input of the objects, are not involved in the ordering. In such case the values of the second parameter have just to be swapped always when the main parameters' values are swapped. For example:

CAKES (NOII, 2014). *Three cakes with different diameters, labeled with 1, 2 and 3 by the order their diameters are given in the input, must be arranged one over the other in such way that cake with bigger diameter is not arranged over cake with smaller. Write a program to output the required order of the labels of the cakes starting with the biggest one.*

In this chapter we also introduce the standard functions `min` and `max`, which could be helpful in some tasks.

3.3. Positional Number Systems

The topic **Positional number systems** is fundamental for education in computer science. But it is also appropriate for making tasks for the early stage of the preparation of contestants. Main difficulty in the topic could be the bad knowledge of the operation exponentiation of the base of the system, which is crucial for understanding the corresponding algorithms. That is why some knowledge about the operation must be taught in the beginning of the topic. Because arrays and loops are not used during this stage the considered numbers are with no more than 3–5 digits. The numbers are usually in decimal system and very rarely in some other – binary, ternary, etc.

Two are the main subtasks that the tasks of this kind could contain. First subtask is to separate the digits of a given number and second is to restore a number from given digits. A task could contain one of the two subtasks or both. For the first task a contestant could consider the given number as value of type `int` or as a sequence of values of type `char`. In first case the knowledge of the topic Quotient and Remainder is necessary because the last digit of the decimal number is the remainder of the number's division by 10 and integer division by 10 will remove the separated digit. If the number itself is necessary for some steps of the algorithm – this is the best way to separate its digits.

If the number itself is not necessary for some steps of the algorithm it could be input in few variables of type `char` which will be the necessary separating of the digits. For

the purpose students must know only how to transform ASCII value of the characters to corresponding digits. This approach could be used in the case when the number is necessary for some steps of the algorithm, but in such case the number must be restored from its digits.

Because the numbers in these tasks are small their restoring could be done by multiplication of the digits by corresponding exponents of 10 and summing of obtained terms. But we prefer to introduce for the purpose the Horner's rule which is the only alternative for the same kind of tasks but for larger numbers. Something more, Horner's rule eliminate the necessity to precalculated exponents of 10 when the number of digits is arbitrary, decreases in such a way the number of multiplications and ameliorate the time complexity.

The following task is typical for this topic:

DIFFERENCE (NOII, 2015). *Let A be the smallest and B the biggest number which are formed by the digits of given number N , $100 \leq N \leq 999$. Write a program to find the difference $B - A$.*

3.4. Metric Units

In tasks of the topic **Metric units** some calculation is usually required that involve different metric units of the same kind – for weight, distance, time, money. The standard approach here is to transform all units to smallest one, mentioned in the statement of the task, to make calculation with smallest unit and then to restore the result in the format required by the task's statement.

As a sample see the following tasks:

SONG (NOII, 2009). *A musical composition which is n minutes and m seconds long must be recorded on a disk. The free space on the disk is k MiB, and to record one second of sound 16 KiB is required. Write a program that outputs YES if the song can be recorded or NO if it cannot. In case the free space on the disk is not enough for recording the song, then the program must print how many KiB are not enough.*

A specific case in this topic are the tasks in which a calculation with dates are necessary. For example:

DATE (NOII, 2009). *Write a program that, given a valid date consisting of day d , month m and year y , finds the date of the next day. A year is a leap year if it is divisible by 4 but not divisible by 100 or divisible by 400.*

Because tasks for manipulating dates appear sometimes in the contest task set we recommend to students to create a function `nextday` that solve the task. The three variables are passed to the function as global and the result is obtained in the same variables to escape the not easy problem of passing parameters by pointers. This is the moment we suggest to student for first time to start creating their own auxiliary functions.

3.5. Ad Hoc Tasks

In competitive programming we call a task *ad hoc* (from Latin – for specific or immediate needs) if it could not be classified in any topic and usually there is no well-known algorithm/approach for its solving. We consider the ability to solve ad hoc tasks very important because it develops the creativity of the contestants – future professional software developers. Something more, most of the tasks given in international programming contest are ad hoc per se.

For the chapter **Ad hoc tasks**, we made thorough analysis of the ad hoc tasks that appeared in contests for the earliest age in Bulgaria, solving of which do not require loops and arrays. The goal is to propose to the beginners idea how to proceed with this kind of tasks.

The first kind of ad hoc tasks that we identified could be called “*does what the statement requires*”. That means the procedure/algorithm that the contestant has to code is described in the statement of the task. It could seem that these kinds of tasks are easy, but it is not always true. Sometime such task requires a perfect programming skill and could take a lot of the contestant’s time (so called time killer tasks). For example:

CONDITIONING (NOII, 2014). *Air conditioner executes inside one hour commands of the format:*

<code> <actual temperature> <required temperature>

where the code is one of the following:

- *f – downgrades the temperature to the required, but if the actual temperature is lower than required, does nothing.*
- *h – increases the temperature to the required, but if the actual temperature is higher than required, does nothing.*
- *a – downgrades the temperature if the actual is higher then required or increases the actual to required otherwise.*
- *v – the air conditioner only ventilates the air and does not change the temperature.*

Write a program that for given actual temperature, required temperature and a code of a command output the actual temperature after an hour.

Most of the other ad hoc tasks of this stage could be called “*consider the different cases*”. For example:

TOURIST (WT, 2010). *Group composed of K students are preparing for a hike in the mountain and must choose one, two or three of the available tents so that the weight of the chosen tents is no more than W kilograms and they are able to accommodate all students. The first tent weighs A_1 kilograms and accommodates B_1 students, the second weighs A_2 kilograms and accommodates B_2 person and the third weighs A_3 kilograms and accommodates B_3 person. Write a program that determines in how many ways can be selected the tents.*

Identifying the different cases that the program must consider is not easy sometime. Some initial knowledge for generation of combinatorial configurations – permutations,

combinations, and variations – over sets with 3–5 elements will be necessary, as well as some skill of splitting the possible input data to equivalent cases.

4. Tasks that Require Loops

For the second half of the season, we include in the preparation the tasks that require loops considering ability of students to organize loops one of the most important skills for the future programmers. With solving tasks that require loops we can start to discuss seriously the time complexity of the used algorithms.

4.1. *Properties of a Set or a Sequence of Elements*

The simplest tasks that require loops are the tasks for finding some properties of numerical sets or sequences of elements. For example, finding the minimal or maximal element, the sum of the elements, the average for a numerical set and so on. Such kind of tasks we have solved for small number of elements in the early stage of the preparation and now we just extend the skills of the students to solve them with using loops. In a similar way we also extend the skills from the early stage for finding optimal/extremal element of a set or sequence when the elements have two or more parameters.

In this category we could also classify the tasks which require to solve many times task that was solved earlier. For example, to include in a loop body code of the function `nextday` (see Subsection 3.4) to so solve the task: for given date find which will be the date after n days.

4.2. *Finding Subsequences*

Many tasks that need using of loops of national or regional contests are for finding subsequences with some properties. These tasks could be solved without using arrays when two conditions are valid: the required subsequences do not overlap and not the subsequence itself is required but some characteristic of it – for example, longest or shortest one. Sample of such task is the following:

LOVABLE (WT 2009). *A sequence of integers is called sympathetic if it contains a subsequence of length at least 2 composed of the one and the same integer. Write a program that checks whether a given sequence is sympathetic and if it is sympathetic to output the integer composing longest subsequence delete it of one and the same integer. If there is more than one such subsequence, then the biggest integer that composes such subsequence has to be output.*

For solving such kind of tasks we suggest to students to keep the status of the search in 5 variables:

- `new` – for the currently considered integer;

- `last` – for the integer considered lately;
- `len` – for the length of currently considered subsequence;
- `maxlen` – for the length of longest found to the moment subsequence;
- `maxnum` – for the integer of the longest found to the moment subsequence.

When `new == last` we just increase `len` by 1, and when is not then we update `maxlen` with `len` when `len > maxlen` or `len == maxlen` but `last > maxnum`.

4.3. Nested Loops

Organizing nested loops is the most difficult topic in this kind of tasks. Especially when the boundaries of changing of the control variable of the inner loop is depending on the value of the control variable of the outer loop. We carefully introduce the approach for building nested loops through tasks for drawing figures with characters. For example:

TRIANGLE. Write a program to output an equilateral triangle of height n as shown on the Fig. 1.

For solving the task first an outer loop with n steps is organized for drawing on i -th step a row of the triangle. In the body of this loop, we must organize two inner loops – one to output sequence of intervals (with control variable j) and one for output sequence of asterisks (with control variable k). Observing the example from statement of the task for $n = 6$ we suggest creation of the Table 1 to identify the boundaries for j and k .

Now it is easy to write the corresponding code.

```

      *
     ***
    *****
   *********
  ***********
 *****

```

Fig. 1.

Table 1

	For j		For k	
	From	To	From	To
1	1	5	1	1
2	1	4	1	3
3	1	3	1	5
4	1	2	1	7
5	1	1	1	9
6	1	0	1	11
i	1	$n - i$	1	$2i - 1$

4.4. More about the Complexity of the Algorithms

We are using tasks that need loops for deeper consideration of the problem for evaluation the complexity of algorithms. In this case the function of complexity is no longer constant like in the tasks that do not need loops. It is relatively easy to explain that the single loop of n steps that has body of constant complexity $O(1)$ is $O(n)$. It is not difficult to understand also that the complexity of two nested loops, outer making m steps and inner – n steps is $O(mn)$.

Difficulties arise by the case when the boundaries of the control variable j (or k) of the inner loop depend on the value of the control variable i of the outer, like in the example above. In this case students must be able to find the complexity $T(i)$ of each inner loop, for each value of i and to sum obtained functions. Let $T_1(i)$ be the time complexity function of first inner loop for the example above and $T_2(i)$ – of the second. From Table 1 we have obviously $T_1(i) = n - i$ and $T_2(i) = 2i - 1$. So, for the complexity $T(n)$ of the algorithm we have:

$$\begin{aligned} T(n) &= T_1(1) + T_1(2) + \dots + T_1(n-1) + T_1(n) + T_2(1) + T_2(2) + \dots + T_2(n) = \\ &= (n-1 + n-2 + \dots + 1 + 0) + (1 + 3 + \dots + 2n-1) = \\ &= n(n-1) / 2 + n^2 = O(n^2). \end{aligned}$$

For explaining how the first sum is calculated we are using the “proof” of the young Gauss, and for the second – just observation of the partial sums: 1, 4, 9, 16, and so on.

4.4. Ad Hoc Tasks

Having the loop construction some more difficult and more interesting tasks could be formulated. For example:

JUMPS (SST, 2022). *A grasshopper is perched on one end of L cm long stick, $L \leq 10^{18}$, and makes successive jumps along the stick towards its other end, until with the last hop falls from her. First jump is m centimeters long, and each subsequent one is longer than the previous by n centimeters. Write a program that determines how many jumps the grasshopper made on the stick before it falls of it.*

Trivial simulation of the process will not pass the tests with very large L and small m and n . For accelerating the simulation let us precompute, using the summation formula that students know, the length P_0 of the first 100 jumps of the grasshopper:

$$P_0 = m + (m + n) + (m + 2n) + \dots + (m + 99n) = 100m + 4450n.$$

The distance P_1 for next 100 jumps will be:

$$\begin{aligned} P_1 &= (m + 100n) + (m + 101n) + (m + 102n) + \dots + (m + 199n) \\ &= 100m + 100 \cdot 100n + 4450n = P_0 + 10000n, \end{aligned}$$

and so on – for each 100 jumps the past distance will be $10000n$ cm longer than the past distance by the previous 100 jumps and the length of the current jump will increase by $100n$. Using this observation, the length of the simulation, which means the complexity of the algorithm also, will decrease about 100 times.

If we precompute the lengths of the first 1000, second 1000, and so on jumps instead 100, which is not so different, we will accelerate the simulation 1000 times.

5. Using Arrays

Arrays are extremely important instrument for future programmers. Including tasks that require arrays we start teaching young contestants to structure its data – correctly and efficiently. Because only the adequate combination of algorithms and data structures could produce efficient programs (Wirth, 1976). Using arrays, we teach the contestants to implement basic abstract data types such as queues, stacks, maps, frequency chart, etc. in static arrays before the start of using the dynamic implementations of STL. This gives to authors possibilities to create more interesting and more edifying tasks, for solving which more sophisticated algorithmic approach are necessary. In this topic we include also different tasks over strings because the strings are *de facto* arrays of characters.

5.1. Sorting and Merging

Ordering 3–5 values at the early stage of the preparation we demonstrated the importance of the sorting of data for efficient solving of some tasks. At this stage it is time to stress this importance, especially for large amounts of data, to demonstrate that classic $O(n^2)$ algorithms are rather not applicable for hundreds of thousands or million elements. We introduce standard sorting function of STL, which use fast $O(n \cdot \log n)$ algorithm, carefully explaining $\log n$ function which the students do not know from mathematic classes yet.

We use the discussions about the different sorting algorithms and their time complexity to introduce the linear $O(n + m)$ algorithm *Counting sort* of n integers when elements are smaller than m . The very simple implementation of this algorithm makes it inevitable alternative of the standard STL method when data is appropriate.

This is the moment also to introduce algorithmic approach known as *merging of sorted areas*. It happens that beside the initial purpose of the approach – to be a part of the efficient $O(n \cdot \log n)$ *Merge sort* – the approach is applicable to some different tasks. For example finding the union and intersection of two sets, represented in sorted areas, etc. Example of such task is the following:

GARDEN (NOI3, 2014). *In one row (bed) n decorative bushes have been sown and in a second – m decorative bushes. If two bushes of the same height are found in each of the two beds, one of them must be moved to a third bed. Write a program that finds the heights of the bushes in the new bed, arranged from the lowest bush to the tallest one. There will be always at least one bush in the new bed.*

Searched set is the intersection of two sets. First, we are sorting the two arrays and then apply a small modification of the classic merging – only when the values of the two considered elements are equal, we move a bush of this height to the third bed. Final loop of the classic merging of two sorted arrays is not necessary at all.

5.2. Finding Subsequences

In subsection 4.2 we excluded from considering the tasks for finding subsequences when the possible subsequences overlap or/and the subsequence itself must be found. With usage of an array this kind of tasks are solvable. If the subsequence must be output as a result we just append to the status of the search, defined in 4.2, two more variables to memorize the begin and the end of the best found to the moment subsequence.

If the subsequences could overlap, we introduce the algorithmic approach that we call *sliding of window* (called by some colleagues a two pointers approach). There are two kinds of such tasks – when the required subsequence is of fixed length and when is of variable length. Searching subsequence of fixed length L , we start with opening of the window over first L elements of the sequence and then we slide the window “closing” it by one element from the beginning and extending it by one element after the end. If subsequences do not overlap sliding a window of fixed length is possible without using an array also. Appropriate for sliding window of fixed length is the following task:

SUMDIGITS. *Write a program to find a subsequence of length k of a sequence of n integers having maximal sum of digits of its elements.*

Crucial for the speed of program in these tasks is the efficient finding the properties of the new window. For this purpose, we recommend to students to keep in appropriate structure the necessary data for the fast calculation properties of the window. For the task above, for example, recommendation is to keep in another array the sum of the digits of each integer so for sliding of the window one subtraction from and one addition to the current sum are enough.

If the searched subsequence is of variable length, then longest or shortest subsequence with the given properties must be found usually. In this case, for sliding the window, more than one element from the beginning could be eliminated (until the necessary property is no more valid) and then one or more elements to be appended to the end (until the property get valid again). Example:

ALL LETTERS. *Write a program that for given text (sequence) composed of small letters find the shortest subsequence of consecutive letters such that each letter is included in the subsequence at least once.*

For fast calculation property of the searched window in this task we recommend keeping a map where for each letter to save how many times it is included in the current window and a counter of the different letters in the window.

5.3. Linear Abstract Data Types

As mentioned above, for solving tasks of this group implementations of some linear abstract types are necessary sometime. Besides the usual queues and stacks especial attention is paid to implementation of different kind of *maps*, that lead to more efficient solutions. Here we clearly differentiate the dynamic implementations of STL from the static implementations that the students must be able to code themselves, stressing the specific features of the two different implementations.

The advantage of dynamic implementations of maps from STL is that they allow a large enough range of the keys, using memory only for couples that are included in the map. The disadvantage is that, because the maintaining of the map is in some tree structure, usual operations (inserting in the map and check for presence of a key) are of complexity $O(\log n)$. Static implementation in an array (or in vector) will need memory proportional of the range of keys and is not applicable when the range of keys is large enough. But in the static implementations the complexity of usual operation is constant. So, the ability of the contestant to choose which of the two kinds of implementations to choose, depending on the task, must be trained systematically.

5.4. Two Dimensional Arrays

In contests for this age group there are not so many tasks that require two dimensional arrays. But it is an important step in teaching contestants to structure their data. And more, including two dimensional arrays in the training give even more possibilities to create interesting tasks with increasing complexity. These tasks almost always require using the nested loops and develop skills of the contestants to organize such loops and to evaluate the time complexity of their algorithms.

Main object in these tasks usually is a rectangular table of cells with given number of rows and columns for which an element (or elements) with some properties is (are) searched. Example is the following task:

MINESWEEPER (AT, 2016). *The board of the Minesweeper game is divided into nine equal cells with quadratic form – three rows with three cells in each row. In some of cells, that are labeled with 9, there is a bomb. The others, labeled with 0, are empty. The player's goal is, for each empty cell, to find the number of bombs in its neighbors – these that share a vertex or side with it. Write a program to find the required numbers.*

Principle difficulty in such a task could be the fact that cells on corners of the board, lying on the sides of the board and the inner cells have different number of neighbors – 3, 5 or 8 respectively. A very helpful approach in such cases is to border the table – up, down, left, and right – with *neutral cells*. In our example these cells must contain zeros. Then all cells of the board will have equal number of neighbors.

For exercising the organization of nested loops we use the classic task:

F	i	n	d
r	i	n	
t	!	g	t
s		e	h

Fig. 2.

BY SPIRAL. In each cell of a table with m rows and n columns one of the letters of a texts is written in spiral order. The spiral starts in upper left corner of the table and go first right, then down, left, and up, then right again and so on, without repeating a cell (see Fig. 2 for $m = n = 4$). Write a program to restore the text.

Solving this task, the approach from Section 4.3 for identifying the borders of four inner loops of the spiral must be applied.

Many other tasks in this subtopic could by classified as ad hoc and need approaches that we discussed above.

6. Divisibility

The chapter **Divisibility** of (Manev, 2003) is dedicated to the divisibility of natural numbers. It is extending in a natural way the topic **Quotient and remainder** that we started with. First, we discuss the main notions of the subject – *prime number*, *divisor* and *multiple*, *greatest common divisor*, *smallest common multiple*, as well as the trivial algorithms that follow from the definitions. For example:

- to check if a given natural number n is prime by finding its remainders of division to the numbers, less than or equal to the square root of n (here we introduce in intuitive level the notion *square root*);
- to factorize a given natural number n to its prime divisors by checking for divisibility to each smaller number as many times as necessary;
- to find the number of different divisors of a natural number n from the degrees of its prime divisors.

Examples of such tasks are:

PRIME (ST, 2008). Write a program that determines how many prime numbers are in given sequence of n positive integers, where $n < 100$ and numbers are less than 200000.

and

PRIMES (WT, 2010). Write a program that determines how many digits in total have the prime numbers in given interval $[A; B]$, $A < B$.

Then we introduce *Euclid's' algorithm* for finding greatest common divisor (GCD) and least common multiple (LCM) of two natural numbers, as well as the Eratosthenes'

sieve for finding the prime numbers less than given n . Showing here that complexity of Euclid's algorithm for finding $\text{GCD}(m, n)$, when $m < n$, is $O(\log n)$ is a real challenge for the mathematical culture of the contestants.

Examples of such tasks are:

CHESTNUTS (NOI3, 2013). *On the main street of the host city of the National Olympics are planted n chestnuts arranged in a straight line. The distances between the different neighboring trees are different. Write a program that finds the minimum number of trees that must be planted between the given so that the distances between every two neighboring trees are equal.*

MIRROR (NOI3, 2013). *Write a program that determines the number of these prime integers p in given interval $[A; B]$, $A < B$, for which is true that p is equal to the number q obtained by reading digits of p in reverse order.*

7. Other Tasks

In the last chapter of the book, we consider tasks that were not classified in the previous chapters. One such kind of tasks are the tasks that require usage of prefix values for solving some range query tasks. For example, maximal sum in the range or minimum/maximum in the range. Such an approach is inevitable when many queries for finding some extremal range must be executed. Example for such task is the following:

MAXIMAL SUM. *Write a program that for given sequence of n integers, absolute value of each of which is less or equal to 1000000, execute q queries for finding the maximal sum in q given intervals, $n \leq 1000000$, $q \leq 1000000$.*

It is obvious that with building prefix sums of the given sequence for $O(n)$ time, we could find each of the required maximal sum with time complexity $O(1)$.

Another, very specific kind are the tasks that excite *scanning of a raster*. In this kind of task, we call a *raster* some (one- or two-dimensional) structure that contains one *pixel* of data for each moment of an interval of time or for each point of some space. The essence of these tasks is to build a raster to keep the given data and then carefully to scan the raster to find the solution.

Example of such task is the following:

AIRPORT (NOI3, 2012). *In one day, N planes land and take off at an airport. It has been known that a subsequent aircraft may re-use a sleeve 10 minutes after the previous aircraft has disengaged from the arm. Write a program that determines the minimum number of sleeves that are required for servicing all aircraft taking off and landing. Times in the schedule are set in hours and minutes.*

For solving the task, we must build a raster, having one pixel for each minute of the day and for each landing airplane to increase by 1 the value of pixels that correspond of its staying in the airport and to decrease them by 1 after the depart of the airplane. Then the required value is the maximal value of the raster recorded in some minute of the day.

In the end of the chapter, we also consider some ad hoc tasks that happened to be difficult according to the results of the contestants from the corresponding contest. Two examples of such tasks are:

DIG (NOI3, 2015). *Given a positive integer n with no more than 30 digits. Write a program that finds the smallest k -digits positive integer that contains at least once each of the decimal digits that are not digits of n .*

EXPFIELD (ST, 2021). *Students of one class had been asked to plant $m \cdot n$ plants labeled from 1 to $m \cdot n$ in the experimental field of their school, in m row lines and n column lines. The students have planted them “by rows” – the plants labeled from 1 to n in the first row, plants labeled from $n + 1$ to $2n$ in the second and so on. Then the teacher of them explained that the correct order is “by columns” – plants labeled from 1 to m in the first column, plants from $m + 1$ to $2m$ in the second and so on. Write a program to find the number of plants that will not be replanted.*

We will leave to the interested reader to find the best solutions of these tasks.

8. Discussion

Bulgarian experience in preparation of young contestants is big enough. From the beginning of 90's separate contests for youngest contestants (5th–7th grade) was organized and some years later the contests for students of 4th–5th was separated too. Since then, the intensity of the contests calendar for the youngest is the same as for all other age groups (with shorter contest's time). We consider that efforts for early preparation are crucial reason small-populated Bulgaria (less than 7 million in the moment) to be still in the top of the countries by the results in IOI, for example – 4th–5th by the total number of medals, and 9th by their quality (see <https://stats.ioinformatics.org/countries/>, links Total and Medals).

Training of very young programmers, especially for participation in programming contests, is a specific activity. Pupils in this age have not clear vision for their future profession yet and just try to find their road in different directions. And the road of the professional programmer is not easy at all. That is why augmentation of difficulties in the training process must be smooth and careful. The main goal of our textbook was to arrange the material in such way that passing from one topic to the next to be as easy as possible.

Some important conclusions for the early preparation of young contestants could be extracted from the analysis of the tasks from Bulgarian national and regional programming contests. On the first place this is the fact that, in the beginning, contestants have not good command of the programming language, and are not still ready to understand and use sophisticated algorithm and data structures. That is why the contest tasks are based mainly on school mathematics. Our analysis of the mathematic skills necessary for successful start in the competitive programming was published in (Manev, 2024).

Our statistic is definitive – most of the school students that demonstrate an early will to participate in programming contests in Bulgaria are coming from the specialized mathematical schools that accept students of 5th grade. Most of the necessary mathematical knowledge and skills mentioned in our analysis the students in some mathematical schools get in math classes or in out of school classes educational forms. These contestants can master the material from the book for one-year being in 5th grade. Students that are not in these mathematical school must start preparation in 4th grade to master necessary material and to have some success in the contests.

We sincerely hope that our experience will be helpful for colleagues that are engaged with the preparation of contestants at an early age and we are ready to discuss more details through personal contact.

References

- Kelevedjiev, E., Branzov, T., Petrov, P., Shalamanov, M. (2020). Bulgarian Platform for Competitions in Informatics. *Mathematics and Education in Mathematics*, pp. 123–130. [In Bulgarian: Келеведжиев, Е., Брънзов, Т., Петров, П., Шаламанов, М., Българска платформа за състезателна информатика. *Математика и математическо образование*, стр. 123–130.]
http://www.math.bas.bg/smb/2020_PK/tom_2020/pdf/123-130.pdf
- Manev, K. (2023). *Introduction to Competitive Programming*. KLMN, Sofia [In Bulgarian, Кр. Манев, *Увод в състезателното програмиране*. КЛМН, София.] ISBN 978-954-8212-12-0
- Manev, K., Karadjova, R. (2024). Mathematics for Beginning Programers. *Mathematics and Education in Mathematics*, pp. 25–35. [In Bulgarian: Манев, К., Караджова, Р., *Математика за начинаещи програмисти*. *Математика и математическо образование*, стр. 25–35.]
http://www.math.bas.bg/smb/2024_PK/tom_2024/pdf/025-035.pdf
- Manev, K., Kelevedjiev, E., Kapralov, S. (2007). Programming Contests for School Students in Bulgaria. *Olympiads in Informatics*, 1, 112–123.
- Verhoeff, T. (2008). Programming Task Packages: Peach Exchange Format. *Olympiads in Informatics*, 2, 192–207.
- Wirtt, N. (1976). *Algorithms + Data structures = Programs*. Prentice-Hall Inc. Englewood Cliffs, New Jersey. ISBN 0-13-022418-9



K. Manev is a retired professor, PhD in Computer Science. He was teaching Discrete mathematics, Programming and Algorithms in many Bulgarian universities – mainly Sofia University, American University in Bulgaria, and New Bulgarian University. He has published over 75 scientific papers and more than 30 textbooks in the fields of Informatics and Information Technologies. He was member of Bulgarian National Committee for Olympiads in Informatics since 1982 and President of the Committee from 1998 to 2002; also leader or deputy leader of Bulgarian team for IOI (International Olympiads in Informatics) in 1989, 1998, 1999, 2000, 2005 and 2014; member of the organizing team of IOI'1989 and IOI'1990; chairman of IOI'2009. From 2001 to 2003 and from 2011 to 2013 he was elected member of International Committee of IOI, from 2005 to 2010 – member of IC, representing the Host country of IOI'2009, and from 2015 to 2017 – a President of IOI. In 2017 he was one of the creators of European Junior Olympiad in Informatics and its President from 2017 to 2020.

Olympiads without Words

Pavel S. PANKOV, Elzat J. BAYALIEVA

Institute of Mathematics, Kyrgyzstan

J. Balasagyn National University, Kyrgyzstan

e-mail: pps5050@mail.ru, elzat.bayalieva@gmail.com

Abstract. To involve as many children as possible from an earlier age in the Olympiad movement, tasks without any conventional signs and denotations are proposed (these tasks can be called “natural”). Drag-and-Drop technique is used for this purpose, with varieties: “n-to-1” (selecting an object to be dragged to the spot); “1-to-m” (selecting the spot or a place the object to be dragged to); “n-to-m” (selecting both an object or a collection of objects and the spot(s) to be dragged to). Parametrizing is necessary: when restarting the program, a slightly different task is to be generated. A sequence of these tasks (in learning mode for any definite language: with announcements after successful completion: “Congratulations, you have mastered the notion of...”) would also be an unobtrusive training course for children. These tasks can be also used for checking and improving AI.

Keywords: Olympiad, children, parametrized task, common sense, guessing, Drag-and-Drop.

1. Introduction

At all times, while learning the essence of any subject one also has to learn the system of symbols (notations) and special terms traditionally used in it. This causes the following disadvantages:

- Pupils and even some teachers confuse the content of a subject with its form, they do not recognize real life applications of its items and cannot apply their skills and knowledge.
- Many persons of good ability and poor initial knowledge are frightened of such a system and do not try to learn at all.
- If the state (official) language of teaching or symbols used are well known by many pupils, then those pupils have a great advantage over those unfamiliar.

Therefore, summarizing teachers’ attempts to make teaching more intelligible and visual, we suggested developing of *independent learning* (Pankov, 1996): ways of teaching and assessment which make minimal use of any media system not related to the content of the subject.

We also proposed to compose Olympiad tasks with minimal use of additional conditions and restrictions (Pankov, 2008), to increase the use of *common sense* by Socrates' method in learning mathematics (Pankov *et al.*, 2015).

In this paper we propose to use this method for expanding the age range for children involved in the Olympiad movement. The problems are designed in such a way that no mathematical knowledge is required to understand and to enter the answer surely. Certainly, each task is to be tested before including to the set of tasks of the Olympiad.

Also, this paper continues the question put in Pankov *et al.* (2023): What mathematical and other topics are present latently in *common sense*?

Sections 2 and 3 contain definitions and classification of objects of Drag-and-Drop technique: Target, Spot and Movable objects, requirements to parametrize tasks.

Sections 4, 5, 6 and 7 consider various types of tasks without words: on similarity; on relations; on equalization; on forecasting.

Remark. Some issues in this article cannot be covered by any general definitions or explanations. They can be demonstrated by examples only.

Remark. Certainly, there are a lot of publications and software touching on this topic and we cannot claim originality for each item. However, many of such publications contain general advice and recommendations only.

For example, a recent work (Hatisaru, 2020), which has, in turn, a vast list of 89 references including 7 ones by the author. But all tasks are in standard form:

Can you solve $7x + 4 = 5x + 8$?

On squared paper, draw as many different parallelograms as you can with an area of 12 square units.

[Here is a mistake: there are infinitely many such parallelograms: (0; 0)-(n; 1)-(n + 12; 1)-(12; 0)].

And a suggestion to the teacher: *Giving students cards depicting the same mathematical idea or concept (e.g., polyhedron) in different ways (e.g., verbal, visual, pictorial descriptions) and asking them to match the cards to enable them to draw links between the different representations of the same concept and to develop new mental images for it.*

Remark. There is a widespread genre: pictures and series of pictures “without words”. This paper may be considered as an “invitation to act” after observing such pictures.

2. Definitions and Classification

Although we try to avoid any conventions, nevertheless some are necessary and will be involved latently.

The Spot (to drag Movable objects to it) will be in the middle of the screen and will be denoted by any color (light green) which will not be used anywhere else.

A Movable object of any color has an outline of the same dark color.

Movable objects to be dragged will be in the lower part of the screen and Targets will be in the upper part.

First tasks will be evident and the user will unconsciously master these conventions.

Varieties for Drag-and-Drop technique:

”n-to-1”: there are several Movable objects and one Target near the Spot. The user is to choose and to move one of Movable objects to the Spot;

”1-to-m”: (without a separate Spot): there are one Movable object and several Targets. The user is to choose and to drag the Movable object to any place (neighbor) of Targets;

”n-to-m”: there are several Movable objects and several Targets. Such complex task is given if the user has passed previous similar tasks successfully. The user is to drag some Movable objects, the announcement (congratulation) appears after all implied Movable objects are dragged to correct Targets.

Movable objects may be

- Solid; only parallel shift is possible by the custom.
- Rotatable (for instance, a segment).
- Transformable (for instance, a chain).

(Rotatable and Transformable objects may be marked by a little circle of dark color at the edge; it is also a clue for the user).

We consider only solid Movable objects in the paper; the only example of Rotatable object is in Task 9.

Remark. Of course, Rotatable and Transformable objects and more difficult for programming and can generate a wide variety of tasks, the proposed method is not limited.

We propose tasks to guess which may be classified as, *searching analogs* (similarity); *using relations* (as often used in IQ tests); *equalization on scales* (visualization of solving linear equations for children); *forecasting* (the simplest example is “continue the sequence ...”). At the same time, it is known that many human ideas cannot be expressed in words (in instructions, descriptions). The simplest example is music.

We hope that some of these tasks proposed by us or invented by others in the framework of the proposed method would cover new ideas which cannot be classified in any verbal form.

3. Requirements

Tasks must be parametrized: when restarting the program, a slightly different task (with the same level of difficulty) is to be generated randomly.

This provides all contestants with different tasks and prevents mutual hints. Also, if the software is used for learning, restarting it yields different tasks that makes learning to be more interesting.

Tasks must be “culturally neutral”.

After successful completion of the task, a nice piece of music and an incentive badge (not a smiley because smileys have definite colors) must be played to the user.

If the software is intended for learning in different languages then after a successful completion of the task an announcement appears:

Congratulations, you have mastered the notion of ...
in the chosen language.

4. Introduction, Examples of Tasks on Similarity

Only the number of the task and drawings (figures, objects) are shown to the user (child). Themes of tasks, words “Target”, “Movable object” below are for programmers and teachers.

Certainly, tasks would be more interesting for children in a colorful, geometric form. Examples are given below in pseudo-graphic for brevity. The Spot is denoted as (?).

(Optional, for younger children; an adult may help)

Task 0. Introduction (1-to-1): The Spot, the Movable object. If the user does nothing then the Movable object moves to the Spot and returns.

Task 1. Colors (n-to-1). The Spot, the Target as a yellow square near (over) the Spot, Movable objects as equal circles of different colors, one of them is yellow.

(After restarting: a blue circle; equal triangles of different colors, one of them is blue, etc.)

This is a particular case of

General Task 2. Property-choose (n-to-1). The Spot, the Target as an image with any Property, Movable objects as similar images (but different from the Target) of different properties, one of them has the same Property.

Particular cases:

Task 3. Notion of natural number (n-to-1).

Example (4-to-1):

Target &&&&& (?)

Four Movable objects: ** **** ***** ***

Task 4. Length (n-to-1). The Spot, the Target as a segment near (over) the Spot, Movable objects as slightly broken lines of sufficiently different lengths, one of them is equal to the length of the Target.

(Congratulations! It is *length*.)

Task 5. Area (n-to-1). The Spot, the Target as a rectangle near (over) the Spot, Movable objects as triangles of sufficiently different areas, one of them is equal to the area of the Target.

(Congratulations! It is *area*.)

Task 6. Symmetry-choose (n-to-1). The Spot, the Target as a symmetrical figure near (over) the Spot, Movable objects as figures, one of them has the same symmetry.

