INFODIDACT (2008), 73–99
tmcs@inf.unideb.hu
http://tmcs.math.klte.hu

Teaching Mathematics and Computer Science

Organizing programming contests

Bence Golda

Abstract. This paper aims to summarize my experience in organizing programming contests. It is an overview of those questions that should be raised and decisions that should be made by organizers, teachers and computer system administrators, who participate "on the other side" of such events.

Key words and phrases: programming contest, program evaluation.

ZDM Subject Classification: D65, U75.

1. Introduction

Following many years of competing in programming contests, since 2003, I have participated in the organization of at least one national or international competition every year. Among these some have a considerable tradition and prestige such as the *Central European Olympiad in Informatics* (2005, Sárospatak), and the *ACM Hungarian Regional Contest* (2003-2007 ACM qualifiers, 2004-2006 ACM-CERC [2]) which is the oldest and most popular contest in its category.

These contests have numerous similarities. They are organized with a similar code of rules, the contestants have to solve algorithmically hard programming tasks on the computers, without external aid, in previously given programming languages, in a limited time. As a result, the competences tested on these events are also quite similar, which is advantageous for the contestants as preparing for one of the events also prepares them for another event. Moreover, due to the fact that the similar conditions generate similar problems, the organization of the contests can be made relatively simple if the organizers share their experience with each-other.

The aim of this paper is to summarize and share in a systematic way the experience I have gained in organizing programming contests, so that the spreading of this practical knowledge finally exceeds the limits of oral tradition and finds its bonds with a scientific background.

2. Programming contests and their attributes

The well known programming contests have practically remained the same throughout the last 20 years, except that they followed the evolution of software and hardware and they offer gradually more complex tasks in several new programming languages. These contests include national and international programming contests (Logo OSzTV [3], Nemes Tihamér [4], OKTV, CEOI [5] and IOI [6]), and events that measure the algorithmic and mathematical thinking and competences with programming as a tool (ACM ICPC [7], ICFP [8]).

In addition, there are contests such as the internationally known Hoshimi-project [9] and Core Wars [10] that were originally set off as playful battles and have lately gathered a considerable number of participants. And lastly there are the contests exploiting innovative materials and new paradigms, usually sponsored by IT-giants, such as TopCoder [11] and ImagineCup [12].

Numerous other IT-related contests are organized in Hungary and abroad, for example the “<19 Freestyle IT Contest” of the C3 Foundation, nevertheless these are not programming-based events and are therefore excluded from the present discussion.

As most of my experience was gathered in the course of organizing a specific type of contest, it is needed to define what type exactly in order to make my suggestions useful for other organizers.

- *Format: paper-based or computer-based.* Programming contests are often paper based when automatic evaluation is problematic—e.g. when measuring the soundness of the program code. However, manual evaluation is a resource wasting method therefore it can not be applied on contests with a high number of participants and if the human power of the organizers is limited, which is rather typical of events organized by universities. Moreover, the objectivity of such evaluation is also questionable. On the contrary, with the help of automatic processes, computer-based contests are capable of managing hundreds of participants with only a few organizers needed for supervision.

As solving and evaluating program codes is basically a computer related task that is quite awkward and somewhat pointless to carry out manually, therefore in the followings I will only deal with issues concerning computer based contests.

- *Aid: closed, open-book, open-net, or free.* Computer based problem solving has a rather unwelcome side-effect, namely that the contestants can access all resources of the computers, including help functions and other documentations [14]. Thus, measuring lexical knowledge in such contests is futile. Nowadays, the custom is that most of the closed contests allow access to these onboard documentations.

  On the other extreme are the contests that allow free access to all aids online, offline or printed—except the help of another human being. Between the extremes of closed and free are the open-book and open-net contests that limit the amount of help allowed; nevertheless, one instructive web-site may prove to be much more useful than a dozen mediocre one.[1]
- *Communication: forbidden, limited or free.* In between forbidden and free communication is limited communication, which means that the contestants are allowed to ask questions from the jury and the organizers, but they receive short answers only. This form of communication also includes in-progress validity judgements about the programs themselves. The rules of the contests often determine the official format of the messages, questions and answers—e.g. a simple one or two word message like 'Yes', 'No', 'No comment', 'Syntax error', etc. is the most frequent answer on ACM-style programming contests. [15]
- *Cooperation: single or team work.* If individuals or small groups (2–3 persons) compete each other there is usually no need for centralized coordination. Nevertheless in case of larger groups and especially lengthy, complex tasks the need for a manager arises [16] whose performance should be indicated by a separate index among the measurement dimensions.
- *Number of contestants.* While this factor has a reduced effect on the preparation strategies of the contestants it is one of the most important factors for

[1] Of course, it is most of the time not manageable to carry more than 10 books to a contest and participants are therefore obliged to make reasoned choices among the available literature.

the organizers. In case of a few participants even 1–2 instructors are enough to successfully conduct an event, while a high number of participants require a well coordinated team of organizers.

