Olympiads in Informatics, 2011, Vol. 5, 12–25

Survey on Informatics Competitions: Developing Tasks

Lasse HAKULINEN

Department of Computer Science and Engineering, Aalto University School of Science
P.O.Box 15400, FI-00076 Aalto, Finland
e-mail: lasse.hakulinen@aalto.fi

Abstract. Informatics competitions offer a motivating way to introduce informatics concepts to students and to find new talents. There are many different competitions in the field of informatics with different objectives. In spite of these differences, they all share the same need for high quality tasks. Tasks can be seen as the heart of the competition, so the effort put in developing new tasks should not be underestimated. In this survey, the development of tasks and different task types are discussed. The focus is on the Olympiads in Informatics competitions, but tasks in other competitions are discussed as well.

Key words: task development, competition tasks, informatics competitions, IOI, competitions and education.

1. Introduction

Motivating students to learn is important for all educators. Competitions offer a convenient way to bring informatics concepts to students in a different fashion than regular teaching in schools and universities. It could be said that the tasks are the heart of the competition. Therefore, designing tasks that support the goals of the competition is an important and demanding undertaking.

Nowadays, there are many different informatics competitions from small to worldwide events. Also, the types of tasks vary from tasks solved with pen and paper to complex problems dealing with large datasets and sophisticated algorithms. Many different types of events offer a wide range of possibilities for students to get involved with informatics.

Competitions can be a place for students to learn new concepts on informatics. They can also be a source of motivation for students interested in problem solving, programming and other aspects of informatics. Students who get interested in programming contests are usually looking for a place for training their skills and to gain some informatics education (Dagienė, 2010).

In this survey, the development of tasks and different task types are discussed. There are a lot of different competitions in the field of informatics and therefore the variety of tasks is wide. This paper describes the tasks used in different competitions, but focuses on

the ones used in Olympiads in Informatics and similar competitions. Also, the educational aspect of competitions is discussed.

Different informatics competitions are introduced in Section 2. Task types are described in Section 3 and the content of the tasks in Section 4. In Section 5, the process of developing new tasks is discussed. The usage of competitions in education and suggestions for task development are discussed in Section 6. Finally, the conclusions of competition tasks and their development are presented in Section 7.

2. Informatics Competitions

There are several informatics competitions held worldwide and the variety of the competitions is wide. Also, the fundamental purposes of the competitions vary. The purpose can be, for example, testing competitors' knowledge, enabling learning, finding especially talented competitors, or promoting informatics. Because the focus of the competitions varies, also the tasks used in different competitions have different characteristics. In addition, the target group and required preliminary knowledge on informatics can be different in each competition.

International Olympiad in Informatics (IOI Website, 2010) is a well known competition that has been organized annually since 1989. Naturally, the competition and tasks have evolved during the last two decades, but the high-level goal is still promoting computer science among the youth, and discovering and stimulating talented students (Verhoeff, 2009). Tasks in IOI concern algorithmic problem solving and they must be implemented in one of the few specified programming languages.

Another well known competition is the ACM International Collegiate Programming Contest (ICPC) where student teams solve programming tasks of varying difficulty (ACM, 2010). In ICPC, teams of three competitors can use one computer to solve several algorithmic programming tasks. The difficulty levels of the tasks vary so that the teams must prioritize the time used for each task.

There are also many other informatics competitions that all have their own characteristics and objectives. Following is a list containing some popular competitions and their characteristics:

- TopCoder: Several different types of competitions from short algorithmic tasks to marathon matches that can last for weeks (TopCoder Website, 2011).
- Google Code Jam: Algorithmic programming problems (Google Code Jam Website, 2011). Initially established in order to find top talents to work at Google (Bigmouth media, 2003).
- ICFP: Programming contest where participating teams can be of any size and they can use any programming languages (ICFP Website, 2011). Teams will have 72 hours time to complete the tasks.
- Imagine cup: Teams are supposed to show how technology can be used to solve the world's toughest problems (Imagine Cup Website, 2011).