Illustrative example 6-1 (5-to-1): Target: Φ ; Movable objects: N, \square , \square , Z, E

Illustrative example 6-2 (5-to-1): Target: Z ; Movable objects: \square , \square , \mathcal{S} , E, H

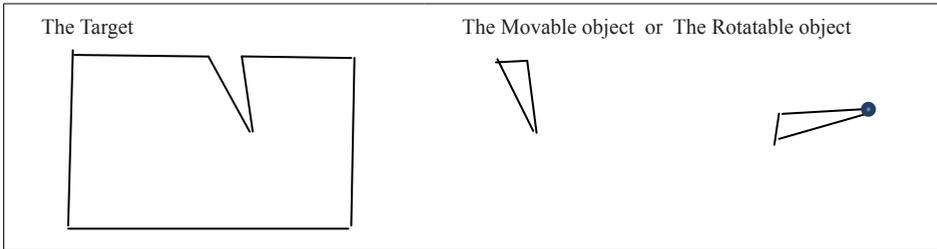
Sub-general Task 7. Linear transformations (n-to-1). The Spot, the Target as a geometrical figure, Movable objects as figures, one of them is a linear transformation of the Target. Illustrative example (4-to-1):

Target: A ; Movable objects: P, \forall , \exists , T

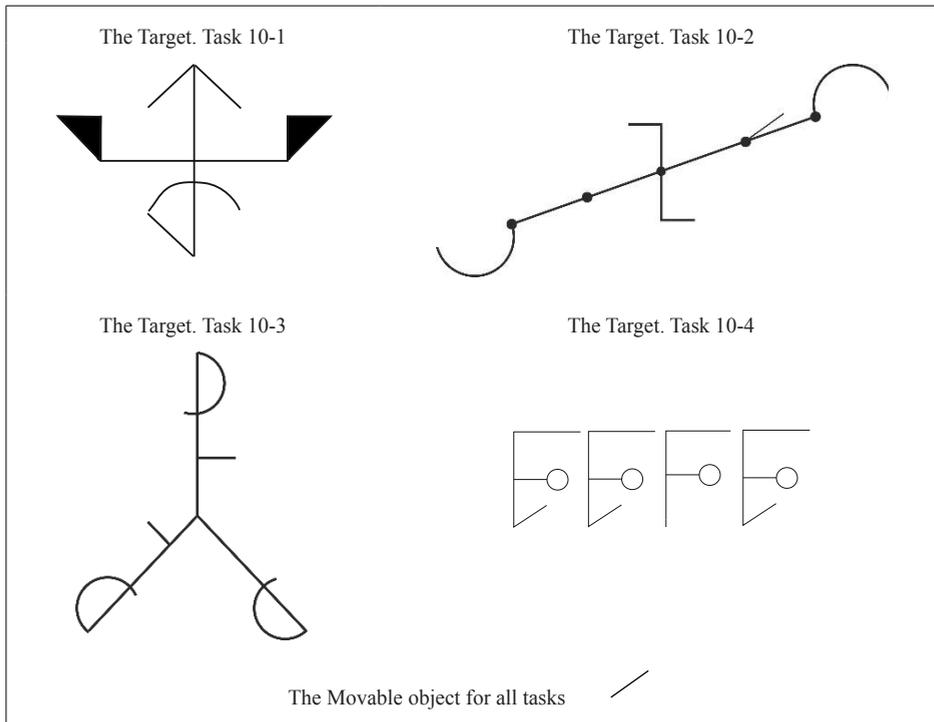
General Task 8. Property-complete (1-to-m). The Target is a geometrical image “almost with any Property”, the Movable object is a little piece which is to complete the Target.

Task 9. Figure-complete (1-to-m).

Example. The Spot is shown as (light green) neighbor around the Target.



Tasks 10. Symmetries (1-to-m). Four tasks in the following drawing.



5. Examples of Tasks on Relations

Set operations either on strings or on drawings:

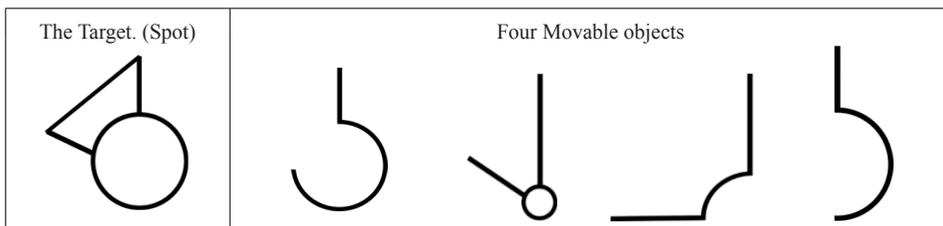
Task 11. Subset (n-to-1).

Example 11-1 (4-to-1):

Target UKDFGE (?)

Four Movable objects: WKD DFG FGZ KSFG

Example 11-2 (4-to-1):



Remark. The radius of the semicircle in the fourth Object is greater than one of the circle in the Target.

By our experience, children note it and choose the first Object surely.

Task 12. Intersection (n-to-1).

Example (5-to-1):

Target UKDFG (?) FDKZPW

Four Movable objects: WKF KUD DGF KFD TDW

Remark. We could not find a suitable way to involve “union of two sets”.

Task 13. New genre? (n-to-n).

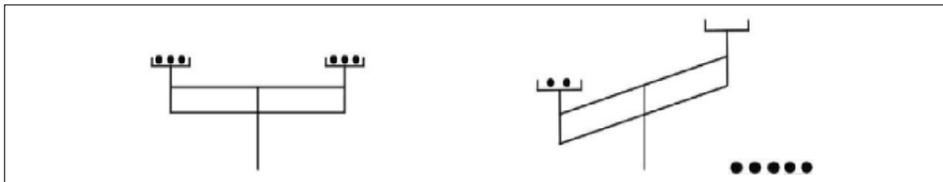
Example (3-to-3):

Targets $\cup\cup\cup\cup$ $\cup\cup\cup\cup\cup\cup$ $\cup\cup\cup\cup$
 (?) (?) (?)

Three Movable objects: **** ***** *****

6. Tasks with Scales (Equalization)

Illustration before the tasks



(the right cup of the right scales is light green)

The following tasks use one set of scales in tilt or two sets of scales: in balance and in tilt and natural numbers as sums of weights on scales. Firstly, the green (right) cup of the scales is up (is lighter in weight than the left one); it is denoted as (>).

Movable objects are (equal) weights.

When the scales become in balance, play affirming beep sounds.

These tasks are of type n(consequently)-to-1).

Remark. Certainly, “the unary system” used here is suitable only for small natural numbers. But it is our goal: we do not teach the child to count, to calculate, to use digits; we give tasks which can be solved without calculations. By our experience, some children solve such tasks successfully.

One set of scales:

Task 14. Introducing the scales.

Example: (*****) > (?)

Task 15. Addition or solving the equation $a = b + x$.

Example: (*****) > (****)(?)

Two sets of scales:

Task 16. Multiplication by 2 or 3.

Example: (Cat (or Apple...)) = (**); (Cat, Cat) > (?)

Task 17. Division by 2 or 3.

Example: (Cat, Cat) = (*****); (Cat) > (?)

Task 18. Linear equation.

Example 18-1: (Cat, **) = (*****); (Cat) > (?)

Example 18-2. (Cat,Cat,*) = (*****); (Cat) > (?)

7. Forecasting Tasks

The well-known

General Task 19. Continue sequence (n-to-1). The Target is three or four members of a sequence.

Remark. Due to the purpose of this paper, digits as conventional signs cannot be used.

Example (3-to-1). Target: *** ***** ***** (?)

Three Movable objects: ***** ***** *****

Fast forecasting with corresponding action is a main component of many computer games.

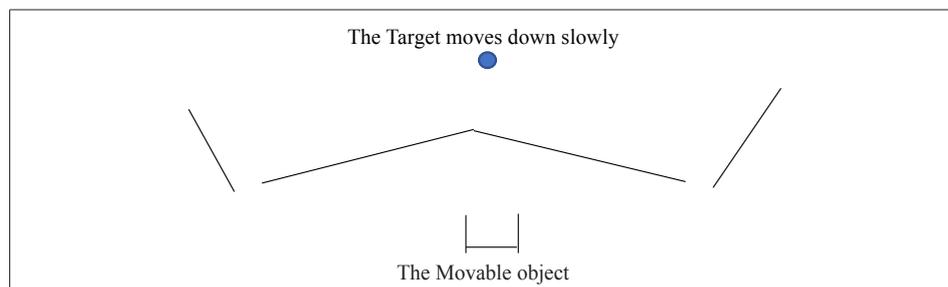
We (Bayachorova *et al.*, 2016) proposed to use slow forecasting for independent presentation of Future Tenses in a language.

We (Pankov *et al.*, 2023) proposed to involve “time” to Olympiad tasks (except the necessary time limit). We did not see such Olympiad tasks.

We propose

General Task 20. Forecasting-action (n-to-1) or (1-to-m). The Target moves slowly. The user is to forecast its further motion and catch it with the Movable object.

The simplest Example (1-to-2).



When the ball touches the roof, there will not be time to shift the box to one of the holes.

8. Conclusion

We hope that this paper will be a source of new tasks for programmers and involve children from an earlier age in the Olympiad movement, promote creation of a new type of software to learn mathematics independently for children in all languages. Also, we hope that these tasks would help to discover new capacities of artificial intelligence. As a consequence of this paper, we can rise the following philosophical problem: what notions or ideas can be expressed without words?

References

- Pankov P.S. (1996). Independent learning for Open society. *Collection of papers as results of seminars conducted within the frames of the program "High Education Support"*. Foundation «Soros-Kyrgyzstan», Bishkek, issue 3, 27–38.
- Pankov, P.S. (2008). Naturalness in Tasks for Olympiads in Informatics. *Olympiads in Informatics: Country Experiences and Developments*, 2, 16–23.
- Pankov, P., Janalieva, J., Naimanova, A. (2015). Inductive and experimental studying of mathematical subjects (mathematical facts and notions which can be discovered independently), *LAP Lambert Academic Publishing*, Saarbrücken.
- Pankov, P.S., Belyaev, A.A. (2023). Latent and evident knowledge to compose and to solve tasks in informatics. *Olympiads in Informatics*, 17, 87–97.
- Hatisaru, V. (2020) Exploring Evidence of Mathematical Tasks and Representations in the Drawings of Middle School Students. *International Electronic Journal of Mathematics Education*, Vol. 15, No. 3, 21 p.
- Bayachorova, B.J., Pankov, P.S. (2016). Mathematical models for independent computer presentation of complex expressions in natural languages. *Bulletin of Kyrgyz-Russian Slavic University*, series natural and technical sciences, 16(5), 19–21.



P.S. Pankov (1950), doctor of physics-mathematics sciences, prof., corr. member of Kyrgyzstani National Academy of Sciences (KR NAS), was the chairman of jury of Bishkek City OIs, 1985–2013, of Republican OIs, 1987–2012, participates in National OIs since 2020, was the leader of Kyrgyzstani teams at IOIs, 2002–2013, 2018–2023. Graduated from the Kyrgyz State University in 1969, is a head of laboratory of Institute of mathematics of KR NAS.



E.J. Bayalieva (1984), Senior Lecturer in the Software Engineering program, Institute of Computer Technologies and Artificial Intelligence, J. Balasagyn National University.

From Concept to Code: A Two-Day Workshop for Secondary Students on Computational Thinking and Programming

Felix STEINERT, Julia KUMMER,
Martina LANDMAN, Lukas LEHNER

*TU Wien, Institute of Information Systems Engineering
Favoritenstraße 9-11, 1040, Vienna, Austria*

*e-mail: felix.steinert@tuwien.ac.at, julia.kummer@tuwien.ac.at,
martina.landman@tuwien.ac.at, lukas.lehner@tuwien.ac.at*

Abstract. Introducing programming and informatics concepts to the next generation of computer scientists is essential. This experience report presents a detailed overview of a two-day informatics workshop for 134 Austrian school kids, ages eleven to thirteen. The workshop program consists of several unplugged activities about algorithms, AI, robotics and coding and block-based programming using Scratch and Sphero BOLT. We evaluated feedback from 110 participants regarding their experience of these two days. Many children reported a significant gain in knowledge. We will also explain the pupils' favourite activities and educational concepts in detail. This report also covers the experience of the ten workshop leaders of the past two days. The workshop was widely positive received and the participants reported a high interest in computer science.

Keywords: computer science education, school outreach, AI, Scratch, algorithms, workshops.

1. Introduction

Incorporating informatics education into secondary schools is essential for preparing students for success in today's technology-driven world.

There is a persistent demand for IT professionals, and it is crucial to broaden participation, particularly among underrepresented groups like girls. Efforts to engage students creatively are vital for fostering interest and participation in computer science, as reported by many institutions, for example Giannakos *et al.* (2013), describing a project with a two-day workshop using scratch and recycled materials or Rottenhofer *et al.* (2022), reporting on a computer science (CS) Workshop in the setting of Circus performance. As Sabitzer *et al.*, (2014) highlight, initiatives like CS Unplugged demonstrate success in engaging students' creativity and fostering interest in technology. They also underscore the value of computational thinking (CT) and early exposure to informatics concepts.

CT is a set of problem-solving skills everyone needs, not only computer scientists, but everyone in their everyday lives (Wing, 2006).

As part of the TU Wien Informatics Didactics group, the eduLAB team provides workshops (Prinzinger, 2021) aimed at enhancing CT and programming skills (Landman *et al.*, 2022; Unkovic & Landman, 2023) among students, starting from the 2nd grade onwards (Landman *et al.*, 2023). Our workshops are designed to prioritize hands-on, age-appropriate activities that empower students to develop and implement their problem-solving strategies, inspired by CS-unplugged activities (Bell *et al.*, 2015). Through engaging tasks, students explore fundamental concepts and methods of computer science, fostering a deeper understanding of computational thinking (CT). CT, understood as a set of problem-solving skills in CS, is crucial for learning informatics, especially including programming education (Grover & Pea, 2013).

Passing on these core skills to the future generation is one of our goals. Through suitable learning settings, we try to increase young people’s interest in programming and the broad field of computer science. As part of a secondary school’s “informatics days” in February 2024, the TU Wien Informatics eduLAB conducted a two-day workshop program. The workshops were held by CS university students in the bachelor’s, master’s, or PhD program. In this article we report on the feedback from the eleven-to thirteen-year-old pupils on this two-day workshop intervention.

2. Program Overview

The program spanned two days, with 134 secondary school pupils from the 6th grade divided into eight groups, each comprising 16 to 17 participants. Following an introductory session during which pupils completed an initial questionnaire, they proceeded to follow a predefined workshop schedule. Each pupil engaged in every session offered as part of the program. Further details regarding the workshop program are provided below.

2.1. Day 1: Concepts of Computer Science

The first day of the workshop was dedicated to unplugged concepts of computer science through a series of four 55-minute interactive workshop sessions. In each session the pupils worked in groups of 4–5 persons, engaging with materials that are designed to be exploratory, playful, and hands-on. With the “unplugged” concept we try to enable interactive learning processes for informatics exercises that do not require computers.

Codes

In this session the topic “Codes” was introduced utilizing everyday life analogies, like crosswalks and traffic lights to effectively convey the concept of symbols having associated meaning.

Pantomime: As introductory task, pupils played a charades-like game, allowing them to uncover characteristics of good codes, like uniqueness and universal understanding. During this task, the pupils guessing the words were not allowed to speak either but could use gestures to communicate their questions before writing down their guesses.

Variable-Length Codes: Following this, the pupils were introduced to the concept of variable-length codes through the exercise “juice bar”, where they were asked to find erroneous codes and determine the binary representation for various fruits based on given combinations.

Uniform-Length Codes: As bonus task for fast pupils, this task let them explore the binary coding system further by playing an adapted version of the memory game “I packed my bag”: instead of verbally adding items to a figurative bag or suitcase, they physically pin different combinations of 3D printed elements onto a board to encode and “store” their items.

Algorithms

Similar to the Codes session, the concept of algorithms was introduced using everyday analogies, such as the step-by-step process of preparing a frozen pizza or making breakfast. Conditionals were explained using the analogy of being vegetarian (“if you are vegetarian use cheese, else use ham”).

Sorting Cards: With the use of the cards of a game called “Ligretto”, the pupils could playfully develop their own sorting strategies for a shuffled stack of cards in a small group (Fig. 1). As described by Landman *et al.* (2023), this task not only represents a hands-on implementation of sorting algorithms but also provides a practical understanding of key concepts in computer science, such as “Divide and Conquer”, distributed computing and resource optimization.

Scheduling: During this task, pupils learned about scheduling algorithms. While planning activities for an entertainment park and a zoo, they discovered the necessity for algorithmic solutions to optimize scheduling processes. After having tried to find and describe their own scheduling algorithm, they were introduced to a “line-sweep” algorithm, which they applied hands-on to solve the scheduling challenges.



Fig. 1. Sorting activity using Ligretto cards.



Fig. 2. An Ozobot robot following a black line.

Artificial Intelligence

Decision Tree: The artificial intelligence (AI) session focused on decision trees with pupils asked to build a decision tree from 3D printed materials that could classify several types of fruit. Pupils were supplied with a data set represented by playing cards. Once they built a model that could correctly classify the given data, the pupils received additional test data that included some intentional stumbling blocks. The outcomes of their models' predictions were discussed in the end, including problems like missing classes or attributes and potential strategies for improving their models were elaborated.

Computational Thinking supported with Ozobots

Ozobot-Maze: The Ozobot-Maze session provided pupils with a hands-on experience with algorithms. The robot follows black lines (Fig. 2) and performs actions based on colour codes. The pupils first needed to decipher the meaning of these codes for the Ozobot. In subsequent steps, the pupils were asked to apply these commands using stickers to navigate the robot through different maps.

2.2. Day 2: Programming Basics

On the second day, the focus shifted to hands-on programming experiences as pupils were introduced to programming using Scratch and Sphero BOLT.

Programming basics with Scratch

Scratch: During this session, the pupils learned programming basics using Scratch, a beginner-friendly, block-based visual programming language. In the first part, the pupils were introduced to the development environment and worked together to complete three exercises. In the second part, they brainstormed and programmed their own unique games, with workshop leaders providing assistance as needed.

Programming basics with Sphero

Sphero: The Sphero BOLT is an educational robot sphere that can be programmed using a block- or text-based programming language. Therefore, it is suitable for beginners in programming. In this session the pupils could work through tasks at their own pace, beginning with very easy tasks (only using commands in a linear sequence) up to intermediate tasks (using nested loops). The robots provide motivation and offer space for creativity, e.g. by letting pupils program colourful pictures onto the integrated LED matrix.

3. Feedback Evaluation

3.1. Method

The pupils were provided one initial questionnaire in the beginning of the two-day workshop program and a second one concluding the workshops. Both questionnaires were conducted anonymously using Microsoft Forms.

The initial questionnaire focused on the pupils' interest in informatics, which they were asked to rate on a scale from 1 to 5.

The final questionnaire, with a total of 12 questions, was divided into three sections: Anonymous personal information, general workshop feedback and questions about informatics.

In the first section participants were asked to provide demographic details such as gender and age.

The section on workshop information allowed participants to offer general feedback on the two-day workshop and indicate whether they would recommend the workshop to other pupils. Additionally, they could select their favourite workshop session of each day and rate the difficulty of all workshop tasks on a Likert scale ranging from "very difficult" to "very easy".

The final section focused on the broader topic of informatics. Here, the pupils could again rate their interest in informatics and provide insights into the extent of new knowledge they gained. Using a Likert scale, participants were asked to express whether they could envision themselves working in a computer science-related field in the future. Various job titles like "Student in Informatics", "Programmer", "Computer Science teacher" and "IT technician" were provided for rating. Lastly, a free text form was included to provide the possibility for individual feedback.

3.2. Participants

The workshop participants were eighth-grade pupils from a secondary school, aged between 11 and 13 years old. All attendees were from the same school. A total number of

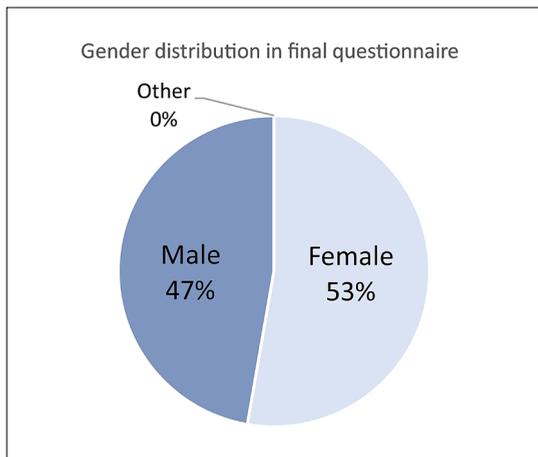


Fig. 3. Distribution of pupils' gender in final questionnaire.

134 pupils participated in the workshop, 125 of them answered the initial questionnaire and 116 the final feedback questionnaire.

From the final questionnaire's responses, six were excluded from the analysis due to suspicion of non-serious answers. Among the remaining 110 responses, the gender distribution was balanced, with 53% identifying as female and 47% as male (Fig. 3. Distribution of pupils' gender in final questionnaire).

4. Evaluation Results

Feedback

Overall, the workshop was well received with 43% of pupils giving the workshop 5 out of 5 stars and the average being 4,07.

By far the most popular task on day one was Ozobot-Maze with 53 pupils voting it as their favourite. In second place was Sorting Cards with 25 votes (Fig. 4). On day two Scratch and Sphero BOLT received roughly the same numbers of votes.

When asked about the difficulty of the tasks over 50% of participants reported "easy" or "very easy" on every exercise (Fig. 5).

Acquired Knowledge

33% of the 110 participants that finished the second questionnaire reported that they learned a lot about informatics.

Attitude towards computer science

We asked the pupils before and after the workshop about their interest in CS on a scale from 1 to 5 stars. Before the workshop 50% of pupils rated their interest a 4 or 5 after the workshop this increased to 63%. On the other side the percent of pupils who rated

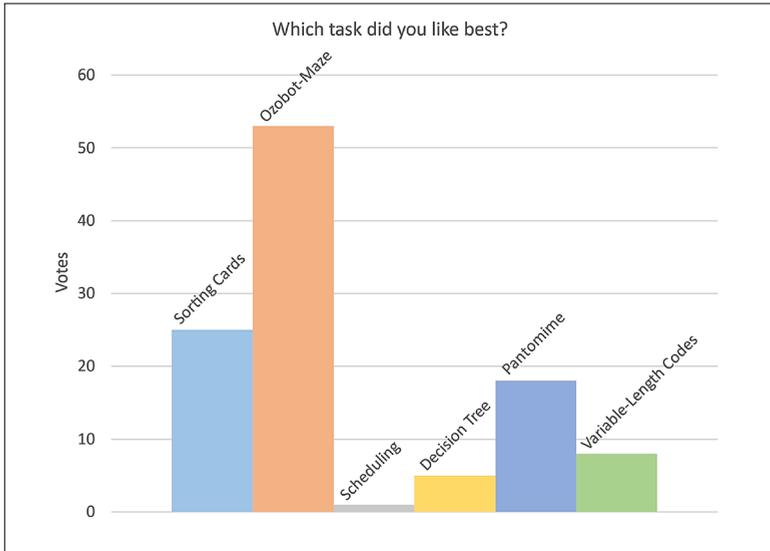


Fig. 4. Favourite workshop tasks as voted by the pupils.

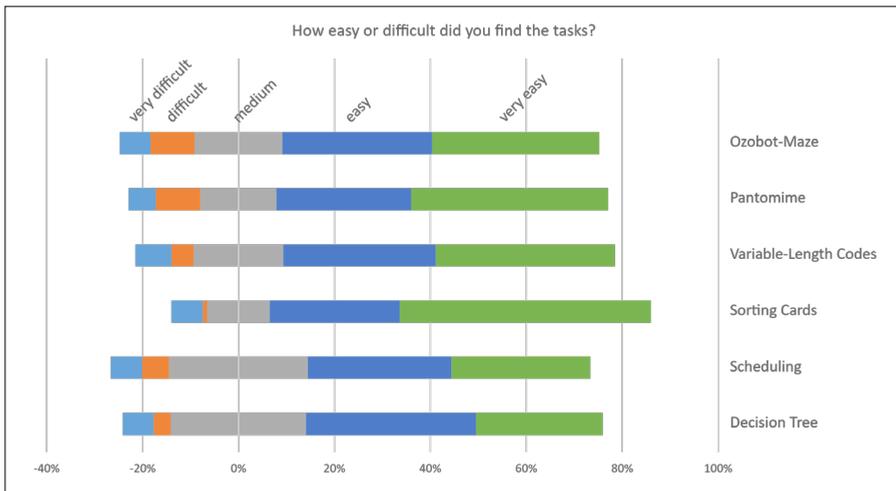


Fig. 5. Difficulty of the workshop tasks as voted by the pupils.

their interest in CS as a 1 or 2 decreases from 32% to 17% (Fig. 6). This shows we were able to increase the interest in CS in the pupils through our workshop directly after the two days.

When pupils were asked if they could imagine to work various IT jobs in the future only 15% answered with “fairly imaginable” and “very easy to imagine” (Fig. 7). A problem with this question could be, that they did not know what those jobs do and require.

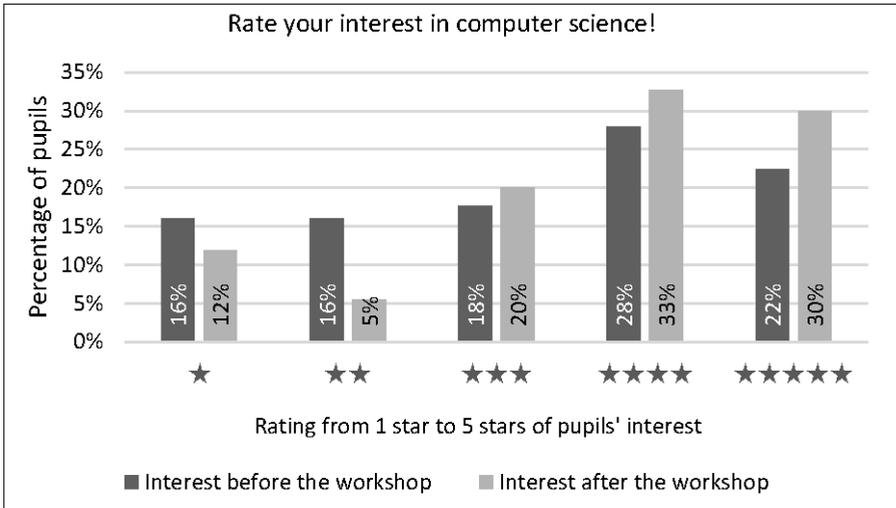


Fig. 6. Pupils' interest in computer science before and after the workshop.

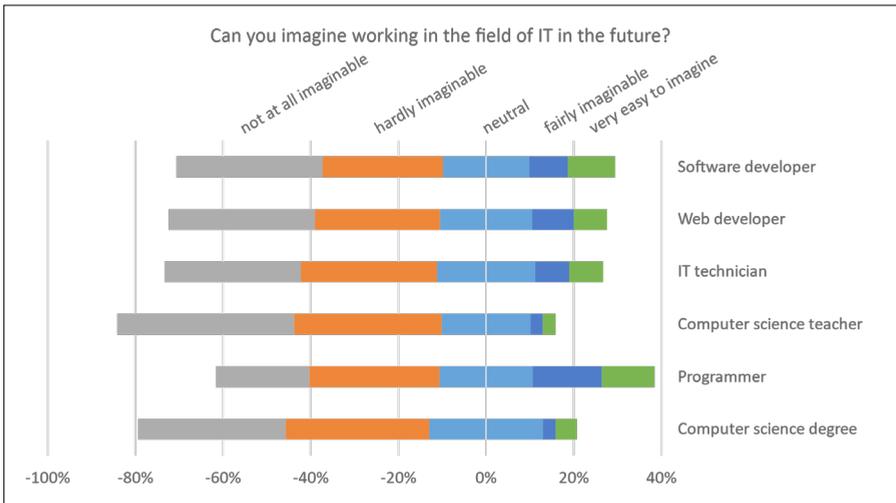


Fig. 7. Pupils' attitude towards their possible futures in IT.

5. Workshop Reflections

Concluding the workshop, we gathered informal feedback from the workshop leaders. The feedback was collected through oral discussions, where observations, insights and

opinions on the interaction with the pupils were described. This feedback was then compared with the results from the questionnaires completed by the participants.

Workshop leaders could observe that the children were more excited about the practical tasks on the second day, where they needed to use their own laptop, compared to the more theoretical tasks on the first day. However, as some workshop leaders noticed, a few pupils faced challenges to stay focused, especially given the many distractions on their laptops. Additionally, it was observed that the later it was the more the children struggled with concentration.

Another observation was that depending on the order in which the sessions were completed, pupils could reproduce concepts they learned in the previous session, potentially relevant for the current one. For example, those who had already completed the algorithm task, involving sorting cards, were able to reproduce algorithm principles in the Ozobot-maze session.

Furthermore, a few workshop leaders noticed that the experience and interest with Scratch varied among the pupils, which made it difficult to engage all pupils in the task. One workshop leader reported that some pupils immediately started to create their own games instead of following the course.

Based on the findings of the questionnaires shown in Fig. 4 many pupils reported that they perceived the AI task easy or very easy. However, a workshop leader shared that when asking the pupils to employ the decision tree model many of them used their human intelligence to solve the problem. This shows up a potential gap in understanding the AI concept.

In comparison to similar initiatives, one aspect that sets us apart is that our workshop offered a broad range of topics, spanning from concepts of CS, like algorithms, coding and AI, to hands-on programming activities. Our workshop provides pupils with a combination of practical programming exercises and “hands-on” learning of theoretical basics. This comprehensive approach within the two-day setup ensures that pupils have the opportunity to explore different areas within the field of CS and discover their strengths and interests.

Moreover, our extensive experience in conducting workshops, based on our regular program at the eduLAB, where over 2500 school children of all ages participate in annually, sets us apart. Based on this experience, our workshop is designed to be accessible to pupils of all skill levels. Whether they are complete beginners or have some prior experience with computer science, our workshop offers multiple difficulty levels in most tasks, which allowed them to proceed at their own pace or in a group setting.

Additionally, our workshop incorporates quantitative data analysis from questionnaires filled out by pupils. This approach enables us to gather valuable insights into the effectiveness of the workshop and adapt future exercises to better meet the needs of participants.

6. Conclusion

In this experience report we showed how we introduced computational thinking through unplugged activities and programming to pupils over a two-day workshop. On the first day we covered informatic concepts like algorithms, codes, and AI with unplugged exercises. The second day was focused on pupils programming themselves with Scratch and Sphero BOLT. We also collected feedback before and after the workshop from the pupils, using two questionnaires. The results showed that there is a high interest in CS but that despite of that not many students can imagine working in the field of IT in the future. Also, the students reported that they learned a lot of new information about CS. The questionnaire responses indicated that many pupils found most tasks to be easy or very easy. This suggests an opportunity to enhance the exercises to make them more challenging. Overall, the workshop was a success and showed that there is a lot of interest in CS.

As limitations to the results, we must consider that we could only collect data from a small age group (eleven- to thirteen-year-old) in one school. We aim to repeat this workshop to get better and more accurate feedback, as well as to increase the sample size. Additionally, the questionnaire we used was not validated. It is based on our curiosity in the pupils' perception of our work. In this experience report, we focused on the feedback of the students. Repeating the workshops but including pre-and post-tests with an existing validated questionnaire instead of our questionnaire, can be considered for future work. Nevertheless, this experience report provides meaningful insights into how to engage pupils into CS and CT. Looking further ahead, it would be desirable to deepen the insights gained and incorporate them into the various CS curricula.

In conclusion, supporting young people through workshop interventions, teaching key computer science concepts, is a big step forward to increase the number of young talents competing in programming challenges and competitions. Increasing the interest in CS is especially important to get young minds participating and contributing to the CS community with their skills and ideas.

References

- Bell, T., Witten, I., Fellows, M. (2015). *CS Unplugged: An Enrichment and Extension Programme for Primary-aged Students*. <https://www.csunplugged.org/>
- Giannakos, M., Jaccheri, L., Proto, R. (2013). Teaching Computer Science to Young Children through Creativity: Lessons Learned from the Case of Norway. In.
- Grover, S. & Pea, R. (2013). Computational Thinking in K–12. *Educational Researcher*, 42(1), 38–43. <https://doi.org/10.3102/0013189X12463051>
- Landman, M., Futschek, G., Unkovic, S. & Voboril, F. (2022). Initial Learning of Textual Programming at School: Evolution of Outreach Activities. *Olympiads in Informatics*, 43–53. <https://doi.org/10.15388/oi.2022.05>
- Landman, M., Rain, S., Kovács, L. & Futschek, G. (2023). Reshaping Unplugged Computer Science Workshops for Primary School Education. In: J.-P. Pellet & G. Parriaux (Ed.), *Lecture Notes in Computer Science. Informatics in Schools. Situation, Evolution, and Perspectives: 16th* (Vol. 14296, p. 139–151). Springer International PU. https://doi.org/10.1007/978-3-031-44900-0_11
- Prinzinger, P. (2021). Informatik für alle! Aktivitäten zu Computational Thinking, Programmieren und „Zaubertricks“. *OCG Journal*, 46(1–2), 13–15. <https://repositum.tuwien.at/handle/20.500.12708/137759> (Erstveröffentlichung 2021)
- Rottenhofer, M., Kuka, L. & Sabitzer, B. (2022). Clear the Ring for Computer Science: A Creative Introduction for Primary Schools. In (p. 103–112). Springer, Cham. https://doi.org/10.1007/978-3-031-15851-3_9
- Sabitzer, B., Antonitsch, P.K. & Pasterk, S. (2014). Informatics concepts for primary education. In: C. Schulte (Ed.), *ACM Digital Library, Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (p. 108–111). ACM. <https://doi.org/10.1145/2670757.2670778>
- Unkovic, S. & Landman, M. (2023). Supporting Non-CS Teachers with Programming Lessons. In: J.-P. Pellet & G. Parriaux (Ed.), *16th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2023, Local Proceedings* (p. 61–74). Zenodo.
- Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35. <https://doi.org/10.1145/1118178.1118215>



F. Steinert studies Software & Information Engineering at TU Wien in the bachelor program. Since the winter semester of 2023 he works at eduLAB as a student assistant conducting workshops for school classes and helping with research.



J. Kummer is a student of the master program Media- and Human Centered Computing at TU Wien. Alongside her studies she works as student assistant at the TU Wien Informatics eduLAB, conducting workshops for school classes and helping with research.



M. Landman Researcher at TU Wien and member of the Informatics eduLAB group in the research unit of Information & Software Engineering since 2021. She has experience in teaching computer science from 5th to 12th grade. She organizes the computer science faculty's school outreach activities, where she develops, organizes, and conducts weekly workshops for school classes. Her research focuses on algorithmic problem solving for kids.



L. Lehner is a researcher and PhD candidate in informatics didactics at TU Wien. His research focuses on how to best promote young people's AI literacy. He develops unplugged learning materials that let learners discover the technical functioning of various machine learning methods without the use of computers.