- *Length: fixed or variable length/long term.* Variable and long term tasks are widely used for practicing, preparation and on casual, informal contests. However, all acknowledged contests use fixed length tasks, usually with short time limits. By picking tasks of the right length, number and difficulty, it is possible to elicit an ideally balanced performance from the contestants that meets the essential requirements of a good contest.
- *Difficulty.* The difficulty of the tasks is probably the most important factor for the contestants; nevertheless in the organization procedure it has only a reduced significance.[2] There is basically no difference in the complexity of organizing a contest with 1000 participants on the difficulty level of 'Hello World!' or a contest that needs well-grounded algorithmic and programming knowledge.

  Nevertheless I think it is important to note that according to my experience, participants feel satisfied with the contest if at least one of them succeeded with solving most—if not all—of the tasks and all of them could manage with at least one.
- *Algorithm or task oriented contests.* On algorithm oriented contests the assignments are built up around a certain algorithm and the main difficulty lies in figuring out how it can be constructed. On task oriented contests the main aim is usually to work out and to write complete software solutions for a life-like problem. To illustrate the difference, the former format is like a work in a laboratory, while the latter is like managing a hospital full of patients.
- *Assessment: source code or output.* IT contests have a great advantage compared to other scientific competitions because here it is possible to automate the evaluation process of the solutions. Nevertheless, it is quite important for the contestants to know in advance whether their solutions should also be appealing on the level of the source code or it is enough if their scripts are capable of solving the task.
- *Evaluation: online or delayed.* For the contestants it is extremely useful if they get feedback during the contests about the validity of their programs as it makes it possible for them to correct their mistakes. Moreover, online

[2] During the past years it happened to me that as the person in charge of the IT infrastructure I have only seen the tasks of the contest a few days after the actual event.

feedback also reduces feelings of insecurity about the quality of the solutions which greatly reduces the level of stress suffered during the contests. However, from the point of view of the organizers online assessment is a great challenge as finding, creating or adjusting an automatic evaluation system is often excessively problematic.

According to the features above, the ACM contests through which I have gathered most of my experience can be described as *computer based, online evaluated, open-book, allowing limited communication, operating with small groups of participants, time limited, validated on the basis of the source code, difficult, algorithm-oriented contests*, on which the amount of participants average around 100–200 persons in each regional round. The practices I share in the upcoming sections can most successfully be applied in similar contests.

3. Preparation

The organization process of a programming contest consists of three main phases, the *preparation*, the *contest* itself and the *post-contest* phase. The preparation phase, ending when the first contestant enters the site of the event or accesses its website, is usually the most demanding period of these three.

In the upcoming paragraphs I will collect and explain all the important actions that have to be carried out and facts that should be observed in order to ensure the conditions needed for a pragmatically (or programatically) successful contest.

3.1. Assembling the IT infrastructure

It is a basic feature of all contests that each and every participant has the same technical background to compete on. Nevertheless ensuring such circumstances is not a trivial task on IT competitions where computers form an integral part of the infrastructure.

3.1.1. Hardware

Independent of whether the evaluation process is handled by a server or by the computers of the contestants, significant differences in the hardware will certainly disadvantage some of the contestants.

(1) *Speed and memory:* The process of solving programming tasks can be divided into three main stages on the computers; the intake of the code, the compilation, and if the contestants follow the general rules of software development, testing.

Of these three the intake of the program code needs only limited resources, while the other two are rather demanding in this aspect. Therefore, during compilation and testing, computers with a restricted amount of memory may deprive the contestants of valuable minutes of thinking.

If it is not possible to use similar computers on a contest, a convenient way for solving the problem is limiting the speed of the faster computers and reducing the available memory space. Under Linux operating systems this can be done by adjusting the *acpi kernel module* and by setting proper user limits with *ulimit*.

If this method is not applicable or the differences of the computers are too significant, equal conditions can be provided for with a central computer, nevertheless it has to be ensured that the contestants will not be able to seize more resources than proportionate.

(2) *Displays:* Nowadays, TFT and LCD displays avoid this problem; however, only 2–3 years ago it was a crucial issue that the frequency settings of the CRT displays were carefully adjusted. A poorly adjusted, flickering screen could easily cause severe headache on 5 hours long contests, and could therefore be seen as a deficiency in the basic circumstances.

