Olympiads in Informatics, 2009, Vol. 3, 149–166

20 Years of IOI Competition Tasks

Tom VERHOEFF

Department of Mathematics and Computing Science, Eindhoven University of Technology
Den Dolech 2, 5612 AZ Eindhoven, The Netherlands
e-mail: t.verhoeff@tue.nl

Abstract. The competition tasks at the International Olympiad in Informatics have evolved over its 20-year history. We distinguish three periods in this evolution and highlight it from various viewpoints. The 101 competition tasks are presented in a table that summarizes their task type and difficulty level, and that classifies them according to concepts involved in their problem and solution domains.

Key words: computer science competition, International Olympiad in Informatics, competition tasks, algorithmics, history.

1. Introduction

The first International Olympiad in Informatics (IOI website, 2009) was held in 1989 in Pravetz, Bulgaria, now 20 years ago. The format of the event has evolved over the years, but its main principles have not changed. The high-level goal of the IOI is still that of promoting computer science (CS) among the youth, and of discovering and stimulating young talent in CS. The IOI is still offering a competition in algorithmic problem solving, where solutions must be implemented in one of a few programming languages (Verhoeff *et al.*, 2006).

Let me remind you of two reasons for restricting the competition to algorithmic programming problems, which nowadays are considered as belonging to a narrow subfield of CS. Back in 1989, CS was not a regular topic in the high school curriculum in most countries. Some school pupils did learn some CS – either by themselves, or through relatives or devoted teachers – mostly in the form of computer programming. This limited the options for a CS competition. Secondly, there were no widely accepted standards for the communication of algorithms. In a competition for high school pupils, who had no formal training in expressing algorithms, it was an obvious choice to require that they write their algorithms as computer programs, which are machine executable. That way, a lot of discussion about the intention and validity of an algorithm can be avoided. It also nicely circumvents the language barrier[1] in one direction, because programming languages are quite universal.

[1] An international contest for high school students cannot rely on a single language for presenting the contest tasks. It must also find a way for the contestants to communicate their work to the jury for evaluation.

It may be comforting to some and sobering to others (myself included) that not much has changed with respect to the educational context. CS is still not equally accepted among other topics in the high school curriculum across the globe, and the means for the communication of algorithms has not much improved. Compare this to the 'maturity' of mathematics. It would be unimaginable to drop math from the high school curriculum (though, no doubt, there would be supporters for that), and it has a universally accepted abstract notation for expressing mathematical objects, questions, and arguments.

What has changed substantially in the IOI since its inception is the difficulty level of the contest problems, usually called 'tasks', and the evaluation mechanism to determine a score for submitted work. The nature of the tasks has seen an increase, albeit small, in diversity.

This article will walk you through the 20-year history of IOI tasks, all of which can be found via (IOI website, 2009). It is not intended as a judgment of the past, though I will take the opportunity to raise some critical notes. I will highlight various aspects, which you can view from different viewpoints, such as

Task author who invents and formulates a task;
Contest director who integrates the whole task *set*;
Team leader who votes about the tasks and translates them;
Contestant who is challenged to solve the tasks;
Judge who prepares the evaluation of the submitted work;
Coach who trains contestants to obtain their best result;
Educator who may later use tasks for pedagogical purposes;
Academia and industry who are potential employers of contestants.

The 20-year history will be broken down into three periods, whose characteristics are summarized in Table 1.

Table 1
The major periods in the evolution of IOI tasks

	1st Lustrum[2] 1989 – 1993	**2nd Lustrum[2] 1994 – 1998**	**2nd Decennium 1999 – 2008**
Hours per day	4[3]	5	
Tasks per day	1[4]	3	
Scoring principle	Partly subjective	Based on test runs only	
Grading method	Manual	Automatic	
Present at grading	Evaluator, leader, and contestant		No-one
Grading platform	Contestant computers		Central Linux server[5]
Task types added	Batch	Interactive	Output-only
Preparation	Host SC only		ISC supervision[6]

[2] A lustrum is a five-year period; a decennium a ten-year period.
[3] Except 5 on IOI'92 and IOI'93
[4] Except 3 on Day One of IOI'93
[5] Except IOI'99 and IOI'00
[6] Except IOI'99

This period of 20 years involved 101 competition tasks. Appendix A provides a tabular overview, which is explained in Section 5. It is impossible to cover all of the tasks here in any detail. Having talked to many IOI stakeholders in the past, it became clear that putting together a top-10 of "best" IOI tasks is not a reasonable endeavor. I have picked one 'representative' task from each period, and I apologize for the necessarily subjective nature of my choices.

2. First Lustrum: 1989–1993

The first IOI featured a single task on a single competition day. The next three IOIs each had two competition days with a single task, while at IOI'93 there were three tasks on Day One and only one on Day Two. The input-output requirements were specified rather loosely. The first three years, I/O was through a console interface (keyboard and screen), and thereafter through disk files. Grading was done manually by an evaluator who operated the program on the contestant's computer.

The first IOI task ever, the problem selected from six candidates for competition at IOI 1989, was described as follows, quoted from the on-line version and (Kenderov and Maneva, 1989; p. 21). It had no title.

Problem 1. Given $2 * N$ boxes in line side by side ($N \leqslant 5$). Two adjacent boxes are empty, and the other boxes contain $N - 1$ symbols `"A"` and $N - 1$ symbols `"B"`.
Example for $N = 5$:

```
-----------------------------------------
| A | B | B | A |   |   | A | B | A | B |
-----------------------------------------
```

Exchanging rule: The content of any two adjacent non-empty boxes can be moved into the two empty ones, preserving their order.

