

# Computer Science Competitions for High School Students – Approaches to Classification and New Task Types

Wolfgang Pohl  
Bundeswettbewerb Informatik, Germany  
pohl@bwinf.de

## 1 Introduction

International computer science (or informatics) competitions for high school students traditionally concentrate on “algorithmic programming”. I.e., students are required to solve algorithmic problems and implement the solutions as computer programs. For grading the students’ work, programs are checked for correctness by automatically testing their input/output behavior.

This approach has its benefits. In particular, the automatic grading process - in contrast to a manual review process - is less prone to subjectivity; mathematically taken, an evaluation function is implemented. However, there are also shortcomings. An automatic process may fail to be fair, too: Small mistakes may lead to complete failure, while other mistakes may not be discovered, and systematically so. Moreover, automatic systems may not be able to evaluate all possible aspects of a solution; e.g., quality of thought and clarity of expression are such problematic dimensions.

In the following, we will first sketch the landscape of computer science competitions for (high school) students. On the national level in particular, there is an interesting variety of contest approaches. Then, some suggestions for augmenting international contests are made.

## 2 The Contest Space

For Germany, an overview of competitions related to computer science [9] organizes the mentioned competitions into the following groups: project contests, task contests, robot contests, others. This grouping is based on the national reality of existing contests, but does not provide a sound classification. We will now describe some dimensions that might be more appropriate for classifying computer science competitions.

### 2.1 Projects vs Tasks

This is a very basic dimension: On the one hand, the organizers of a competition may invite contestants to demonstrate their own projects. In Germany, “Jugend forscht” (similar to the European Union “Young Scientists Contest” [5]) is the classical example of a project contest. In this competition, contestants are free to choose the topic they would like to work on or the goals they want to achieve. Typically, they work at home on their projects, and at a “contest round” they are required to present their work at a presentation booth, which is visited by a jury and sometimes by the general public, too. Hence, besides the scientific quality of the work, presentation abilities may become a factor of success.

On the other hands, the majority of contests in general and computer science contests in particular, have the contestants work on given tasks. For instance, all the computer science contests mentioned in [8] are task-based competitions. Typically, tasks are developed by task committees.

Sometimes, also intermediates between project and task competitions can be found. For instance, the “Javastars” competition [3] calls for projects within a given framework: submissions have to be Java applications for learning, edutainment or infotainment, which can be utilized for school lessons. Such a competition model can be called “conditional project contest”.

### 2.2 Long Time (Homework) vs Short Time (Exam) Rounds

In project competitions, contestants typically have ample time to work on their project and prepare the presentation for a contest round. In many task-based competitions, in contrast, contestants work on the given problems within a typically fairly short period of time. For instance, the programming exams of the International Olympiad in Informatics [2] usually last 5 hours each. Within these five hours, three problems need to be solved.

But there are also task-based contests with homework rounds, like the German “Bundeswettbewerb Informatik” (short: BWINF; Engl.: German Federal Competition in Informatics). BWINF contestants have about ten weeks time to deal with the first round tasks, and about 18 weeks time to get done with the tasks of the second round. Correspondingly, contestants are expected to deliver detailed written material, in most cases accompanied by a substantial piece of software.

### **2.3 Grading: Automatic vs Manual**

Any competition needs to involve a grading procedure. Whereas contributions to project-based competitions need to be judged individually by a jury, task-based contests can incorporate grading schemes that prescribe how solutions to tasks should or even must be graded. In the BWINF (s.a.) , for each task a set of critical issues is defined. For each such issue, its influence on the grading is prescribed. A contestant’s work on a task is analyzed according to such a grading scheme.

In the case of “Bundeswettbewerb Informatik”, grading is done manually by a number of university students of Computer Science. Compared to national contests in other countries and to international competitions, this is an exception. As described in [8], most computer science contests now involve automatic grading, performed by a computer system. Such a computer-based “grading system” nowadays typically receives the source code of the solution to a problem (which required the contestant to develop a short program). Then, it compiles this source code and executes it on a pre-defined set of test data.