Many of the competitions involve quite difficult tasks and are meant for advanced students or even professionals. However, there are also competitions that are meant for any

students interested in informatics regardless of their skill levels. One such competition is Bebras (Dagienė and Futschek, 2008), which is aimed for pupils from lower secondary to upper secondary level.

3. Task Types

3.1. *Tasks with Programming*

Many of the tasks in informatics competitions involve programming. In these tasks, competitors are asked to return a working program code for a certain problem. Typically, the tasks deal with large data sets that have to be manipulated using sophisticated data structures and algorithms. With programming tasks, it is also possible to give instant automatic feedback to the competitors and allow them to return the same task multiple times until they get the right solution.

3.2. *Tasks without Programming*

It is easy to see why programming tasks are a natural choice for task types in informatics competitions. However, there are significant advantages of using non-programming tasks that should not be overlooked. On the other hand, when using non-programming tasks, there are also obstacles that must be addressed different ways than in programming tasks.

One issue in organizing informatics competitions can be the resources, if the use of computers is required. When using tasks that do not require programming, it is possible to organize large scale competitions with pen-and-paper tasks. Burton (2010) describes how the Australian Informatics Competition (AIC) have only pen-and-paper tasks and hence it is accessible to a much broader audience than programming competitions. He says that the greatest difficulty with pen-and-paper informatics competitions is to retain the focus on algorithms and algorithmic thinking. In AIC, the tasks are either multiple-choice questions or the answer can be given as an integer.

Kubica and Radoszewsk (2010) suggest the use of tasks that require algorithmic thinking, but no programming in order to attract students who know nothing about programming or algorithms. They claim that offering tasks or puzzles requiring various levels of algorithmic thinking is a good way to popularize learning programming among young pupils. They provide a couple of problems that require algorithmic thinking but that are formulated in a purely mathematical manner. The problems are designed so that the desired solution is the one that minimizes the total time of inventing it and executing it by hand. The trivial solutions are usually more time consuming to perform. They also suggest that problems that require the knowledge of classical algorithms and advanced techniques should be excluded from the competitions.

Another interesting way to promote informatics among students is described by Bell *et al.* (1998). They introduce a set of activities called *Computer Science Unplugged* that are designed to teach school students about computer science concepts and awake their interest on computer science without using computers at all. They claim that a common

misconception is that computer science is only about programming. Computer Science Unplugged addresses that misconception by presenting the ideas and issues of computer science with activities without computers.

Computer Science Unplugged is not a competition in itself, but its activities can be combined with competition tasks. Voigt *et al.* (2010) combined the Computer Science Unplugged approach to competition-style programming problems. By using both approaches, they wanted to reinforce the concepts students learn using the unplugged approach, and on the other hand, to connect programming with computer science concepts. In the teaching sessions, they first discussed the topic with unplugged activities. After that, the students were given several small competition-style programming task dealing with the same topic. By testing the students' knowledge before and after programming tasks, they found out that the students' performance on the test after the programming activity was significantly improved. Also, students responded mostly positively to the combination of unplugged activities and programming tasks.

3.3. *Code Understanding*

One possibility for a task type that is not widely used in competitions, is using tasks that require code reading skills. In real life, programmers often have to read source code made by other people. In this kind of tasks, memorizing well known algorithms by heart would not be as beneficial as in the traditional programming tasks. The contestants have to really understand the given code in order to improve or analyze it. Code understanding tasks can be used in both, programming and non-programming, types of tasks.

Opmanis (2009) described the use of code understanding tasks in the *Ugale* competition. Competitors were given a code and the assignment was to find an error, construct the worst case counterexample, estimate the complexity, or implement a more efficient version of the same algorithm. Also the Syrian Olympiad in Informatics included tasks that had some initial code, which had to be improved (Idlbi, 2009). They used *Scratch* to motivate younger students to participate and included tasks with some initial game elements that needed to be improved.

3.4. *Subtasks*

With some of the tasks, it might be difficult to get a nice score distribution. It may be that the students who are close to solving the task get as few points as the students who do not have a clue about the right solution. Often it is desirable to give some kind of reward to the competitors who understand the concepts of the tasks even if they are not able to provide completely correct answer.