Aim: Obtain a configuration where all `A`'s are placed to the left of all `B`'s, no matter where the empty boxes are.

Problem: Write a program that:

1. Models the exchanging of boxes, where the number of boxes and the initial state are to be input from the keyboard. Each exchange is input by the number (from 1 to $N - 1$) of the first of the two neighboring boxes which are to be exchanged with the empty ones. The program must find the state of the boxes after the exchange and display it.
2. Given an initial state finds at least one exchanging plan, which reaches the aim (if there is such a plan). A plan includes the initial state and the intermediate states for each step.
3. Finds the minimal exchanging plan which reaches the aim.

Results: Present at least one solution for the example mentioned above.

Note that the task is composed of several related *subtasks*, allowing contestants to earn *partial credit* for partial accomplishments. In the first lustrum, the subtasks are usually based on the processing phases for the 'full' task, such as reading the input data into a suitably defined data structure, carrying out a single state transformation, writing the resulting state in a suitable format, etc.

An IOI task is not complete without a plan for grading the work submitted by the contestants. Kenderov and Maneva (1989, pp. 40-41) present the following *grading scheme* for the first IOI problem.

Four test examples were prepared by the Jury so that to check the program behavior in various cases. Each test example determined the value of N and the initial state as a sequence of A's, B's and zeroes for the empty boxes.

TEST EXAMPLE 1. $N = 5$, `0 0 A B A B A B A B`

The solution had to be obtained in 4 steps.

TEST EXAMPLE 2. $N = 5$, `A B B A 0 0 A B A B`

The minimal number of steps had to be 3.

TEST EXAMPLE 3. $N = 3$, `0 0 A B A B`

A message for no existence of a solution was expected.

TEST EXAMPLE 4. $N = 4$, `0 A B A 0 B A B`

A message for incorrect input data was expected.

The Jury decided the maximum number of points to be 100, which should be distributed as follows:

Subproblem 1. Up to 10 points.

Subproblem 2. Up to 40 points:

- up to 15 points for finding at least one plan or up to 20 points for all the plans found out;
- up to 20 points for reporting the lack of solution;

Subproblem 3.

- up to 15 points for an attempt made for optimization;
- up to 40 points for complete optimization.

Other 10 points were planned to be given in addition – 5 points if some results had been obtained after executing the program and 5 points for good programming style, and original solution, etc. (at decision of the Jury).

To assess the difficulty level of the tasks, one should keep in mind that in the first lustrum we were still dealing with Apple II and MS-DOS computers (CPU clock rates up to approx. 25 MHz[7]), limiting programs to 640 KB RAM (without resorting to trickery for accessing extended memory). The allowed programming languages were: Pascal, Basic, Logo, and later also C and (non-standardized) C++; at IOI'91, FORTRAN was available as well.

The first IOI problem would still make a nice IOI task, but not for an entire (5 hour) competition day. In a high-school programming course, it can provide material for several lessons, even nowadays. It has a concise formulation and its solution involves some important CS concepts, in particular, graphs. Note that Edsger Dijkstra (1959) published his, now famous, shortest path algorithm for graphs, which can be used in this task, precisely 50 years ago.

[7] Unfortunately, it has proved impossible to trace hardware details for the first lustrum.

3. Second Lustrum: 1994–1998

IOI'94 offered three tasks of diverse difficulty on both competition days, thereby establishing a tradition for the years to come (but that may be overturned at IOI'09).

At IOI'95, a new kind of task was introduced, referred to as *reactive* or *interactive* tasks, as opposed to the 'classical' (single-)*batch*-style tasks. In a batch task, all input data is available at the beginning of the run and it does not depend on the program's behavior. In a reactive task, some output must be produced before new input becomes available. That input may depend on the preceding output. The program has a dialogue with a (programmed) environment, which may behave as an *adversary*. When all input is predetermined but not completely available at the start, we speak of an *online* programming problem (the opposite situation is called *offline*).

The first reactive IOI task was named *Wires and Switches* (IOI'95, Day Two) and described as follows.

Wires and Switches. In Fig. 1, a cable with three wires connects side A to side B. On side A, the three wires are labeled 1, 2, and 3. On side B, wires 1 and 3 are connected to switch 3, and wire 2 is connected to switch 1.

In general, the cable contains m wires ($1 \leqslant m \leqslant 90$), labeled 1 through m on side A, and there are m switches on side B, labeled 1 through m. Each wire is connected to exactly one of the switches. Each switch can be connected to zero or more wires.

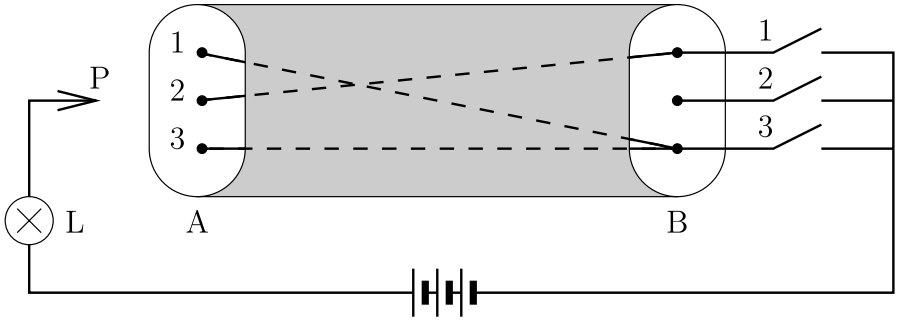


Fig. 1. Cable with three wires and three switches.