A similar issue is that of resolution, as screens with finer resolution enables the display of information on a much higher magnitude. This is especially important in the case of contests where teams compete each other (e.g. ACM), as high-resolution screens enable simultaneous work. For example it is possible that while one of the members is typing the code for a task, the other is checking the solution for another.

If it is not possible to provide similar displays for all contestants than the settings should be adjusted according to the minimum of the maximum possible resolutions of all the displays.

(3) *Keyboards (and mice).* Everybody prefers to work in the most comfortable conditions possible therefore it is not surprising that there was at least one participant who brought his or her own keyboard on each contest I have organized.

Nevertheless, allowing private keyboards have three important consequences. Firstly, local system administrators, quite understandably, usually refuse to break the hardware configurations. Secondly, compatibility is also a sensitive question, as it might be possible that the computers will not be able to recognize special keyboards (e.g. those having USB connectivity) while running the software of the contest.

The third consequence is much less explicit, yet it may be a source of severe problems. On closed and open-book contests where participants are only allowed to use their own knowledge or maximum the electronic resources directly available on the computer, contestants can save precious time if they have at their disposal previously prepared and optimized source codes for solving general tasks right from the beginning of the contest. Therefore, it is possible that contestants would like to get an advantage by using a specialised hardware built into the keyboard being capable of sending such program codes to the input interface of the computer. In addition this type of cheating is virtually impossible to detect, as the codes get to the computer just as if they have been typed in.

These consequences should certainly induce all organizers to seriously consider whether it is the comfort of the contestants or the avoidance of such problems that is more important for them, and they should permit or exclude private peripherals according to their decision.

(4) *Printer.* Ideally, all peripherals should be placed in the close surroundings of the computers; however, the case of the printers is exceptional, as their role during the contests is much less significant than of those mentioned beforehand.

If the computers are connected into a network, it is enough to provide only one or just a few robust printers. If possible the printers should be placed so that the participants can receive their printed documents within one minute with the help of the organizers. This way it is possible to avoid crowding around the printers.

To complete the picture, the possible application of thin client architectures should also be mentioned. In such systems the contestants use low-performance terminals, consisting of essentially no more than a display and a keyboard connected to a server which is responsible for executing the majority of the tasks mentioned before.

However, while such architectures do provide equal hardware support for every participant the terminals have to compete for the resources of the central machine. Therefore, if the organizers want to use thin client architectures, it is crucial that they adjust the maximum of resources available to each of the terminals quite cautiously.

Still, there is another major drawback with such systems which can not be avoided easily. In case the central computer breaks down, the competition is immediately ruined for all contestants. Considering this, I recommend to discard the possibility of using thin client architectures in an early phase of the organization process.

3.1.2. Software

The second step in preparing the computers for a competition is installing the appropriate software. Similar circumstances can only be ensured if besides using comparable hardware; all computers run the same software.

Due to the rapid succession of versions and the constantly changing software market there would be no point in assembling a detailed list of useful software. Nevertheless, it is possible to enumerate the general types of software that are used for programming tasks.

- *Development environment:* For recent programming languages it is customary that the developers create a whole development environment which besides code editing also has automatic compilation, deployment and testing features. Although these development environments are primarily used for industrial purposes, numerous teachers also use them for educational purposes. These teachers base their methods on their own experience and acquaint their students with programming through these frameworks, who, as a result, handle them with great confidence.
- Still, organizers have to be aware of the fact that there are (and will be) participants who do not like complex frameworks, and prefer their own, well-known applications which are also capable of providing solutions for a wide range of problems. Therefore it is a must to install a vast array of the kinds of software listed in the following on the computers of the contestants:
  (1) *Text editors or code editors*—e.g. VIM, Emacs, mcedit, etc.
  (2) *Compilers*—those needed for compiling and testing the submitted program sources, e.g. *GCC/G++*, *Free Pascal*, etc.
  (3) *Scripting facilities*—Although compilation and testing is usually done manually by the contestants, in case of complex tasks it is sometimes practical to use task and goal oriented software for testing the solutions.

For creating such programs popular script languages like *Shell-languages*, *Ruby*, *Python* and *PHP* are the most convenient choice.

(4) *Calculator*—the tests created for the tasks often contain great numbers, which can only be handled with a calculator. Nevertheless, in many contests (e.g. ACM, ICPC, CEOI, IOI, etc.) it is forbidden to use any electronic devices except the computer itself, therefore a calculator software is also among the basic needs.

(5) Tools for measurement and troubleshooting. If the contestants are familiar with the requirements of problem solving (time and memory limits for example), tools measuring the relevant indices (e.g. *ddd*, *gdb*, etc.) could also be of much use for them.