Vegt (2009) explains how subtasks are used in the Dutch Olympiad in Informatics to get a nice score distribution and to reward contestants for what they were able to solve. Using subtasks also opens up new opportunities. For example, Vegt says that when using subtasks, it is easier to slip in a more theoretical question as a subtask.

4. Content of the Tasks

Different task types mentioned in the previous section should be considered when designing new tasks for a competition. In an ideal case, the task type and the content of the task would support each other so that the goals of the competition will be met most efficiently.

When developing tasks for competitions, the wide range of concepts covered in informatics should be borne in mind. Naturally, all competitions have their own goals and the content of the tasks should be build to meet the goals. Nevertheless, when considering the whole spectrum of informatics competitions, it would be desirable to cover all the major concepts of informatics. In real life, some concepts appear in tasks more often than others.

Some task topics that are common in many informatics competitions are, for example: algorithms, programming, problem solving, logic and graphs. Verhoeff (1990) points out that it is important to cover a wide range of informatics aspects in the task set for two reasons. Firstly, it is unfair if "specialists" in certain field have an advantage, because one aspect appears in many of the tasks. Secondly, competitors can work on the types of problems that they like the most. The latter is especially important for the weaker competitors. When the range of topics is wide, weaker contestants have enough topics to choose from allowing them to focus on the tasks that they are most comfortable with.

Constructing a suitable set of tasks for a competition can be a demanding assignment. Verhoeff *et al.* (2006) present a very detailed list of informatics concepts to be used in IOI tasks in their proposal for an IOI syllabus. Opmanis (2009) provides a different type of list of topics used in the *Ugale* competition. The list by Opmanis contains typical topics but also some unusual categories such as a category called *Dominoes* that has tasks using a Dominoes game as a setting for combinatorial tasks.

There are some topics that are rarely used in competitions even though they are an important part of informatics. Tru and Ivanov (2008) state that software testing is a major area of software development that is mostly neglected in the informatics competitions. They examine the main modes of operation of software testers and discuss their suitability as the basis for competition tasks. They discuss the suitability of black-box and white-box testing, as well as static and dynamic testing techniques. They analyze a set of tasks involving testing and conclude that software testing can be a basis for a good competition task.

Criminal activity and misuse of computers is nowadays a big problem worldwide. Therefore, also ethics should be an important part of informatics education. Futschek and Dagienė (2009) address this issue by stating that proper behavior of computers should be part of all Bebras task sets. As an example, they present a task dealing with junk mail where you are asked for the correct way of acting when receiving an e-mail requesting you to forward it to your friends.

4.1. *Games and Puzzles*

The content of the task can also have a big influence on the motivation towards completing it. Carefully selected topic can make the understanding of the problem easier and help the competitors to focus on the core problem of the task rather than on a long task description.

Using games is one way to make the competition tasks more attractive for students. Ninka (2009) discusses the use of reactive and game tasks in informatics competitions. He points out that when using a game, there is no need for a story, and therefore students save some time getting into the actual problem. Also, students have some prior experience of games, which helps them to get motivated. He gives an example of the students' high motivation towards game tasks, where students continued pondering a game used in a competition while waiting for their flight and managed to improve their initial solution. Ninka claims that students are more biased to discuss game tasks after the contest with the aim to discover what they may have missed during the contest.

Ninka also points out how the number of students who are fond of algorithms and programming has reduced. On the other hand, he says that the situation is quite different when students have to program a game. This motivation towards games could be used to promote informatics competitions and also to get students interested in algorithms.

Vegt (2006) explaines how games, algorithmic thinking and competitive programming are combined in CodeCup. CodeCup is an annual game programming competition that is a side event for the Dutch Informatics Olympiad (NIO). In CodeCup, competitors are supposed to write a program that can play a certain game. Vegt says that often the games chosen for the competition do not have a known perfect solution. That way they encourage competitors to develop their own original ideas, rather than implementing known algorithms.

5. Task Development

It is very important for a competition to have a decent set of tasks in order to be successful. Especially when there are so many different informatics competitions held each year, it may not be easy to find new and genuine ideas for interesting and educational tasks. So, how to develop good task and what should be borne in mind when designing new tasks?