Measurements. Your program has to determine how the wires are connected to the switches by doing some measurements. Each switch can be made either conducting or non-conducting. Initially all switches are non-conducting. A wire can be tested on side A with probe P: Lamp L will light up if and only if the sensed wire is connected to a conducting switch.

Your program begins by reading one line with the number m from *standard input*. It then can give three kinds of commands by writing a line to *standard output*. Each command starts with a single uppercase letter: `T` (Test a wire), `C` (Change a switch), and `D` (Done). Command `T` is followed by a wire label, `C` by a switch label, and `D` by a list whose i-th element is the label of the switch to which wire i is connected.

After commands `T` and `C`, your program should read one line from *standard input*. Command `T` returns `Y` (Yes) when the wire's switch is conducting (the lamp lights up), otherwise it returns `N` (No). Command `C` returns `Y` if the new switch state is conducting, and `N` otherwise. The effect of command `C` is to change the state of the switch (if it was conducting then it will be non-conducting afterwards and vice versa); the result is returned just for feedback.

Your program may give commands `T` and `C` mixed in any order. Finally, it gives command `D` and terminates. Your program should give no more than nine hundred (900) commands in total.

Example. Fig. 2 presents an example conversation involving 8 commands relating to Fig. 1.

Standard Output	*Standard Input*
	3
C 3	Y
T 1	Y
T 2	N
T 3	Y
C 3	N
C 2	Y
T 2	N
D 3 1 3	

Fig. 2. Example conversation.

This task was devised by the author when moving into a new home, where the previous owners had lost the chart describing how light fixtures and wall outlets were connected to the switched fuse groups. A key idea for the solution is the binary search, another famous algorithm, which is often not fully appreciated (Feijen and van Gasteren, 1996; §12.3.3). The adversary used in evaluation attempted to minimize the amount of information conveyed at each query. For solutions and further details see (Verhoeff, 1995).

Starting with IOI'94, the grading process was automated so as to ensure that all submitted programs are given the same objective treatment. In preceding years, it had happened that a contestant program was allowed to continue running overnight in the hope that it would produce an answer. Setting an explicit limit on the run time also served to communicate the required efficiency level.

On the programming language front, Logo was dropped as of IOI'95, and Basic disappeared at IOI'98, leaving only Pascal and C/C++.

Another break with the past was the decision to base the score of submitted programs *solely* on a set of automatic *test runs*[8] with carefully constructed secret input data. This was mainly motivated by practical considerations, but is still a controversial issue (Verhoeff, 2006). Although this minimizes the role of human evaluators at the IOI, it does

[8]The organizers use these systematically designed test runs to quantitatively measure program quality. They should not be confused with the – often ad hoc – test runs executed by contestants.

increase the burden of thoroughly preparing the test runs in advance, a job that is often underestimated, and it limits what aspects can be taken into account.

As a consequence of evaluating by automatic test runs only, subtasks had to be defined in a compatible way. Just reading and storing the input or determining some configuration without outputting it cannot be evaluated by test runs. In the second lustrum, subtasks were typically defined by requiring additional output for related but simpler computations, and crediting these separately. Another way to obtain a partial score was to solve a subset of the (secret) input cases.

4. Second Decennium: 1999–2008

The team leaders vote about acceptability of proposed tasks shortly before the actual competition, and then the selected tasks have to be translated. IOI tasks involve a lot of work to prepare. In particular, the choice of bounds and other constraints, and the ingredients for a fair evaluation require ample consideration. The gradual increase in the ability of the contestants and accompanying increase in the diversity and difficulty level of the tasks has only complicated this further. Hence, it is not easy to assess proposed tasks quickly, and thus the team leaders came to face an impossible duty.

At IOI'99 the *International Scientific Committee* (ISC) was established to supervise the preparations for and development of future IOI competitions. The ISC has more time to assess proposed tasks and help ensure their quality. It reports its findings to the General Assembly of all team leaders, who can then take this into account when deciding on tasks.

In the second decennium, another new type of task was introduced. They became known as *output-only* tasks, where the contestants do not submit their programs but only the output files for several given (i.e., non-secret) input files. Of course, creation of the output files requires algorithmic thinking, and in most cases also considerable amounts of programming. In fact, each input file could potentially be tackled by one or more dedicated programs (for instance, a program to analyze and classify the input and a program to handle a particular class of inputs, using the analysis results).

Evaluation of submissions for output-only tasks does not involve programming languages and program compilation and execution. On the other hand, since programs are not collected, there is also no trace of what algorithms the contestants have developed.

The first output-only task was *Double Crypt* at IOI'01. Here is the description of the more interesting output-only task *XOR* (IOI'02, Day One).

XOR. You are implementing an application for a mobile phone, which has a black-and-white screen. The x-coordinates of the screen start from the left and the y-coordinates from the top, as shown in the figures. For the application, you need various images, which are not all of the same size. Instead of storing the images, you want to create the images using the phone's graphics library. You may assume that at the start of drawing an image, all pixels of the screen are white. The only graphics operation in the phone's library is `XOR(L,R,T,B)`, which will reverse the pixel values in the rectangle with top left coordinate (`L`, `T`) and bottom right coordinate (`R`, `B`), where `L` stands for the left, `T` for the top, `R` for the right and `B` for the bottom coordinate. Note that in some other graphics libraries the order of the arguments is different.