- *Time synchroniser and indicator.* One of the most important factors on a competition is the time available for solving the tasks. On most competitions the beginning and the end of this interval is indicated by, for example, a shoot, nevertheless on programming contests it is possible to use the computers themselves for these purposes which in addition, also enables special applications.

  It is advisable to keep the clocks of the computers in synch through a central computer, thus we can ensure that equal opportunities are kept even in the final, pressing period of the competition.

- *Browsers.* The most convenient way of accessing the documentation on the computers and the contest administration software on the central computer is using a graphical browser. Such software, besides being utilizable for browsing the available documentation are also handy for presenting the tasks and if combined with a suitable system, for collecting the solutions and publishing the results.

  Their greatest advantage besides their complex applicability is that they work quite similarly in all environments and operating systems, therefore most of the contestants will be able to handle them routinely. This means that they will not have to waste effort on figuring out the functioning of the software liable for performing the administration of the contest.

This latter aspect has a special importance when assembling the computers used during an event as the primary task of the contestants is to compete. If the administration system controlling the background already poses challenges then the attention is diverted from solving the programming tasks which eventually results reduced performance.

Moreover, in addition to the above mentioned, to provide equal opportunities it is also needed to arrange for the forthcoming safety precautions about the software used by the contestants.

(1) *Limiting the available resources.* Namely, offering only such an amount of resources that we know is needed for solving the tasks. The principle of 'expecting the worse' should be applied, as software or resources unknown to the organizers might easily give unforeseeable advantage to certain contestants. Therefore, if for example the internet is not needed for solving the tasks, all possible measures should be taken in order to avoid the contestants accessing it during the time of the event.

(2) *Only administrators should be able to alter* the basic hardware and software settings. If contestants get unrestricted access to the computers they will be able to cancel or avoid the limitations set beforehand by the organizers. For example, a firewall setting prohibiting network access has no effect if it can be bypassed. In such cases, if the computers are connected with poor quality equipment, this may even result in contestants communicating with each-other through their machines.

(3) *Monitoring.* Keeping the rules can most effectively be enforced by using a comprehensive monitoring application that keeps control of the whole system, including the actions taken by each contestant. Nonetheless, it must not be forgotten, that until cheating is thoroughly proven, all contestants should be handled the same way, and their achievements should not be hindered by any action including excessively slowing their computers with the monitoring processes.

3.1.3. Installation

As an organiser responsible for the technical implementation, most of my time is spent on preparing the computers of the contestants, which is one reason why I consider this phase highly important.

If we aim to carry out all the measures mentioned above we might easily find the time remaining until the contests utterly short compared to that needed for the assembly and installation of the computers. Therefore it is crucial that we carefully observe the time limitations in all cases and start working on time.

Through the years, due to limited resources and/or manpower I had to discover and develop numerous methods and tricks that reduce the time needed for the installation phase.

(1) *Know the hardware!* Although being familiar with the details of the computers sounds trivial, the hardware on which a contest is organized is often not known well enough to meet all the needs.

This is because in most cases the organizers do not have enough computers at their disposal, and therefore they rent them from universities, schools, or other institutions. This, in turn, also means that the local system administrator has to be also involved, as they are the only persons who are familiar with the fine details of the system.[3] My experience tells that the administrator-organizer dialogue should begin months before the actual event.

In the course of the negotiations it is advisable to clarify the characteristics of the computers especially those that were mentioned beforehand. Moreover, the planned course of installation before the event, the possible repairing actions and the restoration process after the contest should also be negotiated.

These steps are usually essential as the computers are generally not used for competing and are therefore optimized for rather different needs. Thus, the changes that have to be carried out are often radical and such operations need thorough preparation.

(2) *Know the system!* Besides knowing the computers, it is also needed to have an extensive knowledge about the system as a whole, including the network, its inner and outer connection points, the speed and capacity of the interconnecting lines and subnets, the different defensive measures (firewalls, etc.), and the server or servers responsible for the network logic.

Moreover it is also advisable to get acquainted with the equipment responsible for the electric supply as it might be possible that there are periods—for example through the weekends—when the supply is normally off. This could easily interfere with contests that are often held on weekends and in other unusual terms.

(3) *Install in advance, and in one turn!* Equipped with the essential knowledge about the computers and the network, it is *theoretically* possible to prepare a software configuration for the contest at home, without going near any of the actual contest computers. This technique enables to save time for the local

[3] Nowadays it is more and more common that schools apply external system administrator instead of the informatics teacher. As organizers also need to rely on the knowledge of the administrator they hire them in order to ease the organizational needs of the contests or exams.

administrator and to save our own time also, as we will have to work much less on the spot.