5.1. *Features of a Good Task*

Burton and Hiron (2008) define several features of a good informatics olympiad task. However, they remind that the desired features may vary depending on the goals and target group of the competition. Similar features are also discussed by Diks *et al.* (2007). Following is a summary of the features defined by Burton and Hiron:

- The problem statement should be short and easy to understand.
- The algorithms that solve the task should not directly resemble a classic algorithm or a known problem. However, it can be a modification of a classic algorithm.
- There should be several different valid solutions of varying difficulty and efficiency in order to allow weaker students to gain partial marks for the task.
- The official solution should be reasonably concise.
- In most cases (depending on the difficulty of the task), the official solution should also be the best known solution for the task.

- For the experienced students, it is nice to have tasks where it is not obvious in what category the desired algorithm belongs to.

Tasks do not need to be difficult or complex in order to be useful in competitions. Dagienė and Futschek (2008) discuss criteria for good tasks in the Bebras International Contest on Informatics and Computer Literacy. The tasks are not programming tasks, but they are solved using a computer. The two task types in Bebras competitions are interactive and multiple-choice questions. The tasks are generally fairly short because one of the criteria of a good Bebras task is that it can be solved in 3 minutes. The main idea behind Bebras is not to ask for already learned facts but to give problems that allow students to learn something about concepts that might be new for them (Futschek and Dagienė, 2009).

5.2. *Sources of Inspiration for New Tasks*

Burton and Hiron (2008) discuss different techniques that can be used when searching for new ideas for tasks and refine these ideas into problems suitable for an informatics olympiad. Perhaps the most difficult part of creating a new task is getting the initial idea of the task. They say that one of the most common techniques is to look around and take inspiration from things that you see in real life. They also suggest an alternative approach where you could find somebody unfamiliar with computer science and ask them for ideas. They also point out that one of the disadvantages of looking around for inspiration is that you do not have a solution in mind. Therefore, the task creation process can be very time consuming when you have to also find the solutions for all the possible task candidates. Third source of inspiration they introduce, is to use and modify a problem one faces in a daily work, e.g., a small piece of a complex research problem or an unconventional algorithm. An advantage of this method is that often the solution is in the research paper or you have solved it yourself.

Tasks do not need to be generated from scratch every time. Burton and Hiron also suggest combining old tasks when designing new tasks. They illustrate how to find the characteristics of old tasks and find a combinations of characteristics that have not yet been used when designing a new task. Another point of view they pointed out when designing new task is to start with a standard algorithm and set a task that modifies it in some way.

Maybe one of the most exotic ideas by Burton and Hiron is to start by drawing random things on a blank piece of paper. By adding objects and relations you might start to get vague ideas of where you are heading and what could be the problem statement of the task. They also propose searching ideas from other disciplines such as mathematics or from games and puzzles. They state two ways games and puzzles are useful. First, they provide ready-made tasks such as playing a game optimally or solving a puzzle using the smallest number of moves. Second, they can supply interesting sets of objects and rules that can act as starting points for other tasks.

Also Pankov (2008) suggests the use of real things around us to get inspiration for new tasks. He claims that using a real situation or task is more preferable than starting

with an effective algorithm and composing a task with a corresponding subject. In his opinion, that is because using real life situations can yield original tasks with natural, short and elegant formulations and also give less advantage to experienced participants.

Pankov (2010) also says that natural sciences contain many interesting laws and facts that could be used as a base for competition tasks in informatics. He offers three types of tasks that can be set if the properties of an object are given: combinatory, optimization and interaction. As an example, he shows how the conservation laws can act as a basis for a competition task where competitors must find the smallest possible velocity of merged pointwise objects.

Many of the data structures used in competition tasks relate closely to real life. This way students do not necessarily need to be familiar with the data structures beforehand. Graphs are widely used in tasks of all difficulty levels. Graphs are an important origin of tasks as they are modeling real life and therefore the tasks become easy to understand also for younger students (Manev, 2008).