As an example, consider the image in Fig. 3 (right). Applying `XOR(2,4,2,6)` to an all white image gives the image on the left. Applying `XOR(3,6,4,7)` to the left image gives the image in the middle, and applying `XOR(1,3,3,5)` to the middle image finally gives the image on the right.

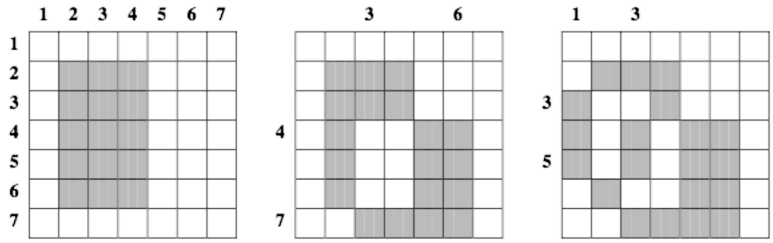


Fig. 3. Screen after each of three successive `XOR` operations.

Given a set of black-and-white pictures, your task is to generate each picture from an initially white screen using as few `XOR` calls as you can. You are given ten input files describing the images, and you are to submit files including the required `XOR` call parameters, not a program to create these files.

Output-only tasks were introduced to make certain classes of 'hard' problems acceptable at the IOI. Evaluation based on a limited set of test runs with secret inputs has the danger of biased results when the input space is 'convoluted'. In that situation, the test runs can explore only a – often very limited – subspace of allowed inputs, thereby possibly not covering corners where the contestant made mistakes or did exceptionally well (Forisek, 2006). An output-only task involves some specific non-secret inputs, so that the contestants know exactly which cases need to be handled.

XOR is such a 'hard' problem. In fact, the organizers did not know an efficient optimal algorithm. It is an *open-ended* task according to (Kemkes *et al.*, 2002). It does, however, have a nice approximation algorithm that is guaranteed to be no more than a factor 2 off the optimum. That is why this task used *relative scoring*, where contestant scores were based on how well they did compared to other contestants. Relative scoring was first featured in the memorable task *Toxic iShongololo* (IOI'97, Day One).

Subtasks as a means of offering an opportunity for partial credit came into disuse in the second decennium. Instead, partial credit could be obtained depending on program efficiency, by defining two or more *subclasses of inputs*, all of them requiring the same 'full' output. A typical way of classifying inputs is by their 'size'. 'Smaller' or 'simpler' inputs could be handled by less efficient, easier programs, whereas 'larger' or more 'complex' inputs required more sophisticated programs.

Test run clustering was introduced at IOI 2005 to reduce the opportunity for harvesting undeserved points by guessing and other forms of opportunistic programming that is not aimed at solving the actual computational task. The points for a cluster of test runs are awarded only if all test runs in the cluster are successful. More generally, the score for a cluster is defined as the minimum of the scores for the constituent test runs.

Another major change in this period (as of IOI'01) concerns the use of *centralized Linux servers* for grading. This allows better control over resources (time, memory, files,

network) used by submitted programs during the evaluation test runs. The contestants submit their work through a web interface to the contest support system, where evaluation takes place. Previously, test runs were executed on the contestant computers after the contest. Note that contestants would not necessarily be required to develop their programs under Linux. Since the central contest server and the contestants' development computers can differ (if only in configuration details), the contestants need a facility to do their own 'test runs' on the server, that is, have their program executed on the server with their own test input.

5. The 101 IOI Tasks

Appendix A presents an overview of the 101 competition tasks that appeared in the past 20 IOIs. There are many aspects one could want to summarize in such an overview, depending on one's background.

Task author: how much effort was needed for task creation; what alternatives, bounds, and other parameters were considered;
Contest director: how balanced was the task *set* as a whole;
Team leader: how much effort was needed for understanding, assessing the quality, and doing the translation;
Contestant: how much time was spent on understanding the task, on designing an abstract solution, on implementing it as an executable program;
Judge: how much effort was needed to prepare test data, checkers, and a summary of the design decisions behind the grading approach;
Coach: how easy were the results to understand and explain; what topics need attention in training;
Educator: how useful were the analysis and solutions; which parts could be used in school, covering what topics;
Academia and industry: to what extent did the tasks contribute to a better image of computer science.

Most of that information is hard to obtain or no longer available. Concerning the translation and comprehension effort, for example, one could measure the word, line, or page count of the task description. There is some variation in this length, ranging from half a page to three pages or more. But I find this metric too superficial. I have chosen to restrict myself to presenting information on

- task type,
- difficulty level (if data was available), and
- classification of technical features.

5.1. *Difficulty Levels*

Difficulty levels are distinguished on the basis of what percentage of contestants were able to 'fully' solve the task. We consider a submission scoring 90% or more as 'fully'

solving the task, i.e., modulo a 'small' mistake. The three main difficulty levels are: **easy** ($> 40\%$ 'fully' solved), **medium** (between 40% and 10%), and **hard** ($< 10\%$). The medium level is subdivided into three sublevels: medium-easy, medium-medium, and medium-hard; see Table 3.

This is admittedly a somewhat arbitrary definition of difficulty level and it is not an absolute measure, but relative to the actual population of contestants for a particular IOI. It is, however, objective and these same criteria have been used in various IOI questionnaires.