### **2.4 Delivery: Hardware vs Software – vs Explicit Answer**

Traditional computer science competitions for high school students involve the creation of some piece of software. In recent years, however, many competitions were founded that require the contestants to work with or even design and construct hardware. The most prominent example perhaps is the Robocup competition with its Robocup Junior offspring. Robocup Junior [4] has three divisions: RoboSoccer – “teams” of one or two robots play football against each other, RoboRescue – robots perform a specific task within an emergency scenario, and RoboDance – groups of robots are involved in an “artistic performance”. Several robot competitions have been founded in recent years; [9] mentions several robot competitions that are held in Germany, Europe, and the world. Another competition in Germany is “Invent A Chip”, where students are required to answer

questions about microelectronics, develop ideas for new chips and (the best are given that opportunity) do a software-aided chip design.

On that same dimension, i.e. the dimension of what students are asked to deliver, a third item is needed: explicit answer. There are several competitions that mainly require the students to answer questions explicitly. One example is the Lithuanian "Beaver" contest, which is currently being transferred to other countries as well [7]. To answer Beaver questions correctly, contestants sometimes need specific knowledge, but more often have to prove their understanding of informatics concepts as well as their ability of algorithmic and logical thinking.

The contests of the American Computer Science League (ACSL, [1]) mostly consist of a "short answer test" and a programming problem, so that they do not take a clear position within this dimension. A short answer test contains five questions from categories like Number Systems, Boolean (Logic), LISP, Data Structures, Graph Theory, Digital Electronics, and WDTPD (What Does This Program Do). Answers typically are really short and may often consist of a single number only. The programming problem is solved by submitting a program source code. Overall, an ACSL contest does not take a clear position according to this dimension. A programming problem may be solved within 72 hours, so that an ACSL contest as a whole cannot be classified clearly along the homework vs exam category either.

## **2.5 Answer Format**

This dimension applies to task competitions only, and it may be correlated to the delivery dimension mentioned above. In typical programming competitions like IOI, other regional olympiads, and many national competitions as well, contestants answer the given tasks by submitting a program. In former years, executables were required, but recently, source code submission has taken over. In other competitions (typically those where students have more time to work on the tasks), students also are required to submit a description and explanation of their work in writing. Furthermore, there are quiz competitions, the questions of which can be answered in a multiple choice fashion or in other ways that allow for automatic grading.

## **2.6 Age Divisions**

Some competitions are divided according to the age of their contestants (e.g., Robocup Junior, the ACSL contests, and the "Jugend forscht" project contest of

Germany), while others – like IOI – do not. This is perhaps not a dimension for contest classification, since you can view the introduction of age divisions as (virtual) introduction of a set of different contests (of the same kind) with different age limits. However, dividing a contest based on age is a feature that I consider worth mentioning. Actually, the age limit of a competition is perhaps one of its most crucial features. In task contests, problem setters will have to consider the age and assumed previous knowledge of contestants very seriously. Typically, for contests with only an upper age limit, it can be observed that tasks are designed to discover the most excellent students, whose age mostly is close to the limit; examples are IOI and BWINF. In a contest with upper and lower limit to the age of participants (which may be an age division of a larger contest), tasks are much more tailored to fit the expected contestant population; ACSL contests are an example for that.

Note that the USA Computing Olympiad (USACO) implements the concept of “difficulty divisions”. Contestants are not separated by age, but by their ability to solve the given problems. Based on their success, contestants may be promoted to the next more difficult division, and the main achievement in one year’s USACO is to be among the best contestants of the most difficult division. Hence, the lower divisions mainly serve to lower the threshold for entering into the whole competition and to practise for the most difficult division. BWINF has recently pursued a similar idea by giving a “junior problem” in the first contest round, which is more easy than the other problems (but, however, may be solved by younger contestants only; i.e., age and difficulty get mixed here).

### **3 Suggestions for international task competitions with exam setting and automatic grading of software submissions**

International Computer Science competitions like IOI and its regional siblings aim to satisfy several conditions:

**short duration** It is expensive to host the delegations, so the contest must be over within a short period of time (typically a week, but there are also more compact “versions” like the Baltic Olympiad in Informatics, which typically lasts about five days).

**control** Organizers want to make sure that contest performance was achieved by

the contestants only. Together with the duration condition mentioned above, this prohibits both projects and homework solutions. Not surprisingly, IOI and the other OIs are task contests with exams.