Algorithmic Problem-Solving Advancements: A Comprehensive Exploration across Diverse Domains

Aadesh TANEJA¹, Anurima KOTHARI²

¹*Saint MSG Glorious International School, Sirsa, India*

²*DSB International Public School, Rishikesh, India*

e-mail: aadesh47taneja@gmail.com, kotharianurima@gmail.com

Abstract. Recent developments in algorithmic problem-solving techniques have significantly influenced diverse domains, from mathematical computations to real-world problem-solving. This paper explores the advancements in algorithm development, emphasizing the application of mathematical reasoning and rigorous design in functions and recursive functions. Additionally, the review spans the landscape of solving mathematical word problems (MWP), analysing methodologies, and providing insights into the challenges and complexities inherent in natural language processing, machine learning, and artificial intelligence. In a comparative study, computational and algorithmic advances for solving Richards' equation are evaluated, revealing their joint contributions to a substantial improvement in efficiency. The collective insights from these perspectives underscore the transformative impact of algorithmic advancements across interdisciplinary domains.

Keywords: algorithmic advancements, mathematical word problems (MWP), natural language processing (NLP), machine learning, artificial intelligence (AI), Richards' equation, hydrology, soil science, finite difference method (FDM), finite element method (FEM), numerical optimization techniques, Surrogate models, genetic algorithms

1. Introduction

In recent years, algorithmic problem-solving has witnessed remarkable advancements, revolutionizing diverse domains ranging from mathematical computations to real-world problem-solving scenarios. These advancements have been fuelled by breakthroughs in mathematical reasoning, rigorous algorithm design, and the ever-evolving landscape of computational methodologies. The importance of algorithm development across various domains cannot be overstated, as it underpins the efficiency, accuracy, and scalability of solutions to complex problems encountered in fields as varied as finance, healthcare, engineering, and beyond.

This paper aims to provide a comprehensive exploration of the recent developments in algorithmic problem-solving techniques, shedding light on their significance and impact across interdisciplinary domains.

The structure of this paper is organized to offer a systematic analysis of algorithmic problem-solving advancements. Firstly, we delve into the fundamental principles of mathematical reasoning and rigorous design that serve as the bedrock of effective algorithm development. Subsequently, we look at methodologies employed in solving mathematical word problems (MWPs), discovering the complexities inherent in natural language processing (NLP), machine learning, and artificial intelligence (AI) as they intersect with algorithmic approaches.

Furthermore, this paper conducts a comparative study evaluating computational and algorithmic advances in tackling Richards' equation, a pivotal problem with wide-ranging applications in fields such as hydrology and soil science. Through this comparative analysis, we aim to underscore the collective contributions of algorithmic innovations towards enhancing computational efficiency and solution accuracy.

```
quicksort(array, low, high):
    if low < high:
        pivot_index = partition(array, low, high)
        quicksort(array, low, pivot_index - 1)
        quicksort(array, pivot_index + 1, high)

partition(array, low, high):
    pivot = array[high]
    i = low - 1
    for j = low to high - 1:
        if array[j] <= pivot:
            i = i + 1
            swap(array, i, j)
    swap(array, i + 1, high)
    return i + 1

swap(array, i, j):
    temp = array[i]
    array[i] = array[j]
    array[j] = temp
```

This pseudocode snippet illustrates the Quicksort algorithm, where the '**quicksort**' function recursively sorts subarrays by partitioning elements around a pivot, and the '**partition**' function partitions the array into two halves. The '**swap**' function is used to swap elements within the array.

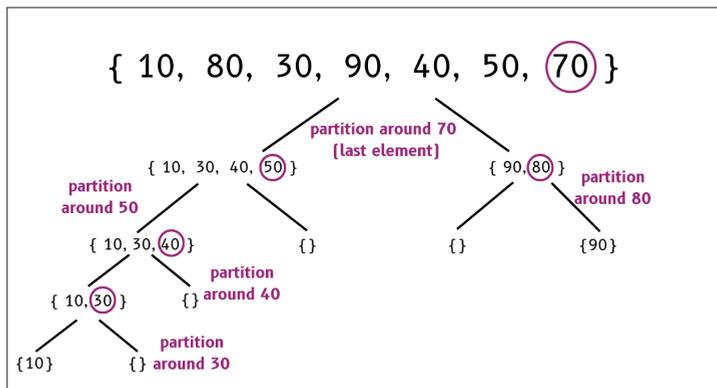


Fig. 1. QuickSort Algorithm.

Source: <https://www.geeksforgeeks.org/quick-sort/>

2. Mathematical Reasoning and Rigorous Design in Algorithm Development

Algorithm development relies heavily on mathematical reasoning, which involves the systematic application of logical principles to solve problems. Mathematical reasoning plays a crucial role in guiding the design and analysis of algorithms, ensuring their efficiency, correctness, and scalability. By leveraging mathematical concepts such as logic, probability theory, graph theory, and combinatorics, algorithm designers can formulate precise solutions to complex problems.

The importance of rigorous design principles cannot be overstated in algorithm development. Rigorous design principles encompass techniques for ensuring the correctness and efficiency of algorithms. This includes strategies such as divide and conquer, dynamic programming, greedy algorithms, and backtracking, which provide structured approaches to problem-solving while adhering to mathematical rigor.

Examples illustrating the application of mathematical reasoning in algorithm development abound across various domains. One such example is the use of algorithms in cryptography, where mathematical principles such as number theory and algebra are employed to design secure encryption and decryption schemes. Another example is the application of algorithms in optimization problems, where mathematical optimization techniques are utilized to find the most efficient solution among a set of feasible options.

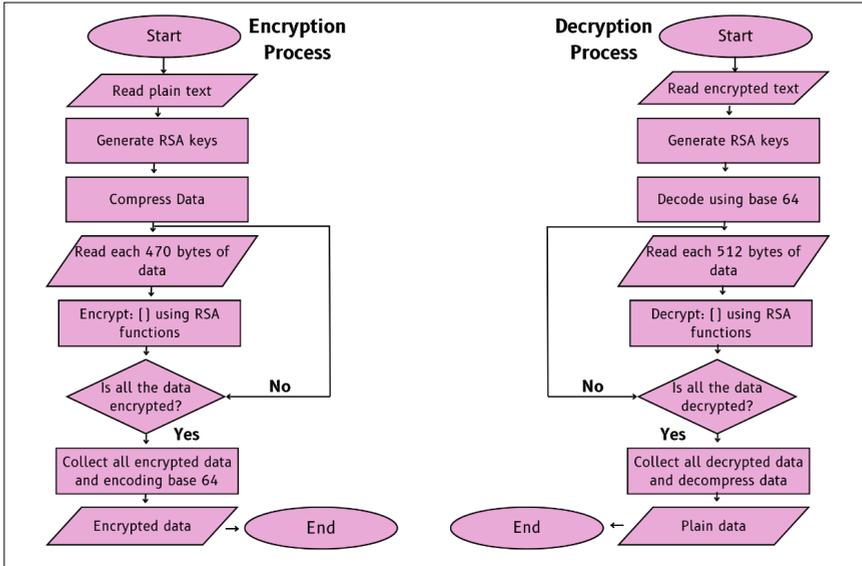


Fig. 2. RSA's encryption and decryption process.

Source: https://www.researchgate.net/figure/Flowchart-of-RSA-encryption-and-decryption-operations_fig1_353809093

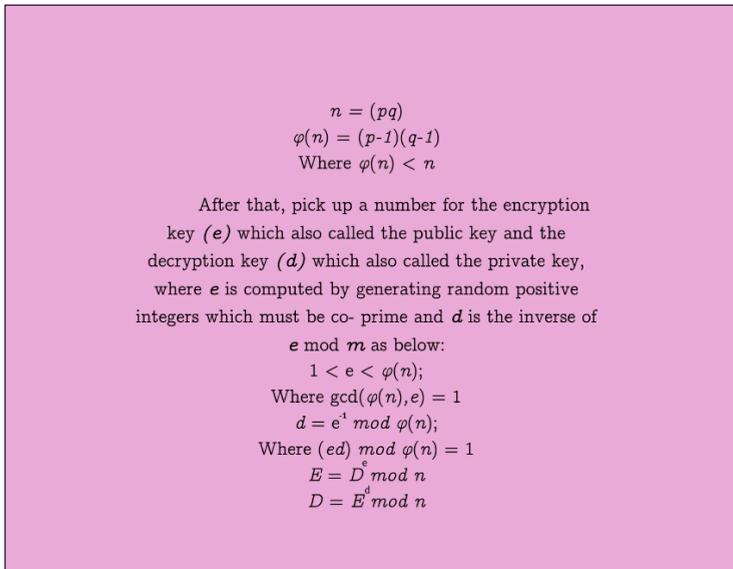


Fig. 3. Flowchart of Genetic Algorithm, a computational technique inspired by natural selection that involves processes such as selection, crossover, and mutation to optimize solutions.

Source: https://www.researchgate.net/figure/Complete-steps-of-RSA-algorithm-22-Mathematical-Proof-of-RSA-Algorithm-RSA-computations_fig1_318978830

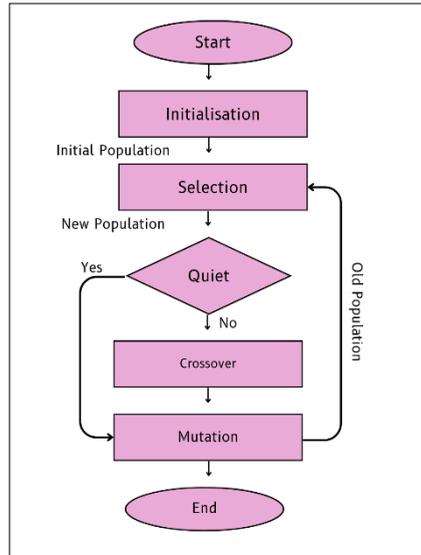


Fig. 4. Mathematical Proof of RSA Algorithm RSA computations can be mathematically proved by forward substitution of the encryption process of plaintext message M to get the ciphered message C and then by backward substitution of Ciphertext C to get back the plaintext message M. Source: https://www.researchgate.net/figure/Complete-steps-of-RSA-algorithm-22-Mathematical-Proof-of-RSA-Algorithm-RSA-computations_fig1_318978830

Sorting Algorithms	Time Complexity			Space Complexity
	Best Case	Average Case	Worst Case	Worst Case
Bubble Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$	$O(1)$
Selection Sort	$\Omega(N^2)$	$\Theta(N^2)$	$O(N^2)$	$O(1)$
Insertion Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$	$O(1)$
Quick Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N^2)$	$O(N)$
Merge Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$	$O(N)$
Heap Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$	$O(1)$

Fig. 5. Comparison of performance metrics (e.g., time complexity, space complexity, accuracy) of different algorithmic approaches in solving specific problems.

Source: <https://afteracademy.com/blog/comparison-of-sorting-algorithms/>

3. Solving Mathematical Word Problems (MWPs)

Mathematical Word Problems (MWPs) pose unique challenges due to their requirement for interpreting natural language and translating it into mathematical expressions or equations. Despite their ubiquity in educational settings and real-world applications, MWPs remain notoriously difficult for many individuals to solve. Understanding and solving MWPs are crucial skills that are applicable across various domains, from education to engineering and beyond.

Overview of MWPs and their significance: MWPs typically involve extracting mathematical information from natural language texts and formulating equations or expressions to represent the problem. These problems often require critical thinking and problem-solving skills, as well as a deep understanding of mathematical concepts and their real-world applications. Solving MWPs is essential for developing mathematical proficiency and problem-solving abilities, making them a fundamental aspect of mathematics education.

Methodologies for solving MWPs: Several methodologies are employed for solving MWPs, ranging from heuristic approaches to formal algorithmic techniques. Heuristic methods involve using problem-solving strategies, such as identifying key words or phrases, drawing diagrams, or breaking down complex problems into simpler components. Algorithmic techniques, on the other hand, leverage formal mathematical and computational approaches to analyse and solve MWPs. These techniques may involve symbolic manipulation, equation solving, or mathematical modelling to represent and solve the problem systematically.

Challenges in natural language processing (NLP) for MWP solving: One of the primary challenges in solving MWPs is the ambiguity and complexity inherent in natural language. NLP techniques are often employed to parse and understand the meaning of text, extracting relevant information and identifying mathematical relationships. However, NLP systems may struggle with linguistic ambiguities, figurative language, or domain-specific terminology present in MWPs, leading to errors or inaccuracies in interpretation.

Insights into machine learning and artificial intelligence approaches for MWP solving: Machine learning and artificial intelligence (AI) techniques offer promising avenues for improving MWP solving capabilities. These approaches involve training models on large datasets of MWPs and their corresponding solutions, allowing algorithms to learn patterns and relationships between natural language text and mathematical representations. AI systems can then be used to automatically generate solutions to MWPs or assist human users in solving them more efficiently. However, challenges remain in developing AI systems that can generalize effectively across diverse problem domains and accurately interpret complex natural language inputs.

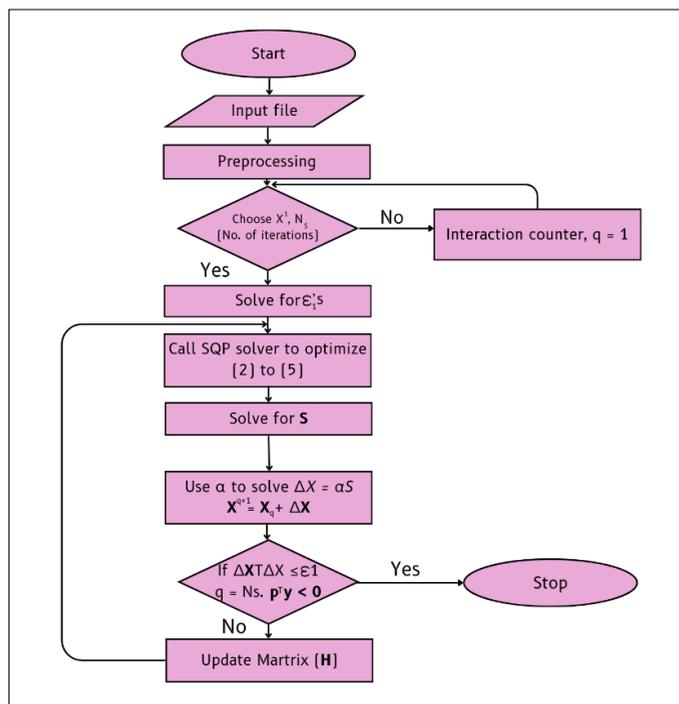


Fig. 6. Flowchart of NLP Algorithm for SQP.

Source: https://www.researchgate.net/figure/Flowchart-of-NLP-Algorithm-for-SQP_fig1_222229919

4. Computational and Algorithmic Advances for Solving Richards' Equation

Introduction to Richards' Equation and its Significance: Richards' equation is a partial differential equation that describes the movement of water in unsaturated soils. It is widely used in hydrology, soil science, and agriculture to model processes such as infiltration, drainage, and groundwater recharge. The equation takes into account factors such as soil properties, boundary conditions, and external forcings to simulate the flow of water through the soil profile. Solving Richards' equation accurately is crucial for understanding and predicting water movement in natural and engineered systems, making it a fundamental tool in various scientific and engineering applications.

4.1. Comparative Study of Computational and Algorithmic Approaches for Solving Richards' Equation:

Richards' equation, a non-linear partial differential equation, describes the movement of water through unsaturated soils. It governs the water content as a function of space and

time in the soil profile. The equation is mathematically expressed as:

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial z} \left(\mathbf{K}(\theta) \left(\frac{\partial h}{\partial z} + 1 \right) \right) - S$$

Where:

\vec{q} is the volumetric flux;

θ is the volumetric water content;

h is the liquid pressure head, which is negative for unsaturated porous media;

$\mathbf{K}(h)$ is the unsaturated hydraulic conductivity;

∇z is the geodetic head gradient, which is assumed as $\nabla z = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ for three-dimensional problems.

S is the sink term [T^{-1}], typically root water uptake

4.2. Methodologies for Solving Richards' Equation:

Numerous computational and algorithmic approaches have been developed for solving Richards' equation to simulate water movement in unsaturated soils. These approaches range from traditional numerical methods to more recent machine learning-based techniques. Common methods include:

1. **Finite Difference Method (FDM):** This approach discretizes the spatial and temporal domains and approximates the derivatives in the equation using finite differences. It is widely used due to its simplicity and effectiveness in capturing soil water dynamics.
2. **Finite Element Method (FEM):** FEM divides the soil domain into finite elements and formulates a system of algebraic equations based on variational principles. It offers flexibility in handling complex geometries and material properties but may require more computational resources.
3. **Numerical Optimization Techniques:** Optimization methods such as genetic algorithms or gradient-based optimization are used to estimate model parameters and calibrate Richards' equation to observational data. These techniques help improve the accuracy of simulations by adjusting model parameters to match field measurements.
4. **Machine Learning-Based Surrogate Models:** Recent advancements in machine learning have introduced surrogate models trained on observational data to approximate the solution of Richards' equation. Neural networks, support vec-

tor machines, and Gaussian processes are examples of machine learning models used to emulate the behaviour of complex physical systems.

```

import numpy as np
import matplotlib.pyplot as plt

# Parameters
L = 1.0 # Length of soil profile (m)
T = 100.0 # Total simulation time (s)
N = 100 # Number of spatial grid points
M = 1000 # Number of time steps
Ks = 1e-4 # Saturated hydraulic conductivity (m/s)
n = 2.0 # Porosity
alpha = 0.01 # Brooks-Corey parameter
theta_i = 0.1 # Initial water content
theta_s = 0.4 # Saturated water content

# Spatial and temporal discretization
dx = L / N
dt = T / M

# Initialize water content array
theta = np.zeros((N+1, M+1))
theta[:, 0] = theta_i

# Finite difference method
for k in range(M):
    for i in range(1, N):
        theta[i, k+1] = theta[i, k] + (Ks * dt / n) * ((theta[i+1, k] - theta[i, k]) / dx)**alpha

# Plot results
x = np.linspace(0, L, N+1)
t = np.linspace(0, T, M+1)
X, T = np.meshgrid(x, t)
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, T, theta.T, cmap='viridis')
ax.set_xlabel('Distance (m)')
ax.set_ylabel('Time (s)')
ax.set_zlabel('Water Content')
ax.set_title('Numerical Solution of Richards\' Equation')
plt.show()

```

This code numerically solves Richards' equation using the finite difference method (FDM) and then visualizes the water content distribution over time in a one-dimensional soil profile.

5. Impact of Algorithmic Advancements Across Interdisciplinary Domains

Summarization of Insights from Previous Sections: Throughout this paper, we have explored various facets of algorithmic advancements, ranging from mathematical reasoning and rigorous design principles to computational methodologies and algorithmic approaches for solving complex problems. We have discussed the significance of algorithm development across diverse domains, emphasizing its transformative potential in addressing multifaceted challenges.

Discussion on the Transformative Impact of Algorithmic Advancements: Algorithmic advancements have profoundly influenced interdisciplinary domains, revolutionizing the way problems are solved and insights are gained across fields such as mathematics, engineering, environmental science, and beyond. By leveraging mathematical reasoning and rigorous design principles, algorithms have enabled the efficient and accurate solution of complex problems that were once considered intractable. Moreover, the integration of machine learning and artificial intelligence techniques has further expanded the capabilities of algorithms, allowing for the automation of tasks, the discovery of patterns in data, and the optimization of processes.

Case Studies Highlighting Interdisciplinary Applications of Algorithmic Advancements: Several case studies exemplify the interdisciplinary applications of algorithmic advancements. For instance, in hydrology and soil science, computational and algorithmic approaches have been instrumental in simulating water movement in unsaturated soils, as demonstrated by the numerical solution of Richards' equation. In finance, algorithmic trading strategies leverage mathematical models and computational algorithms to make data-driven investment decisions in real-time. Similarly, in healthcare, machine learning algorithms analyse medical imaging data to assist in disease diagnosis and treatment planning. These examples underscore the diverse range of applications where algorithmic advancements have made significant contributions.

Future Directions and Potential Areas for Further Research: Looking ahead, there are several promising avenues for further research in algorithmic advancements. Future studies could focus on enhancing the efficiency and scalability of algorithms, improving their robustness to uncertainties and variability in real-world data, and exploring novel algorithmic approaches for addressing emerging challenges. Additionally, interdisciplinary collaboration and the integration of diverse perspectives from mathematics, computer science, and domain-specific fields will be crucial for driving innovation and unlocking the full potential of algorithmic advancements in addressing complex societal and scientific problems.

6. Conclusion

In conclusion, this paper has provided a comprehensive exploration of algorithmic advancements and their transformative impact across diverse interdisciplinary domains. Through our investigation, several key findings have emerged, highlighting the significance of algorithm development in addressing complex problems and driving innovation in various fields.

We began by delving into the fundamental principles of mathematical reasoning and rigorous design, emphasizing their critical role in the development of effective algorithms. By leveraging mathematical concepts and computational methodologies, algorithms have facilitated the solution of mathematical word problems, optimization tasks, and computational challenges such as the Richards' equation.

As we look to the future, the implications of algorithmic advancements for research and applications are profound. Continued innovation in algorithm development holds the potential to revolutionize industries, streamline processes, and address pressing societal challenges. Interdisciplinary collaboration and the integration of diverse perspectives will be essential for the full potential of algorithmic advancements and driving progress in the years to come.

References

- Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Brooks, R.H., Corey, A.T. (1964). Hydraulic properties of porous media. *Hydrology Papers*, 3.
- Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.
- Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press.
- Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.
- Loh, W.Y., Shih, Y.S. (1997). Split selection methods for classification trees. *Statistica Sinica*, 7(4), 815–840.
- Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P. (2007). *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). Cambridge University Press.
- Rasmussen, C.E., Williams, C.K.I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.
- Sutton, R.S., Barto, A.G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.
- Wolfram, S. (1991). *Mathematica: A System for Doing Mathematics by Computer*. Addison-Wesley.



A. Taneja – a high school senior recognized as the India Topper in IGCSE HSL. Aadesh is the founder of edumadeasy since April 2023, where they have helped over 15,000 students with IGCSE preparation. He has also been involved in research in the fields of Mathematics, AI, and Computer Science, at institutions such as the Mathematical Association of America (MAA), the New York Academy of Sciences (NYAS), and the Stanford Asian Liver Center.



A. Kothari – a dedicated enthusiast in the field of technology and an expert in coding across various languages. Anurima is a high school junior participating in diverse extracurricular pursuits. She has co-founded edumadeasy; a company which has helped over 15,000 students with their IGCSE examinations by providing them resources. Moreover, she's actively participating in various research projects in the field of AI, ML and computer science.

Staying DRY with OO and FP

Tom VERHOEFF

*Mathematics and Computer Science, Eindhoven University of Technology
Groene Loper 5, 5612 AE, Eindhoven, Netherlands
e-mail: t.verhoeff@tue.nl*

Abstract. We discuss the coding principle Don't Repeat Yourself (DRY) and compare various tactics to achieve DRY code, both in the context of object-oriented (OO) programming and functional programming (FP). Neither OO nor FP play a significant role in the International Olympiad in Informatics (IOI), but these paradigms are highly relevant in industry. This article aims to make clear that they offer useful computational insights that should appeal to talented students and that could somehow make their appearance in the IOI. We conclude with an AHA insight and some WET advice for talent development and programming contests.

Keywords:

1. A DRY Introduction

Like any product, software goes through a life cycle. In the case of programming contests, the life of a program can be pretty short: a couple of hours. And during that life, depending on the format of the contest, one person or at most only a few people have seen the code, while it was conceived. Nowadays, evaluation in contests is done without anyone actually seeing that code. If, however, you earn a living by writing source code, then the life cycle extends quite a bit further. In fact, initial development then only accounts for a small fraction of the life cycle. The rest concerns operation and maintenance. Especially maintenance brings other qualities to the game, other than functional correctness, speed, and memory usage.

When programming under time pressure (as in a contest), it is tempting to copy-paste-edit code, also known as code *cloning*. For one thing, in a contest, code size often hardly matters, and finding a more clever solution can cost valuable time. When earning a living with software, cloning code might be tempting if you are paid by number of lines of code delivered. In general, however, copy-pasting hurts in the longer run. Imagine that you have ten clones of the same piece of code in your software. If you find a defect, or a way to improve performance, then you will want to update all copies (first you need

to find them all). If the copies were edited after pasting, then updating can become a challenge. You don't want that under time pressure either.

Hunt and Thomas (2019) introduced the software engineering principle *Don't Repeat Yourself* (DRY) in 1999 to create awareness of the problems of code cloning. Note that code clones are a purely syntactic phenomenon, that is, they concern a static property of code. This is also why it is not too hard to build clone detection into IDEs. DRY should not be confused with the Single Responsibility Principle (SRP), which concerns a semantic property. Solving the same problem twice is also a different concern, as can be seen from these two exercises:

1. Give two functions, solving the same problem, but with minimal code duplication.
2. Give two functions, solving different problems, but with considerable code duplication.

Here is a solution to the first exercise, in Python (we use version 3.12; all source code is available (Verhoeff, 2024); Java and C++ would look similar):

```

1  def fac_iter(n: int) -> int:
2      """Compute factorial of n, iteratively.
3      """
4      result = 1
5
6      for i in range(1, n + 1): # see footnote 1
7          result = result * i
8
9      return result
10
11 def fac_rec(n: int) -> int:
12     """Compute factorial of n, recursively.
13     """
14     if n == 0:
15         return 1
16     else:
17         return n * fac_rec(n - 1)

```

Apart from the header and the docstring, the code for `fac_iter` and `fac_rec` couldn't be more different, even though they compute the same thing. For the second exercise, compare `fac_iter` and `sum_array` defined by

```

18 def sum_array(array: list[int], n: int) -> int:
19     """Sum first n items in array, iteratively.
20     """
21     result = 0
22
23     for i in range(0, n):
24         result = result + array[i]
25
26     return result

```

¹ `i in range(a, b)` implies $a \leq i < b$; that is, upper bound `b` is excluded.

Now we have two functions that compute two very different things, but there is significant overlap in code: they are near code clones.

2. DRY via Object-Oriented Design Patterns

What can one do to avoid code clones, such as we have seen above? Typically this is done by defining a function that has the copied code as body, and replacing each copy by a call to that function. In case of near clones, one can introduce parameters, so that the body becomes more general, and the calls can be specialized by providing appropriate arguments to the calls.

For instance, the two statements on lines 4 and 21 are near clones, and values 0 and 1 can be generalized to integer parameter `initial`. The statement in the body then would become `result = initial`. The same holds for lines 6 and 23, which could be generalized to `for i range(start, start + n)`.

But for the statements on lines 7 and 24 this won't work so easily, because these involve two different ways of *updating* `result`. In the world of object-oriented (OO) programming, the language mechanisms involving classes, inheritance, overriding, and polymorphism can be used effectively to address this situation. Because these OO language mechanisms are easy to abuse, various standard ways have evolved to use them safely, known as *Design Patterns* (Gamma *et al.*, 1994).

2.1. Template Method Design Pattern

For our example, the *Template Method* pattern can be used, because it captures a common algorithmic structure with some variation points. The common algorithm is programmed in a so-called *template method* in an *abstract base class*. That template method calls so-called *hook methods* at the variation points. These hook methods are *abstract*, that is, they are not implemented in the base class. Each clone is now replaced by a call to the template method in a *subclass*, *inheriting from the base class*, while *overriding* the abstract hook methods with the code that is specific for each clone.

Let's apply this to our example. Here is the abstract base class that captures the common part of the iteration algorithm:

```

27 from abc import ABC, abstractmethod
28 from typing import override # requires Python 3.12
29
30 class Iteration(ABC):
31     """Abstract Base Class for iteration
32     using the Template Method design pattern.
33     """
34     @abstractmethod
35     def initialize(self) -> int:
36         """Abstract hook method to initialize result."""

```

```

37
38     @abstractmethod
39     def update(self, i: int, result: int) -> int:
40         """Abstract hook method to update result."""
41
42     def iterate(self, start: int, n: int) -> int:
43         """The template method with the common algorithm.
44         """
45         result = self.initialize()
46
47         for i in range(start, start + n):
48             result = self.update(i, result)
49
50         return result

```

Note that we used a hook method to initialize `result`, but avoided a hook method to provide the start value for `i`, and instead passed it to the template method directly as argument. That way, both options are illustrated.

The base class can be specialized as follows for computing factorials.

```

51 class Factorial(Iteration):
52     """Specialize the iteration algorithm for factorials.
53     """
54     @override
55     def initialize(self) -> int:
56         """Override abstract initialization hook method.
57         """
58         return 1
59
60     @override
61     def update(self, i: int, result: int) -> int:
62         """Override abstract update hook method.
63         """
64         result = result * i
65         return result

```

And the subclass `Factorial` is then used as:

```

66 def fac_iter_TM(n: int) -> int:
67     return Factorial().iterate(1, n)

```

For summing an array, the following specialization can be used, where the array is provided via the constructor:

```

68 class ArraySummer(Iteration):
69     """Specialize the iteration algorithm for summing an array.
70     """
71     def __init__(self, array: list[int]) -> None:
72         self.array = array
73
74     @override
75     def initialize(self) -> int:

```

```

76         return 0
77
78     @override
79     def update(self, i: int, result: int) -> int:
80         return result + self.array[i]
81
82     def sum_array_TM(array: list[int], n: int) -> int:
83         return ArraySummer(array).iterate(0, n)

```

There are several reasons to be less happy with this way of avoiding copied code. For one thing, the code is now considerably longer. In fact, you could argue that classes `Factorial` (lines 51–65) and `ArraySummer` (lines 68–80) are near code clones and in practice probably would have been programmed by copy-paste-edit. The code is also harder to understand, because the control flow is rather contorted due to inheritance and overriding. Consider the call `fac_iter_TM(5)`. The following happens during its execution:

1. `Factorial()` instantiates an object of type `Factorial`.
2. Its method `iterate(1, 5)` is called.
3. The definition of this method isn't found in `Factorial`.
4. The search for an implementation proceeds up the inheritance chain (this is done at runtime), arriving at the implementation in the base class `Iteration`. This is known as *dynamic dispatch*.
5. The body of that method executes, behaving like

```

84     result = self.initialize()
85
86     for i in range(1, 1 + 5): # start == 1
87         result = self.update(i, result)
88
89     return result

```

6. Inside the body, hook methods `self.initialize()` and `self.update()` are called. But these aren't implemented in the base class. However, keep in mind that `iterate()` was called in the scope of `Factorial`. Thus, their implementations are first looked for there, so that the body of `iterate` in fact behaves like:

```

90     result = 1
91
92     for i in range(1, 1 + 5):
93         result *= i
94
95     return result

```

There are even more objections, such as the runtime overhead, in terms of function calls, in particular of the hook methods. Note that `update()` occurs inside the loop. Finally, this solution is rather rigid. Whenever, you want to specialize `Iteration`, you “hard code” the initialization and update functionality into a new subclass. You cannot easily reuse hook methods separately from one specialization in another. There is, what we call, *tight coupling*: concrete hook methods are bound in the concrete subclass *at design time*.

2.2. Strategy Design Pattern

To allow specialization at runtime and independent variation, we can use the *Strategy* design pattern. The code at each variation point (hook) is considered a strategy obeying a fixed interface but admitting multiple implementations. This lets one inject concrete hook “strategies” via the constructor. First, we define interfaces (abstract base classes) to specify the signatures of the initialization and update hook strategies:

```

96 class Initializer(ABC):
97     """Interface (Abstract Base Class) for initialization.
98     """
99     @abstractmethod
100     def initialize(self) -> int:
101         """Return initial value."""
102
103 class Updater(ABC):
104     """Interface (Abstract Base Class) for updating.
105     """
106     @abstractmethod
107     def update(self, i: int, result: int) -> int:
108         """Return updated value."""

```

Next, we define the class with the template method (that class is no longer abstract and is also no longer subclassed):

```

109 class IterationS:
110     """Class for iteration with a template method,
111     using the Strategy design pattern for hook strategies.
112     """
113     def __init__(self, initializer: Initializer,
114                 updater: Updater) -> None:
115         """Store the hook strategies.
116         """
117         self.initializer = initializer
118         self.updater = updater
119
120     def iterate(self, start: int, n: int) -> int:
121         """The template method with the common algorithm.
122         """
123         result = self.initializer.initialize()
124
125         for i in range(start, start + n):
126             result = self.updater.update(i, result)
127
128         return result

```

These three classes are used as follows to implement factorial:

```

129 class Initializer1(Initializer):
130     """Initialize with 1.
131     """
132     @override
133     def initialize(self) -> int:
134         return 1
135
136 class FacUpdater(Updater):
137     """Updater for factorial.
138     """
139     @override
140     def update(self, i: int, result: int) -> int:
141         return result * i
142
143 def fac_iter_S(n: int) -> int:
144     return IterationS(Initializer1(), FacUpdater()
145                      ).iterate(1, n)

```

The array summer is implemented in a similar way:

```

146 class Initialize0(Initializer):
147     """Initialize with 0.
148     """
149     @override
150     def initialize(self) -> int:
151         return 0
152
153 class ArraySumUpdater(Updater):
154     """Updater for array_sum.
155     """
156     def __init__(self, array: list[int]) -> None:
157         self.array = array
158
159     @override
160     def update(self, i: int, result: int) -> int:
161         return result + self.array[i]
162
163 def sum_array_S(array: list[int], n: int) -> int:
164     return IterationS(Initialize0(),
165                     ArraySumUpdater(array)
166                     ).iterate(0, n)

```

We achieved increased flexibility (*loose coupling*), because the various initializers and updaters are more general and can now easily be mixed and matched at runtime. But we did pay a price for this: still more code and also more runtime overhead (but a simpler control flow). Concerning more runtime overhead, note the extra indirection in the calls of the hook strategies (lines 123 and 126): `self.initializer.initialize()` and `self.updater.update()`.

This may leave you wondering why object-oriented programming and OO design patterns were invented in the first place. And it also clarifies why these are not used in (most) programming contests.

3. DRY via Functional Programming

Rather than using classes, methods, and objects, we now stick to *pure functions*, that is, functions *without side effects*. For some, that may feel like tying your hands behind your back. But it is interesting to see where it gets you.

The class `IterationS` above is not really needed if we can inject the initialization and update functionality directly into its method (function) `iterate`. For updating, that requires a parameter of a function type, mapping two integers to an integer, typed in Python as `Callable[[int, int], int]`. Here is the definition of `iterate` as a pure function (not inside any class):

```

167 from typing import Callable # to type functions
168
169 def iterate(initialize: Callable[[], int], start: int,
170            update: Callable[[int, int], int]
171            ) -> Callable[[int], int]:
172     """Return a function that iterates, using
173     given start value and initialize and update functions.
174     """
175     def f(n: int) -> int:
176         result = initialize()
177
178         for i in range(start, start + n):
179             result = update(i, result)
180
181         return result
182
183     return f

```

To stay close to the object-oriented solutions, we have also made `initialize` a (parameterless) function parameter. Furthermore, note that `iterate` returns a function, in this case mapping an integer `n` to an integer. So, you could think of it as a *function factory*: from `initialize`, `start`, and `update`, it constructs a function (of `n`). This is how to use it to get the factorial function:

```

185 from operator import mul # multiplication
186
187 # type: Callable[[int], int]
188 fac_iter_FP = iterate(lambda: 1, 1, mul)

```

We can call our newly created function simply as `fac_iter_FP(5)`. But what about letting this factory function create a function to sum an array? We don't want to give

iterate an extra parameter, because that would be in the way when defining functions like factorial. Think about it. There is no object to inject the array into via a constructor. Therefore, the generated function needs to take the array as extra parameter. See if you can find a way out.