Unfortunately, scores per contestant per task are not available for all IOIs. The difficulty level of tasks for a particular IOI can be assessed together as a *set*, by considering the cut-off scores for the medals. Table 9 shows these scores relativized to the maximum score. Keep in mind that according to the IOI Regulations (no more than) half of contestants receive a medal, where the ratio of the number of bronze, silver, and gold medals is $3 : 2 : 1$. That is, approximately one half of all contestants receive no medal, one quarter a bronze medal, one sixth a silver, and one twelfth a gold medal. The lower the relative medal cut-off scores, the fewer points were needed to obtain medals, the harder the task set was for that population of contestants.

5.2. *Task Classification*

The classification concerns these three aspects:

1) the given context and input,
2) the computational task and output,
3) the (algorithmic) ingredients of a full-scoring solution.

The first two items together characterize the *problem domain*, and the third item the *solution domain*. In the overview tables, these three items are separated by semicolons. The terminology follows common practice; for instance, see (Skiena, 2008). It is worthwhile to consider integration into (Verhoeff, 2004) and (Verhoeff *et al.*, 2006). The classification is not entirely satisfactory; some tasks are hard to classify concisely.

Note that Kyryukhin and Okulov (2007) provides technical information (in Russian) on all competition tasks of the first 18 IOIs, including the task descriptions, analyses, solution guidance, classifications, and code snippets (both pseudo code and Pascal implementations). I have occasionally consulted this useful reference when making the classification in Appendix A, but my classification is not the same.

6. Conclusion

The past 20 IOIs involved 101 competition tasks, covering a wide range of CS topics, task types, and difficulty levels. I have summarized them in Tables 5 through 8. I recommend that these tables are verified, refined, and extended with further information, and are kept up to date. In particular, it would be interesting to refine the difficulty assessment, by separately measuring the difficulty of

- *comprehension*, for instance, in terms of number of concepts and definitions involved;
- *mathematical analysis*, for instance, by number and nature of key properties (lemmata) to be discovered;
- *algorithm design* (the focus of this article);
- *implementation* (mostly ignored in this article).

It would also be useful to have a cross-reference of the classification, which lists for each class all related tasks. An on-line data base comes to mind.

Many of the tasks are too hard to use 'as is' in regular CS courses for secondary education. The kind of algorithmics that nowadays plays a role at the IOI is too advanced for incorporation in the high-school curriculum. When looking at the difficulty level of the individual tasks and at the cut-off scores for medals, there is an obvious trend towards relatively harder problems, that is, towards problems and problem sets that are solved by a decreasing percentage of contestants. I do not think this is a good development.

Table 2 lists the five tasks that I have contributed to the IOI competitions.

While analyzing the past, it became painfully clear that the IOI community needs to do a better job at preserving its historic record. All task descriptions are available online[10], but other information is often lacking. In particular, it is desirable to know

- computing platforms and other constraints;
- results *per task* for *all* contestants (possibly anonymized);
- test data (preferably in digital form), checkers, and motivation;
- problem analyses, design options and decisions, exemplary[11] pseudo code and documented program texts[12].

In most cases it is not easy or plainly impossible to (re)evaluate your own attempt at a solution according to the rules of that IOI. Consult (Verhoeff, 2008) for further suggestions on this topic.

The next decennium will certainly see new developments. Computer hardware and programming languages evolve: object-oriented and component-based software on multi-

Table 2
IOI tasks contributed by the author

Task	**Year**	**Remarks**
Wires and Switches	IOI'95	Interactive, see (Verhoeff, 1995)
Median Strength	IOI'00	Interactive, see (Horváth and Verhoeff, 2002)
Double Crypt	IOI'01	Output only, uses AES[9]
Reverse	IOI'03	Output only, output is a 'program'
Mean Sequence	IOI'05	Easy, non-trivial

[9] Advanced Encryption Standard (established as a NIST standard in 2002).
[10] For IOI'90 it unclear on the (IOI website, 2009) which were the actual competition tasks.
[11] worthy of imitation
[12] Kyryukhin and Okulov (2007) come a long way in providing this information for the first 18 IOIs.

core networked processors with interactive multimedia user interfaces require multi-threading, distributed algorithms, and network protocols. The IOI does not have to follow these trends, but if the aim is to stimulate youthful talent, then it is advisable to investigate the possibility of attractive tasks involving newer technologies.

The programming languages that may be used in the IOI competition are now restricted to Pascal and C/C++. Java and Python have been around for more than ten years and are quite popular and accessible. It will be interesting to see how the IOI evolves on the language front, striking a balance between expressing algorithms elegantly and implementing them efficiently for actual execution.

Acknowledgments

The anonymous reviewers provided critical and helpful feedback on the first version of this paper. I would like to thank Gyula Horváth for assisting in the task classification and for collecting the score data for Table 9.

A Tabular Overview of Tasks and Task Sets

Tables 5 through 8 list all 101 IOI competition tasks of the past 20 years together with task type, difficulty level (where objectively assessable), and a classification. Table 3 explains the codes for task types and difficulty levels. Abbreviations for the classification are listed in Table 4. Table 9 shows the relative cut-off scores for medals, which can be used to assess the difficulty level of the task sets as a whole. For more detailed explanations, see Section 5.