5.3. *Task Preparation Process*

Getting the initial idea or creating the first prototype of the task is just the beginning when preparing a new task for a competition. Diks *et al.* (2008) propose best practices that should be applied in a task preparation process for any programming contest. They present the task preparation process used in the Polish Olympiad in Informatics (POI) and say that a rigorous implementation of the process is the key to assure good quality of tasks and to ensure that mistakes in the tasks are discovered before the contest, when it would be too late to correct them. Also Verhoeff (1990) gives detailed guidelines for the process of constructing a task set for a competition. He focuses on "ACM-style" competitions and suggests that each task should have one person who is responsible for the task.

Diks *et al.* (2008) suggest a list of aspects that should be taken into account when judging the appropriateness of a task. The listing is very similar to the features of a good olympiad task by Burton and Hiron (2008). In addition, Diks *et al.* emphasize the significance of task analysis and verification in the preparation face. For instance, the task analysis report should contain a set of source codes of solutions in all competition languages with model, suboptimal and wrong algorithms. Similar process is suggested by Verhoeff (1990), who points out that in addition to the correct solution, the solutions that are considered too inefficient should be implemented during the task preparation process. This way the time limits and the input data can be fixed so that undesired solutions will fail in the tests even if they produce the correct output.

It is also important to take advantage of the previous competitions. Wang *et al.* (2010) discuss the methods used to assure the task quality in the China National Olympiad in Informatics (CNOI). In CNOI, after each competition there is a session where the tasks and their solutions are discussed. Competitors are encouraged to join in and ask questions about the tasks. Wang *et al.* say that the session helps to discover the imperfection of the tasks, which usually leads to improvements in the future. Also Verhoeff (1990) suggests having an evaluation after the competition in order to improve the arrangements for the following competitions.

The quantity of needed tasks varies among the competition organizers. Kolstad (2009) explaines how the constant need for new tasks in the USA Computing Olympiad (USACO) was addressed. USACO organizes annually six internet-based programming competitions in three divisions so they typically need 75 new tasks of different difficulty each year. They tackled the issue by creating a web-based system, *probgate*, where the task creating community can upload and edit their tasks. This has speeded up the process of developing large amounts of new tasks that meet the quality criteria set for them. However, Kolstad states that USACO's way of creating tasks emphasizes tasks throughput and not detailed analysis of tasks or extremely thorough black-box testing.

Dagienė and Futschek (2008) discuss the development of new tasks for the Bebras competition. They point out that attraction, invention, tricks and surprise should be desirable features of each problem presented to competitors. They also remind that the problems have to be selected carefully bearing in mind the different aspects of each problem, i.e., how to interpret task's attractiveness to students.

5.4. *Choosing the Task Type*

Choosing the most suitable task type is also one thing that should be considered when designing new tasks. The purpose of the competition and the content of the tasks often derives the design towards some task types. For example, programming tasks are a natural way to test programming skills. However, one should keep an open mind for different task types as well.

Opmanis (2009) describes different task types used in the *Ugale* competition. The task in the *Ugale* competitions differ from the task used in most of the informatics competitions. For example, there is no constraint that written program must run in particular time and space limits during program execution. The contestants can also use any software during the contest. Some of the tasks are supposed to be solved with a computer and some of them are pen-and-paper type of tasks. One task can also be both. Opmanis states that it is really challenging to develop a task, which could be solved either by using pure mathematic skills without a computer, as well as by writing a correct computer program that gives the answer.

6. Discussion

In many cases, informatics competitions are closely related to education. They are not necessarily part of the formal curricula, but many of the competitions are aimed for students. Therefore, the potential of exploiting competitions in education should not be underestimated. The educational point of view should be borne in mind also when designing the task set for a competition. If the goal is to encourage many students to join, it should be made easy for students to participate even if they are not the top talents. On the other hand, competitions could be integrated to the traditional education so that they are not seen just as a separate activity for a small marginal group.

6.1. *Competitions and Education*

The motivation level of students is clearly a significant factor in learning, regardless of the discipline in hand. Informatics competitions and competition-type assignments can be one way to motivate students to learn informatics. Verhoeff (1997) gives a historic overview of competitions and education. He states that as the society and cultures constantly change, the educational practices must change as well.