As Nikita Sobolev (2020) wrote: “Functional programmers are smart people. Really. They can do literally everything with just pure functions.” And so it is. The trick is to *generalize* the result type `int` (of the functions that our factory creates) to an arbitrary type `X`:

```

189 def iterateG[X](initialize: Callable[[], X], start: int,
190                update: Callable[[int, X], X]
191                ) -> Callable[[int], X]:
192     """Return a generic function int -> X that iterates
193     using given start value and initialize and update functions.
194     (Generic function definitions require Python 3.12.)
195     """
196     def f(n: int) -> X:
197         result: X = initialize()
198
199         for i in range(start, start + n):
200             result = update(i, result)
201
202         return result
203
204     return f

```

The types of the function parameters `initialize` and `update` had to be generalized accordingly. Function `iterateG` is known as a *generic* factory function, parameterized by type `X`. For factorial, `X` is simply `int` and we can still define:

```

205 fac_iter_FPG = iterateG(lambda: 1, 1, mul)

```

For array summing, we use the following *function* type as result type `X`:

```

206 type IntFromArray = Callable[[list[int]], int]

```

Values of this type are functions that map an array of integers to an integer. So, the factory now can create a function that takes integer `n` as parameter and produces a *function that takes an array as parameter*, which in turn computes the array sum. The factory creates what is known as a *curried* function: you need to feed it two parameters *in sequence*, rather than together.

Thus we have sneaked in a way of injecting the array into the iteration that repeatedly updates the variable `result`, which also has that function type. So, the loop now computes with functions of type `IntFromArray`. The initial value of `result` is

```

207 ifa_0: IntFromArray = lambda array: 0

```

because the initial value does not depend on the array (though for other specializations it could). Using lambda expressions, the `result` variable can now be updated by

```

208 # type: Callable[[int, IntFromArray], IntFromArray]
209 update_ifa = (lambda i, result_ifa:
210               lambda array: result_ifa(array) + array[i])

```

This may be a bit tricky to read at first, but by considering the types, the definition should make sense (hang on for an example). Finally, we define

```

211 # type: Callable[[int], IntFromArray]
212 sum_array_FPG = iterateG(lambda: ifa_0, 0, update_ifa)

```

The call `sum_array_FPG(n)(array)` sums the first `n` values of `array`. Let's reason through the three updates in the call `sum_array_FPG(3)`. The initial value of `result` is `ifa_0`:

```

213 1: update_ifa(0, ifa_0)
214 == { definition of ifa_0 }
215    update_ifa(0, lambda array: 0)
216 == { definition of update_ifa }
217    (lambda array:
218      (lambda array:
219        0
220      )(array) + array[0]
221    )
222
223 2: update_ifa(1, (lambda array:
224                  (lambda array:
225                    0
226                  )(array) + array[0]
227                ))
228 == { definition of update_ifa }
229    (lambda array:
230      (lambda array:
231        (lambda array:
232          0
233        )(array) + array[0]
234      )(array) + array[1]
235    )
236
237 3: update_ifa(2, (lambda array:
238                  (lambda array:
239                    (lambda array:
240                      0
241                    )(array) + array[0]
242                  )(array) + array[1]
243                ))
244 == { definition of update_ifa }
245    (lambda array:
246      (lambda array:
247        (lambda array:
248          (lambda array:
249            0

```

```

250         )(array) + array[0]
251         )(array) + array[1])
252         )(array) + array[2]
253     )

```

This can be considered *meta-programming*, where the factory function actually creates a program to solve the array summing problem for a specific value of n . When the Python interpreter evaluates a plain lambda expression like `lambda array: ...array...`, it won't evaluate the body `...array...`. It just produces a code object that is only put to work when the lambda expression is *called* on a specific argument. A plain lambda expression is a function treated as data. In languages that have better support for functional programming, the expressions above could be evaluated (simplified) further (at runtime). For instance, we have:

```

254     (lambda array:
255         (lambda array:
256             0
257         )(array) + array[0]
258     )
259 == { beta-reduction on inner lambda expression }
260     (lambda array:
261         0 + array[0]
262     )

```

Repeated beta-reductions would simplify the expression on lines 245–253 to

```

263     (lambda array:
264         0 + array[0] + array[1] + array[2]
265     )

```

which is clearly a program that sums the first three items of an array that is given as argument. In fact, the beta-reductions would be done during each update, so that the value of `result` would never be as complicated as above. That value would always have the shape

```

266     (lambda array:
267         0 + array[0] + ... + array[i]
268     )

```

since

```

269     update_ifa(i + 1, (lambda array:
270         0 + array[0] + ... + array[i]
271     ))
272 == { definition of update_ifa }
273     (lambda array:
274         (lambda array:
275             0 + array[0] + ... + array[i]
276         )(array) + array[i + 1]

```

```

277     )
278 ==   { beta-reduction }
279     (lambda array:
280     0 + array[0] + ... + array[i] + array[i + 1]
281     )

```

In a truly functional programming language, we could have written the factory such that it grows the program from the *inside*, rather than the *outside*. That way, if the array argument is already available before the factory starts, then the evaluation of the updates could already use that array to simplify the accumulating result expression even further.²

In conclusion, the DRY functional code is pretty and relatively short. The definition of the generic factory function `iterateG`, isn't much longer than the duplicated code (see lines 196–202), it is more general, and it includes documentation (type hints and a docstring). There is still execution overhead when doing this in a language like Python (or Java or C++). But by writing in a language better suitable for functional programming, that execution overhead can be reduced considerably by a good compiler.

The evaluation of a functional program can look contorted in its own way. So, you might think that it isn't much better than the contorted control flow in the execution of an object-oriented program. But here we need to remind you that the evaluation order of a program involving only pure functions (so, no mutable data either) does not matter. You can leave it to the compiler and runtime system to find an efficient order. In particular, function arguments need not be (fully) evaluated before putting the function body to work. Python won't do that without explicit help and its default eager execution order is not so good for functional programs. Moreover, pure function evaluation is easy to parallelize.

Due to their higher generality, reading and writing of functional programs does place higher demands on the programmer's abstraction skills. But for the talented participants of olympiads that should not be an obstacle. See Appendix A for more Python examples.

4. A WET Conclusion with an AHA Insight

The examples that we gave are clearly out of proportion. Functions `fac_iter` and `sum_array` have only five nearly duplicated lines of code. You see such loops everywhere in code. It would seem to make little sense to eliminate them all. For the object-oriented style that is certainly the case, because the overhead of abstraction is considerable. It only pays off when code clones are larger.

In the functional style, the situation is quite different. There, one can have small building blocks that are very general. Since purely functional programming languages

² See <https://t-verhoeff-software.pages.tue.nl/code-for-staying-dry-with-oo-and-fp> for an interactive evaluation explorer.

have no loops, one needs to use recursion (which is not an easy thing (Verhoeff, 2023)). Many forms of recursion have been encapsulated in *recursion schemes* such as *folds* and *unfolds*. These act like our factory function `iterateG`, and these are preferred over writing your own recursive function definitions. You hardly see explicit recursion in good functional programs.

Once a beginning programmer has seen the benefits of abstraction, and how it can help avoid code duplication, there lurks the temptation to apply it at every opportunity to ensure that code is DRY from the start. Creating abstractions is indeed intellectually satisfying, but one needs to be careful.

Programmers with more experience will know that abstractions are hard to get right and that they may not fit a future situation that you had envisioned. Adjusting an abstraction and then also all its instantiations is about as dangerous as updating duplicated code. Therefore, another common advice is to *Avoid Hasty Abstractions* (AHA). If you generalize code too early, you may end up with abstractions that later turn out to be less useful and that need adjusting. In fact, function `iterateG` could be considered a hasty abstraction. In functional programming, a more appropriate abstraction turns out to be `foldN` (see Appendix A).

Don't be afraid to write some duplicated code. That way you can better see in what direction and how far it makes sense to generalize your code. This is captured in yet another common advice: *Write Everything Twice* (WET), or even better *Write Everything Thrice*. By explicitly writing out similar code multiple times, it becomes easier to recognize the relevant patterns and how to abstract from the differences.

That brings me to programming contests. Are these techniques to achieve DRY code of any use there? Code written in a contest is throw-away code. But generalization is a great problem solving technique. So, getting better at that can help. Also, applying abstraction in your code can be helpful to stay in control. It would then be useful if you can use a programming language with good support for abstraction. Unfortunately, on the theoretical side we have much more advanced notions for abstraction than are available in common practical programming languages. The mind can see things that are hard to express in code. I find this a shortcoming of the current format of the IOI.

I hope that team leaders are going to study the examples in this article together with their contestants. Higher-order functions, functions with functions as parameters and returning functions, are very powerful devices. It takes some practice to get used to the functional style, but I am convinced that clever contestants will enjoy it.

Acknowledgment

I would like to thank Berry Schoenmakers (TU Eindhoven, Netherlands) and Mārtinš Opmanis (Latvia) for helping me improve this article.

References

- Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.
- Hunt, A., Thomas, D. (20th Anniversary Ed. 20219; 1st ed. 1999). *The Pragmatic Programmer: From Journeyman to Master*, Addison-Wesley.
- Sobolev, N. (2020). “Typed Functional Dependency Injection in Python”, blog post, 17 Feb. 2020. <https://dev.to/wemake-services/typed-functional-dependency-injection-in-python-4e7b>
- Verhoeff, T. (2023). Understanding and designing recursive functions via syntactic rewriting. *IOI Journal*, 17, 99–119.
- Verhoeff, T. (2024). Git repository with source code for “Staying DRY with OO and FP”. <https://gitlab.tue.nl/t-verhoeff-software/code-for-staying-dry-with-oo-and-fp> (Accessed 27 April 2024)



T. Verhoeff is Assistant Professor in Computer Science at Eindhoven University of Technology, where he works in the group Software Engineering & Technology. His research interests are support tools for verified software development and model driven engineering. He received the IOI Distinguished Service Award at IOI 2007 in Zagreb, Croatia, in particular for his role in setting up and maintaining a web archive of IOI-related material and facilities for communication in the IOI community, and in establishing, developing, chairing, and contributing to the IOI Scientific Committee from 1999 until 2007.

Appendix A

Two more Python examples, using foldN

Generic factory function `iteratG` was a hasty abstraction. It does not need both parameters `initialize` and `start`. They can be combined into one parameter, say `initial`. A better generalization is generic factory function `foldN`:

```

282 def foldN[X](initial: X,
283             update: Callable[[X], X]
284             ) -> Callable[[int], X]:
285     """Return a generic function int -> X
286     that applies update repeatedly to initial.
287     """
288     def f(n: int) -> X:
289         result: X = initial
290
291         for i in range(0, n):
292             result = update(result)
293
294         return result
295
296     return f

```

It cleanly captures the essence of the natural numbers. Each natural number has a unique construction using `zero = 0` and `succ = lambda n: n + 1`:

```

297     n == succ(succ(...succ(zero)...)) # n copies of succ

```

Similarly,

```

298     foldN(i, u)(n) == u(u(...u(i)...)) # n copies of u

```

Thus, folding replaces constructor `zero` by `i` and `succ` by `u`.

Let's use `foldN` to compute the famous Fibonacci numbers:

```

299 def fib(n: int) -> int:
300     return foldN((0, 1),
301                lambda t: (t[1], t[0] + t[1]))(n)[0]

```

Here, the the generic type parameter `X` is instantiated by `tuple[int, int]`.

And what do you think of this application of `foldN`:

```

302 type Converter = Callable[[list[int]],
303                          tuple[int, list[str], int, int]]
304
305 init_c: Converter = lambda bits: (1, [], 1, 0)
306
307 def update_c(result_c: Converter) -> Converter:
308     def f(bits: list[int]) -> tuple[int, list[str], int, int]:
309         index, strings, power, value = result_c(bits)
310         bit = bits[-index] # indexed from the end

```

```

311         strings.append(f"{bit} * 2^{index - 1}")
312         return index + 1, strings, power * 2, value + bit*power
313
314     return f
315
316 def convert(bits: list[int]) -> str:
317     _, strings, _, value = foldN(init_c,
318                                update_c)(len(bits))(bits)
319     return ' + '.join(strings) + f" = {value}"

```

Generic type parameter `X` is now instantiated by `Converter`, which is a function type. That is why the function produced by `foldN` (see lines 320–321) takes two parameters in succession: an integer and then an array of bits. Here is an example usage of `convert`:

```

320     print(convert([1, 1, 0, 1]))

```

It produces as output:

$$1 * 2^0 + 0 * 2^1 + 1 * 2^2 + 1 * 2^3 = 13$$

The Impact of Non-Formal Educational Approach on the Academic Performance and Employability of Engineering and Computer Science Students

Eslam M. WAGEED, Yousry S. ELGAMAL,
Ossama M. ISMAIL, Mohamed H. ABDRABOU

Arab Academy for Science, Technology and Maritime Transport (AASTMT), Egypt
e-mail: eslam@aast.edu, yelgamal@aast.edu, ossama@aast.edu, mhabdrabou@egypt.aast.edu

Abstract. Education is no longer a one-time event due to the quick changes in the modern world, the diversity of knowledge it contains, and the ongoing need for personal growth. It turns into a dynamic process that happens all through life. The old curricula and systems of formal education are finding it difficult to keep up with the quick changes occurring in the field of computer science. Non-formal education is a solution to the requirements of the era and can be implemented through seminars, training camps and workshops. It could be additional activities like competitions or extracurricular learning. The aim for this study is to investigate the impact of the non-formal educational approaches on engineering and computer science students' academic performance as well as their chances to obtain a job after graduation.

Keywords: non-formal education, academic performance, employability.

1. Introduction

Education is an essential component of human development, shaping people, societies, and nations. For decades, education has been considered as a transformational force, allowing people to gain the knowledge, skills, and attitudes required for personal development, economic growth, and improvements in society.

Education is not about sitting in classrooms and recollecting facts. It's not about texts or tests. Education focuses on developing critical thinking and problem-solving skills. Education encourages the ability to face real-world difficulties by stimulating curiosity, analysis, and innovation.

In 2015, the United Nations (UN) adopted a set of 17 goals as a universal call to action to end poverty, protect the planet, and ensure that people will enjoy peace and prosperity by 2030. These 17 goals are the Sustainable Development Goals (SDGs). The SDGs place priority on education because of its ability to empower individuals,

encourage sustainable development, and promote a more just and equitable world. According to the UN website, education is the key that will allow many other SDGs to be achieved. When people are able to get quality education they can break from the cycle of poverty.

1.1. *Types of Education: Informal, Formal, Non-Formal*

According to the International Commission on the Development of Education (commonly known as the Faure Commission) which was established by UNESCO in 1972, education was classified into three categories: formal, non-formal, and informal education (International Commission..., 1972).

- **Formal Education** refers to the highly institutionalized, systematically graded, and hierarchically structured “education system,” which extends from primary school to the upper levels of university.
- **Non-Formal Education** is any organized educational activity performed outside the formal educational system to give specific types of learning to certain groups of people, including adults and children.
- **Informal Education** is a lifelong process in which individuals gain and accumulate knowledge, skills, attitudes, and insights through daily experiences and exposure to the environment at home, work, play, family, and friends.

The rapidly changes in the world of technology with its diversified nature of knowledge and constant need for personal development, education is no longer a one-time event. It becomes a dynamic process that occurs throughout life. Formal education, with its traditional structures and curricula, is struggling to keep up with the rapid pace of change in the computer science world.

On the other hand, Non-formal education empowers individuals to have lifelong learning throughout their lives and stay up to date with the newest advancements in technology. It is characterized by flexibility in subjects and schedules. It focuses on practical skills without diving too far into theories. Also, it is accessible across a variety of platforms for free or at affordable prices.

1.2. *Characteristics of Non-Formal Education*

Faure’s [1] main characteristics about non-formal education are as follows:

- **Flexibility:** The non-formal education is flexible and adaptable, catering to the diverse needs and circumstances of learners.
- **Voluntary Participation:** Participation in non-formal education is usually voluntary. Learners willingly engage in these educational activities based on their interests, needs, and motivations.

- **Diversity of Content:** Non-formal education covers a wide range of topics, many of which go beyond what's covered in traditional education. It embraces areas for personal development, as well as practical knowledge as well as skills.
- **Accessibility:** Non-formal education seeks to reach people who might not have had easy access to traditional educational because of a variety of factors, such as geography, schedule conflicts, or unique personal situations.
- **Informality in Structure:** Non-formal education is frequently more casual than formal education, which has strict curricula and structures. It supports a variety of arrangements and formats, enabling innovative and imaginative teaching strategies.
- **Lifelong Learning:** Non-formal education is in favor of lifetime learning, which holds that opportunities for learning should not be restricted to a person's age or stage of development but should be always available to them.
- **Flexibility:** The non-formal education is flexible and adaptable, catering to the diverse needs and circumstances of learners.
- **Voluntary Participation:** Participation in non-formal education is usually voluntary. Learners willingly engage in these educational activities based on their interests, needs, and motivations.
- **Diversity of Content:** Non-formal education covers a wide range of topics, many of which go beyond what's covered in traditional education. It embraces areas for personal development, as well as practical knowledge as well as skills.
- **Accessibility:** Non-formal education seeks to reach people who might not have had easy access to traditional educational because of a variety of factors, such as geography, schedule conflicts, or unique personal situations.
- **Informality in Structure:** Non-formal education is frequently more casual than formal education, which has strict curricula and structures. It supports a variety of arrangements and formats, enabling innovative and imaginative teaching strategies.
- **Lifelong Learning:** Non-formal education is in favor of lifetime learning, which holds that opportunities for learning should not be restricted to a person's age or stage of development but should be always available to them.

The function of non-formal education is to develop the potential of students with an emphasis on mastering functional knowledge and skills and developing professional attitudes and personalities (Elice *et al.*, 2023). Non-formal education responds to the learning needs of a group and can be carried out, in seminars, training sessions, workshops, through partnerships between facilitators and participants, in groups or communities or in other organizations other than in the education system. It does not end with the granting of certificates but usually it takes place within an institutionalized framework, outside the school system, comprising extra class or extra didactic activities like competitions, extracurricular education, and training activities (Vilceanu, 2019).

Non-formal educational activities have a positive impact university students' academic success and employability, enhancing their readiness for the job market through holistic development beyond the curriculum (Norberto *et al.*, 2023).

2. Research Method

Depending on the Statistical Yearbook – Education 2023 of the Central Agency for Public Mobilization and Statistics in Egypt (CAPMAS) which is considered a comprehensive statistical reference for all official statistics, the annual number of graduates from engineering and computer science faculties is almost 35 thousand graduates. It is increasing every year by 15%. A small percentage of these graduates are joining the non-formal educational system that operates in various institutions such as the AAST Regional Informatics Center (RIC).

The aim for this study is to investigate the impact of these non-formal educational activities on engineering and computer science students' academic performance as well as their chances to obtain a job after graduation.

The study will examine different aspects may be impacted by participating in these non-formal educational approaches, such as the ability to work in groups, time management skills, problem-solving skills, and creativity, as well as how engagement in non-formal activities may lead to job opportunities.

A questionnaire of two sections was used to collect the data. The first section of the questionnaire introduces the research's goal and the criteria that questionnaire participants should meet. Following this introduction, some personal information is gathered, such as the participant's age, gender, education level, university, and prior RIC activities.

The questionnaire's second section consisted of 12 statements and separated into two parts.

The first part has 6 statements that measure the impact of the RIC activities on the academic performance:

1. GPA Improvement.
2. Self-Learning Skills.
3. Problem-Solving and Creativity.
4. Time Management Skills.
5. Encouragement to join engineering field.
6. Ability to get a scholarship.

The second part of the questionnaire contains 6 statements that measure the impact of the RIC activities on the employability skills and opportunities as follows:

1. Chances for getting a job or internship.
2. Ability to work under stress.
3. Communication skills improvement.
4. Ability to work in groups.
5. Creating social networks.
6. Early participation giving an edge.

The research population consists of undergraduates and graduates who participated in non-formal educational approaches during their high school or university studies. The questionnaire was not open to everyone. The Purposive Sample Technique was applied

to identify the individuals that best suited to answer the research question. The questionnaire participants should meet the following requirements:

1. The participant must be a student or graduate of an engineering or computer science faculty.
2. The participant must have joined at least one of the RIC activities or similar activities while in high school, university, or both.

The RIC activities are the Egyptian/International Olympiad in Informatics (EOI/IOI), Egyptian/International Collegiate Programming Contest (ECPC/ICPC), RoboCup, Remotely Operated Vehicle (ROV), International Challenge on Informatics and Computational Thinking (Bebras), Formula Students and Robocon or similar activities like the Egyptian/International Math Olympiad (EMO/IMO), Catch the Flag (CTF) and Intel ISEF.

3. Results and Findings

As mentioned above, the number of graduates from engineering and computer science faculties in Egypt in 2023 is almost 35 thousand graduates and increasing every year by 15%. A small percentage of these graduates are joining the non-formal educational system that operates in various institutions such as the Arab Academy for Science, Technology, and Maritime Transport, Regional Informatics Center (AASMT-RIC). These non-formal educational activities are directed towards high school students and undergraduates. All the activities are focusing on the computer science and robotics fields.

In order to get accurate responses, the questionnaire was distributed to 800 individuals that participate in the AASTMT-RIC activities or similar activities while we receive 270 responses.

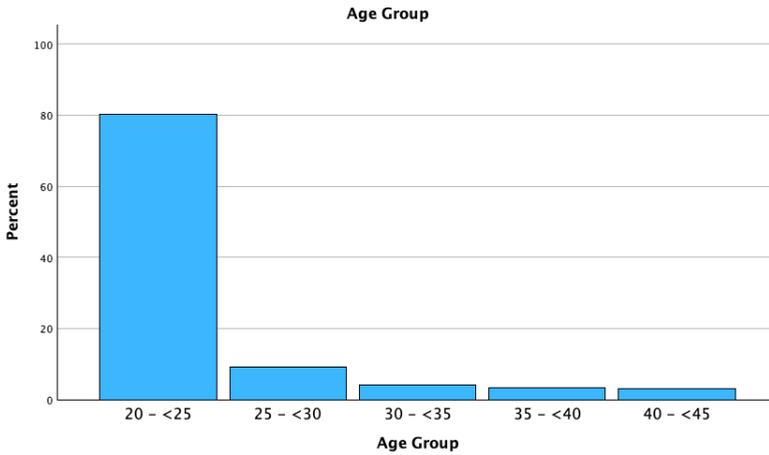
The following section shows the analysis for the demographic part of the questionnaire in addition to the academic performance and employability.

3.1. Demographic Analysis

3.1.1. Age Group

The majority of survey respondents – more than 80% – are in the 20–25 age range. They continue to attend the institution to study. The primary concern that came from

Age Group	N	%
20 – <25	217	80.4%
25 – <30	25	9.3%
30 – <35	11	4.1%
35 – <40	9	3.3%
40 – <45	8	3.0%

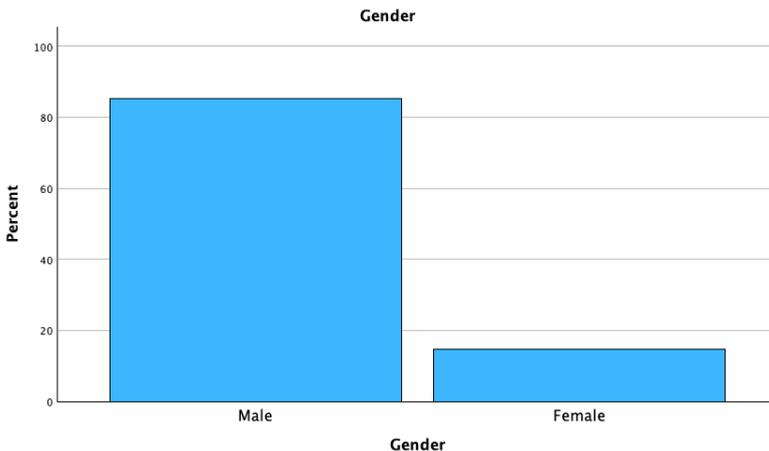


this inquiry is that we don't maintain close relationships with our graduate students, or, to put it another way, we don't communicate well with them. This could indicate how crucial it is to have alumni involved in every activity to maintain the flow of knowledge by putting former students in touch with current ones.

3.1.2. Gender

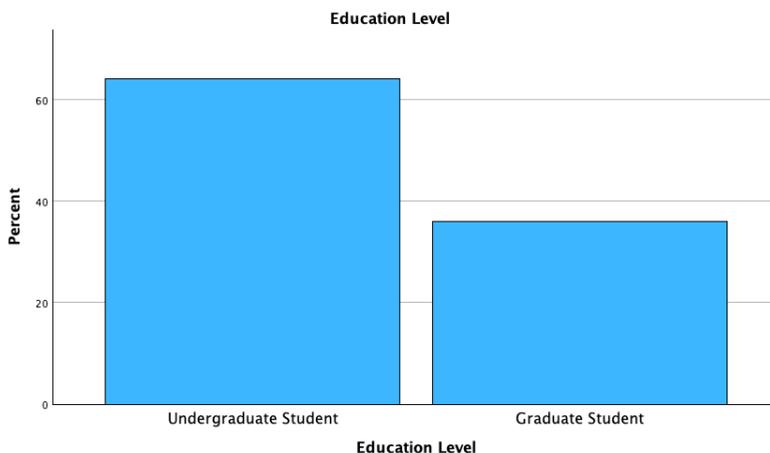
Given that most participants in the non-formal activities are males, the male to female ratio is reasonable. To prevent the discouragement that could result from not winning or achieving the top level in the competitions, we might need to make more effort to engage more girls or split the prizes between males and females.

Gender	N	%
Male	230	85.2%
Female	40	14.8%



3.1.3. Education Level

Education Level	N	%
Undergraduate Student	173	64.1%
Graduate Student	97	35.9%



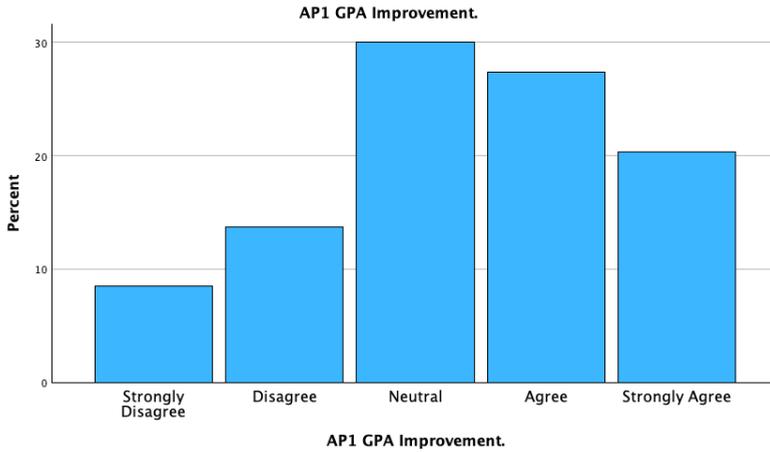
Undergraduates make up the majority of participants, according to the Age group responses. Of the participants, around 64 percent are still enrolled in university courses. This could be considered as a limitation in the research.

3.2. Academic Performance

3.2.1. Participating in RIC Activities Helped me to Improve my GPA and Grades

GPA stands for Grade Point Average. It's a numerical way to summarize a student's academic performance over time. Grade point average (GPA), obtained credits (ECTS) and gender to be the most consistent and decisive predictors of academic performance (Kocsis & Molnár, 2024). **Nearly 48% of participants stated that their GPA is positively impacted by their participation in extracurricular activities.** A 30% of the participants believe that extracurricular activities have no impact directly on their GPA, while the remaining 22% believe that involvement in these activities lowers their GPA. This could be because of the extra time needed for non-formal activities, which could result in less study hours for formal education courses. This inquiry should have an impact on the fourth question which is related to the improvement of the Time-Management skills.

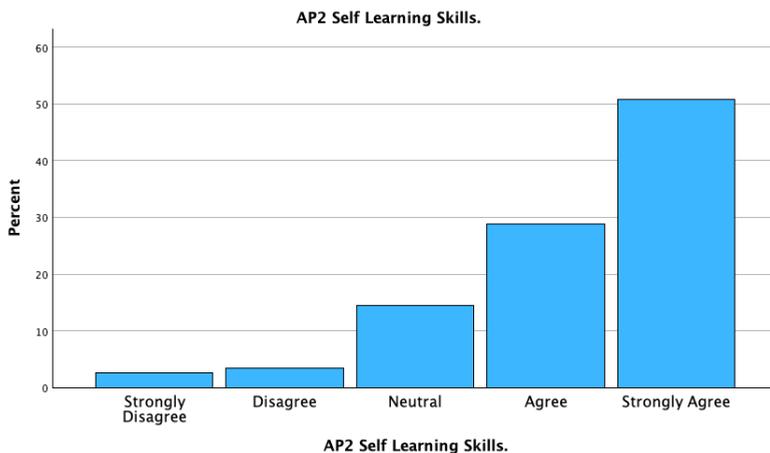
API GPA Improvement	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
N	23	37	81	74	55
%	8.50%	13.70%	30.00%	27.40%	20.40%



3.2.2. *I became a self-learner after I joined the RIC activities*

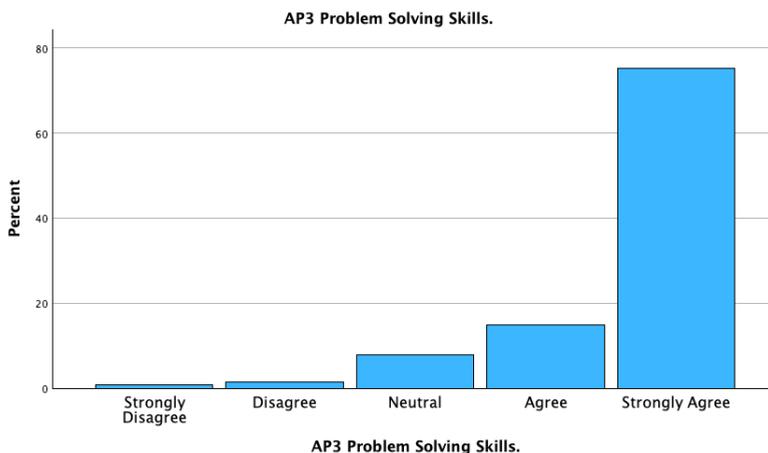
Self-learning skills empower students to become independent, adaptable, and lifelong learners. It’s a valuable asset that benefits them throughout their academic journey and beyond. Self-regulated online learning skills were a significant predictor of academic success (Tijen, 2022). The questionnaire responses make it clearly evident that taking part in non-formal educational events has a significant influence on students’ abilities for self-learning. Nearly 80% of participants said that after participating in extracurricular activities, they turned into self-learners. Of the respondents, about 6% believe that the activities had no influence on their ability to learn on their own, while the remaining respondents observe no change in either direction.

AP2 Self Learning Skills	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
N	7	9	39	78	137
%	2.60%	3.30%	14.40%	28.90%	50.70%



3.2.3. Participating in RIC Activities Improved my Problem-Solving Skills and Creativity

AP3 Problem Solving Skills	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
N	2	4	21	40	203
%	0.70%	1.50%	7.80%	14.80%	75.20%

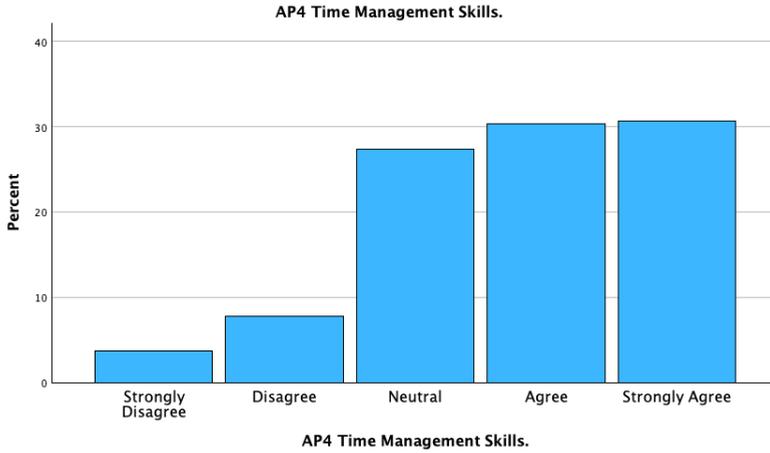


Extracurricular activities are considered essential for optimal intellectual development, complementing school activities (Miltiadis, 2022). **Problem-solving skills are key for students.** They boost academic success, critical thinking, confidence, and teamwork. They prepare students for a world full of challenges. The graph above makes it clearly obvious that engaging in extracurricular activities greatly enhances the ability for creativity and problem-solving. 87% of the participants agreed that participating in competitions improves their creativity and problem-solving abilities.

3.2.4. Joining the RIC Activities Improved my Time Management Skills

The goal of time management skills is to maximize every day. They are crucial since it seems like we never have enough time to accomplish what we need to or want to. You can accomplish your goals, be more productive, and experience less stress if you have good time management abilities. The effective coping strategies, such as good time management, can lead to better academic outcomes and career development for nursing students (Achamma & Nirmala, 2023). As we could see from the results, time-management skill is affected positively by engaging in non-formal activities. More than 88% of the responses indicate that the students gain the ability to divide the daytime between the formal and non-formal education.