In *practice* this means the application of up-to-date virtualization techniques (e.g. QEMU, Virtual PC, VMware Workstation, etc.) [17] or 'system within system' installation, which means that a second operating system capable of working on its own is installed into another one. Nevertheless, this requires great skills from the technical organizers.[4]

(4) *Install in one turn!* This might first seem to contradict the previous point in certain ways, yet while that was about creating a functionally whole configuration, this proposition means the actual practice of copying ready-made bundles of software (so called images, that is a packet containing the operating system and other software also) onto the computers themselves. Nevertheless, even this simple-looking task may turn out to be a source of numerous problems in the case of too many computers and overtly big *image* files.

- In case the programs and the operating system need more capacity than a CD or DVD, even if compressed.
- If there is no suitable input device in the computers.
- If the number of computers is so high that direct installation is impossible, network installation remains the only possible alternative.
- In case of much data, the capacity of the network may also delimit performance. No matter whether we do the copying successively or simultaneously, the time required for completing the task increases proportionally.[5]

Lucky for us, we are not the first to face similar problems therefore there is a ready-made technology at our disposal, the multicast-based software distribution method. The main feature of the technology is that if we want to install the same image to all computers we can do it within a single network load, setting up all contest machines simultaneously.

In practice, due to error handling and other additional network traffic the interval of installation will be around the twice of a single installation, yet it will still be far from the original length that would be proportional to

[4] In case of Linux (the choice of mine) systems, this meant the application of the method called *chrooted* or *jailed* installation.

[5] According to my experience installing 4 GB image files onto 50 computers through a guaranteed 100 Mbit full-duplex network takes approximately 5 hours. Taking into consideration the technical parameters, this is more of an optimistic estimate than an average.

the number of computers. This can be achieved with software such as *udpcast* for Linux or the well-known *Norton Ghost*.

(5) *Modify in one turn!* Just like installing, possible modifications are also rather demanding, as they have to be carried out similarly on all computers (successively or simultaneously).

One effective way to remedy the situation in Linux environment is to install a remote access service (e.g. *OpenSSH*) together with the software enabling synchronised copying (*rsync*) to the computers of the contestants. With these, installing files and software can be handled easily, and with the help of simple scripts it is also possible to carry out tasks like automatic saving on all computers after the contest is over.

The only inconvenience with remote access is that passwords are required for each and every login, or for each computer on which we would like to execute a given command. It is nevertheless quite essential that we keep safety more important than convenience, therefore instead of using empty passwords it is advisable to review the login methods of OpenSSH, which is capable of applying private and public keys (e.g. *RSA keys*) for authentication purposes. With these keys it can be avoided that the contestants attack and hack each-other's computers through the channels used for administration.[6]

(6) *Always have spare parts!* There should always be spare hardware and software ready for use. Unfortunately this is often forgotten by the organizers.

As a matter of fact the techniques mentioned above can also be successfully applied in other areas of the IT sector. For example in case of a homogeneous machine pool the time needed for software maintenance or reinstallation can be reduced considerably.

For example, in schools the maturity exam is often rather problematic, especially if it is taken by numerous students. In most cases normal teaching is continued on the computers until the day before the examinations take place and the teachers and administrators have to tackle with installing and setting up the system needed for the exams during the night, which could be avoided by using the methods mentioned above.

[6] It is recommended to set up a firewall protection that inhibits contestants to administer each other's computer.

3.1.4. Server(s) and evaluation machine(s)

If the organizers are the backbone of the contest, and the contestants are the muscles, then the heart and the lungs are the central server and the evaluation computers (computer judges).

Besides the already mentioned suggestions there are numerous other rules that are known the best by system administrators, therefore if there are no practicing administrators among the organizers it is recommended to ask the help of a professional.

During the years I used the following computer services on the servers:

- *Serving and translating network addresses.* On web-based contests each computer needs an individual identifier for network communication. The service handling out the IP addresses within the network is called the DHCP-service. It is closely connected to the DNS-service that is responsible for translation between the logical designations (e.g. computer_001.acm) and the IP-addresses (consisting of numbers, e.g. 192.168.1.10).
- *Firewall.* The need for prohibiting any connection between contestants' computers has already been discussed, nevertheless if more than one contests take place simultaneously at the same location, it might be needed to create groups of computers with different permissions. For example, if there is a closed and a free, internet-based competition organised to the same location at the same time, it is needed that the computers involved in the closed competition are prohibited from accessing the Internet while they should still be able to reach the server computer.