Verhoeff brings out that there are conflicting opinions about bringing competitions into education. One view is that competitions are a part of every culture and therefore students should be introduced to them, because they will need the competitive skills later in life. The other view is that competition can be seen as opposed to collaboration and therefore should be avoided in education. However, Verhoeff reminds that different views of competitions should be distinguished. The focus of the competition can be, for example, in defeating other contestant, or in some external entity. The latter can, in fact, encourage students to teamwork.

Also Lawrence (2010) points out that competitions may be highly beneficial for some students, while for others it can be a negative factor. He also describes the use of competitive programming in several computer science courses. He says that pedagogical results from the courses have shown benefits, but it is critical to make sure that the competition is presented in a proper way. Students must have the opportunity to excel also without the competition setting because not all students are motivated by it. However, competitive programming assignments can provide an ongoing feedback throughout the assignment and encourage friendly competition with the instructor and other students.

Dagienė and Skupienė (2010) also discuss how competitive programming can be used in informatics education. They point out that problem solving is one of the most important parts of teaching of cognitive skills. They say that programming provides a challenging environment for learning problem solving and therefore teaching programming should be brought back to the secondary education curricula. Dagienė and Skupiene state that competitive learning is an important source of motivation for students to improve their programming skills.

Programming competitions do not necessarily have to be separate from the more traditional forms of informatics education. In National University of Singapore (NUS), competitive programming have been integrated to the curricula (Halim and Halim, 2010). They offer a special module called *Competitive Programming* where students can put their theoretical knowledge in use in a competition style setting. However, the module is not available for all students since there are pre-requisites that the students must satisfy.

Vasiga *et al.* (2008) ponder what do the tasks in olympiads actually measure? They state that the principal focus of olympiad tasks should be problem solving. They suggest criteria for analyzing tasks and claim that the goal of focusing on the problem solving skills is often hard to attain due to detailed information processing, mystery and esoteric prior knowledge of algorithms. Also Kiryukhin (2010) discussed the content of the tasks in informatics olympiads and their relationship to national informatics education. He compared the informatics olympiads and the requirements of the Russian State School

Education Standard (SSES) in Informatics. He concluded that there is a large difference between SSES and the content of the informatics olympiads, which reduces the possibilities of major portion of the students to participate and succeed in the competition.

Combining informatics competitions and education is not a trivial task. Often the competitions might be seen as a separate activity for a small group of informatics enthusiasts. If the aim is to get majority of the students to participate, collaboration between competition organizers and educators is crucial. In order to get many of the students to join in, the competition tasks and the content of the school curricula should not differ too much.

6.2. *Suggestions for Task Development*

Task development can be time-consuming and the demand for new unique tasks is big. Each task type has its advantages and disadvantages that should be considered. Programming tasks are well suited for complex algorithms and large data sets. They can also be graded automatically and instantly. However, they require competitors to learn and use some of the supported programming languages. On the other hand, non-programming tasks can lower the bar for participation and encourage different people to join in. Nevertheless, with non-programming tasks the data sets have to be smaller and keeping the focus on algorithms might be challenging.

Tasks that require code reading skills could be used more widely. They offer a great potential to test different kind of topics, for example:

- Finding an error in existing code can be used to test debugging and testing skills.
- Improving a given solution can be used to test code understanding.
- Algorithm analysis can be tested by requiring an input, which results to the worst case running time of the algorithm.

It should be made sure that the tasks measure the objectives that are set for them. For example, does the task measure some preliminary knowledge the contestants should have, or is it testing their inventiveness and problem solving skills? Typically, it is not an intention to measure only the facts and methods that are memorized in advance. Often there is a classical algorithm that can be used in the solution even if the focus is on problem solving. However, if the task is designed well, it requires problem solving to figure out how to use the known algorithm.

Games can be one way to tackle the challenge of creating tasks that focus more on problem solving than memorizing classical algorithms. Each game has a unique set of rules. These rules provide a unique basis that might help designing the task in a way that finding a known algorithm that solves the problem is not obvious. Games can also help the competitors to get interested in algorithms.