**fairness** The conditions that the organizers impose on contestants should be identical to all contestants. Therefore, contestants face a standardized competition environment (hardware and software). Moreover, tasks are translated into the contestants' languages, so that performance does not depend on the knowledge of a selected contest language. Finally, submissions are graded automatically, to avoid subjectivity in manual grading and to save time.

That is, in the terms that were set before, the OIs are task competitions with exam setting and automatic grading of software submissions.

Within this framework, different types of tasks have been tried. The main task types are:

**standard** The task requires the contestant to submit a program that reads input and produces output, according to given formats for both input and output. Grading is done by executing the program on input data that is unknown and equal to all contestants.

**library** The task requires the contestant to submit a program that interacts with an unknown program using a given interface. Input may be read and output may be written in addition. The unknown program may be given as a library that is linked to the contestant's program. Grading is done by executing the contestant's program and the unknown program, the latter behaving in a number of pre-defined ways (possibly reacting to the behavior of the contestant's program), equally for all contestants.

**output** The task requires the contestant to submit output data for given input data. Grading is done by comparing the contestant's output to pre-computed output data.

The realization of these task types are based on the availability of a grading system that is (basically; a grading system is a complex and complicated piece of software that needs to fulfill many and often difficult and/or subtle requirements) able to compile and execute programs in a controlled way, and compare files. In this section, I suggest some new task types that could possibly be realized with such a grading system, too. Note that these are ideas only that have not been validated yet.

**test data** The task requires the contestant to submit test data (both input and output) for a given problem. Grading is done by executing the organizers' solution to the problem for correctness and/or efficiency on that data. A test run is successful (for the contestant) when the output of the "master program" on the contestant's input corresponds to the contestant's output. It is more difficult to have the contestants considering efficiency, too. One possible way is to have several "master programs" with different efficiency levels and see which (if any) of these programs is made exceed limits by the contestant's test data. Note that a similar suggestion is made in [6].

**test data debug** For a given problem, a program is presented to the contestant that is faulty and/or sub-optimal. The task requires the contestant to submit test data which will make the given program fail or exceed limits.

**library debug** Similarly, the contestant may be required to submit a library or a program that the given (faulty/sub-optimal) program will interact with, and that will make that program fail or exceed limits.

**re-engineer debug** The contestant is given a program only. This program solves some problem correctly, but sub-optimally. The contestant is required to submit a program that solves that problem optimally. Hence, contestants must be able to re-engineer the source code, discover the problem and solve it on their own.

## 4 Conclusion and Further Work

Several dimensions for distinguishing computer science contests were presented that may lead to a classification scheme. Using these dimensions, the international olympiads in informatics were classified as task competitions with exam setting and automatic grading of software submissions. Finally, several new task types for competitions of that kind were suggested. Now, example tasks have to be developed that illustrate and validate those suggestions.

## References

- [1] ACSL home page [online, last checked January 7, 2006]. Available from: <http://www.acsl.org/>.

- [2] IOI home page [online, last checked November 7, 2005]. Available from: <http://www.ioinformatics.org/>.
- [3] Javastars home page [online, last checked November 7, 2005]. Available from: <http://www.javastars.de/>.
- [4] Robocup Junior home page [online, last checked November 7, 2005]. Available from: <http://www.robocupjunior.org/>.
- [5] Young scientists contest [online, last checked November 7, 2005]. Available from: <http://europa.eu.int/comm/research/youngscientists/>.
- [6] Gordon Cormack, Graeme Kemkes, Ian Munro, and Troy Vasiga. Structure, scoring, and purpose of computing competitions. Accepted for Workshop on Computer Science Competitions, January 2006.
- [7] Valentina Dagiene. Competition in information technology – learning in an attractive way. Accepted for Workshop on Computer Science Competitions, January 2006.
- [8] Wolfgang Pohl. National computer science contests, 2004. Available from: <http://www.bwinf.de/olympiade/national-contests.pdf>.
- [9] Wolfgang Pohl. Informatik-Wettbewerbe in Deutschland. *LOG IN*, (133):10–23, 2005.