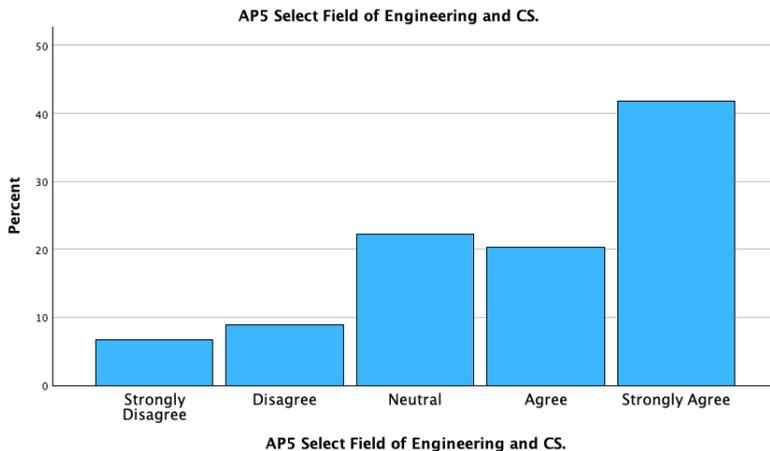
AP4 Time Management Skills	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
N	10	21	74	82	83
%	3.70%	7.80%	27.40%	30.40%	30.70%



3.2.5. Participating in the RIC Activities Helped me to Early Select my Field of Study and Join the Engineering and Computer Science Career

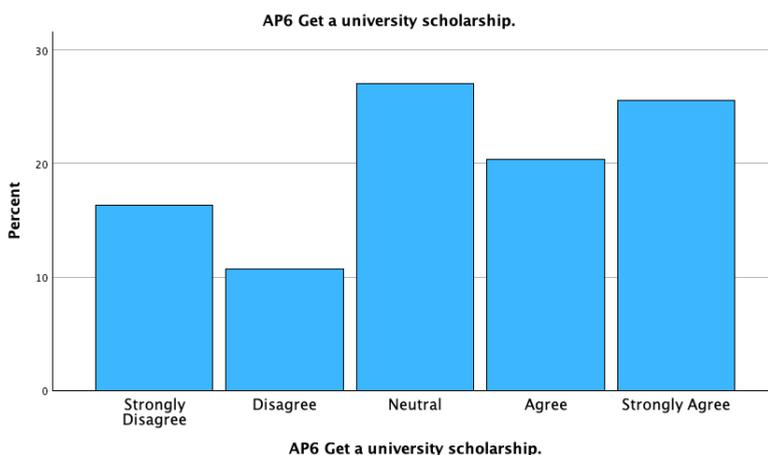
Early career thinking helps students discover their strengths and interests, setting goals and staying motivated in school. Out-of-school STEM activities positively influenced the STEM career choices of female students (İsmail, 2021). As we can see in the graph, more than 60% agreed that participating in the non-formal activities encourage more students to join the engineering or computer science fields. A research by (Viacheslav *et al.*, 2020) found that Graduates with STEM training closely connected to their future profession show a higher tendency to choose relevant study directions.

AP5 Select Field of Engineering and CS	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
N	18	24	60	55	113
%	6.70%	8.90%	22.20%	20.40%	41.90%



3.2.6. *Joining the RIC activities enhances my chances for getting a university scholarship.*

AP6 Get a university scholarship	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
N	44	29	73	55	69
%	16.30%	10.70%	27.00%	20.40%	25.60%



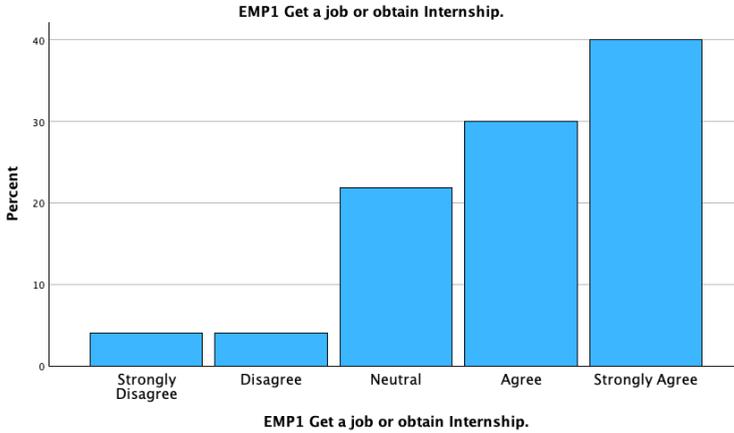
Not every high school student who participated in extracurricular activities will be awarded a scholarship to attend a university. Several other criteria also have a role, such as the degree of student participation and the universities’ ability to award grants. The graph indicates that there isn’t a pattern in the replies. The majority of universities award scholarships to students who participate at the highest levels, such as the IMO and IOI. Very few young people were able to participate to this extent, and even fewer were able to receive a medal. This appears to be one of the causes of the equal percentage of answers without a dominant response.

3.3. *Employability*

3.3.1. *By Taking Part in RIC Activities, my Chances of Getting a Job or Obtaining an Internship Increased*

Extracurricular activities (ECAs) mostly have a positive impact on university students’ academic success and employability, with only a few showing negative effects (Vilceanu, 2019). Employers perceive extracurricular involvement as a moderate influencer of graduate employability (Bodunrin, 2017). According to the responses, 70% of the students believed their chances of obtaining a job or an internship were improved by their involvement in non-formal activities. The activities outside of school help students become more competent and ready for the workforce.

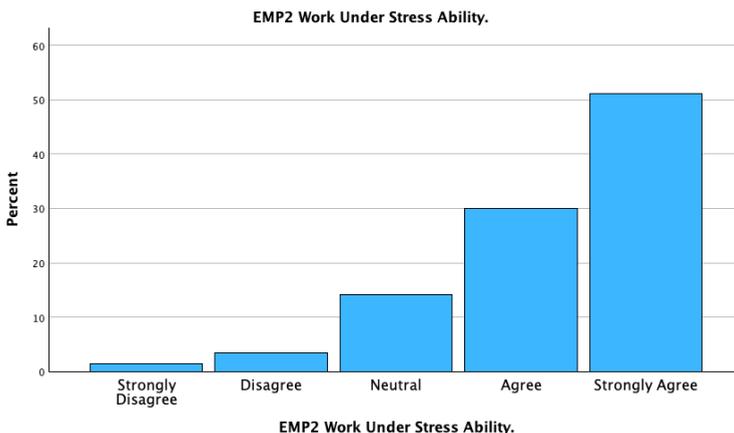
EMP1 Get a job or obtain Internship	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
N	11	11	59	81	108
%	4.10%	4.10%	21.90%	30.00%	40.00%



3.3.2. Participating in the RIC Activities Helped me to Improve my Ability to Work under Stress

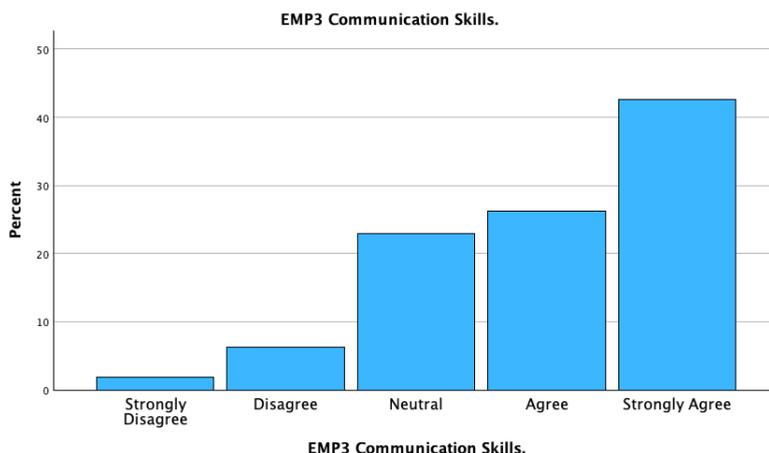
The ability to work under stress is a valuable skill that many employers highly value. University students have a high willingness to develop all soft skills, with stress management being prioritized as a significant developmental need (Esra, 2023). The results show that more than 80% of the students agreed that non-formal educational activities developed their work under stress skill. The ability to work under stress is a skill that can be developed and improved over time with practice and self-awareness.

EMP2 Work Under Stress Ability	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
N	4	9	38	81	138
%	1.50%	3.30%	14.10%	30.00%	51.10%



3.3.3. Joining the RIC Activities Improved my Communication Skills

EMP3 Communication Skills	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
N	5	17	62	71	115
%	1.90%	6.30%	23.00%	26.30%	42.60%

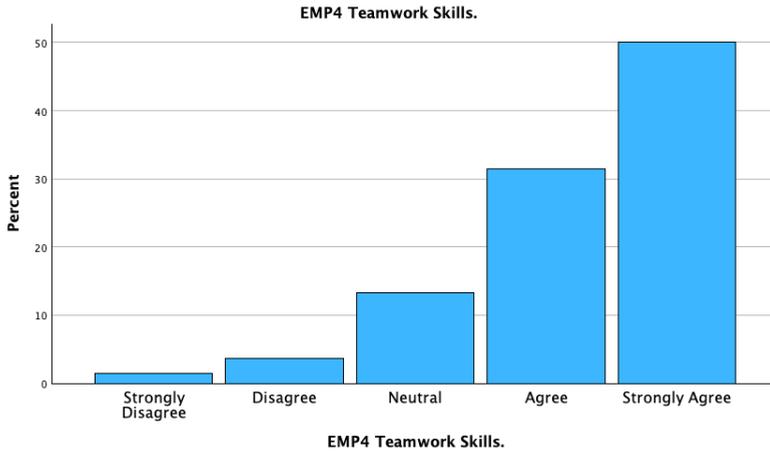


Improving communication skills can have a significant impact on both personal and professional life. It is obvious that participating in the non-formal activities positively affect the students' communication skills. Improving communication skills takes time and effort, but the benefits are invaluable in personal relationships, teamwork, leadership, and overall career success.

3.3.4. Taking Part in RIC Activities Improved my Ability to Work in a Team and Respect others' Opinions

The ability to work effectively in groups significantly impacts an individual's career advancement opportunities. Teamwork is a vital part of learning. If the major of students is software engineering, working in group is their daily working way. When they are students, mastering how to work in group is crucial to their careers (Chenyang, 2021). Majority of students agree – more than 81% of survey respondents – that participation in extracurricular activities improves group work. It also improves the acceptance of different points of view and cultural traditions. Employers in multinational companies place a high value on this ability. Working across cultural and linguistic divides has become imperative in today's world.

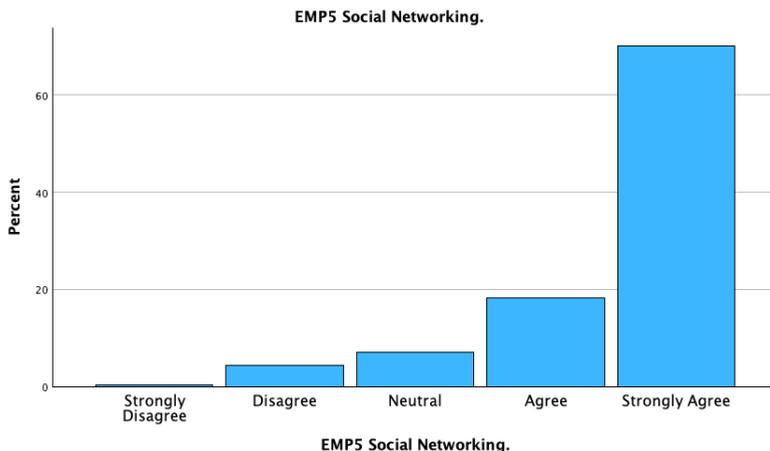
EMP4 Teamwork Skills	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
N	4	10	36	85	135
%	1.50%	3.70%	13.30%	31.50%	50.00%



3.3.5. By Participating in the RIC Activities, I Created a Social Network of People who had the Same Interest in my Studying Field

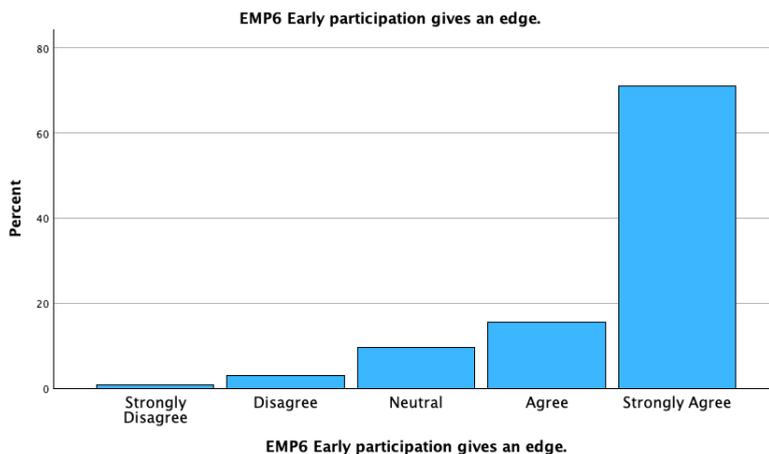
Social connectedness is crucial for employability and career success, benefiting individuals in job acquisition, career advancement, and professional learning (Ruth *et al.*, 2019). Almost 88% of the questionnaire responders confirm the positive impact of joining the RIC activities on creating social networks of people who have the same interest from the same studying field. Passion sharing among professionals can result in information sharing about current trends, employment openings, and industry expertise. This network may prove to be an invaluable asset in discovering forthcoming prospects or obtaining recommendations.

EMP5 Social Networking	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
N	1	12	19	49	189
%	0.40%	4.40%	7.00%	18.10%	70.00%



3.3.6. Early Participation in the RIC Activities Gives an Edge to the Students

EMP6 Early participation gives an edge	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
N	2	8	26	42	192
%	0.70%	3.00%	9.60%	15.60%	71.10%



Students participate in non-formal education programs to complement their formal studies and enhance their qualifications. Students also engage in these programs to utilize leisure time, develop social networks, and enjoy learning new things, focusing on acquiring attitudes and skills (Coombs & Ahmed, 1974). More than 85% of respondents said they thought being involved in RIC activities early on gave them an advantage over their peers.

4. Conclusion

In conclusion, the evidence presented underscores the significant positive impact of non-formal education on both academic performance and employability among students. Through various non-traditional learning experiences, such as workshops, internships, and skill-based programs, students have been able to enhance their cognitive abilities, practical skills, and overall competencies. This exposure not only enriches their academic journey by complementing formal education but also equips them with the necessary tools and attributes sought after by employers in the professional arena.

Furthermore, the versatility of non-formal education allows students to cultivate a broader skill set that extends beyond conventional academic subjects, fostering adaptability and innovation. By participating in diverse learning environments outside the classroom, students gain practical insights and hands-on experiences that are instrumental in preparing them for the dynamic demands of the workforce.

The outcomes are clear: students who engage in non-formal education demonstrate improved academic outcomes, while also showcasing enhanced employability through the development of critical soft skills, problem-solving abilities, and professional networks.

As education continues to evolve, incorporating non-formal educational opportunities into mainstream curricula and career development strategies should be prioritized. This integration not only enriches the educational experience but also cultivates a generation of students better equipped to thrive academically and professionally in an ever-changing global landscape. Thus, the promotion and support of non-formal education initiatives are essential for empowering students and maximizing their potential in both academic pursuits and future career endeavors.

References

- Achamma, V., Nirmala, D.R. (2023). Impact of time management program on stress and coping strategies adopted by nursing students with regard to academic performance. *IP Journal of Paediatrics and Nursing Science*, 48–56.
- Bodunrin, I., Akinrinmade, A., Ayeni, O. (2017). Influence of extracurricular involvement on graduate employability. *Malaysian Online Journal of Educational Management*, 29–31.
- Chenyang, Z. (2021). Explore Ways to Study Effectively in Groups from Data Scienc. *IEEE International Conference on Educational Technology (ICET)*, pp. 26–30.
- Coombs, P.H., Ahmed, M. (1974). *Attacking Rural Poverty: How Non-Formal Education Can Help*. John Hopkins Press, Baltimore.
- Elice, D., Maseleno, A., Pahrudin, A. (2023). Formal, Informaland Non-Formal Education Systems. *Journal of Learning and Educational Policy*, 4(1).
- Esra, A.B. (2013). Stress Management: A Priority Developmental Need for University Students. *Acibadem üniversitesi sağlık bilimleri dergisi*.
- International Commission on the Development of Education, Edgar Faure. *Learning to be: The world of education today and tomorrow*, Paris: UNESCO, 1972.
- İsmail, D. (2021). Impact of Out-of-School STEM Activities on STEM Career Choices of Female Students. *Eurasian Journal of Educational Research*.
- Kocsis, A., Molnár, G. (2024). Oxford Review of Education. *Oxford Review of Education*, pp. 1–19.
- Maria, P.-K. (2022). Participation of university students in non-formal lifelong learning programs: types of programs, reasons for participation and the importance of learning outcomes in their student, professional, personal, and social life. *European Journal of Education Studies*.
- Miltiadis, Z. (2022). Extracurricular Activities in the Function of Intellectual Education. *Reflexia*, pp. 9–18.
- Norberto, R., Malafaia, C., Neves, T., Menezes, I. (2023). The Impact of Extracurricular Activities on University Students' Academic Success and Employability. *European Journal of Higher Education*, p. 1–21.
- Ruth, B., Denise, J., Kate, L., Matalena, T. (2019). Social connectedness and graduate employability: exploring the professional networks of graduates from business and creative industries. *Higher Education and the Future of Graduate Employability*, pp. 70–89.
- Tijen, T. (2022). The Effect of Self-regulated Online Learning Skills on Academic Achievement,” *Anadolu Journal of Educational Sciences International*, pp. 389–416.
- Viacheslav, O., Nataliia, V., Liudmyla, K., & Nataliya, A. (2020). Studies of impact of specialized STEM training on choice further education. 75:04014–. DOI: 10.1051/SHSCONF/20207504014, *SHS Web Conference*.
- Vilceanu, F. V. (2019). Startegic References for Non-Formal Education. *Horizons for sustainability, Constantin Brâncuși” University of Târgu-Jiu*, no. 2.



E. Wageed is the head of school programs department at the Regional Informatics Center (RIC) at the AASTMT. He is the executive director of the Egyptian Olympiad in Informatics since 2008, IOI IC member since 2014 and the co-founder of RoboCupJunior Egypt and Beaver Challenge in Egypt. He promotes the non-formal educational activities in many countries in the middle east. His doctoral degree on the impact of non-formal educational activities.



Y. Elgamal is a Professor of Computer Engineering, senior advisor at The Arab Academy for Science and Technology, and Chairman of The Computer Scientific Society (CSS), Alexandria-Egypt. He served as The Minister of Education of Egypt 2005–2010, Chairman, Board of Trustees, Egypt Japan University of Science and Technology (E-JUST) 2010–2014, and the senior consultant of the National Telecommunications Institute of Egypt. He is a member of the group of experts preparing The Global Knowledge Index, and The Chairman of The Information and Communication Committee at The National Committee of Education, Culture, and Science (UNESCO, ALECSO, ISESCO). Elgamal has also served in a number of capacities at The Arab Academy for Science and Technology and Maritime Transport including Vice-President for Education and Research, Founding Dean of College of Engineering and Technology, Founding Chairman of Electronics and Communication Department, and Assistant to the President for Informatics. He served also as a Lecture of Nuclear Electronics at The Atomic Energy Agency (IAEA).



O. Ismail is the Founding Dean of the Regional Informatics Center (RIC) at AASTMT. The RIC was conceived to advance Robotics and Competitive Programming in the MENA region, promoting student talent and fostering a skilled cohort to keep pace with global advancements in these fields. Oussama was the director and head of the scientific committee of the Egyptian Olympiad in Informatics (EOI) since 2003 and head of the HSC of IOI 2008 Egypt.



M. Abdrabou is the Dean of Productivity and Quality Institute (PQI) at AASTMT. He specialized in Quality and Organizational Excellence as well as excelled in various academic and administrative positions in PQI of higher learning during his career. He is deeply involved in engagements with regulatory bodies and accreditation agencies for furtherance of institutional interests. He has performed a key role in improving quality in education.

REPORTS

Palestine at the International Olympiad in Informatics: Advancing Computational Thinking Among K-12 Students

Musa ALREFAYA, Samed ALHAJAJLA

*Palestinian Olympiad in Informatics, Palestine Polytechnic University, Meshka
e-mail: mousa@ppu.edu, haasamed@gmail.com*

Abstract. The Palestinian Olympiad in Informatics has been a significant endeavour since its inception in 2017. It has steadily grown, with programming clubs established in all schools to foster student interest and skill development. The milestone of winning the first bronze medal in 2022 marked a notable achievement in our journey. This report provides an overview of Palestine's participation in the International Olympiad in Informatics, outlining our objectives, preparation efforts, performance, and the profound impact of our involvement on the local and international levels, sparking increased interest in computer science across Palestine and fostering international collaboration and innovation in informatics.

Keywords: Palestine, education technology, Palestinian Olympiad in Informatics, computational problem-solving.

Introduction

The state of Palestine has joined the family of the International Olympiad in Informatics through an initiative of the College of Information Technology and Computer Engineering at Palestine Polytechnic University led by Dr Musa Alrefaya since 2016. As a result of its successful experience in training school students by holding summer camps in the field of competitive programming, A special team of trainers from Palestine and the Arab countries in the Palestinian Collegiate Programming Contest for Universities (PCPC) are training the participants in the International Olympiad in Informatics.

Computational problem-solving initiatives began in Palestine in 2012, with the inaugural Palestinian Collegiate Programming Contest (PCPC) held at Palestine Polytechnic University (PPU). The event drew participation from 10 universities and 81 students from the West Bank. The following year saw the launch of the first training summer school at PPU, aimed at nurturing talent in problem-solving skills. Out of over 1000 students, 65 were selected to participate in this intensive program.

As the years progressed, the scope of problem-solving training widened, with more students engaging in various educational activities. Multiple summer schools were organised across different universities and affiliated associations, providing opportunities for students to enhance their problem-solving abilities and computational thinking skills.

Recognising the importance of fostering programming proficiency among students, the Ministry of Education introduced a local competition called the Student Coding Initiative. This competition encouraged students to explore computer programming and computational concepts. The Student Coding competition witnessed significant participation, attracting hundreds of students eager to expand their programming knowledge and showcase their skills.

In 2020, Samed AlHajajla, a former participant at the IOI and the deputy leader of the Palestinian team, founded Meshka, a social startup on a mission to enhance the computational problem-solving skills of K12 students through computer science, math, science, and design thinking. Since 2020, Meshka has trained thousands of K12 students in computational problem-solving, successfully fulfilling its goal of improving students' problem-solving abilities.

The landscape of problem-solving education in Palestine has evolved significantly since its inception in 2012. Initiatives such as the PCPC, Meshka's computational problem-solving programs, training summer schools, and the Student Coding competition have empowered Palestinian students to develop their problem-solving abilities and excel in computer science. This steady growth and progress inspires us to continue our efforts and strive for significant achievements.

The inception of summer schools in 2013 at Palestine Polytechnic University marked a significant milestone in advancing computer science education in Palestine. These summer schools aimed to enhance students' informatics and computational thinking skills. The selection of trainers for these schools was meticulously done through logical exams conducted by the Ministry of Education department in each city, ensuring the quality of education imparted to the students.

The levels of participating students have consistently improved, reflecting the effectiveness of the summer school programs. Additionally, more universities have joined in training students through the International Collegiate Programming Contest (ICPC) community, contributing to a broader reach and impact.

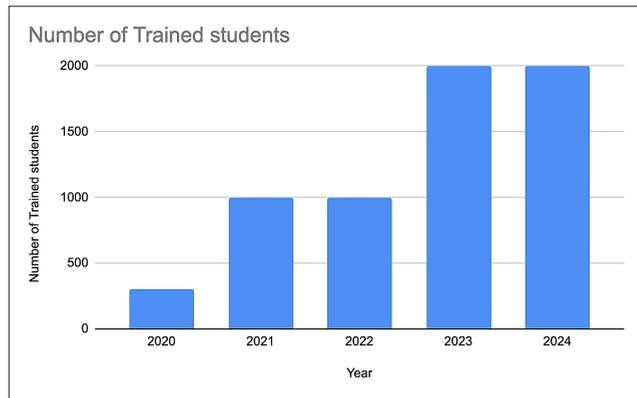
On average, each summer school at every university attracted between 60 and 100 students, indicating a growing interest and demand for such educational initiatives. Simultaneously, efforts to raise awareness among schools and students continued, promoting the importance of computer science education.

The Palestinian Olympiad in Informatics team has partnered with Meshka to develop advanced computational problem-solving programs for K12 students. One of these pro-

grams is Solve for Palestine, which offers three levels: beginner, intermediate, and advanced. All students are welcome to participate at the beginner level and will go through an online learning experience based on peer-to-peer learning. Facilitators are also available to provide group office hours and assist with projects and exercises every week. After one month, students who pass the final exam will be selected to move on to the intermediate level. Here, they will be exposed to IOI-style problems, which will ultimately help them develop the skills necessary to solve IOI problems. After that, in partnership with Code.X, advanced training is tailored to each selected student to prepare them for the national and international informatics competition.

The following table shows the number of students trained in a joint effort of the Palestinian Olympiad in Informatics and Meshka:

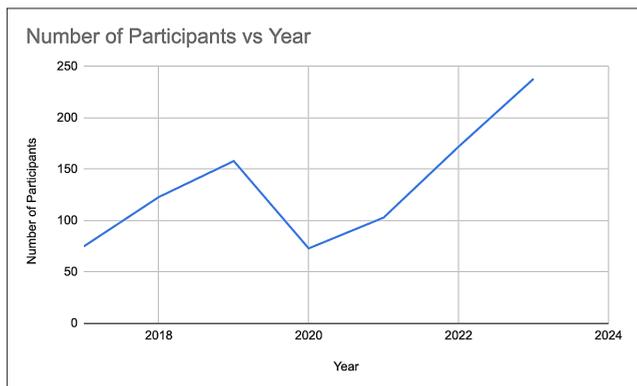
Year	Number of Trained students
2020	300
2021	1000
2022	1000
2023	2000
2024	2000



In 2020, the number of students interested in computational problem-solving has increased significantly. This can be attributed to the growing awareness and advocacy for the importance of technology and learning coding. Students who have joined this program have had access to mre opportunities to compete, win competitions, receive scholarships, gain outstanding recognition, and further their learning. This has consequently encouraged more students to develop an interest in this field of study.

The following table shows the number of participants in the national IOI competition during previous years:

Year	Number of Participants
2017	75
2018	123
2019	158
2020	73
2021	103
2022	172
2023	238
2024	Expected number (300–350)



Through the analysis of data collected on the number of participants in the annual competition from 2017 to 2023, with forecasts for 2024, we can observe significant developments and essential changes in the interest to participate. The competition began in 2017 with 75 participants and saw a noticeable increase in numbers in the following years, reaching 123 participants in 2018 and then rising to 158 in 2019; this is due to the ongoing work of the Palestinian Olympiad in informatics to advocate and raise awareness about informatics in Palestine.

The year 2020 witnessed a sharp decline in numbers, likely due to the impacts of the COVID-19 pandemic, where the number of participants fell to 73. However, the numbers began to recover gradually in 2021, with 103 participants and continued to rise strongly in the following two years, with 172 participants in 2022 and 238 in 2023. For 2024, organisers expect this increase to continue, with estimates ranging from 300 to 350 participants.

These figures reflect the success of the organisers' strategies in enhancing the event and increasing its appeal, as well as the sector's recovery from external shocks such as the pandemic. The expected growth for 2024 shows the organisers' confidence in continuity and expansion, indicating that the competition has established itself as a significant event that attracts increasing interest each year.

Preparation for International IOI

To effectively enhance and expand participation in the International Olympiad in Informatics (IOI), strategic steps include developing early programming skills in school curricula, strengthening training through advanced programs and mentorship, broadening participation via inclusive policies and regional contests, leveraging technology for broader engagement, promoting informatics education through awareness campaigns, continuously evaluating and adapting strategies, and building a supportive community for sustained excellence. These efforts aim to increase the quality of individual performances and the scope of international collaboration and innovation in informatics.

The journey to international IOI began with a nationwide selection process, culminating in identifying Palestine's most talented young programmers. Four students were chosen to represent our nation through competitive exams and performance evaluations.

The national team comprised four exceptional students, each with a proven track record in national programming contests and a deep passion for computer science.

The training process for students to participate in the competition begins through a strategic partnership with the Palestinian Olympiad in Informatics, Meshka, and the Ministry of Education, where agreements are made to identify and select outstanding students from various schools. This selection is based on specific criteria that ensure the choice of students who are most capable and prepared for competition. After selection, these students are enrolled in intensive technological summer camps designed

to enhance their programming and analytical skills. In these camps, students receive training from specialists and experts in programming and technology, where they are guided and trained on the latest methods and techniques that enable them to excel in international competitions.

The expertise of graduates from this competition and the PCPC programming contest was utilised to enhance the student's skills and prepare them effectively for the competition. These alumni, who possess extensive practical experience and a deep understanding of competition challenges, provided training and guidance to new participants. The training sessions were designed to cover a wide range of technical topics and necessary skills, which increased the students' efficiency and improved their ability to compete at a high level in these contests. Thanks to this approach, the competition created a dynamic educational environment that continuously develops students' skills and equips them with the tools needed for success.

This program focused on advanced algorithms, data structures, and problem-solving strategies essential for international success.

Participation in IOI

Experience

The Palestinian team engaged fully in the IOI experience, from the intellectually challenging competition to cultural exchange activities. This exposure to the global informatics community was invaluable, providing our students with new perspectives and inspiration.

Performance and Achievements

Our team performed admirably, with each member demonstrating exceptional skill and determination. While we did not secure a medal this year, the team ranked competitively, and their achievements have made our nation proud.

In 2017, Palestine debuted in the International Olympiad in Informatics (IOI) held in Iran. This was a significant milestone for the Palestinian informatics community. The breakthrough came in 2022 when Nicola Abu Saad secured Palestine's first medal in the competition, bringing recognition and pride to the nation.

Since 2020, independent training sessions have been conducted through a Meshka dedicated to nurturing young talent in informatics. Over the last two years, the Ministry of Telecom and Information Technology has stepped in to support and sponsor training sessions, further bolstering the country's participation in international competitions.

Outcomes and Impact

Learning and Development

IOI experience has significantly contributed to our participants' personal and academic growth, equipping them with skills far beyond the competition. Students who represent Palestine in the IOI study with scholarships in university, win every programming competition in Palestine, have opportunities to study at Ivy League schools, and intern at big tech companies such as Facebook and Google.

Future Participation

Reflecting on our IOI journey, we recognise areas for improvement, particularly in our training program and resource allocation. These insights will be invaluable in enhancing our future performance.

National Impact

Participation in IOI has stimulated increased interest in computer science across Palestine, encouraging more students to pursue excellence in this field. It has also highlighted the importance of supporting STEM education at a national level.

Challenges and Recommendations

Despite logistical and financial challenges, our team's resilience and dedication were unwavering. We recommend establishing robust support structures for future teams, including securing long-term funding and expanding our training resources.

Despite the progress made, several challenges persist. One major obstacle is the difficulty for students in attending on-site training sessions due to obstacles and checkpoints between cities. Financial constraints remain a significant challenge, as most training activities rely heavily on volunteer efforts.

Another formidable challenge is the inclusion of students from Gaza, given the restrictions on movement. However, a breakthrough occurred last year with the participation of a contestant from Gaza in IOI 2023, marking a significant step towards inclusivity.

Another challenge for many students is the language barrier, particularly the adoption of English in training sessions. Efforts to overcome this hurdle are ongoing but require sustained support and resources.

This year, students from Gaza faced significant obstacles that directly affected their participation in the competition due to the complex political situation in the region. The frequent closures of schools and universities due to tensions and security incidents have severely limited their opportunities for continuous education and proper preparation for international competitions. Additionally, difficulties in internet communication, wheth-

er due to disruptions or technological restrictions, posed a significant barrier to their effective communication with organisers and other participants. Together, these factors presented a tremendous challenge for the students of Gaza, who strive to demonstrate their capabilities and achieve excellence despite the overwhelming circumstances surrounding them.

Future Endeavors

Despite the challenges, there is optimism for the future. Initiatives like Bebras, introduced recently, hold promise for further engaging students and fostering interest in informatics. Additionally, ongoing efforts to overcome logistical and financial barriers will pave the way for greater participation and success in international competitions.

Conclusion

The Palestinian team's journey at the IOI was one of learning, growth, and inspiration. We extend our deepest gratitude to everyone who supported this endeavour. We are committed to building on this experience, nurturing our young talents, and aiming for even more outstanding achievements in future IOIs.

The Palestinian Olympiad in Informatics has come a long way since its inception, thanks to the dedication of educators, volunteers, and supporters. While challenges persist, the achievements and progress made serve as a testament to the resilience and determination of the Palestinian informatics community. With continued support and collaboration, Palestine is poised to make even more significant strides in the field of informatics and excel on the international stage.



M. Alrefaya – Dean of Dual Studies Deanship since 2021 and has been a full-time senior lecturer at the College of Information Technology and Computer Engineering college-Palestine Polytechnic University in Palestine since 2004. He was a researcher at the Electronic and Informatics Department at the Vrije University of Brussels from 2007 to 2015. Alrefaya has many publications in medical image processing. He is the director and founder of the Palestinian Olympiad in Informatics, Palestinian Bebras community, and the Palestinian Collegiate Programming Contest (PCPC).



S. AlHajjla – Social Entrepreneur, Software Engineer, and Social Service Designer, he is the CEO and Co-Founder of Meshka, a platform working on integrating problem-solving skills in K12 education. Director and Founder of the Palestinian Mathematical Olympiad, official representative of Palestine at the International Mathematical Olympiad, deputy leader and coach of the Palestinian team at the International Olympiad in Informatics.

High School Programming Olympiads in Gomel Region

Michael DOLINSKY

*Faculty of Mathematics and Technologies of Programming, F. Skorina Gomel State University
Sovetskaya str., 104, Gomel. 246019. Republic of Belarus
e-mail: dolinsky@gsu.by*

Abstract. This article describes the content of programming competitions for students in grades 9–11 of the Gomel region. A general idea of the thematic content of the tasks and examples of tasks by topic are given. The methodology of teaching and preparing schoolchildren for such Olympiads is also briefly described. A serious technical basis is the instrumental system of distance learning developed under the supervision of the author (<http://dl.gsu.by>). The paper presents very good results of Gomel schoolchildren in International Olympiad in Informatics. The main contribution of the paper is share our successful long-term experience in preparing schoolchildren for Olympiads in informatics.

Keywords: programming Olympiads, high school, instrumental system of distance learning.

1. Introduction

In many cases programming starts in elementary school (Dagienė *et al.*, 2019). It then continues into high school in a variety of ways: it could be unplugged learning (Plugar, 2021; van der Vegt, 2016), game learning (Combefis *et al.*, 2016), using Scratch (Fagerlund *et al.*, 2020), the use of specialized software development environments (Kabátová *et al.*, 2016; Tsvetkova *et al.*, 2021; Alemany *et al.*, 2016), robot programming (Kanemune *et al.*, 2017; Panskyi *et al.*, 2021).

Since September 1996, on the basis of secondary school 27 in Gomel, and in September 1999, additionally and on the basis of the distance learning site DL.GSU.BY (hereinafter referred to as DL), work is being carried out on the optional study of computer science and programming for schoolchildren of different ages (Dolinsky, 2016). The key feature of this training is the early start of programming education – actually from the 1st grade, and in some cases from kindergarten (Dolinsky, 2018). For such students, special programming Olympiads are held in order to increase motivation for classes, as well as for the early acquisition of competitive experience. Problems for Olympiad in programming for 1–4 grades students of primary school are described at (Dolinsky, 2022).

Problems for programming Olympiads for students in grades 5–8 of secondary school are described at (Dolinsky, 2023). Verification of solutions is carried out automatically on the DL.GSU.BY website (Dolinsky, 2017). This article offers materials for programming Olympiads and a brief description of teaching programming and preparing for such Olympiads for students in grades 9–11 of high school.