  In addition, besides these, a firewall computer may also serve as a further weapon in the struggle against malicious contestants.
- *Central synchronization of time.* The time of the beginning and the end is an important factor of all contests. It is the task of the time-service to keep the clocks of the computers in sync.
- *Web service.* The easiest way of publishing the data on contests is using a web-based service. The biggest advantages of web data are that it can be displayed in the same way on all kinds of hardware and software, and it can provide safe, bidirectional communication. Thus, collecting the solutions given to the tasks of contests can be made as simple as publishing task descriptions.
- *Central authentication service.* This makes it possible to administer and verify all users of a system, including their permissions and authority. As it

will be detailed in section 4.3, this service has a special importance during the few minutes before the beginning of the contest.

As an addition to the authentication service the *single sign on* technology makes it possible that the contestants have to type in their username and password only once, during the first login. On all other occasions when identification would be needed—for example printing, or uploading the solutions for the tasks—this professional method will solve the task without assistance from the contestants.

- *Providing backup data space.* It is possible that due to certain problems the computer of a contestant has to be replaced. Besides lost time, the amount of lost work is also a crucial problem, which can be prevented by providing redundant and 'remote' data space. However, in the case of remotely stored data it is even more critical to pay attention to safety and permissions.
- *Logging.* Supervising the contestants through monitoring the computers and collecting log files is an essential ingredients to forestall cheating and can only be done centrally, which means that it requires a service on the server.
- *Printing.* As usually there are much less printers available than any other peripherals it can not be avoided that a number of contestants share a printer. In order to avoid confusion and unwanted information flow, also including program codes getting into undue hands, the organizers should take care of delivering the printed material to the right person and place. In order to ensure flawless functioning, the documents should be clearly identified.

  It can be easily tackled with for the benefit of everybody that the printing software supports other fancy features like source code syntax highlighting. In Linux systems the *a2ps* software package can bring excellent results.

These system and network services are not compulsory; nevertheless they are rather helpful tools for easing the administration processes, ensuring flawless operation during the contests and for detecting the incidental problems.

However, the following components should be considered required on an up-to-date programming contest:

- *A service for publishing the tasks and collecting the solutions*—a web-based framework can successfully fulfil this task.
- *Evaluation service*—possibly ran simultaneously on several computers that are used exclusively for this errand. Objectivity and fairness can only be ensured if the evaluation software work in identical conditions.

For these two interwoven tasks there has been numerous software solutions developed at different locations all around the world (*SIO* [18], *OnlineJudge* [19], etc.), therefore it is possible to find a suitable system for covering all needs that will be introduced in the upcoming sections.

3.1.5. The tasks

The measures mentioned so far are of no use if the tasks given to the contestants are not compiled with great care. The type of tasks and the actual tasks given on a contest are largely determined by the previously set goals of the event, therefore pedagogical decisions that have to be made in connection to these will not be discussed here.

Nevertheless, when creating the tasks it is advisable to take into consideration the following points.

(1) *Difficulty.* It has already been touched upon during the discussion of the types of contests that a competition can only be considered successful if the contestants enjoy taking part on the event. A basic requirement for this is that the difficulty of the tasks should be adequate both for the organizers and the participants, namely,

a) none of the participants should leave without solving at least one of the tasks,

b) there should be no tasks that are not solved by at least one of the participants,

c) there should be no, or only a few participants who manages to solve all tasks and spend most of their time doing nothing,

d) the tasks should be able to discriminate between the participants, or in other words they should elicit results that discriminate participants according to the aims of the contest—for example a qualifier should be capable of creating two or more groups of participants while a final should select the best,

e) the number of simple, drudging tasks should not be overtly high.

It should also be considered, that we would primarily like to measure the swiftness of thoughts and therefore the speed of typing should not be a significant measure. According to Khera, Astrachan, Kotz and others who have been involved in creating the tasks of the ACM contests during the eighties, adequate tasks should be solvable within 200 lines or 150 function calls. [20]

(2) *Assessment—the quality of evaluation determines the quality of the whole contest.* It can be taken for granted that if the process of evaluation or assessment fails, the contest will end in a failure.

As numerous tasks and types of tasks rely on automatic evaluation during assessment, the framework that evaluates and the program that verifies the solutions should also be subjected to thorough quality control.