7. Conclusions

Problem solving is clearly an important skill for all students. Different types of competitions can be used to offer students opportunities to practise and develop their problem

solving skills. Competitions offer a great potential to enrich informatics education. However, integrating competitions and education is not an easy task and it should be done keeping all types of learners in mind. In this survey, several different types of tasks that have been used in informatics competitions were discussed. The tasks vary from pen-and-paper types of tasks to demanding programming tasks involving large data sets and complex algorithms. This variability in task types surely offers challenges for many different types of learners to benefit from the competitions.

It is a richness that we have so many different types of competitions involving informatics. There is no need to include all the task types in one competition as the goals of the competitions differ as well. However, it should be borne in mind that the topics of the tasks would cover the goals set for the competition. For some topics it is easier to create new tasks than for others. Therefore, attention should be paid also to the rarely used topics such as testing and ethics.

Also the task type should be tailored for the participating competitors. Short non-programming tasks can be used to attract new students to the competitions. Tasks can be formed in a way that there is no need for preliminary knowledge, which lowers the bar for participating. On the other hand, complex programming tasks with sophisticated algorithms and data structures can be used to find the top talents and to push the skillful competitors even further in their competence.

Although the difficulty levels and the types of tasks can vary significantly, the basic concepts of a good task stay mostly the same. First of all, the task should be easy to understand and unambiguous. Usually the competitions strive for tasks that do not require memorizing existing algorithms or solution patterns. The difficulty level should be suitable for the target group and in a competition there should be tasks that are easy and hard enough for each contestant. An ideal task is also interesting for the students and so exiting that it inspires students to study the subject even further.

The development of tasks is an essential aspect when organizing a competition. The tasks must be different from the previous tasks and naturally it is difficult and time consuming to come up with a lot of unique tasks. Many different methods for task development were presented in this survey. Combining these methods and task types could be a useful way to tackle the challenge of creating fresh and interesting questions for future competitions.

References

ACM (2010). *Fact Sheet*, 1st edn. `http://iiu.edu.my/acmicpc/upload/ACM-ICPC-Factsheet.pdf`. Accessed 12.11.2010.

Bell, T., Witten, I., Fellows, M. (1998). *Computer Science Unplugged: Off-Line Activities and Games for All Ages*. Citeseer.

Bigmouth media (2003). Google Launches Code Jam 2003. `http://www.bigmouthmedia.com/live/articles/google-launches-code-jam-2003.asp/1305/`. Accessed 17.02.2011.

Burton, B. (2010). Encouraging algorithmic thinking without a computer. *Olympiads in Informatics*, 4, 3–14.

Burton, B., Hiron, M. (2008). Creating informatics olympiad tasks: exploring the black art. *Olympiads in Informatics*, 2, 26–33.

Dagienė, V. (2010). Sustaining Informatics Education by Contests. *Teaching Fundamentals Concepts of Informatics*, 1–12.

Dagienė, V., Futschek, G. (2008). Bebras international contest on informatics and computer literacy: criteria for good tasks. *Informatics Education-Supporting Computational Thinking*, 19–30.

Dagienė, V.. Skupienė, J. (2010). Olympiads in informatics: competitive learning of programming for secondary school students. In: Verdú, E., Lorenzo, R., Revilla, M., Requeras, L. (Eds.), *A New Learning Paradigm: Competition Supported by Technology*, 107–126. Sello Editorial.

Diks, K., Kubica, M., Radoszewski, J., Stencel, K. (2008). A proposal for a task preparation process. *Olympiads in Informatics*, 2, 64–74.

Diks, K., Kubica, M., Stencel, K. (2007). Polish olympiad in informatics – 14 years of experience. *Olympiads in Informatics*, 1, 50–56.

Futschek, G., Dagienė, V. (2009). A contest on informatics and computer fluency attracts school students to learn basic technology concepts. In: *Proceedings 9th WCCE 2009, Education and Technology for a Better World*.

Google Code Jam Website (2011). `http://code.google.com/codejam/`. Accessed 17.02.2011.