Training includes a consistent study of the necessary information and solidifying them by solving the proposed problems from the systematically collected problems of Olympiads of past years. Solutions are checked automatically on the DL.GSU.BY website. Note that all the Olympiad tasks of the current academic year are included in the systematic training immediately after the last Olympiad. We also note how ideas for new problems appear. On the one hand, we focus on the IOI-curriculum (IOI Syllabus, 2023), and on the other hand, every year we solve the problems of the USACO, COCI, St. Petersburg individual and team Olympiads. These tasks (or their subtasks) that serve as a source for the tasks of our Olympiads in subsequent years. It is necessary to note the inextricable connection between regional Olympiads and training. Olympiads are held to show students and teachers what topics need to be studied in order to successfully participate in Olympiads. On the other hand, all the tasks of the Olympiads upon their completion are naturally integrated into the education system. We also note that the content of Olympiads and training is expanding in accordance with the development trends of Olympiads in the world, focusing on new theoretical topics offered at the Olympiads IOI, USACO, COCI, Russian and Belarussian Olympiads in informatics and programming. When asked why we don't use many other resources in preparation for the Olympiads, we answer this way:

1. The DL.GSU.BY website has been developing since 1999, when many of the currently existing successful resources did not even exist, and we were forced to pave “our own path.” As such, the author began teaching programming to schoolchildren, as well as the development and accumulation of teaching methods in the 80s of the last century.
2. We have our own teaching methodology, and we teach text-based programming, starting from elementary school (and even from kindergarten).
3. We use ideas and even tasks from other resources, but we study them in the sequence we build for studying theoretical and practical material.
4. Our best schoolchildren actively participate in Olympiads held on third-party sites, both in real time and in the mode of virtual contests and after-Olympiads solving.

2. Content of the Olympiads

Tasks for grades 9–11 include 15 tasks in ascending order of complexity (each student is invited to solve all these tasks). The first 10 tasks (the same with the corresponding Olympiad for grades 5–8) on the following topics:

1. Introduction to programming.
2. One-dimensional array.

3. Two-dimensional array.
4. Geometry.
5. Strings.
6. Sorting.
7. Text task.
8. Elements of number theory.
9. Greedy.
10. Queue.

Their purpose and content are described in more detail in (Dolinsky, 2023).

Additionally, in the Olympiads of grades 9–11, 5 more tasks are given on the following topics:

1. Recursion.
2. Dynamic programming and recurrence relations.
3. Graphs.
4. Complex data structures.
5. Complex dynamic programming.

Let us describe in more detail the topics of these tasks. In most cases the condition of the task is formulated as concisely as possible, without a legend (description of the task with many sentences, that have not importance for the task). This is done in order to determine whether the student knows the corresponding algorithm and is able to write and debug a program for it or not. For every theme 11–15 theme example of theme task is given to show difficulty level of the tasks. The list of subtopics studied in each topic directly indicates the composition and order of the theory being studied. The subtopics are arranged in order of increasing difficulty, as are the tasks in each subtopic. This is what, in the opinion of the author of the article, every participant in the Olympiads needs to know. Every year the list of subtopics and tasks in them is replenished. There is no need to prove why studying such subtopics and in this order is better than other constructions. Each reader, like the author, can have his own opinion on this matter.

Topic 11. "Recursion"

Topic "Recursion" includes tasks in which a recursive call to a procedure or function will be required to solve the problem. Currently, the "Recursion" topic contains tasks for the following subtopics. Set of all subsets: derivation of one of the ways to sum M , derivation of all ways to sum M , number of ways to sum M , number of ways to sum at least M , maximum sum not greater than M , subset with maximum number of matching elements, forbidden subsets, the sum of K subsets. Combinations: quantity, output. Accommodations. Permutations. Permutations with repetitions. Bracket expressions. Gray code. Fast exponentiation. Number generation. By definition. All subsets of rows of a two-dimensional array. On a two-dimensional array. Divide and conquer. Recursion with memoization. Recursion with memoization by profile.

An example of a task on the topic “Recursion”:

Problem “Decomposition into a sum of different”

Count the number of different representations of a given natural number N as a sum of at least two pairwise distinct positive terms. Print out all of these.

Input Format

number N ($1 \leq N \leq 10$)

Output Format

all possible representations of the number N and the number of such representations.

Note: the output should be carried out in descending order of the terms (see the example of input – output).

Input example:

7

Sample output:

6 + 1

5 + 2

4 + 3

4 + 2 + 1

4

Topic 12. “Dynamic Programming and Recurrence Relations”

Topic “Dynamic Programming and Recurrence Relations” includes subtopics: one-dimensional array: all sums, maximum length subsequence, number of maximum length subsequences, maximum length subsequence in $O(N \cdot \log N)$, knapsack, sum of several previous elements, sum of several previous elements with recovery response, maximum of sums, splitting into subarrays, prefix sums, prefix maximums, suffix minimums, permutation-number; two-dimensional array: sum of several previous elements, maximum of several previous elements, minimum sum of several previous elements, sum of maximums of several previous elements, maximum frame, prefix sums.

An example of a task on the topic “Dynamic programming and recurrence relations”:

Problem “Fibo-other numbers”

Fibo-other numbers are built according to the formula:

$$f(0) = 1;$$

$$f(1) = 1;$$

...

$$f(n) = f(n - 1) + f(n - 2) + g(n),$$

where $g(n)$ is the number of digits in the number n .

You need to write a program that determines the number of Fibo-other numbers in a given numerical interval.

Input format:

The input consists of two numbers A and B that define the boundaries of the segment.

Output Format:

Print one number – the number of Fibon-ther numbers in the interval [A, B].

Limits: $1 \leq A \leq B \leq 1,000,000$

Input example Output example

2 10

3

Topic 13. “Graphs”

Topic “Graphs” includes tasks in which it is required to perform the analysis and processing of graphs and contains the following subtopics. Vertices enumeration: vertex degree, adjacency matrix. Queue: bipartite graph check, shortest paths, articulation points. Dijkstra’s and Floyd’s algorithms. Recursion: path with maximum number of edges, reachability matrix, sources and sinks, vertex reachability, unreachable vertices, connectivity, connected components, strongly connected components, dominant sets, cycle search, Euler cycle, Hamilton cycle, directed graph path, topological sort, maximum matching, Kuhn’s algorithm. Tree definition, number of edges per vertex, tree diameter, least common ancestor, vertex visit order by depth-first search, centroid decomposition, Huffman character coding, binary tree, quaternary tree. Minimum spanning tree: Prim’s and Kruskal’s algorithms. Disjoint set units. Strategic games. Hidden graphs. Euler formula. Maximum flow.

An example of a task on the topic “Graphs”:

Problem “Diameter”

Given an undirected graph G with n vertices and $n - 1$ edges. This graph is connected and each edge has a non-negative integer length. Let $d(x, y)$ be the length of the shortest path between vertices x and y in graph G. The diameter of graph G is defined as the maximum of all possible distances $d(x, y)$, where x and y are two arbitrary vertices of graph G. Write a program that which calculates the diameter of the graph G.

Input format:

Your program must read input from standard input. The first line contains a number ($0 < n < 1000$). The vertices of the graph are numbered from 1 to n. Each of the following $n - 1$ lines describes one edge: the first two numbers are the numbers of the vertices connected by this edge, and the third number is the length of this edge. The length of any edge is an integer less than 1000.

Output Format:

The single line of the standard input must contain the diameter of the graph.

Input example Output example

10

15

4 5 5

4 3 2

4 2 1

5 6 4
 5 1 0
 5 7 4
 3 8 4
 3 9 3
 3 10 3

Topic 14. “Complex data structures”

Topic “Complex data structures” includes tasks for which solution it is required to have the corresponding theoretical knowledge on the following topics. Segment tree: no modification (max, min, sum, max sums); single assignment (changing of one element of array): sum, minimum, maximum, minimum segment where there are all numbers from 1 to K); single increment (sum); increment on segment directly, increment on segment lazy propagation (sum, minimum, maximum, number of positives, element access); assignment on a segment (sum, number of segments from units). Fenwick tree. Trie. Bit trie. Search tree. Suffix array.

An example of a task on the topic “Complex data structures”:

Problem “Tree of maximums”

Implement a data structure that stores information about N ($1 \leq N \leq 100000$) integers A_1, \dots, A_N . The structure must support the following operations.

- (1) INIT(N) Initialization with the number N . In this case, each number A_i is assigned the value 0.
- (2) MODIFY(L, R, value) For each $i, L \leq i \leq R$, change A_i to $A_i + \text{value}$.
- (3) FINDMAX(L, R) Output to the output file the maximum $\max\{A[L], A[L + 1], \dots, A[R]\}$.

Input Format

The input file contains no more than 100,000 operations. Each operation is described on a separate line. The operation description starts with an integer from 1 to 3 – it is number from the list above. The operation parameters follow in the order they are listed in parentheses. The numbers on each line are separated by spaces.

All operations are correct. It means that:

- The INIT operation is the very first operation in the input file and does not appear anywhere else in it.
- For the MODIFY operation, the constraints $1 \leq L \leq R \leq N$ and $-10000 \leq \text{value} \leq 10000$.
- The FINDMAX operation satisfies the constraints $1 \leq L \leq R \leq N$.

Output Format

Perform the operations in the order they are listed in the input file. If you need to output some information to the output file to perform the operation, then output this information. Write the output for each operation on a separate line.

Input example	Output example
15	0
2 1 1 -6	0
3 2 4	0
3 1 2	0
3 1 3	-4
3 1 5	-4
2 2 5 -4	-8
2 4 5 -4	-8
3 1 2	
3 2 5	
2 1 3 -4	
3 4 5	
3 5 5	
2 1 1 -10	
2 1 3 3	

Topic 15. «Complex dynamic programming»

Topic «Complex dynamic programming» includes tasks that require knowledge of the relevant theory and the ability to come up with a way to apply it to the following subtopics: Bitmask DP, DP on tree, Binary Lifting, DP on profile, DP on strings, DP on numbers, DP on bit numbers, the include-exclude method.

An example of a task on the topic “Complex dynamic programming”:

Problem “KT-number 2”

Numbers K and T are given. You need to find out how many T-digit numbers exist that do not contain K successive odd digits (these are the numbers 1, 3, 5, 7, 9).

Input format:

K, T, M – three numbers separated by a space.

Restrictions:

$1 \leq K; T \leq 30; 10 \leq M \leq 1000000$.

Output Format:

The number of T-digit numbers that do not contain K consecutive odd digits, taken modulo M (the remainder of dividing the desired number by M is implied).

Input example:

2 3 100

Output example:

50

Systematic and purposeful preparation of regional Olympiads is an important means of developing the Olympiad movement in the region. Regional Olympiads are held in the Gomel region five times a year: in October–November, school and city grades 1–11, and in March–April, school, city and regional (zonal) for students in grades 1–9. When conducting these Olympiads, Internet technologies and the DL.GSU.BY website are used, which allows not only schoolchildren from the Gomel region, but also everyone

to participate in all the Olympiads. And, it should be noted, there are dozens of such people from all regions of Belarus and Minsk.

3. Training and Motivation System

It is important to note that, despite the focus on programming, training is essentially developing in nature and therefore it is very useful both for those who later choose information technology as their professional field, and for everyone who will be engaged in at least some time.

Practice also shows that training is built in a rather interesting form. All classes are conducted only on a voluntary basis during extracurricular time.

Another equally important aspect is a differentiated approach. The use of Internet technologies makes it possible to provide individual training along a personal educational trajectory. If the student is unable to solve the problem, he is consulted by other students or the teacher. Face-to-face classes are held on Wednesdays and Sundays on the basis of the computer science cabinet of secondary school 27 in Gomel.

In addition, weekly from Thursday 8.00 to Wednesday 20.00, one of the regional Olympiads that took place earlier in 2010–2023 opens for solution, solving which (at a convenient time for himself) each student can check how well he knows the material he has studied, as well as what other topics to be studied. The teacher receives similar information about each of his students.

All of the above-described educational and competitive work is carried out on the distance learning website (Dolinsky, 2016). There, access to theory is provided, assignments are given; solutions are checked. All tests except the last one are given for educational purposes; all kinds of tables of results and ratings are built; using the forum, interactive interactions between students and each other and the teacher are supported.

To motivate the best students to become more active, they are encouraged to participate in competitions on the platform (Codeforces, 2010–2023), where they can compete with thousands of schoolchildren from all over the world. In addition, the Codeforces platform automatically recalculates the ratings of all competition participants.

In order for schoolchildren of the Gomel region to understand their level of preparation relative to their opponents, we have developed and continue to develop a website (Ratings at Codeforces of Gomel region schoolchildren, 2023). The best 25 students from Gomel are nominated to participate in the regional Olympiad. The best 25 students of the Gomel region are competing to win diplomas at the regional Olympiad. The best 15 in the Gomel region are competing to be included in the Gomel region team for the Belarusian Republican Olympiad in Informatics, the best of 11 of them are to win diplomas of the Republican Olympiad. Since 2018, schoolchildren in the Gomel region have taken 11 or more diplomas from the Republican Olympiad for 15 participants (Results of diplomas of Belarus regions, 1997–2021).

In order to make it more convenient for teachers and students to track the results of competitive and educational work of schoolchildren in the Gomel region on Codeforces (and in the future on other similar sites), we began to develop a website (Results of testing at external resources, 2023).

4. Results On International Competitions

Such a good system of training and motivation bears fruit. For the period from 1997 (when the Gomel schoolchild Kuzniatsou Artsiom first entered the IOI and won a silver medal there) to 2021 (in 2022 and 2023 Belarusian schoolchildren competed without indicating the country they represent), Belarusian schoolchildren won 88 medals, 34 of which were won by Gomel schoolchildren, the results for the regions of Belarus are presented in Table 1. (Results of Belarus regions in IOI, 1997–2021).

Twenty Gomel region schoolchildren win the medals of IOI during 1997–2021 (Results of private achievements of Gomel region schoolchildren, 1997–2021).

Gomel schoolboy, 8th grade student Mikhail Brel won a bronze medal at IOI 2023 (Results of Mikhail Brel at IOI, 2023).

The Gomel region team won a silver medal (absolute 6th place) in the twenty-fourth open All-Russian team Olympiad for schoolchildren in programming (Results of the twenty-fourth open All-Russian team Olympiad for schoolchildren in programming, 2023).

In 2016 and 2018, the author of the article became a diploma winner of the All-Russian competitions of scientific and practical works on methods of teaching computer science and informatization of education INFO-2016 and INFO-2018, held by the All-Russian Scientific and Methodological Society of Teachers and the Education and Informatics publishing house (Results XV All-Russian competition of scientific and practical works INFO (2016, 2018)).

Table 1
Results of Belarus regions in IOI (1997–2021)

Region	Total	Gold	Silver	Bronze
Gomel region	34	9	15	10
Minsk region	13	1	6	6
Vitebsk region	13	0	6	7
Lyceum BSU	12	1	4	7
Brest region	6	1	0	5
Minsk (capital of Belarus)	4	0	3	1
Grodno region	4	0	2	2
Mogilev region	2	1	1	0
Total	88	13	37	38

The respected reviewer requested a comparison of Belarus' results with other countries at the IOI. I consider this to be unlawful; the author does not teach all schoolchildren in Belarus. But in 2006–2009, the author was the coach of the Belarusian national team at IOI. And during that period, the Belarusian team won 14 medals for 16 participants (6 gold, 6 silver and 2 bronze), showing the sixth result during this period after China, Poland, Russia, USA and Taiwan (Results of countries on IOI, 2006–2009).

By the way, preparation for Olympiads in computer science and programming goes very far beyond the scope of the material studied in computer science classes in Belarusian schools.

5. Conclusion

This article presents the materials of Olympiads in programming for students 9–11 grades and briefly presents the methodology for teaching and preparing these students for such Olympiads. The Olympiad includes 15 problems on the topics: introduction to programming, one-dimensional array, two-dimensional array, geometry, strings, sorting, text problem, elements of number theory, greedy algorithm, queue, recursion, dynamic programming and recurrence relations, graphs, complex data structures, dynamic programming is hard. This approach allows us to provide the paradigm “each student will solve at least one problem”, “no student will solve all problems”, although in practice there are cases of violation of both rules. But in general, it turns out a balanced Olympiad for students with any level of training. The annual transfer of tasks from past Olympiads to the course “Olympiads 9–11” provides an opportunity for a systematic study of theory in preparation for subsequent Olympiads. And the weekly training Olympiads (based on the materials of the Olympiads of previous years) provide training in practical skills for solving problems at the olympiad and quality control of assimilation of the material. The paper also presented successes of Gomel region schoolchildren at international contests in informatics and programming.

References

- Alemaný, F.J., Vilahur, V.J. (2016). eSeeCode: Creating a Computer Language from Teaching Experiences. *Olympiads in Informatics*, 10, 3–18.
- Combéfis, S., Beresnevičius, G., Dagienė, V. (2016). Learning Programming through Games and Contests: Overview, Characterization and Discussion. *Olympiads in Informatics*, 10, 39–60
- Codeforces (2010–2023). <https://codeforces.com/>
- Dolinsky, M. (2013). An approach to teach introductory-level computer programming. *Olympiads in Informatics*, 7, 14–22.
- Dolinsky, M. (2014). Technology for the development of thinking of preschool children and primary school children. *Olympiads in Informatics*, 8, 63–68.
- Dolinsky, M. (2016). Gomel training school for Olympiads in Informatics. *Olympiads in Informatics*, 10, 237–247.

- Dolinsky, M. (2017). A new generation distance learning system for programming and Olympiads in Informatics. *Olympiads in Informatics*, 11, 29–39
- Dolinsky, M., Dolinskaya, M. (2018). How to start teaching programming at Primary School. *Olympiads in Informatics*, 12, 13–24.
- Dolinsky, M., Dolinskaya, M. (2019). Training in writing the simplest programs from early ages. *Olympiads in Informatics*, 13, 21–30.
- Dolinsky, M., Dolinskaya, M. (2020). The technology of differentiated instruction in text programming in Elementary School based on the website dl.gsu.by. *Olympiads in Informatics*, 14, 37–46.
- Dolinsky, M. (2022). Primary School Programming Olympiads in Gomel region (Belarus). *Olympiads in Informatics*, 16, 107–123.
- Dolinsky, M. (2023). Secondary School Programming Olympiads in Gomel Region (Belarus). *Olympiads in Informatics*, 17, 107–123.
- Fagerlund, J., Hakkinen, P., Vesisenano, M., Viiri, J. (2020). Assessing 4th Grade Students' Computational Thinking through Scratch Programming Projects. *Informatics in Education*, 19(4), 611–640. DOI: 10.15388/infedu.2020.27.
- Kabátová, M., Kala, I., Tomcsányiová, M. (2016). Programming in Slovak Primary Schools. *Olympiads in Informatics*, 10, 125–159.
- IOI Syllabus (2023). <https://ioinformatics.org/page/syllabus/12>
- Kanemune, S., Shirai, S., Tani, S. (2017). Informatics and programming education at Primary and Secondary Schools in Japan. *Olympiads in Informatics*, 11, 143–150.
- Panskyi, T., Rowinska, Z. (2021). A Holistic Digital Game-Based Learning Approach to Out-of-School Primary Programming Education. *Informatics in Education*, 20(2), 255–276. DOI: 10.15388/infedu.2021.12.
- Plugar, Z. (2021). Extending Computational Thinking Activities. *Olympiads in Informatics*, 15, 83–89.
- Pozdniakov, S., Dagienė, V. (eds) (2019). Informatics in schools. New ideas in school informatics. ISSEP 2019. *Lecture Notes in Computer Science*, vol 11913. Springer, Cham.
https://doi.org/10.1007/978-3-030-33759-9_7
- Ratings at Codeforces of Gomel region schoolchildren (2023). <https://dl.gsu.by/codeforces/>
- Results of Belarus regions (1997–2021). (In Russian).
<https://dl.gsu.by/olymp/result/ioi/region.asp>
- Results of diplomas of Belarus regions (1997–2021). <https://dl.gsu.by/olymp/rgomel.asp>
- Results of countries on IOI (2006–2009). <https://dl.gsu.by/olymp/result/ioi2006/>
- Results of Mikhail Brel at IOI (2023). <https://stats.ioinformatics.org/people/7971>
- Results of private achievement of Gomel region schoolchildren (1997–2021).
<https://dl.gsu.by/servlet/olympResultsPersonalMedal?c.id=1&u.c=25&lng=rus&r.id=3&a.r=3>
- Results of testing at external resources (2023). <https://dl.gsu.by/etr/>
- Results of the twenty-fourth open All-Russian team Olympiad for schoolchildren in programming (2023). (In Russian).
<https://neerc.ifmo.ru/school/archive/2023-2024/ru-olymp-team-russia-2023-standings.html>
- Results of XV All-Russian competition of scientific and practical works INFO-2018 (2018). (In Russian).
<https://infojournal.ru/competition/info-2018-result/>
- Results of XV All-Russian competition of scientific and practical works INFO-2016 (2016). (In Russian).
<https://infojournal.ru/competition/info-2016-result/>
- Tsvetkova, MS, Kiryukhin VM (2021). Algorithmic thinking and new digital literacy. *Olympiads in Informatics*, 15, 105–118.
- van der Vegt, W. (2016). Bridging the Gap Between Bebras and Olympiad; Experiences from the Netherlands. *Olympiads in Informatics*, 10, 223–230



M. Dolinsky is a lecturer in Gomel State University “Fr. Skoryna” from 1993. Since 1999 he is a leading developer of the educational site of the University (dl.gsu.by). Since 1997 he is heading preparation of the scholars in Gomel to participate in programming contests and Olympiad in informatics. He was a deputy leader of the team of Belarus for IOI’2006, IOI’2007, IOI’2008 and IOI’2009. His PhD is devoted to the tools for digital system design. His current research is in teaching Computer Science and Mathematics from early age.

omegaUp: A Decade of Growth and Impact in Latin American Coding Education

Hugo E. DUEÑAS OROZCO, Tania AVALOS PIÑÓN

omegaUp.org

Bellevue, Washington

e-mail: hugo@omegaup.org, vanessa@omegaup.org

Abstract. Ten years ago we presented in this journal omegaUp, an open-source contest management platform for the Mexican Olympiad in Informatics. Today, it has grown into a comprehensive education technology tool empowering tens of thousands of students in Latin America, from beginners to competitive programmers. In this article we discuss the main achievements in the past decade including free online courses, content quality assurance, improved UX, as well as becoming the host of important competitions such as the Iberoamerican Olympiad in Informatics and the national olympiads in Ecuador, Mexico, and Peru. .

Keywords: contest management system, omegaUp, cloud, user-experience, online judge, informatics education.

1. Introduction

omegaUp is an online platform that provides educational resources for competitive programming. It was originally developed as a contest management and training platform for the Mexican Olympiad in Informatics and other competitions in Latin America (Chávez *et al.*, 2014), but it has since grown into a comprehensive tool for teaching and learning computer science. This report provides an overview of the history and development of OmegaUp, as well as its current features and offerings.

2. omegaUp's Growth

Over the past decade omegaUp has seen tremendous growth as measured by many metrics that we explore in this section:

- omegaUp has over 221,000 registered users, over 9,000 public problems and has hosted over 11,000 contests as of January 2024.
- On a typical month omegaUp sees about 27,000 active users.

→ omegaUp has graded over 5 million submissions since its launch in 2012 and has seen sustained growth over the years as can be observed in Fig. 1.

At omegaUp we have 4 main types of user:

- Contestant – Those users who use omegaUp to train their competitive programming skills and/or to compete in programming competitions supported by the platform.
- School Student – Those users who use omegaUp for their school classes.
- Coach – Those users who coach students for programming competitions and use omegaUp for that purpose.
- School Teacher – Those users who are school teachers and use omegaUp to manage their classes.

Observations:

- 90% of our users are students, 10% are teachers.
- We have about 22.6 students per teacher in the school segment but only 5.8 contestants per coach in the competitive segment. Which is expected given that competitive programming groups tend to be smaller than school classes.

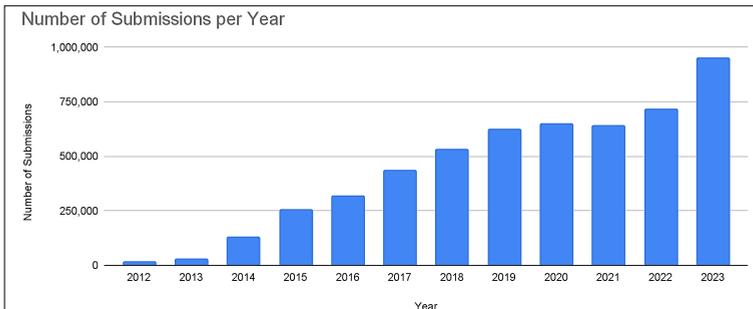


Fig. 1. Number of code submissions to omegaUp per year.

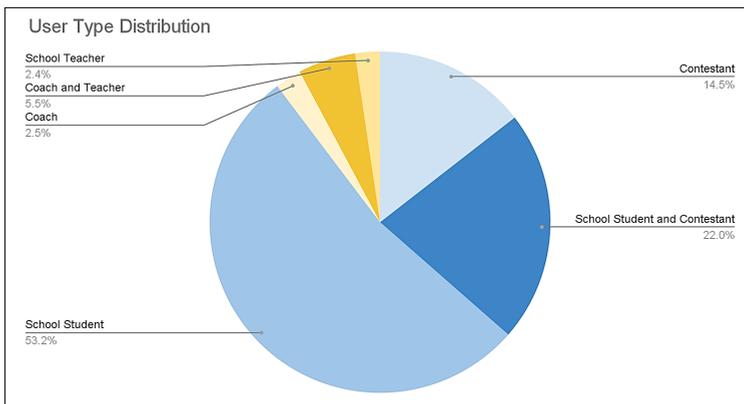


Fig. 2. Distribution of omegaUp users by type.

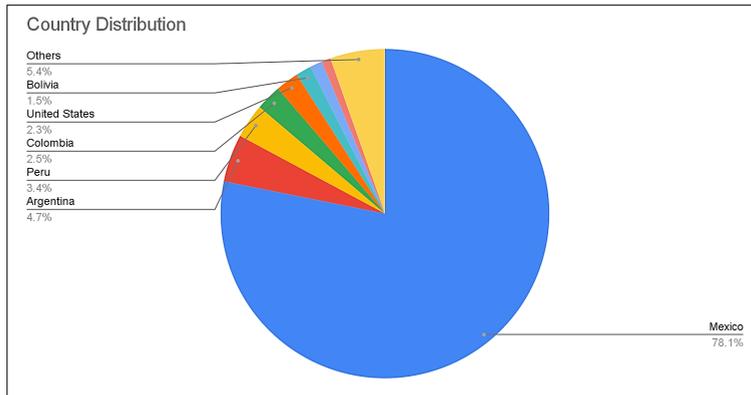


Fig. 3. Distribution of omegaUp users by country of residence.

- 70% of school teachers are also coaches. This is expected given that omegaUp started out as a competitive programming platform exclusively and that's how it gained its original user base.

According to Google Analytics, 78% of omegaUp's traffic comes from Mexico and the rest comes mostly from Latin American countries as shown in Fig. 3.

3. Enhancing User Experience

While the core online judging functionality was strong, omegaUp.com lacked user-friendliness. To address this and attract new users, we conducted in-depth research with our three main user groups: students, teachers, and competitive programmers.

Identifying User Needs

Students: Struggled to find active contests and suitable practice problems due to usability issues.

- Teachers: Required a user-friendly interface (UI) for ease of access and content creation, considering some might not be tech-savvy.
- Competitive Programmers: Sought a modern UI in line with current online judge standards.

Prioritising Resources

Given resource limitations, we focused on these key projects:

- UI Stack Migration: Before making user-facing changes, we upgraded the technical foundation of the UI. This involved migrating from Javascript to Typescript, Smarty to Vue.js, and integrated Bootstrap for a more robust framework.

→ Redesigning Core Components: After the technical migration, a complete redesign of the main platform pages commenced. Usability and user-friendliness for all age groups (students and teachers) were the top priorities.

Homepage Redesign

The landing page was restructured for intuitive navigation and the navbar was updated to reflect current features with clearer and more descriptive names.

Contest List Redesign

The previous contest list, displayed as a table with confusing tabs, only showed contest names. The new design addresses user needs:

- Identifying current and past contests.
- Start dates and filtering by time frame.
- Upcoming contest visibility.
- Additional details like organiser, participant count, scoreboard access, and contest mode.

Problem List Redesign

OmegaUp allows users to create coding problems in a specific format. Initially, these problems were displayed in a simple table. However, as the number of problems grew, this approach became unwieldy. Filtering and searching for specific problems proved increasingly difficult.

To address this, we implemented a two-pronged solution:

- Problem Classification: Discussed in detail in the “Problem Classification” section, this initiative involved classifying existing problems and establishing a semi-automated system for classifying new ones.

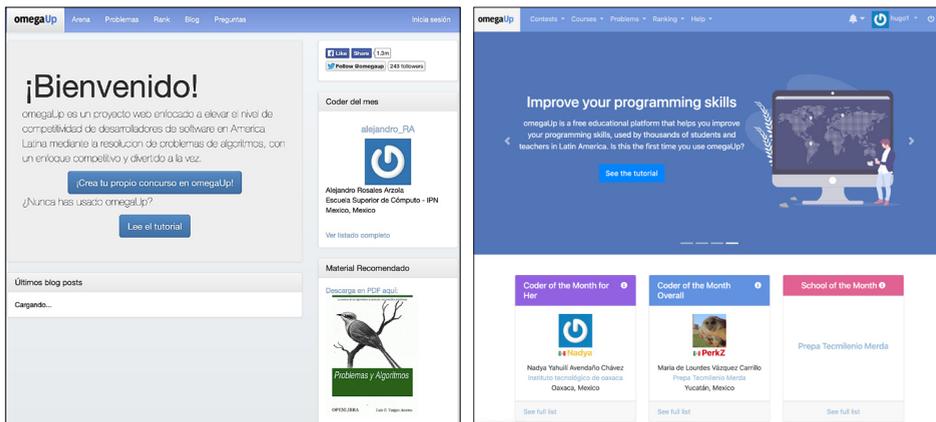


Fig. 4. omegaUp homepage in March 2019 (left) compared to January 2024 (right).

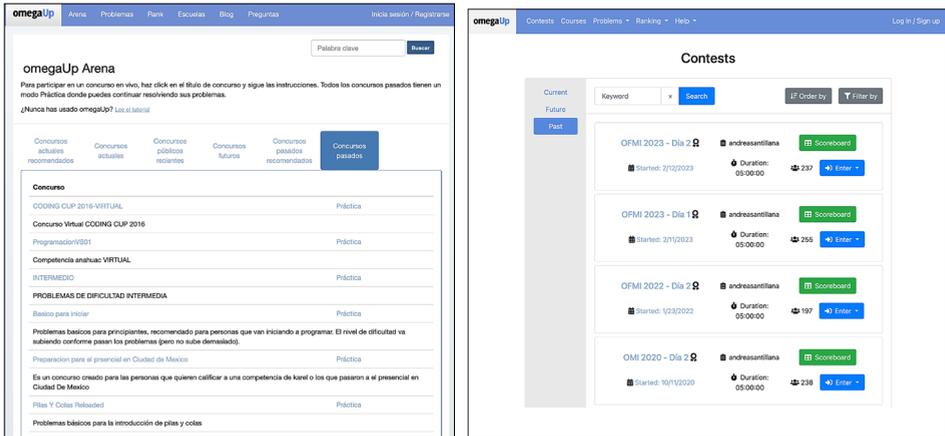


Fig. 5. omegaUp contest list in March 2019 (left) compared to January 2024 (right).

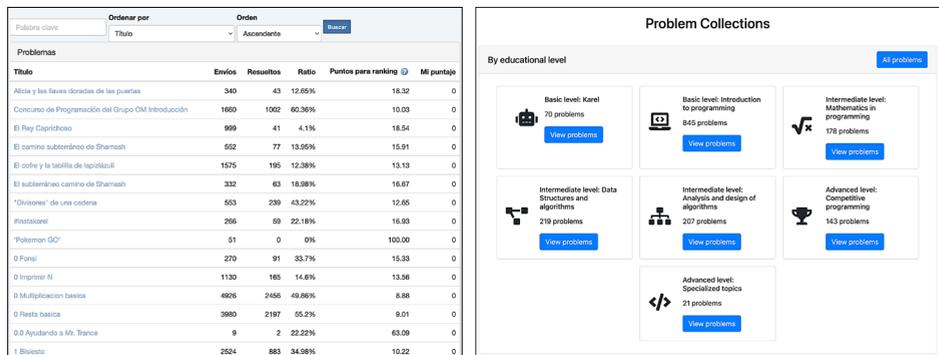


Fig. 6. omegaUp problem list in March 2019 (left) compared to January 2024 (right).

- Curated Collections: Instead of a single, overwhelming list, we now present problems in curated collections based on educational level and difficulty. This makes it easier for users to find problems that suit their needs.

Inclusive Language

A well-documented challenge in STEM education and careers is the lack of diversity (Verdugo-Castro *et al.*, 2022), (Card & Payne, 2020). The organisation recognizes that there are multiple ways to address this and create a more welcoming environment for all learners. One of the steps we followed was to review and modify the text used in our platform that can be interpreted as biased.

We undertook a comprehensive review process for the text used in the platform to identify text messages with gender bias. We identified about 150 such strings and started an effort to replace all of them with gender-free alternatives, which was a big challenge given that Spanish is a highly gendered language (Rosenblat, 1962), (Barrera Linares,

2019). An example string that we found frequently used was “profesor” (male teacher) when referring to any teacher, we replaced that with the gender-free alternative “docente”. Another example is “juez” (male judge), replaced with “jurado” (jury) which is gender-free.

Currently, 99% of existing platform text has been reviewed and revised to ensure gender neutrality, and all newly added texts are also required to adhere to these guidelines. In the future, we are also committed to expanding these efforts to include our Portuguese dictionary, ensuring a welcoming platform for all users regardless of language.

4. Problem Classification

In the past, our users struggled to find problems that matched their needs, both in difficulty and required knowledge. To address this challenge, we developed a robust system for problem classification:

- **Standardized Tags:**
 - Authors can assign relevant public tags to problems (e.g., “Binary Search,” “Shortest Paths,” “Dynamic Programming”).
- **Expert Review Process:**
 - A dedicated team of reviewers analyzes every public problem submitted to the platform.
 - Reviewers assess problem quality, promote suitable problems, and add/remove relevant tags for better searchability.
 - For problems requiring improvement, reviewers provide constructive feedback to the author.
- **Maintaining a Safe Platform:**
 - Reviewers also address reports of inappropriate content (offensive, spam, etc.).
 - If necessary, reviewers can mark problems as “banned” to ensure a positive user experience.