Table 3

Type and difficulty codes used in Tables 5 through 8

Code	Task Type
B	Batch
I	Interactive
O	Output only
T	Theoretical

Code	Difficulty	'Fully' solved by[13]
E	Easy	40% – 100%
M	Medium-Easy	30% – 40%
M	Medium-Medium	20% – 30%
M	Medium-Hard	10% – 20%
H	Hard	0% – 10%

[13] Percentage of contestants scoring $\geqslant 90\%$ on the task.

Table 4

Classification abbreviations used in Tables 5 through 8

Code	Problem Features
Ari	Arithmetic
CG	Computational Geometry
Cnt	Counting
Comb	Combinatorial
Dist	Distance
DS	Data Structure
Enum	Enumerating
FSM	Finite State Machine
Gm	(Combinatorial) Game
Gr	Graph
Ham	Hamilton
i	implicit/implied
Mh	Manhattan
Mtch	Matching
Num	Number (integer)
Opt	Optimization
(un)Rank	(Un)Ranking
Rect	Rectangle
Sch	Scheduling
Srch	Searching
Seq	Sequence
Srt	Sorting

Code	Solution Features
Approx	Approximation
BB	Branch & Bound
BFS	Breadth First Search
BS	Binary Search
BT	Backtracking
DC	Divide & Conquer
DFS	Depth First Search
DP	Dynamic Programming
ES	Exhaustive Search
Exp	Exponential
Grdy	Greedy
MI	Mathematical Insight
MM	Meet in the Middle
MST	Min/Max Spanning Tree
(N)P	(Nondet.) Polynomial
Heu	Heuristics
Hash	Hashing
L(A)	Linear (Algebra)
Pc	Precomputation
Rec	Recursive
SP	Shortest Path
SL	Sweep/Scan Line

Table 5

Overview of competition tasks in first lustrum

1. IOI 1989, Pravetz, Bulgaria			
[Exchanging Boxes]	B	**M**	Seq; Srt, Opt; iGr, BFS
2. IOI 1990, Minsk, Belorussian Republic, SU			
[Sliding Puzzle]	B	**H**	2d Num Grid; Comb Opt; iGr, BT
[Watchmen]	B	**H**	Num Seq; Sch, Opt; ES
3. IOI 1991, Athens, Greece			
Square Problem	B	*E*	2d Grid; Ham cycle, Cnt; iGr, BT
S-Terms Problem	B	**H**	Grammar; DS, Enum, String rewriting; _
4. IOI 1992, Bonn, Germany			
Islands in the Sea	B		Nonogram (logic puzzle); Solve; BT
Climbing a Mountain	B		Constraints; Sch, Opt; BT
5. IOI 1993, Mendoza, Argentina			
[Necklace]	B		Color cycle; Cut Opt; L
[Company Shares]	B		Weighted Gr; Enum; all-pair SP
[Rectangles]	B		2d CG, Rect; Areas; z-Buffer, DFS
[Itinerary]	B		Gr; Cycle Opt; DP

Table 6
Overview of competition tasks in second lustrum

6. IOI 1994, Haninge, Sweden			
The Triangle	B		Num triangle; Opt; iGr, DP
The Castle	B		2d Grid; Cnt, Opt; iGr, DFS
The Primes	B		Magic digit square; Primes, Enum; Pc, BT
The Clocks	B		FSM, Sch, Opt; MI, LA
The Buses	B		Num Seq; Sch, Opt; BB
The Circle	B		Num cycle, Opt, Enum; BT
7. IOI 1995, Eindhoven, The Netherlands			
Packing Rectangles	B	*M*	2d CG, Rect; Area, Opt, Enum; ES
Shopping Offers	B	*M*	Constraints; Comb Opt; DP
Printing	T	M	Program analysis & modification
Letter Game	B	*E*	String list; Opt, Enum; ES
Street Race	B	*M*	Gr; Vertex Cnt, Enum; P
Wires and Switches	I	*M*	Find mapping; BS
8. IOI 1996, Veszprém, Hungary			
A Game	I	*E*	2p Gm; Sum Opt; MI+L or DP
Job Processing	B	**H**	Constraints; Sch, Opt; Grdy, MM
Network of Schools	B	**M**	Gr; Opt; DFS
Sorting a 3-valued Sequence	B	*E*	Num Seq; Srt, Opt; Srt, P
Longest Prefix	B	*M*	String list; Opt; Pc, P
Magic Squares	B	*M*	FSM; Event Seq; iGr, BFS
9. IOI 1997, Cape Town, South Africa			
Mars Explorer	B		2d Grid; Opt; iGr
Game of Hex	I		2p Gm; Moves; Heu
Toxic iShongololo	B		3d Grid; Opt; Heu
Map Labeling	B		2d CG; Rect placement; Heu
Character Recognition	B		2d Image; Approx Mtch; Heu
Stacking Containers	I		3d CG, unit cubes; Heu
10. IOI 1998, Setúbal, Portugal			
Contact	B		String; Cnt, Enum; Histogram, Srt
Starry Night	B		2d Image; Subshape Mtch; ES
Party Lamps	B		FSM; Enum; Linear Algebra, Brute Force
Picture	B		2d CG, Rect; Perimeter; SL
Camelot	B		2d Grid, 1p Gm; Opt; iGr, ES
Polygon	B		Number cycle, Ari; Opt, Enum; DP