During the past years I have met the following problems related to evaluation:

- Hardware failure—besides obvious hardware problems such as a defective hard drive there are certain deficiencies that do not surface during normal use because the operating system automatically remedies them or because the users simply do not recognise them. Such deficiencies can for example occur in memory modules; they do not interfere with everyday applications and are usually caused by other components. Nevertheless, on a programming contest, all bits are of importance therefore it is a basic requirement that all machines used for evaluation should be in perfect condition.
  This can be ensured by testing the hardware with memory tests (such as *memtest86*), hard drive tests (such as *smartmontools* for Linux), and other applications.
- *Differing architecture and standard libraries*—in case of interactive tasks or tasks that require the contestants to integrate an external module prepared by the organizers into their programs it is important to ensure that the modules will function just as well on the computers of the contestants and the machines used for evaluation.
  It might easily result a fatal error if the modules are compiled onto an inadequate processor type, or if we forget to attach the library containing the external function calls. However, determining a single processor family in advance, for example i386 which is compatible with most computer types, and the static compilation of the modules, which renders external libraries needless, can provide a relatively easy way to prevent these errors.
- *Faults in the test data*—For different operating systems, due to the differences in the system libraries, the encoding of the text data and the character or string representing line feed might also differ. As during the evaluation of the solutions the programs use textual data, it is worthwhile to settle the expected format in advance—similarly to how the

UTF-8 encoding has practically became a standard. An effective solution to the problem of differing interpretation by different software might be the application of return and line feed characters together.

Besides fixed format, it is also important to keep in mind the basic aims of programming contests and create tasks that can be solved with basic input and output routines in any kind of programming languages, except if the task itself is deciphering the input data. In case of simple test data it is easier to avoid errors on the organizers' side and, in addition, the contest will not primarily be about loading, recognizing and interpreting characters.

In general terms there is no universal solution against faulty test data generated by defective examples, therefore preparing the tasks and the test data requires special expertise and an exceptional level of attention.

4. The contest

"The best for us is if Bence is sound asleep during the contest"
—Balázs Benedek, 2006, managing organiser, ACM CERC

Ideally, a person like me, who is responsible for the IT infrastructure has nothing to do during a contest, except for feeding the printers with paper. The short epigraph above is to emphasise that until the IT administrators and technical organizers only observe the course of the contest and has nothing important to do, the managing organizers can generally be reassured that everything is all right.

In order to ensure such flawless operation we need to take several more preemptive measures besides those already mentioned.

4.1. Before the contest

Clarifying the rules is an essential element of all contests, irrelevant of whether the contest is a few hours or several days long. Although all contests are unique, there are several things that are relevant for all events and should certainly be explained to the participants.

- The schedule of the contest including the timing and place of the programmes.
- The rules concerning the type of the contest—e.g. open book, team-based, limited help allowed, etc.

- Instructions of computer usage:
  - The operating system,
  - Installed programs,
  - Information on logging in and logging out,
  - Information on user data storage and archiving—guarantees and emergency plans.
- Information on the evaluation system:
  - Login procedure and credentials,
  - Location of tasks and documents,
  - Information on uploading and sending solutions,
  - Accessibility of results,
  - Information on the use of the allowed communication channels—e.g. the electronic availability of the jury.

There are several ways to introduce these points; nevertheless it is crucial that the rules of the contest are documented in written form so that in case of complaints and misunderstandings the organizers can rely on physical documentation.

If we have the time, for example on longer events, we should also present these rules in oral form to the participants, giving them the chance to give feedback and ensuring that they have heard the important details at least once.

Giving information in spoken form is also quite useful when explaining the computer system. This way we can call the attention of the participants to the peculiarities of the evaluation system, the software installed on the machines, and the method of logging in.

It is quite advisable to arrange the seating of contestants before they actually get to any of the computers. Thus, they will have enough time to find their own computers and our administrative tasks are also somewhat simplified. If we arrange for the seating well in advance it might also be possible to divide the participants according to countries of origin and spoken languages, preventing forbidden communication and cheating.

4.2. Warm-up contests

It is useful for both organizers and contestants if everybody gets the chance of familiarizing with the location and the devices. In sports this is called the perambulation, while on programming contest this is the warm-up contest.

On these, the participants can obtain hands-on experience about what they have already heard or read about, while the organizers can locate and mend the problems that only surface during actual use. It is important that we simulate as many of the processes of the real contest as possible during the warm-up period, therefore it is useful to consider the followings.

(1) The warm-up contest *should not be longer* than the quarter of the length of the actual contest. Thus it will not become tedious but will still provide enough time for the contestants to familiarize themselves with the situation.

(2) We should pay much attention to the tasks we appoint on these phases, as too difficult tasks might induce the participants to concentrate on the task rather than on getting to know the details of the system.

(3) The *rules of the real contest* should also be consistently kept during the warm-up phase, as this helps the contestants to get used to them.

(4) *'Do not panic!'*[7]—The problems that are discovered should not be corrected immediately. They should rather be collected first, and then corrected in order of importance in the time remaining between the warm-up and the actual contest.