As of January 2024 we have 1678 promoted problems and have banned 137 problems due to being inappropriate.

5. Courses

Many omegaUp users who manage coding contests for students also leverage the platform for classroom instruction. Recognizing this need, we developed a comprehensive course management system with the following features:

- **Rich Content Delivery:**
 - **Lectures:** Create video-embedded lectures to explain course materials.
 - **Text and Multimedia:** Supplement lectures with text, images, and additional videos.

- Engaging Assessments:
 - Homework Assignments: Design problem sets with deadlines and point values.
 - Exams: Create timed exams to assess student understanding.
 - Automated Grading: Save time with automatic grading based on point values assigned to problems.
- Streamlined Collaboration:
 - Teaching Assistants: Assign TAs to address student questions and provide feedback.
 - Line-by-Line Code Reviews: Offer detailed feedback on student code submissions.
 - (Coming Soon) AI Teaching Assistant: We're developing an AI-powered teaching assistant to provide additional support for students.
- Prestigious Users:
 - Leading institutions like CIMAT, UAM, ITESM, and TecNM utilize omegaUp Courses.

Free MOOCs for Everyone

omegaUp, in collaboration with other organizations, offers a range of free Massive Open Online Courses (MOOCs) on the platform, including:

- Programming Languages: C++, Python, Java.
- Algorithmic Problem Solving: Introduction to Algorithms I & II.
- Olympiad Preparation: Mexican Olympiad in Informatics, Peruvian Olympiad in Informatics, Karel Programming (for beginners).

6. Important Contests Hosted

Over the past few years, omegaUp has become the go-to platform for managing coding contests in the Spanish-speaking world. Here are some of the high-profile events hosted on omegaUp:

- Regional Mexican Olympiads: Jalisco, Guanajuato, Veracruz, Nuevo Leon, Aguascalientes, and more.
- National and International Olympiads.
 - Mexican Olympiad in Informatics (OMI) since 2012.
 - Iberoamerican Olympiad in Informatics (CIIC) since 2015.
 - Peruvian Olympiad in Informatics since 2017.
 - Ecuadorian Olympiad in Informatics since 2020.
- Major Programming Competitions:
 - Coding Cup TecNM (flagship competition of the National Technological Institute of Mexico) since 2015. In 2019, it attracted over 500 teams of 3 students each.
 - Central American Programming Cup since 2020.

Acknowledgments

Special thanks to Juan Pablo Gómez and Carlos Abel Córdova for the many code contributions that they have done and all the interns that they have managed and mentored. Special thanks also to professor Rodrigo Castro for spearheading the efforts to classify and promote high quality content in the platform. Shout out also to the amazing interns that have delivered impactful contributions to the platform: Aarón, Alexia, Anmol, Eduardo, Ingrid, Karyme, Luis Abraham, Luis Alberto, Mauricio, Miguel, Mohit, Nicole, Omar, Óscar, Ruiz, Shivam, Vincent. Kudos to the omegaUp co-founders Luis Héctor Chávez, Alan González and Joe Ponce, none of omegaUp's impact could have been done without them.

References

- Chávez, L.H., González, A., & Ponce, J. (2014). omegaUp: Cloud-Based Contest Management System and Training Platform in the Mexican Olympiad in Informatics. *Olympiads in Informatics*, 8.
- Cepeda, A. and García, M. (2011). Mexican Olympiad in Informatics. *Olympiads in Informatics*, 5, 128–130. omegaUp codebase on github, <https://github.com/omegaup/omegaup>
- Verdugo-Castro, S., García-Holgado, A., & Sánchez-Gómez, M. C. (2022). The gender gap in higher STEM studies: A systematic literature review. *Heliyon*, 8(8), e10300. <https://doi.org/10.1016/j.heliyon.2022.e10300>
- Card, D., Payne, A.A. (2020). High school choices and the gender gap in STEM. *Economic Inquiry*, 59(1), 9–28. <https://doi.org/10.1111/ecin.12934>
- Gualtieri, M. (2009). Best practices in user experience (UX) design. Design compelling user experiences to wow your customers, 1(1), 1–17.
- Hartson, R., Pyla, P.S. (2018). *The UX Book: Agile UX Design for a Quality User Experience*. Morgan Kaufmann.
- Rosenblat, Á. (1962). Morfología del género en español: Comportamiento de las terminaciones -o, -a. *Nueva Revista de Filología Hispánica*, 16(1/2), 31–80. <http://www.jstor.org/stable/40297584>
- Barrera Linares, L. (2019). Gender/sex relationship and plural inclusive masculine in Spanish. *Literatura y lingüística*, 40, 327–354. <https://dx.doi.org/10.29344/0717621x.40.2070>



H.E Dueñas Orozco – participated in the IOI 2010. Has a bachelor's degree in computer science (2015) from Universidad de Guanajuato. Serves as engineering manager at omegaUp since 2019. Currently works as a Software Engineer at Google.



T. Avalos Piñon – has a bachelor's degree in mathematics from Universidad de Guanajuato (2018). Serves as engineering manager at omegaUp since 2019.

IOI Project Report on Improving TPS (Task Preparation System)

Kian MIRJALALI¹, Ali BEHJATI²

¹*Department of Computer Engineering, Sharif University of Technology, Tehran, Iran*

²*Douro Labs, Porto, Portugal*

e-mail: mirjalali@ce.sharif.edu, ali@dourolabs.xyz

Abstract. The Task Preparation System (TPS) is primarily designed for preparing IOI tasks. Initially developed and successfully utilized during IOI 2017, it has since been employed in various nationwide and international programming contests, such as IOI 2019~2024. Based on feedback received over the years, the tool required further development to enhance its usability, functionality, and maintainability. This article is the conclusion report of an IOI project defined for a specific set of improvements on TPS.

Keywords: IOI project, competitive programming, task preparation, Olympiad in Informatics, programming contest.

1. Introduction

The host technical and scientific committees of IOI 2017 developed several software tools during their preparations for the contest. One of these tools, the Task Preparation System (TPS), which was specifically created and utilized for preparing the contest tasks, received highly positive feedback from both the HSC and ISC members. The tool is publicly available on GitHub¹ under the MIT License² and has been widely used in numerous programming contests, including IOI 2019~2024, Iran's national IOI team selection contests, and the ICPC Regional contests (Tehran site). Due to the extensive use of TPS, we presented its application and shared insights with other members of the IOI community during a talk at the IOI conference in 2019, as well as by publishing an article in the IOI journal (Mirjalali *et al.*, 2019). We recommend reading that article first to become acquainted with the specific details and features of TPS.

¹ <https://github.com/ioi-2017/tps>

² <https://github.com/ioi-2017/tps/blob/master/LICENSE.txt>

TPS is a standalone collection of tools primarily written in Python and Bash scripts, specifically designed for preparing the tasks (also known as problems) in programming contests. Typically, TPS operates through a command-line interface and is employed in an offline environment. However, the task/contest directory is often shared with collaborators using version control systems like `git`. The process of preparing a high-quality contest problem is intricate and requires careful handling of multiple steps and components. Some of the key elements involved, but not limited to, include:

- Task statement, usually written in latex or markdown format.
- Designing function signature and IO format.
- Specifying the task constraints such as time/memory limits and restrictions on input values.
- Designing subtasks and their score.
- Graders: programs (written for each programming language allowed in the contest, such as C++ and Java) that link with the contestant's solution and provide it with the grading interface, say for reading the input and writing to the output.
- Task (public) attachment: the set of files provided to the contestants during the contest, such as sample test data, compilation scripts, and basic graders for local testing.
- Input generators and validators.
- Parameters for generating the test data.
- Assignment of test data to subtasks.
- Solutions; including correct, wrong, slow, and suboptimal solutions written for specific subtasks.
- Checker: a program that verifies the output of the contestant's solution per test case and specifies its score.

TPS, being a command-line interface, streamlines the preparation of programming problems by automating various error-prone tasks and effectively detecting errors and warnings. Fig. 1 provides an example of how TPS generates the test data for a task. Moreover, as an open-source project, TPS offers easy customization options, allowing users to tailor it to their specific requirements.

```

> tps gen
generator          compile[wARN]
solution           compile[OK]
validator          compile[OK]
0-01               gen[OK]      val[OK]     sol[OK]
1-01               gen[OK]      val[FAIL]   sol[OK]
1-02               gen[OK]      val[OK]     sol[OK]
2-01               gen[OK]      val[OK]     sol[OK]
2-02               gen[OK]      val[OK]     sol[OK]
3-01               gen[OK]      val[OK]     sol[OK]
3-02               gen[OK]      val[OK]     sol[OK]

Finished.

```

Fig. 1. An example of executing “tps gen”.

A sign of a software being alive and under usage is the flow of bug reports, suggestions, and feature requests. According to Lehman's laws of software evolution (Lehman *et al.*, 1997):

- A system must be continually adapted and its functional content must be continually increased, or else it becomes progressively less satisfactory over its lifetime.
- As a system evolves, its complexity increases unless work is done to reduce it.
- The quality of a system will decline unless it is rigorously maintained and adapted to operational environment changes.

As the original developers, we have voluntarily maintained TPS since 2017, finding it enjoyable and meaningful work. Based on user feedback, we recognized the need to enhance the usability of TPS's existing features, introduce new functionalities, and ensure its maintainability through code refactorings. However, accomplishing these tasks proved to be time-consuming, highlighting the necessity for dedicated resources to tackle more substantial improvements. After thorough consultations, we decided to propose this further development of TPS as a project supported by the IOI. Our project proposal was accepted, granting us the opportunity of making significant enhancements to TPS. This article serves as a report on the progress made during this IOI project. We will first provide a brief overview of the software state before the project commencement, and then, we will explain the improvements made throughout the project.

2. Software State before the Project

Before the start of this IOI project, TPS had already undergone several changes since 2017. The following is a summary of the improvements made during this period. Please refer to the GitHub history³ for more details.

- Recurring refactorings to keep the software maintainable.
- Resolved several reported bugs.
- Some updates on the offline markdown viewer tool⁴.
- Improvements and bug fixes for Windows users.
- Better error handling, including detection of missing generated tests.
- Improved the compilation process; detecting compile warnings and adding the option for showing verbose details.
- Added Python as a language for solutions.
- Generalized the scripts to handle output-only, two-step, and communication tasks out of the box (without the need to customize the scripts per task).
- Added the testing framework with more than 300 tests for the “tps” command.

³ <https://github.com/ioi-2017/tps/commits/master>

⁴ <https://github.com/ioi-2017/markdown-viewer>

- Added multiple configurable settings for tasks: grader name, [not] having checker, [not] availability of {C++, Java, Python, Pascal} as solution languages.
- Added the general “export” command to produce a package for importing into contest systems in order to reduce the manual work; specifically, added the exporter script for CMS.

3. TPS Improvements in the IOI Project

An initial list of technical tasks was created as a starting point for the project. However, as is typical in software projects, some tasks were removed from the list after conducting more thorough cost-benefit analyses, while new tasks were added due to circumstances. Although the initial estimation was around 500 hours, it ultimately required over 700 hours to complete all the tasks on the updated list. We now go through the major tasks that were accomplished as part of this project.

Improvements in Software Design and Behavior

Several improvements have been made in the TPS behavior, addressing bugs, handling user-induced errors, and enhancing the user interface. For instance, there are now more informative details available regarding the behavior of a solution when invoked against the provided test data. Furthermore, over 70 refactoring commits have been made throughout the project in order to improve the code quality and maintainability. These refactorings played a crucial role in keeping the codebase clean while developing other features.

Improving the Test Suites and Testing Infrastructure for the TPS Software

Having automated tests is crucial for achieving acceptable software delivery performance in terms of tempo and stability (Forsgren *et al.*, 2018). Without them, making software modifications can gradually become as challenging as walking through a minefield. Throughout this project, nearly 1500 tests have been incorporated in the TPS git repository, encompassing unit tests for common utility functions, tests on “tps” command itself, and tests on subcommands such as “tps gen” and “tps invoke”. Additionally, behavioral tests have been added for a modified version of the “testlib” header, tailored specifically for CMS and IOI tasks⁵.

In order to accomplish this goal, we made more than 45 commits dedicated to improving the testing infrastructure. These changes encompass a range of enhancements, including the addition of tools for probing the state of variables and files after the execution of a command. Fig. 2 provides usage examples to illustrate these improvements. In the first test of this example, it is expected that running the command “set_variable my_new_var "my new value"” will set the value of variable “my_new_var” to “my new value” without printing anything in the standard output/error streams. The second test states that executing the command “tps gen” (in a predefined environment)

⁵ <https://github.com/ioi-2017/tps/tree/master/extra-assets/testlib>

```

expect_exec -vs my_new_var "my new value" \
-oempty -eempty \
set_variable my_new_var "my new value"

expect_exec -f "tests" "td3/probed_files/0_tests" \
-o "td3/stdout" -eempty \
tps gen

```

Fig. 2. Examples of probing the state of variables and files in TPS tests.

shall print the contents of file “td3/stdout” in the standard output, and nothing in the standard error stream. Furthermore, it should create a directory with name “tests” and contents exactly matching the directory “td3/probed_files/0_tests”. Please refer to the testing documentation⁶ for more comprehensive details.

Completing/Updating the Documentation

The official documentation is now up-to-date, thoroughly explaining all features. Moreover, a brief technical documentation has been provided, covering internal code styles, patterns, and conventions. Separate comprehensive documentations have also been added for creating TPS task templates and for testing the TPS software itself. Furthermore, the “extra-assets” directory contains appropriate versions of Makefile, gitignore, and the “testlib” header specifically tailored for CMS and IOI tasks. These additions aim to provide users with the necessary resources and guidelines to effectively utilize TPS.

Easier Installation Process

An online installer has been successfully implemented and released for TPS. This installer is prominently introduced in the first-page README file of the project. Users can now easily install TPS by executing the following command. This streamlined installation process eliminates the need for manual cloning of the project from GitHub and running the installation script. As a result, the installation process for new TPS users is greatly simplified.

```

bash -c "$(curl -fsSL
https://raw.githubusercontent.com/ioi-2017/tps/master/online-installer/install.sh)"

```

Extending the Task Exporters

A task exporter is a script for transforming the data of a prepared task into a package with a predefined format suitable for importing into an online judge system. Using task exporters reduces the manual work and automates the error-prone process of adding problems to contest systems. A task exporter for CMS was already implemented in TPS before this IOI project. In order to enhance the usability of TPS, it is crucial to implement exporters for other online judge systems as well. In this regard, we have now implemented the

⁶ <https://github.com/ioi-2017/tps/blob/master/tests/README.md>

exporter for DOMjudge, the online contest system primarily used in ICPC. To use the exporter, users can run the command “`tps export DOMjudge`” in the directory of a prepared task. The exporter will then create an archive that can be uploaded to the administration system of DOMjudge. TPS architecture is designed to be flexible, allowing for easy addition of exporter scripts for other online judge systems in the future.

We have also introduced a second protocol for the existing task exporter for CMS. This new protocol aims to enhance the integration between TPS and CMS by providing increased configurability through the TPS directory structure. To export a prepared task for CMS, users shall now run the command “`tps export CMS <protocol-version>`” within the task directory, where the parameter “`<protocol-version>`” can be specified as either “1” for the old protocol, or “2” for the new protocol.

Adding the Command “stress”

This command is designed to subject a solution to stress testing. Specifically, it executes the solution against a series of randomly generated test cases with the aim of identifying a test case that causes the solution to fail, commonly known as being *hacked*. This tool is particularly helpful in the process of finding tests for distinguishing incorrect solutions. The stress testing procedure is conducted in a series of rounds, with the following steps being carried out in each round:

1. A “test case generation string” is randomly produced; we will later explain how this is done. This string is a single-line text similar to the test generation lines written in the file “`gen/data`”.
2. The test case input is generated from the test case generation string, with the same method as the process of generating the task test cases using “`tps gen`”.
3. The generated test case input is validated by the input validators.
4. The corresponding test case output is produced by the model solution.
5. The stressed solution is invoked with the generated test case as input. The score and verdict of the invocation is specified through a process similar to the command “`tps invoke`”.
6. The stressed solution is considered to be hacked by the generated test case if it does not get the required score.

A sample execution of the command is depicted in Fig. 3. Alongside the information displayed in the terminal output, the test case generation strings which expose faults in the solution, are also recorded in a separate file. This allows for further analysis and examination of the specific test cases that triggered the failure, or using them as the task test data.

The stress command gets two positional arguments. The first argument specifies the path of the solution file to be stressed. The second positional argument is one of the following:

- The path to a test case generation file; a python file which produces the test case generation strings. The python file shall implement a function “`gen_command()`” that returns a test case generation string upon each call. A sample test case generation file is shown in Fig. 4.

```

> tps stress "my-solution.cpp" "gen1 80 {random.randint(1, 70)} {ustr(8, 9)}"
test-gen-file      create[OK]
test-gen-file      verify[OK]
generator          compile[OK]
validator          compile[OK]
model solution     compile[OK]
stressed solution  compile[OK]
checker           compile[OK]
Round 1:
gen1 80 56 KjqdZ77ZT
gen[OK] val[OK] model[OK] stressed[OK] 0.016 check[OK] 1 [Correct]
Round 2:
gen1 80 7 4wjafMy_c
gen[OK] val[OK] model[OK] stressed[OK] 0.017 check[OK] 1 [Correct]
Round 3:
gen1 80 59 0d6Uo8i00
gen[OK] val[OK] model[OK] stressed[OK] 0.018 check[OK] 0 [Wrong Answer]
Hacked!
Round 4:
gen1 80 4 ISFYrwLk4
gen[OK] val[OK] model[OK] stressed[OK] 0.019 check[OK] 1 [Correct]
Round 5:
gen1 80 18 JClqyX58d
gen[OK] val[OK] model[OK] stressed[OK] 0.015 check[OK] 0 [Wrong Answer]
Hacked!
:

```

Fig. 3. A sample execution of “tps stress”.

```

from stress_test_gen_utils import *

def gen_command():
    return "gen1 80 {} {}".format(
        random.randint(1, 70),
        ustr(8, 9),
    )

```

Fig. 4. A sample test case generation file for “tps stress”.

- A test case generation format string; a general string used for producing test case generation strings. The string must be in the shape of a Python *format string* that produces a test case generation string upon each evaluation. Below is the test case generation format string equivalent to the sample test case generation file in Fig. 4.

```
"gen1 80 {random.randint(1, 70)} {ustr(8, 9)}"
```

The second positional argument of the stress command is interpreted as a test case generation file path if an ordinary file exists with the same path as that argument. Otherwise, it will be interpreted as a test case generation format string.

Adding the Command “init”

Creating the TPS directory structure manually for a new task can be error-prone and cumbersome. However, this process has now been automated with the introduction of the command “`tps init`”, which is similar to the widely known command “`git init`”. By executing “`tps init`”, users can simply initialize a new task directory based on a specified task template. Currently, the directory “`task-templates`” in the TPS git repository contains a ready task template named “`default`” which is specifically designed for IOI batch tasks. However, it is also easy for users to create and use their own custom task templates. Comprehensive documentation on task templates is available to provide guidance in this regard⁷.

The process of initiating a new task using “`tps init`” generally starts with a user interaction, prompting for a few task template parameters needed for building the correct directory structure. Such an interaction is depicted in Fig. 5 (user inputs are in boldface and blue color for clarity). It initializes a task in a new directory “`day1-book`” using the template “`default`” located at “`tps/task-templates`”. Additionally, the option “`-D has_java=false`” defines the variable “`has_java`” as “`false`” and bypasses prompting the user for this variable during the interaction.

```

> tps init "day1-book" -T "tps/task-templates" -t "default" -D has_java=false
Running the instantiation script 'tps/task-templates/default/task-template-instantiate.sh'...
Template parameter 'short_name'...
Enter a value of type 'identifier' for 'short_name':
book
Template parameter 'task_title' (Shown as heading of statement)...
Enter a value of type 'string' for 'task_title':
The Book
Template parameter 'has_grader' (Are solutions linked with graders)...
Enter a value of type 'bool' for 'has_grader':
y
Template parameter 'grader_function_name'...
Enter a value of type 'identifier' for 'grader_function_name':
solve
Template parameter 'has_java' (Is Java language available for solutions)...
Parameter 'has_java' has predefined value 'false'.
Template parameter 'has_public' (Is public data provided to the contestants)...
Enter a value of type 'bool' for 'has_public':
y
Template parameter 'statement_format' (Is statement in markdown or tex format)...
Enter a value among {md, tex, none} for 'statement_format':
md
Copying task template 'tps/task-templates/default' to the new directory 'day1-book'...
Done.
Entering the new directory 'day1-book'
Replacing '__TPARAM_HAS_JAVA__' with 'false' in content of file 'problem.json'...
Done.
Removing files related to language Java
Replacing '__TPARAM_SHORT_NAME__' with 'book' in all file contents under '.'...
:
The instantiation script execution finished successfully.
Finished. Task directory 'day1-book' is ready.

```

Fig. 5. An example of interacting with “`tps init`”.

⁷ https://github.com/ioi-2017/tps/blob/master/docs/task_templates.md

4. Conclusion

Despite the conclusion of this IOI project, the influx of bug reports, feature requests, and improvements for TPS continues, as is typical for any live project. We hope to engage more collaborators and contributors for the project in the future. Additionally, we are interested in establishing correspondence with programming contest organizers to showcase TPS, offer assistance in its usage, and gather feedback.

Acknowledgments

The members of the development team for this IOI project were Kian Mirjalali, Ali Behjati, Peyman Jabbarzade, and Mahdi Shokri. We would like to thank Amir Keivan Mohtashami, Amir Mohammad Dehghan, Ali Sharifi Zarchi, Mohammad Ali Abam, Hamid Zarrabi-Zadeh, and ISC members, especially Jonathan Irvin Gunawan for their useful comments on this project. We should also acknowledge Farid Ahmadov (Azerbaijan), Ali Sharifi Zarchi (Iran), Mohammad Mahdian (Iran), Mohammad Ali Abam (Iran), Kresimir Malnar (Croatia), Madhavan Mukund (India), Eslam Wageed (Egypt), Musa Alrefaya (Palestine), and Haris Gavranovic (Bosnia and Herzegovina) for their kind testimonials and endorsements for this IOI project.

References

- Forsgren, N., Humble, J., Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution Press.
- Lehman, M.M., Ramil, J.F., Wernick, P.D., Perry, D.E., Turski, W.M. (1997). Metrics and laws of software evolution-the nineties view. In: *Proc. 4th International Software Metrics Symposium (METRICS '97)*. pp. 20–32.
- Mirjalali, K., Mohtashami, A.K., Roghani, M., Zarrabi-Zadeh, H. (2019). TPS (task preparation system): A tool for developing tasks in programming contests. *Olympiads in Informatics*, 13, 209–215.



K. Mirjalali is a software engineer with a PhD in Computer Engineering Department from Sharif University of Technology. He was a member of the International Technical Committee (ITC) in IOI 2015 and also a member of the Host Technical and Scientific Committees (HTC, HSC) in IOI 2017. He was also an invited HSC member for IOI 2019~2022, 2024. He won a silver medal in CEOI 2003 and became a world-finalist in ICPC 2007. He has been a scientific committee member of Iranian National Olympiad in Informatics (INOI) since 2003, and ICPC in the west Asia region Tehran site since 2009.



A. Behjati is a Software Engineer at Douro Labs. He was a gold medalist in IOI 2015 and 2016. He also has been awarded a bronze medal in ICPC 2019. He was Iran's team deputy leader in IOI 2017 and an invited HSC member in IOI 2020.

The First Step Towards Increasing Female Participants in the Olympiads in Informatics in Japan

Rie Shigetomi YAMAGUCHI^{1,2*}, Tetsushi ITO^{2,3†}

¹*Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1, Hongo, Bunkyo, 113-8656, Tokyo, Japan*

²*Commitee, The Japanese Committee for International Olympiad in Informatics, 1-10-7-301, Shibuya, Shibuya, 150-0002, Tokyo, Japan*

³*Graduate School of Science, Kyoto University, Kyoto, 606-8502, Kyoto, Japan*
e-mail: yamaguchi.rie@i.u-tokyo.ac.jp, tetsushi.ito@ioi-jp.org

Abstract. In recent years, with women’s social advancement, we have been asked in various locations to “increase the proportion of women in society.” Of course, the voices are particularly loud in the field of information technology in which we are engaged. However, direct use of female quotas, that is, blatantly lowering the threshold only for women, may lead to reverse discrimination. On the other hand, since including women themselves may change the criteria for selection, it is difficult to strike a balance between the two. In Japanese Olympiads in Informatics (JOI), we have received similar questions from various quarters. We are all trying to think hard about it. In this article, we shall explain the first step towards increasing the number of female participants in JOI.

Keywords: Science Olympiad, diversity, female

1. Background: Status of Women in Informatics in Japan

- a. The number of female students in informatics departments in the universities is only slightly more than 10 % in the 2016 edition of the “White Paper on IT Human Resources” (IPA, 2016), although the exact number is unknown because the Basic School Survey by the Ministry of Education, Culture, Sports, Science and Technology does not include the category of “informatics” (Science council of Japan, 2020). As of 2019, the percentage of female members of the Information Processing Society of Japan (IPSJ) is approximately 12 % for student members and 7% for regular members. Since higher education related to informatics is often linked

*Corresponding author.

†These authors contributed equally to this work. Contributing author.

to engineering, and there are few role models for women, female students are often stuck in following stereotyped idea

Engineering \approx { male-oriented themes, few job opportunities for women }

before entering into the field of informatics.

The White Paper on Gender Equality (Gender Equality Bureau Cabinet Office, 2000) points out that female students tend to avoid mathematics and science because of the environment rather than gender differences in ability. While more than 60% of both male and female elementary school students like mathematics and science, the percentage of female junior high school students who like mathematics and science decreased by 13.6 and 27.8 points, respectively, which is more pronounced than that of male students. This indicates that interest in mathematics and science declined for some reason between elementary and junior high school.

In Japan, according to the survey of Programme for International Student Assessment (PISA) on the academic achievements of 15-years-old students, the scientific and mathematical literacy of both male and female students are higher than that of the international average (NIER, 2015). However, a high percentage of female students with high mathematical literacy do not pursue fields in which they can make use of their scientific and mathematical abilities such as IT. Rather, they tend to make safety-oriented choices, and pursue fields in which they can obtain qualifications or licenses such as medical schools.

Reflecting this, the percentage of women in science and engineering fields at universities is low. In particular, the percentage of female researchers in the fields of engineering and science, which account for the majority of researchers, is low at 12.6 % (11.1 % in engineering and 14.6 % in science) for researchers in universities and other research institutes and 8.1 % (5.6 % in engineering and 14.8 % in science) for researchers in companies. The percentage of corporate researchers was 8.1 % (5.6 % in engineering, 14.8 % in science), a low level.

Based on the results of the PISA survey, it is thought that these are not necessarily due to a lack of girls' academic ability in science and mathematics subjects, but are influenced by the environment, such as the trend of girls around them going on to higher education, parents' intentions, and the absence of role models, etc. It is necessary to foster an environment that allows students to understand the connection between their knowledge and the real world, and to provide an environment in which students can understand the connection between what they have learned and the real world. In addition, it is also important to provide support not only to students but also to their families and guardians. An environment is needed that helps students understand the connection between the knowledge they have learned and the real world, and to provide support not only to students but also to their families and guardians.

2. The Information Olympiad is a Programming Contest

The International Olympiad in Informatics (IOI) is one of the International Science Olympiads for high school students and below. Japan currently sends delegations to seven International Olympiads (mathematics, physics, chemistry, informatics, biology, geography, and geology).

As with the other International Science Olympiads, the IOI is designed for students up to high school age to identify and help develop problem-solving skills in mathematics and information science, and to promote international exchange among athletes and educators from different countries. The IOI is an internationally well-known programming contest, with approximately 90 countries participating each year. Up to four competitors from each country are allowed to participate, and Japan has participated in four of these contests every year.

The competition is an individual competition, with five hours per day to work on three to four problems. The competition lasts for two days (twice), and the final total score determines the ranking. There are partial points, and points are not awarded for fastest solutions.

The problem-solving task is to devise an effective algorithm to solve a given problem, write a program based on the algorithm, and compete on the correctness of the output from the execution on the computer. Since there are limitations on memory usage and execution time, high mathematical skills are required to design efficient algorithms (Fig. 1).

The Japan Information Olympiad is a place to select members to be sent to the world competition, and the number of participants is rapidly increasing from 978 in 2017 to

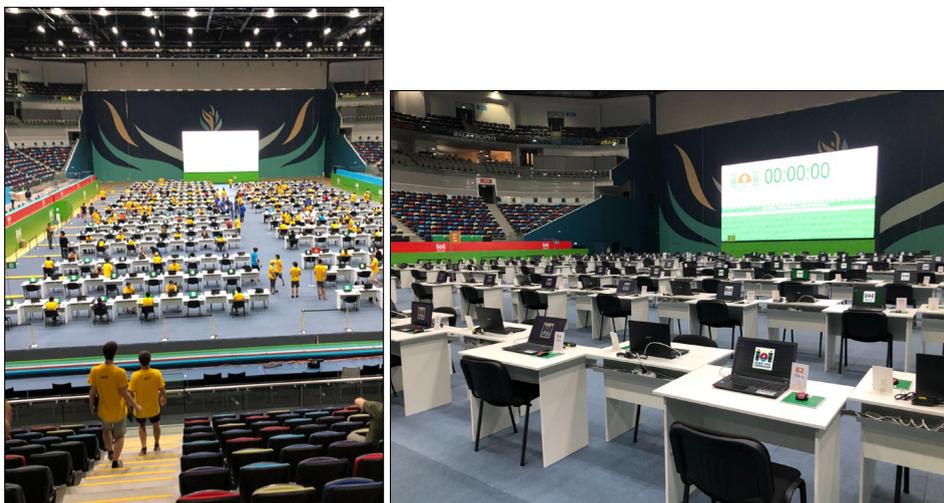


Fig. 1. The 2019 IOI in Azerbaijan (One computer per person on a desk in a large hall. The athletes sit at each of these desks and compete for 5 hours per day for a total of 2 days. The competition lasts for a total of two days with five hours of competition per day).

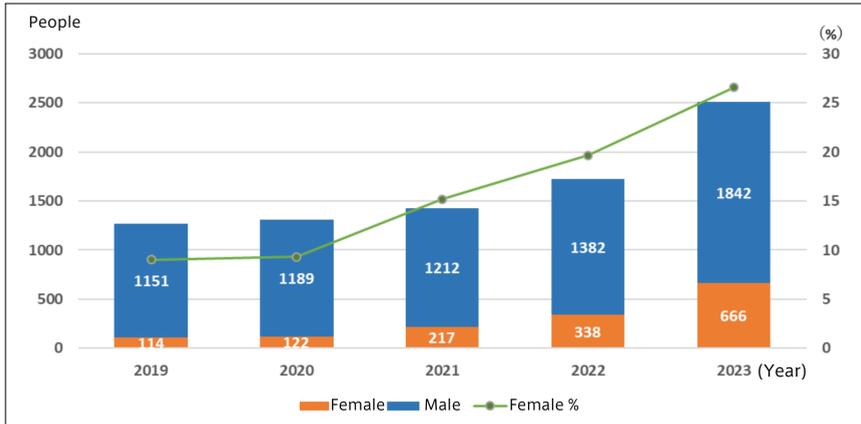


Fig. 2. Number of participants in the Japan Information Olympiad qualifiers and percentage of female participants (prepared by the Japan Committee for the Information Olympiad).

1,720 in 2022. As shown in Fig. 2, the number of participants has been increasing despite the Corona disaster, and both the number of female participants and the ratio of female participants have increased significantly since 2021, partly due to efforts to increase the number of female participants in particular.

3. Establishment of the European Girls' Olympiad in Informatics

The European Girls' Olympiad in Informatics (EGOI) (EGOI, 2023; EGOI, 2024) was launched in 2021. The EGOI aims to encourage women to enter the field of informatics by targeting only women, whereas the IOI has been held for both men and women. At the time of its inception, this competition was intended to provide a platform for young women to enjoy and deepen their interest in computer science.

The basic rules themselves are almost the same as those of the IOI, with the difference that, as the name suggests, one of the eligibility criteria is gender, and only women can participate. In addition, the number of participating countries is limited to Europe, and countries outside Europe, such as Japan and the United States, are allowed to participate only when there is room for more participants. On the other hand, this is the only programming contest for women on a global scale, making it an international "Women's" Information Olympiad to determine the actual world's best.

Japan has been sending its delegation to the EGOI since 2021, with online participation in 2021 due to the Corona disaster (Fig. 3), but in 2022, a delegation was sent to Turkey, where delegation participated on-site for the first time (Fig. 4). In 2021, Japan established the Japanese Olympiad in Informatics for Girls (JOIG), which is a programming contest for female high school students and younger. From 2022, Step 1 is the same as the Japan Olympiad in Informatics First Preliminary Round, but Step 2 (JOIG Main Round) and Step 3 (JOIG Spring Training) are open to female participants only, and the contests are conducted.



Fig. 3. EGOI 2021 Switzerland (online participation from hotels in Japan).

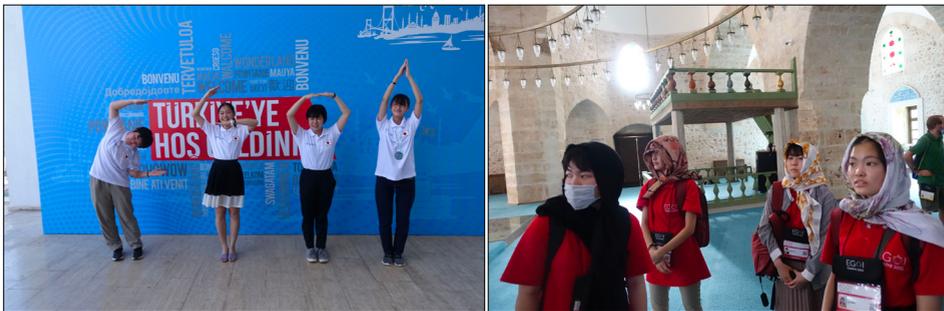


Fig. 4. EGOI 2022 Turkey (Left: EGOI human script by participating athletes, Right: mosque experience).

The existence of such competitions for women has encouraged more active participation by female contestants, and the participation rate of women has increased rapidly since 2021. We believe that participation in competitions for women, as well as the existence of international competitions, has become a clear goal for female students participating in the Information Olympiad. In conjunction with this, introductory courses for women have been held, and online programming courses (JOI introductory courses) not limited to women have been held regularly to create a framework in which interested students can more easily participate.

4. Women Role Models in Informatics

As described in the chapter “Status of Women in Informatics,” there are few role models for women in informatics, and we started a role model course in October 2021. This lecture is recorded on video in advance and distributed on YouTube (Japanese Committee for the IOI, 2024).



Fig. 5. “Ask a senior colleague! The road map to becoming a programmer Vol. 1”
(booklet of interviews from the Role Models course).