Table 7
Overview of competition tasks in third lustrum

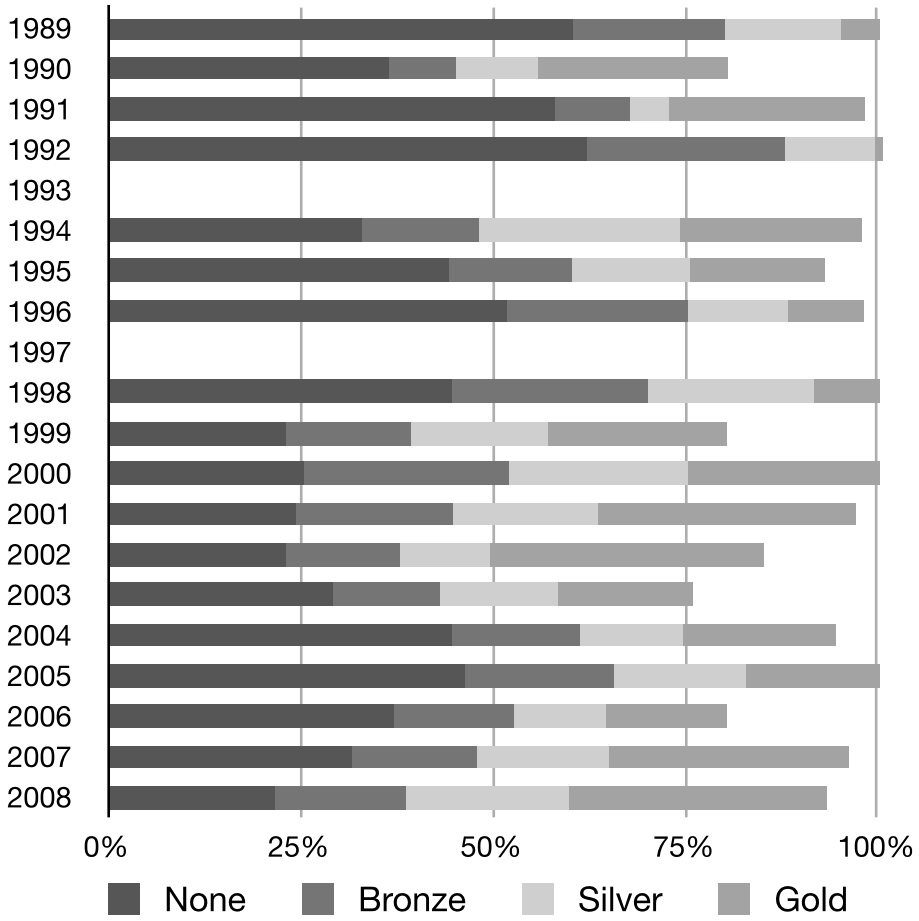
11. IOI 1999, Antalya-Belek, Turkey			
Little Shop of Flowers	B		Constraints; Comb Opt; DP
Hidden Codes	B		String list; Mtch, Opt; P
Underground City	I		2d Grid, implicit maze; Find loc; ES
Traffic Lights	B		Weighted Gr; Path Opt; iGr, SP
Flatten	B		1p Gm, Num Seq; Move Seq, Opt; iGr, LA, NP?
A Strip of Land	B		2d Num Grid; Rect, Area Opt; P
12. IOI 2000, Beijing, China			
Palindrome	B	M	String; Opt; DP
Car Parking	B	**M**	Num Seq; Srt, Opt; Srt, Grdy
Median Strength	I	**M**	Implicit Num Seq; Find median; Heap
Walls	B	M	Planar iGr; Opt; Dual Gr, all-pair SP
Post Office	B	M	1D CG; Comb Opt; DP
Building with Blocks	B	**H**	3d Grid; Set Packing Opt, Rot/Refl; BB
13. IOI 2001, Tampere, Finland			
Mobile Phones	I	**H**	2d Grid; Rect Cnt; Binary Tree
Ioiwari Game	I	M	2p Gm; Win; iGr, minimax
Twofive	B	**H**	String, Num, iSeq; (un)Rank; MI, Pc, DP
Score	I	**M**	2p Gm, Labeled Gr; Win; DFS, minimax
Double Crypt	O	M	Crypto func; Find keys; Hash, MM
Depot	B	**H**	2d Grid, Num Seq; Inverse, Enum; BT
14. IOI 2002, Yong-In, Korean Republic			
The Troublesome Frog	B	**M**	2d CG; Line, Opt; DP, Hash
Utopia Divided	B	**H**	2d CG, Num Seq; Sch; MI, DC, Srt
XOR	O	**H**	2d Grid; Rect, Opt; NP?, Approx
Batch Scheduling	B	**H**	Constraints; Sch; DP
Bus Terminals	B	**H**	2d CG, Mh Dist; Opt; P
Two Rods	I	**H**	2d CG; Rect, Find 2 segments; BS
15. IOI 2003, University of Wisconsin Parkside, U.S.A.			
Trail Maintenance	I	*M*	Gr; Subgraph Opt; MST
Comparing Code	B	**H**	String list; Mtch, Opt; P
Reverse	O	**H**	FSM; program; NP
Guess Which Cow	I	**H**	2d Grid; Query Opt; BFS, bit level
Amazing Robots	B	**H**	2d Grid, Sch; Solve maze; iGr, BFS
Seeing the Boundary	B	**H**	2d CG, Polygons; Visibility; polar SL