Halim, S., Halim, F. (2010). Competitive programming in National University of Singapore. In: Verdú, E., Lorenzo, R., Revilla, M., Requeras, L. (Eds.), *A New Learning Paradigm: Competition Supported by Technology*, 173–206. Sello Editorial.

ICFP Website (2011). The ICFP Programming Contest. `http://icfpcontest.org/`. Accessed 17.02.2011.

Idlbi, A. (2009). Taking kids into programming (contests) with scratch. *Olympiads in Informatics*, 3, 17–25.

Imagine Cup Website (2011). `http://www.imaginecup.com/`. Accessed 17.02.2011.

IOI Website (2010). International Olympiad in Informatics. `http://www.ioinformatics.org`. Accessed 12.11.2010.

Kiryukhin, V.M. (2010). Mutual influence of the national educational standard and olympiad in informatics contests. *Olympiads in Informatics*, 4, 15–29.

Kolstad, R. (2009). Infrastucture for contest task development. *Olympiads in Informatics*, 3, 38–59.

Kubica, M., Radoszewsk, J. (2010). Algorithms without programming. *Olympiads in Informatics*, 4, 52–66.

Lawrence, R. (2010). Motivating students using competitive programming. In: Verdú, E., Lorenzo, R., Revilla, M., Requeras, L. (Eds.), *A New Learning Paradigm: Competition Supported by Technology*, 157–172. Sello Editorial.

Manev, K. (2008). Tasks on graphs. *Olympiads in Informatics*, 2, 90–104.

Ninka, I. (2009). The role of reactive and game tasks in competitions. *Olympiads in Informatics*, 3, 74–79.

Opmanis, M. (2009). Team competition in mathematics and informatics "Ugale" – finding new task types. *Olympiads in Informatics*, 3, 80–100.

Pankov, P. (2008). Naturalness in tasks for olympiads in informatics. *Olympiads in Informatics*, 2, 115–121.

Pankov, P. (2010). Real processes as sources for tasks in informatics. *Olympiads in Informatics*, 4, 95–103.

TopCoder Website (2011). `http://www.topcoder.com`. Accessed 16.02.2011.

Truu, A., Ivanov, H. (2008). On using testing-related tasks in the IOI. *Olympiads in Informatics*, 2, 171–180.

Vasiga, T., Cormack, G., Kemkes, G. (2008). What do Olympiad Tasks Measure? *Olympiads in Informatics*, 2, 181–191.

Vegt, W. (2006). The CodeCup, an annual game programming competition. In: *Perspectives on Computer Science Competitions for (High School) Students*. `http://www.bwinf.de/competition-workshop/papers.html`.

Vegt, W. (2009). Using subtasks. *Olympiads in Informatics*, 3, 144–148.

Verhoeff, T. (1990). *Guidelines for Producing a Programming Contest Problem Set*.

Verhoeff, T. (1997). The role of competitions in education. In: *Future World: Educating for the 21st Century, a Conference and Exhibition at IOI*.

Verhoeff, T. (2009). 20 years of ioi competition tasks. *Olympiads in Informatics*, 3, 149–166.

Verhoeff, T., Horváth, G., Diks, K., Cormack, G. (2006). A proposal for an IOI syllabus. *Teaching Mathematics and Computer Science*, 4(1), 193–216.

Voigt, J., Bell, T., Aspvall, B. (2010). Competition-style programming problems for computer science unplugged activities. In: Verdú, E., Lorenzo, R., Revilla, M., Requeras, L. (Eds.), *A New Learning Paradigm: Competition Supported by Technology*, 207–234. Sello Editorial.

Wang, H., Yin, B., Liu, R., Tang, W., Hu, W. (2010). Selection mechanism and task creation of Chinese national olympiad in informatics. *Olympiads in Informatics*, 4, 142–150.

L. Hakulinen is a doctoral student at the Department of Computer Science and Engineering, Aalto University School of Science. He received his master's degree in computer science in 2010. Currently he works at the Learning + Technology Group (LeTech) and his research focuses on using serious games in computer science education.