The interviewees are women who use programming and related technologies and knowledge. We interviewed these women about their work and student days, and asked them why they decided to pursue a career in the information field. The contents of this report are also published in book form as shown in Fig. 5, and sent to high schools throughout Japan.

All of the stories are very interesting we have been able to hear from people in the information field who are now making positive efforts despite having had to face setbacks at various points in their careers. For example, a person who once found a job that he had dreamed of since high school but returned to university to study informatics again, or a person who came from an experimental research background but found that the informatics field, which does not require midnight experiments, was the best choice for conducting research and working while raising a child. Many of the stories convey the advantages of working in the information field.

5. Manga Booklet for Female Participants

Many people are inspired by popular manga and TV dramas to pursue careers in the information field. As a side note, the year after the broadcast of *Dragon Zakura*, the ratio of entrance examinations for the University of Tokyo increased significantly, and after the publication of *Animal Doctor, Dobutsu-no-Oishasan*, the number of students who aim to become veterinarians at Hokkaido University increased. Manga and TV dramas



Fig. 6. Cover of the third manga booklet (the story of two girls in their first year of high school, who participated in the course and aimed to participate in Japanese Olympiad in Informatics).

have been shown to have a direct impact on the future dreams of high school students, especially in Japan.

The reason why more female students aspire to become doctors or pharmacists than those in the information-related fields is not only because of their confidence in their qualifications, but also because they can easily visualize their professions and student life through manga and dramas. On the other hand, women in the information-related fields who are portrayed in manga and TV dramas are sometimes treated as “geeks,” and it is difficult to imagine them aiming for such positions in a cheerful and attractive manner.

Although we would like to create a more grandiose manga, due to budget and resource constraints, we decided to start small story manga, and have already published three issues of a manga booklet (Fig. 7). We have already published three issues (Fig. 6). We are not only sending these to high schools nationwide, but are also sending a large number of copies to girls’ schools to encourage their participation.

6. We Need to Advertise the Whiteness of the Information Field

The Information Olympiad has never refused the participation of women, but the participation rate has increased significantly in recent years after a long period of slow growth. The main reasons for this increase are the existence of a clear international

competition, EGOI, and the establishment of a women's division in the national competition. At the same time, there may be a slight improvement in the understanding of society as pointed out by the Science Council of Japan and other organizations. One of the most impressive stories in the Role Model Lecture was that many people talked about the "ease of returning to work after childbirth" in the information-related field. If this kind of understanding is promoted in society as a whole, it will help programmers escape from the black image of simple work and show that all workplaces are white.

7. After a Young Girl was Even Slightly Interested

Once the various promotions reach young women and non-binary people, the next challenge is how to get them to settle into our field. The Japan Information Olympiad Association has been trying to establish this by offering introductory courses, but it is still not enough. Unfortunately, Japan has long been said to be a country where women's participation in society is weak (OECD, 2023). There are various reasons for this, but in particular, it is said that there is a large generation gap in science education for women.

We believe that there are various factors that hinder the young generation from taking an interest in science. In order to solve this problem, Japan should study the methods used in other parts of the world. Japan has a tendency to favor uniformity in education, which makes it difficult for new types of education, such as information education, to make inroads. We have high hopes that the Information Olympiad will help to break through the above problems, as the name of the Olympiad gives a clear image of the activities that will follow.

The Information Olympiad is a part of primary and secondary education that is closely related to the family and society, and there is a need to turn the cycle of human resource development into a virtuous circle through various publicity activities.

References

- IPA (2016). Information-technology promotion agency (IPA), IT human resource development division: *IT Human Resource White Paper 2016*, Preparedness to step out into diverse cultures -Hurry up to cope with digital transformation (in Japanese) (27 April, 2016)
- Science council of Japan (2020). *Report of the Science Council of Japan: Current status and challenges of gender balance in science and engineering* (in Japanese) (5 June, 2020)
- Gender Equality Bureau Cabinet Office (2000). The cabinet office, gender equality bureau cabinet office: *White Paper on Gender Equality 2000*.
https://www.gender.go.jp/english_contents/about_danjo/whitepaper/plan2000/2000/3.html
- NIER (2015). National institute for educational policy research (NIER): The organisation for economic co-operation and development (OECD) PISA 2015 survey.
<https://www.nier.go.jp/kokusai/pisa/index.html#PISA2015>
- EGOI (2023). *European Girls' Olympiad in Informatics 2023*. Retrieved at 31 May, 2024:
<https://egoi23.se/>
- EGOI (2024). *European Girls' Olympiad in Informatics 2024*. Retrieved at 31 May, 2024:
<https://egoi2024.nl/>

Japanese Committee for the IOI (2024). Japanese Committee for The International Olympiad in Informatics: Role model course (in Japanese). Retrived at 31 May, 2024 (2022):

<https://joi.ioi-jp.org/support-message>

OECD (2023). Organisation for Economic Cooperation and Development: Joining Forces for Gender Equality What is holding us back? Retrived at 31 May, 2024 (2023):

<https://www.oecd.org/japan/Gender2023-JPN-En.pdf>



R. Yamaguchi, Associate Professor of Graduate School of Information Science and Technology, The University of Tokyo. She received Master degree in mathematics from Tsuda College, now Tsuda University, in 2003, and PhD degree in Information Science and Technology from the University of Tokyo in 2006. She joined Information Security Center, National Institute of Advanced Industrial Science and Technology, AIST, in 2006 and concurrently serve in National Information Security Center at Cabinet Secretariat, now National center of Information Incident readiness and Strategy for Cybersecurity from 2007 to 2011. Current position since 2013.



T. Ito is an associate professor of Graduate School of Science at Kyoto University since 2009. He earned a Ph.D. (Mathematical Sciences) from the University of Tokyo in 2003. His research interest is in number theory, algebraic geometry and related areas. He was a participant of IOI in 1994 and 1995. He is now a board member of the Japanese Committee for International Olympiad in Informatics.

The Official IOI Website: The Good, the Bad and the Ugly

Araz YUSUBOV

*School of IT and Engineering, ADA University, Baku, Azerbaijan
e-mail: ayusubov@ada.edu.az*

Abstract. This paper is an extended report from one of the discussion groups at the 35th International Olympiad in Informatics (IOI) in Hungary. It looks back at the evolution of the official IOI website, provides an overview of the current web resources and reflection on their strengths and weaknesses. The report also proposes short-term and long-term suggestions for further development of the IOI web services.

Keywords: IOI, website, web services, customer journey, sitemap, social media.

Introduction

It is a longstanding tradition now that every International Olympiad in Informatics (IOI) program includes group discussions. The IOI community members are called to propose the topics and sometimes the resulting discussion reports lead to proposals for changes and new procedures, for example, the recent introduction of the Honorable Mention award (Jovanov and Stankov, 2020).

One of the seven group discussions at the IOI 2023 in Hungary was “Face of the IOI” with the question posed as “What can be improved on the IOI official website and other online infrastructure (country websites, social media, etc.)?” The motivation for the discussion was the anecdotal evidence of difficulties in finding the required information as some of the materials are not immediately available through the current official IOI website. For example, in 2022 there were no submissions for the IOI call for projects, but apparently the relevant announcement was not prominently visible and easily accessible at the website.

The goal of this report is to expand on the discussions of this workshop with a hope that the short-term and long-term recommendations will be taken on board for further improvement of the IOI web services.

1. The Look Back

The Wayback Machine¹ holds the snapshot records for the official IOI website hosted at <https://ioinformatics.org/> as old as back from 5 October 2002. The second version of the website was launched around 26 June 2005 and credits Don Piele², an IOI pioneer from the USA, who sadly passed away in 2014 (Donald Piele Obituary, 2014), as the webmaster. The longest serving third version was created in January 2008, as it is apparent from the page source of the homepage, by Scott Greenlay of the University of Waterloo, who at the time worked with Troy Vasiga³ of Team Canada. Since 2012, the website has been updated by Martins Opmanis⁴, an IOI veteran from Latvia. He continues maintaining this version, which is hosted at <https://ioi.lv/>, in parallel with the official IOI website after the introduction of the newest version in August 2018. This website, also accessible through <https://history.ioinformatics.org>, is still popular with 600+ peak monthly visits increasing every year (650 in August 2019, 702 in September and 724 in December 2020, 806 in June 2021 and 936 in August 2022).



Fig. 1. Snapshots of the IOI website homepage (from left to right) in 1998 and 2002 (top row), in 2005 and 2008 (bottom row).

¹ The Wayback Machine is an initiative of the Internet Archive non-profit, building a digital library of Internet sites and other cultural artifacts in digital form. <https://web.archive.org/>

² IOI Stats – United States of America – People – Donald Theodore Piele. <https://stats.ioinformatics.org/people/4185>

³ IOI Stats – Canada – People – Troy Vasiga. <https://stats.ioinformatics.org/people/3254>

⁴ IOI Stats – Latvia – People – Mārtiņš Opmanis. <https://stats.ioinformatics.org/people/2645>

There is another website hosted at <https://olympiads.win.tue.nl/ioi/> and maintained at the time by Tom Verhoeff⁵ of Eindhoven University of Technology (TU/e), which predates the current IOI website with an earliest snapshot from 5 December 1998. This site is still up and is an invaluable resource if you want to deep-dive into the history of IOI, the times when the diskettes were used to pass the solutions.

One more website is the IOI Statistics website with the earliest Wayback Machine snapshot from April 2014, hosted at <https://stats.ioinformatics.org/>. Arguably the most popular IOI website that provides convenient access to comprehensive contestant data, it is moderated by colleagues from Latvia, Eduard Kaliničenko⁶, Martins Opmanis and Oleg Oshmyan⁷.

1.1. Mailing Lists

As per the Final Report (Heyderhoff *et al.*, 1992) for the IOI 1992 in Bonn, Germany, one of the decisions was “to install an electronic discussion list, in the form of a list server, as a means of communication among the participating countries in the period between two Olympiads.” The proposal came from the Hungary and the USA delegations, and as it is apparent from the domain name (inf.bme.hu) the mailing list was hosted by the Budapest University of Technology and Economics. In October 1997, it was replaced by the new IOI-list at TU/e with more specific mailing lists added over the years.

Currently the International Technical Committee (ITC) maintains the mailing lists for the IOI General Assembly, announcements, discussions, for all three international committees, and for national/regional training camps⁸.

1.2. Social Media Channels

Back in August 2019, the IOI pages were launched at major social media channels i.e. Facebook, X (formerly Twitter), Instagram and YouTube. Apparently all relevant social media profiles are associated with the IOI Secretariat email. They are maintained by Eslam Wageed⁹ of Team Egypt.

⁵ IOI Stats – Netherlands – People – Tom Verhoeff. <https://stats.ioinformatics.org/people/3553>

⁶ IOI Stats – Latvia – People – Eduards Kaļiņičenko. <https://stats.ioinformatics.org/people/681>

⁷ IOI Stats – Latvia – People – Oļegs Ošmjans. <https://stats.ioinformatics.org/people/1731>

⁸ IOI Mailing Lists. <https://lists.ioinformatics.org/>

⁹ IOI Stats – Egypt – People – Eslam Wageed. <https://stats.ioinformatics.org/people/3269>

2. The Current State

The interest in “a redesign of the look and feel” of the official IOI website had been expressed as early as in 2011 (International Committee, 2011). In August 2018, a project led by Bojan Kostadinov¹⁰ of Team North Macedonia concluded with the launch of the new version of the website. There was an unfortunate miscommunication during the unintentionally abrupt transition to the new version of the website, as it is apparent from discussions at the IOI 2018 General Assembly (IOI, 2018) and in the International Committee (International Committee, 2019) during the IOI 2019. The intention of the redesign was primarily to change the look and feel while retaining all the information.

The current IOI website uses Bootstrap, a popular free, open-source front-end development framework for building websites with responsive design, easily viewable primarily on mobile devices. The new design has also greater potential for more visual, media-rich content. However, this potential still is to be used fully, as initially the primary intent was only to transfer the existing content.

While having a powerful mission and an inspiring imagery IOI still to more proactively engage new members to join the IOI countries, new long-term sponsors to sustainably support the organization and new nations to host future IOI contests. The IOI website has an unused potential to help with these.

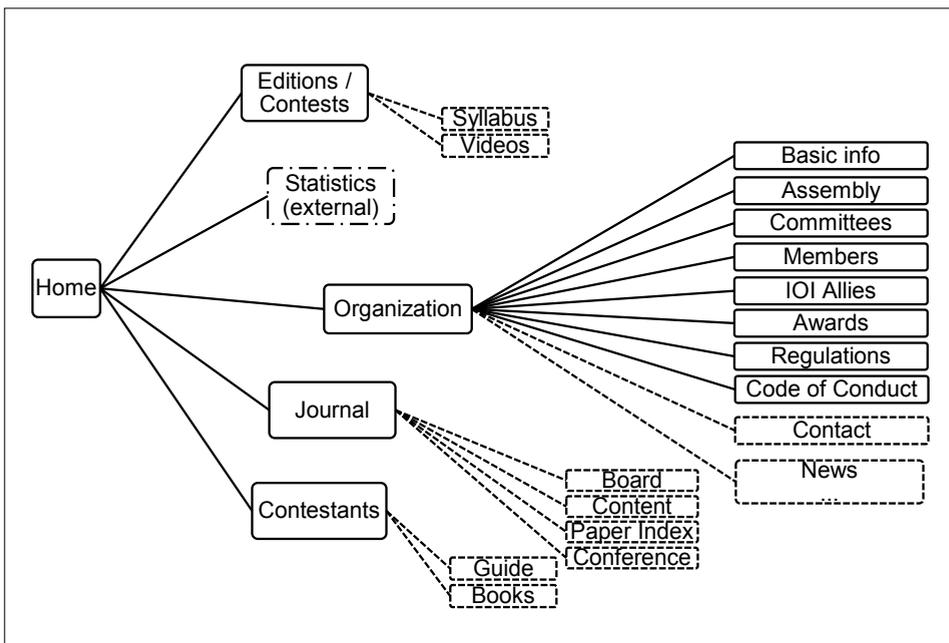


Fig. 2. The current sitemap of the official IOI website.

¹⁰IOI Stats–Macedonia–People–Bojan Kostadinov. <https://stats.ioinformatics.org/people/4451>

In addition to the look and feel changes, over the years there were suggestions for the IOI website to go through structural redesign. For example, a call to address the concerns about “some information on the website [being] hard to find,” was voiced as early as back in 2009 (International Committee, 2009).

It is apparent from the sitemap (Fig. 2) that there are useful resources that are not easily accessible or not permanently available (denoted by dashed borders), such as getting-started guideline or recommended readings for the new contestants or even contacting details, which are buried as links at the bottom of the homepage with no alternative paths to them.

This need for structural redesign was also the main motivation behind the “Face of the IOI” workshop at the IOI 2023. The following section summarizes the discussion points and outlines potential recommendations.

3. The Way Forward

In the early 2000’s many organizations went through strategic rethinking of their web services and moving out from an informational, archival view towards using them as a means for engagement and communication.

In 2003, the author led a country phase for the multi-million pound British Council project to revamp its web strategy (Telecompaper, 2003). The news articles back then were mostly focused on how this organization that operates in 110 countries will achieve almost an immediate return on investment and 1.3 million GBP savings right the next year after the completion of the project by centralizing and harmonizing its infrastructure. However, the success of the project was also defined by the professional central team, which put together a clear vision for the goal of every web service, clear standards for customer-centered website structure, web page templates to be used for predefined page types, comprehensive artistic and technical guidelines for developing textual and media content.

The central vision was that any content should eventually serve the goal of bringing the visitor in contact with the organization for procuring its services or products. Understanding potential visitors and building a customer journey map is crucial for its implementation. A customer journey is broadly described (Lemon and Verhoeff, 2016) as the flow of iterative and dynamic customer experience process from pre-purchase to purchase and post-purchase, the interactions that occur before, during, or after the customer experiences a product or service.

Ultimately the sitemap will reflect the customer journeys with an aim to minimize the effort for reaching the required destination, for example the web page with the sought information. This also informs the requirements towards the visual content that is meant to be engaging, showing people in action, focusing on personal experiences, as well as towards the textual content that should be clear and direct with calls to action, formal and friendly with a readable structure.

Table 1
The list of identified visitor groups and their primary intentions
(available and – easy to find, – not easy to find, – not available)

Who are they?	What do they want?	Call to action
1. School students, interested	What is IOI? What are the costs? What my friend is doing there? What are the benefits?	Learn about us <input type="checkbox"/>
	What is the organization, committees?	Learn about us <input checked="" type="checkbox"/>
	How to join IOI? Is my country there?	Join us <input type="checkbox"/>
	What are the procedures?	Learn about us <input checked="" type="checkbox"/>
	How to prepare for the competition? What are the resources?	Join us <input type="checkbox"/>
2. School students, returning	Learn about statistics.	Learn about us <input checked="" type="checkbox"/>
	Learn about future IOI.	Join us <input checked="" type="checkbox"/>
	Is IOI safe? What is the safety advice for this year?	Join us <input checked="" type="checkbox"/>
3. Parents / teachers of the above 1 or 2	– same as above 1 or 2.	
4. Companies	Hire the top talent.	Grow with us <input checked="" type="checkbox"/>
	Sponsor the next IOI.	Support us <input checked="" type="checkbox"/>
5. Universities	Recruit the top talent.	Grow with us <input checked="" type="checkbox"/>
	Research competitive programming.	Grow with us <input checked="" type="checkbox"/>
	Participate at the IOI conference.	Grow with us <input type="checkbox"/>
6. Potential hosts, e.g. country government	How to become a host? e.g. processes, steps, contacts	Support us <input checked="" type="checkbox"/>
7. Other Olympiads	Compare processes and procedures	Learn about us <input checked="" type="checkbox"/>
8. Activists, e.g. disability, gender	Does IOI support their cause?	Learn about us <input type="checkbox"/>
9. Larger IOI community	How to submit a project?	Grow with us <input type="checkbox"/>
	How to submit a task?	Grow with us <input type="checkbox"/>
	How to submit an article (to the IOI conference)?	Grow with us <input checked="" type="checkbox"/>

Short Term Reflections

The group went through an exercise of taking a step back for a fresh look at the IOI website from the perspective of what are the main groups of visitors and what is their primary intention. Having access to website statistics (e.g. visitor data, search phrases) and surveying a larger number of people could help with potentially more visitor groups identified.

Quick examination shows that answers to a number of the inquiries listed in Table 1 are not easily accessible. Ironically, to have a laconic answer to the simple question about what the IOI is, one now needs to navigate to the IOI regulations, while many country websites include immediately visible few standard paragraphs about the IOI.

The IOI Regulations (IOI, n.d.) list “To encourage countries to organise a future IOI in their country” among the five “main objectives to be accomplished by the IOI”, however there is not much to accommodate related inquiries at the website.

Another objective is “to foster friendly international relationships among computer scientists and informatics educators.” IOI Conference and Journal are very much speaking to this objective. Researching competitive programming can be further promoted, as there are very few efforts to analyze the competition data (Hasanov *et al.*, 2021). Promoting the IOI Conference more outside the IOI community, perhaps through partnerships with relevant global professional societies, may also increase the number of guest participants from potential new countries.

The group proposed a second exercise about checking if those visitor groups easily land at the IOI website by searching for related keywords e.g. IOI, programming/coding/computing/algorithmic competition/contest/award for high-school kids/children/students. A quick examination showed that for a direct search for “IOI” Google recommends the IOI official website and Wikipedia article on top of the search results, accompanying it with references to the I.O.I Korean pop music girls’ band. The search for variations of the “programming competition for high-school kids” returns the reference to IOI among top nine results, except when “award” is used.

Long Term Reflections

The need for “a centralized content management system for building official IOI websites” every year was already noted in conclusions of the IOI 2019 team’s report (Yusubov *et al.*, 2022) as a way to “save host team’s efforts for setting up this important communication channel, resolve the issue of archiving the historical content, and ensure a consistent look and feel.”

The idea is to arrange a centralized hosting for the official IOI main website, as well as IOI editions. Many global organizations or conferences use this approach. A good example is the ACM Celebration of Women in Computing womENcourage™ websites (womENcourage, 2020). The system uses the centrally hosted WordPress platform with a generic ACM template, which provides some degree of flexibility for customization. Each year, the global administrator creates a new site with default structure and content for the upcoming edition. The local host gets administrator access with the possibility of adding local editors.

There are many immediate benefits of this approach:

- Helping hosts: hosts will not have to think about arranging the web hosting, choosing the content management system, etc. they will immediately focus on adding the content and managing the communications.
- Consistency: using a centralized general template, some preset color schemes and graphical design themes will help to build a strong identity.
- Harmonization: some content can be reused between the websites, e.g. news about this year’s IOI can automatically appear at the main IOI website.
- Auto-archiving: past IOI edition websites will not face the danger of being lost, as well as not rely on individual initiatives for ensuring proper archiving.

The group also discussed potential risks that need to be considered and mitigated:

- A considerable effort will need to be put to complete the project with the technical infrastructure and processes successfully developed.

- The assigned IOI organizational structure will have to handle the additional responsibility of maintaining the centralized hosting.
- There may be a need for a special functionality that will exceed the limitations of the common template or the platform.
- A host may have wishes dictated by local arrangements or specific requirements that will conflict with the central arrangements.

Further Thoughts

Domain name reservation is another matter to keep an eye on. Many organizations make sure to secure their names (e.g. ioiXXXX, ioinformatics) on all major general and country top-level domain names. Perhaps, this can also be centralized.

Once the centralized approach is piloted for the websites, it can be used also for media hosting/social media channels. For example, ACM gives limited access for conference organizers to the official ACM YouTube channel (YouTube, n.d.) for uploading their video material. Many videos are popular years after an IOI edition is over. For example, in 2023, the Team USA interview at IOI 2019 got up to 1,177 views monthly. Perhaps, if relevant agreements and licensing is arranged, monetization of these videos (and other media) may open a new income stream in the IOI budget.

Conclusion

There is a big potential of the official IOI website for turning into a powerful engagement tool that will help with achieving the organizational goals. The previous section mentions a number of recommendations for further improvement of the IOI web services.

A prerequisite for a successful implementation of those is having accessible web statistics from simple visit and page view numbers to website visitor journey maps, as the famous quote says “you can’t manage what you don’t measure.” Above all, as people and procedures are considered among major components of any information system (Bourgeois, 2014), it is important to establish clear procedures for what the main operations (e.g. uploading the contest tasks in a week after the IOI is closed) are and who the responsible person is for each operation.

Acknowledgements

The author thanks the workshop participants, Sandra Schumann¹¹, Team Estonia leader, and Steven Halim¹², Team Singapore leader, for their active contribution to the initial discussions. Martins Opmanis and Troy Vasiga were very kind to share some historical data. The author is also grateful to Martins Opmanis for reviewing the draft of the paper.

¹¹ IOI Stats – Estonia – People – Sandra Schumann. <https://stats.ioinformatics.org/people/6520>

¹² IOI Stats – Singapore – People – Steven Halim. <https://stats.ioinformatics.org/people/3313>

References

- Bourgeois, D.T. (2014). *Information Systems for Business and Beyond*.
- Donald Piele Obituary (2014). Accessed 27 June 2024, <https://obits.columbian.com/us/obituaries/columbian/name/donald-piele-obituary?id=22685778>
- Hasanov, J., Gadirli, H., Bagiyev, A. (2021). On Using Real-Time and Post-Contest Data to Improve the Contest Organization, Technical/Scientific Procedures and Build an Efficient Contestant Preparation Strategy. *Olympiads in Informatics*, 2022, 16, 23–36. https://ioinformatics.org/journal/v15_2021_23_36.pdf
- Heyderhoff, P. (Editor), Hein, H.-W., Krückeberg, F., Miklitz, G., Widmayer, P. (1992). *Final Report IOI 1992 Bonn / Germany* <https://olympiads.win.tue.nl/ioi/ioi92/report.html>
- International Committee (2009). Minutes of the Meetings held in Plovdiv, Bulgaria / 8–15 August, 2009. Accessed 27 June 2024, <https://ioinformatics.org/minutes/ic/ic-2009-aug-minutes.pdf>
- International Committee (2011). Minutes of the Meetings held in Pattaya, Thailand / 17–20 February, 2011. <https://ioinformatics.org/minutes/ic/ic-2011-feb-minutes.pdf>
- International Committee (2019). Minutes of the Meetings held in Baku, Azerbaijan / 4–11 August 2019. <https://ioinformatics.org/minutes/ic/ic-2019-aug-minutes.pdf>
- IOI (n.d.). Regulations. Accessed 27 June 2024, <https://ioinformatics.org/page/regulations/>
- IOI (2018). IOI 2018 General Assembly Minutes. Tsukuba, Japan September 1–8, 2018 Convention Hall 300. Accessed 27 June 2024, <https://ioinformatics.org/files/ioi2018minutes.pdf>
- Jovanov, M., Stankov, E. (2020). Introduction of “Honorable Mention” Award at the International Olympiad in Informatics. *Olympiads in Informatics*, 2020, 14, 87–104. https://ioinformatics.org/journal/v14_2020_87_104.pdf
- Lemon, K.N., Verhoeff, P.C. (2016). Understanding customer experience throughout the customer journey. *Journal of Marketing*, 80(6), 69–96. <http://www.jstor.org/stable/44134974>
- Telecompaper (2003). British Council invests in Web strategy revamp (11 December 2003). <https://www.telecompaper.com/news/british-council-invests-in-web-strategy-revamp--369458>
- womENCourage (2020). ACM Celebration of Women in Computing: womENCourage 2020. 24–27 September 2020, Baku, Azerbaijan. Accessed 27 June 2024, <https://womencourage.acm.org/2020>
- YouTube (n.d.). YouTube – Association for Computing Machinery (ACM). Accessed 27 June 2024, <https://www.youtube.com/@theofficialacm>
- Yusubov, A., Ahmadi, F., Hasanov, J. (2022). Hosting IOI 2019 Azerbaijan: Back to the Future. *Olympiads in Informatics*, 2022, 16, 173–195. https://ioinformatics.org/journal/v16_2022_173_195.pdf



A. Yusubov is an Assistant Professor of Computer and Information Sciences in the School of IT and Engineering at ADA University. Dr. Yusubov worked in academia, industry and international organizations, led and contributed to various educational projects, including the national FIRST LEGO League robotics tournaments for school children. He is an ACM Senior Member and the founding member of the Azerbaijan ACM/ACM-W Chapter. Dr. Yusubov has been an IC member for the period of 2017–2020, elected again for 2021–2024, was local Host Coordination Committee Manager during IOI 2019. <http://stats.ioinformatics.org/people/6418>

About Journal and Instructions to Authors

OLYMPIADS IN INFORMATICS is a peer-reviewed scholarly journal that provides an international forum for presenting research and developments in the specific scope of teaching and learning informatics through olympiads and other competitions. The journal is focused on the research and practice of professionals who are working in the field of teaching informatics to talented student. OLYMPIADS IN INFORMATICS is published annually (in the summer).

The format for the journal follows the tracks:

- the primary section of the journal focuses on research
- the second report section is devoted to sharing experiences of countries in informatics olympiads
- the last smallest section presents books reviews or other information

The journal is closely connected to the scientific conference annually organized during the International Olympiad in Informatics (IOI).

Abstracting/Indexing

OLYMPIADS IN INFORMATICS is abstracted/indexed by:

- Cabell Publishing
- Central and Eastern European Online Library (CEEOL)
- EBSCO
- Educational Research Abstracts (ERA)
- ERIC
- InfoBase Index
- INSPEC
- SCOPUS – Elsevier Bibliographic Databases

Submission of Manuscripts

All research papers submitted for publication in this journal must contain original unpublished work and must not have been submitted for publication elsewhere. Any manuscript which does not conform to the requirements will be returned.

The journal language is English. No formal limit is placed on the length of a paper, but the editors may recommend the shortening of a long paper.

Each paper submitted for the journal should be prepared according to the following structure:

- concise and informative title
- full names and affiliations of all authors, including e-mail addresses

- informative abstract of 70–150 words
- list of relevant keywords
- full text of the paper
- list of references
- biographic information about the author(s) including photography

All illustrations should be numbered consecutively and supplied with captions. They must fit on a 124 × 194 mm sheet of paper, including the title.

The references cited in the text should be indicated in brackets:

- for one author – (Johnson, 1999)
- for two authors – (Johnson and Peterson, 2002)
- for three or more authors – (Johnson *et al.*, 2002)
- the page number can be indicated as (Hubwieser, 2001, p. 25)

The list of references should be presented at the end of the paper in alphabetic order. Papers by the same author(s) in the same year should be distinguished by the letters a, b, etc. Only Latin characters should be used in references.

Please adhere closely to the following format in the list of references:

For books:

Hubwieser, P. (2001). *Didaktik der Informatik*. Springer-Verlag, Berlin.

Schwartz, J.E., Beichner, R.J. (1999). *Essentials of Educational Technology*. Allyn and Bacon, Boston.

For contribution to collective works:

Batissta, M.T., Clements, D.H. (2000). Mathematics curriculum development as a scientific endeavor. In: Kelly, A.E., Lesh, R.A. (Eds.), *Handbook of Research Design in Mathematics and Science Education*. Lawrence Erlbaum Associates Pub., London, 737–760.

Plomp, T., Reinen, I.J. (1996). Computer literacy. In: Plomp, T., Ely, A.D. (Eds.), *International Encyclopedia for Educational Technology*. Pergamon Press, London, 626–630.

For journal papers:

McCormick, R. (1992). Curriculum development and new information technology. *Journal of Information Technology for Teacher Education*, 1(1), 23–49.

<http://rice.edn.deakin.edu.au/archives/JITTE/j113.htm>

Burton, B.A. (2010). Encouraging algorithmic thinking without a computer. *Olympiads in Informatics*, 4, 3–14.

For documents on Internet:

IOI (2008). *International Olympiads in Informatics*

<http://www.IOInformatics.org/>

Hassinen, P., Elomaa, J., Ronkko, J., Halme, J., Hodju, P. (1999). *Neural Networks Tool – Nenet (Version 1.1)*.

<http://koti.mbnet.fi/~phodju/nenet/Nenet/General.html>

Authors must submit electronic versions of manuscripts in PDF to the editors. The manuscripts should conform all the requirements above.

If a paper is accepted for publication, the authors will be asked for a computer-processed text of the final version of the paper, supplemented with illustrations and tables, prepared as a Microsoft Word or LaTeX document. The illustrations are to be presented in TIF, WMF, BMP, PCX or PNG formats (the resolution of point graphics pictures is 300 dots per inch).

Contacts for communication

Valentina Dagiene
Vilnius University
Akademijos str. 4, LT-08663 Vilnius, Lithuania
Phone: +370 5 2109 732
Fax: +370 52 729 209
E-mail: valentina.dagiene@mif.vu.lt

Internet Address

All the information about the journal can be found at:

<https://ioinformatics.org/page/ioi-journal>

Olympiads in Informatics

Volume 18, 2024

Foreword	i
G. AUDRITO, S. CAPECCHI, M. G. CIOBANU, L. LAURA <i>Giochi di Fibonacci</i> Year II: Competitive Blocks Programming for Young Students	1
M. KAYKOBAD Popularizing Science and Science Competitions	25
E. LEE, T. REIZIN, F.E. WU, F.E. WU Trends on Returning Contestants and Geography at the International Olympiad in Informatics	33
M. MAMMADLI, N. MAMMADLI, J. HASANOV Analysis and Evaluation of the Contestant's Progress in Real-time Coding Contests	51
K. MANEV Preparing of Youngest Students for Participation in Programming Contests	63
P.S. PANKOV, E.J. BAYALIEVA Olympiads without Words	81
F. STEINERT, J. KUMMER, M. LANDMAN, L. LEHNER From Concept to Code: A Two-Day Workshop for Secondary Students on Computational Thinking and Programming	89
A. TANEJA, A. KOTHARI Algorithmic Problem-Solving Advancements: A Comprehensive Exploration across Diverse Domains	101
T. VERHOEFF Staying DRY with OO and FP	113
E.M. WAGEED, Y.S. ELGAMAL, O.M. ISMAIL, M.H. ABDRABOU The Impact of Non-Formal Educational Approach on the Academic Performance and Employability of Engineering and Computer Science Students	129
REPORTS	
M. ALREFAYA, S. ALHAJAJLA Palestine at the International Olympiad in Informatics: Advancing Computational Thinking Among K-12 Students	147
M. DOLINSKY High School Programming Olympiads in Gomel Region	155
H.E. DUEÑAS OROZCO, T. AVALOS PIÑON omegaUp: A Decade of Growth and Impact in Latin American Coding Education	167
K. MIRJALALI, A. BEHJATI IOI Project Report on Improving TPS (Task Preparation System)	175
R.S. YAMAGUCHI, T. ITO The First Step Towards Increasing Female Participants in the Olympiads in Informatics in Japan	185
A. YUSUBOV The Official IOI Website: The Good, the Bad and the Ugly	195

Publisher office: Vilnius University
Akademijos str. 4, LT-08663 Vilnius, Lithuania
July, 2024

Olympiads in Informatics

Volume 18, 2024

Foreword	i
G. AUDRITO, S. CAPECCHI, M. G. CIOBANU, L. LAURA <i>Giochi di Fibonacci</i> Year II: Competitive Blocks Programming for Young Students	1
M. KAYKOBAD Popularizing Science and Science Competitions	25
E. LEE, T. REIZIN, F.E. WU, F.E. WU Trends on Returning Contestants and Geography at the International Olympiad in Informatics	33
M. MAMMADLI, N. MAMMADLI, J. HASANOV Analysis and Evaluation of the Contestant's Progress in Real-time Coding Contests	51
K. MANEV Preparing of Youngest Students for Participation in Programming Contests	63
P.S. PANKOV, E.J. BAYALIEVA Olympiads without Words	81
F. STEINERT, J. KUMMER, M. LANDMAN, L. LEHNER From Concept to Code: A Two-Day Workshop for Secondary Students on Computational Thinking and Programming	89
A. TANEJA, A. KOTHARI Algorithmic Problem-Solving Advancements: A Comprehensive Exploration across Diverse Domains	101
T. VERHOEFF Staying DRY with OO and FP	113
E.M. WAGEED, Y.S. ELGAMAL, O.M. ISMAIL, M.H. ABDRABOU The Impact of Non-Formal Educational Approach on the Academic Performance and Employability of Engineering and Computer Science Students	129
REPORTS	
M. ALREFAYA, S. ALHAJAJLA Palestine at the International Olympiad in Informatics: Advancing Computational Thinking Among K-12 Students	147
M. DOLINSKY High School Programming Olympiads in Gomel Region	155
H.E. DUEÑAS OROZCO, T. AVALOS PIÑON omegaUp: A Decade of Growth and Impact in Latin American Coding Education	167
K. MIRJALALI, A. BEHJATI IOI Project Report on Improving TPS (Task Preparation System)	175
R.S. YAMAGUCHI, T. ITO The First Step Towards Increasing Female Participants in the Olympiads in Informatics in Japan	185
A. YUSUBOV The Official IOI Website: The Good, the Bad and the Ugly	195