Table 8
Overview of competition tasks in fourth lustrum

16. IOI 2004, Athens, Greece			
Artemis	B	**H**	2d Bit Grid; Rect, Opt; Pc, DC, Srt
Hermes	B	M	2d CG; Dist Opt; DP
Polygon	O	**H**	2d CG; inverse Minkowski sum; NP
Empodia	B	M	Num Seq; Subsequence Enum; MI, L
Farmer	B	*M*	Num Seq; Comb Opt; DP
Phidias	B	*M*	2d CG, Rect; Cut Opt; DP
17. IOI 2005, Nowy Sącz, Poland			
Garden	B	**M**	2d Grid; Rect, Opt; Pc, SL
Mean Sequence	B	*E*	Num Seq; Cnt Num Seq; L
Mountain	B	**H**	Num Seq; Answer queries; Binary Tree
Birthday	B	**M**	Num Cycle; Comb Opt; L
Rectangle Game	I	*M*	Gm, Rect, Cut; Win; MI
Rivers	B	**M**	Num Tree; Comb Opt; DP
18. IOI 2006, Mérida, Yucatán, Mexico			
Forbidden Subgraph	O	**H**	Gr; Subgraph Opt; NP
Pyramid	B	**M**	2d Num Grid; Rect, Opt; Pc, Binary Tree
Mayan Writing	B	*E*	String; Cnt; L
A Black Box Game	IO	**H**	2d Grid; Find configuration; Rec, Exp
The Valley of Mexico	B	M	Gr; Planar Ham path; DP
Joining Points	B	**H**	2d CG, 2 Point sets; 1p Gm; Rec, DC
19. IOI 2007, Zagreb, Croatia			
Aliens	I	M	2d Grid; Find center; BS
Flood	B	**H**	2d Mh CG; Segment Enum; dual iGr, BFS
Sails	B	**H**	Num Seq; Comb Opt; Grdy, Diff Tree
Miners	B	*E*	String; Sch, Opt; DP
Pairs	B	**H**	1d/2d/3d Mh CG; Pair Cnt; Srt, SL
Trainings	B	**H**	Gr, Tree; Even Cycle, Opt; DP
20. IOI 2008, Cairo, Egypt			
Type Printer	B	*E*	String list; Sch, Opt; Trie, DFS
Islands	B	**H**	Weighted Gr; Path Opt; MI, Tree diameter
Fish	B	**H**	iGr; Comb, Cnt; MI, Srt, Binary Tree
Linear	B	*M*	String, iSeq; Comb, Rank; MI, Pc, DP
Teleporters	B	**H**	1d CG; Motion Opt; iGr, DFS/BFS, Grdy
Pyramid Base	B	**H**	2d CG; Rect, Area Opt; MI, BS, SL, DS

Table 9

Relative cut-off scores for medals (no data available for 1993, 1997)



References

Dijkstra, E.W.D. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**, 269–271.

Feijen, W.H.J. and van Gasteren, A.J.M. (1996). Programming, proving, and calculation. In C.N. Dean and M.G. Hinchey (Eds.), *Teaching and Learning Formal Methods*, Academic Press, pp. 197–243. §12.3.3: `http://www.mathmeth.com/wf/files/wf2xx/wf214t.ps` (accessed May 2009)

Forisek, M. (2006) On the suitability of programming tasks for automated evaluation. *Informatics in Education*, **5**(1), 63–76.

Horváth, G. and Verhoeff, T. (2002). Finding the median under IOI conditions. *Informatics in Education*, **1**(1), 73–92.

International Olympiad in Informatics Website. `http://www.IOInformatics.org/` (accessed May 2009)

Kemkes, G., Cormack, G., Munro, I. and Vasiga, T. (2007). New task types at the Canadian computing competition. *Olympiads in Informatics*, **1**, 79–89.

Kenderov, P.S. and Maneva, M.N. (Eds.) (1989). In *Proceedings of the International Olympiad in Informatics*,

Pravetz, Bulgaria, May 16–19. Union of the Mathematicians in Bulgaria, Sofia.

Kiryukhin, V. and Okulov, S. (2007). *Methods of Problem Solving in Informatics: International Olympiads*. LBZ (BINOM. Knowledge Lab), Moscow (in Russian). `http://www.lbz.ru/` (accessed May 2009)

Skiena, S.S. (2008). *The Algorithm Design Manual* (2nd ed.). Springer-Verlag. `http://www.algorist.com/` (accessed May 2009)

Verhoeff, T. (1995). The lost group chart and related problems. In *Simplex Sigillum Veri, A Liber Amicorum for Prof. Dr. F.E.J. Kruseman Aretz*. Faculty of Mathematics and Computing Science, Eindhoven University of Technology. December 1995, pp. 308-313. `http://www.win.tue.nl/ wstomv/publications/kruseman.pdf` (accessed May 2009).

Verhoeff, T. (2004). *Concepts, Terminology, and Notations for IOI Competition Tasks*. `http://www.win.tue.nl/wstomv/publications/terminology.pdf` (accessed May 2009)

Verhoeff, T. (2006). The IOI is (not) a science olympiad. *Informatics in Education*, **5**(1), 147–159.

Verhoeff, T., Horváth, G., Diks, K. and Cormack, G. (2006). A proposal for an IOI Syllabus. *Teaching Mathematics and Computer Science*, **IV**(1), 193–216. `http://www.win.tue.nl/ wstomv/publications/ioi-syllabus-proposal.pdf` (accessed May 2009)

Verhoeff, T. (2008). Programming task packages: peach exchange format. *Olympiads in Informatics*, **2**, 192–207.

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.