(5) After the warm-up contest all computers should be *restored to their original contest state*. This means that all alteration, changes in the settings and sample program codes that were made by the contestants shall be removed. The easiest way to do this is to reinstall all computers; for example with the methods detailed earlier.

    This is needed in order to provide equal opportunities for all the participants, who should compete with similar configurations and within similar environments.

4.3. The contest

After the warm-up period the next phase is the actual contest. Besides launching the restored and amended system there are two more important things to do before the start.

(1) *Printing the task sheets.* Though it is computer-based contests that are discussed here, printed task sheets that can be touched, crumpled or scribbled over are also necessities on these events.

[7] '. . . and always bring a towel!'

According to Salenieks et al. [16] in case of an average contest where 5 contestants work in a team it is enough if we provide 2 sheets per team, because limited resources support teamwork and reduce individuality.

However, my own experience shows that on the ACM contests of the last few years several teams have asked for at least 3 sheets (1 sheet for each contestant) on both the warm-up and the real contest. The reason for this might lie in the preparation strategies of the contestants. That is, most of the successful participants of the ACM contests have already competed on the IT Olympiad, and although they observe and keep all the rules, they still prefer to solve the tasks individually. If economizing with paper is important, we should print the different tasks onto individual sheets, thus the contestants can divide the tasks easily between each-other.

Maximal convenience is provided if both printing methods are combined and we provide each task on separate sheets for all participants. Moreover, in case we would like to provide full service for participants speaking different languages, besides having task sheets written in a common language we may also prepare them in the mother tongues of the contestants. This is especially customary on contests where the participants are of a young age.

(2) *Forbidding login until the beginning of the contest.* This is needed so that none of the contestants can carry out preparations during the time of arrangements as that might enable them to gain advantage over the others.

Besides having codified this prohibition in the rules and administering its observance personally, it should also be supported in terms of software by a central authentication service which only enables logging in after the start of the contest.

Now, as I have already stated beforehand, with the start of the contest, the whole situation changes. Ideally, it is the organizers who can ease up a bit, while the contestants have to work with all their energies on solving the tasks.

Nevertheless, there are numerous technical details that have to be taken care of during the contests, the nature of which mainly depends on the type and the rules of the contest. Moreover, these details also provide aspects that help selecting the most appropriate evaluation system. Proper decisions can only be made through giving adequate answers to adequate questions, for which the followings may be of some aid.

- *Supported programming languages.* The more the better. In case of only one supported language, it is advisable to consider the popularity of it among the potential contestants. Supporting dead or out-of-date languages is obviously unneeded.

It is worthwhile to discard languages that significantly differ from those used on previous contests[8] and those that oversimplify the solution of the tasks; for example, it is much easier to handle a list with the newest high-level architecture than with C or Pascal.

- *Evaluation.* Both the rules of the contest and the characteristics of IT background pose strong requirements for the features of the solutions meant for submitting, therefore evaluation is one of the most complex areas on a programming contest.

  It is worthwhile to firstly determine the inward and outward route of the data used by the solutions. Earlier, reading and writing files were the most accepted method; however, newer systems support communication through the *standard input and output.* The advantage of this method is that it is not needed to explicitly codify the names of the files read and written, but the program can read directly from the input and write the solutions onto the output. Moreover, besides decreasing the possibility of mistakes this method also reduces the administrative burdens.

  Secondly, decisions have to be made about the technique of producing executable files from the source code. On large competitions with numerous participants and tasks, separating the compilation and the execution of the programs is essential. Thus, one computer should do the compilations while the other 'juries' involved in evaluation should be occupied with running the tests.

  Thirdly, based on the tests and the time available for their running we have to make a decision about whether simultaneous evaluation is needed. In case of large tests improved execution time and resource utilization can be achieved without posing any further requirements to the contestants by breaking down the processes.

  – In the case of related tests a closing character might signify the end of the tests; for example number '0' (single-round testing, can not transform to parallel execution)

  – We may also give the number of test right on the first line (single-round testing, can not transform to parallel execution), or

[8] In the case of ACM, it is rather probable that no languages suitable for logical programming (e.g. Prolog) will ever be used on the contests.

  – In case of tests broken down into different files we may simply use the end of file signal for signifying the end of the examination (multiple-round testing; can run parallel)

  Each of these has both advantages and disadvantages in terms of pedagogy and from the point of view of the contestants. For example, while with single-round testing much care has to be exercised to correctly allocate and free the memory, there are no significant differences in handling the input and output.

- The maximum number of uploads. There are two factors that might require the maximization of the number of uploads, the available resources and the rules of the contest. While the former protects the evaluation system from being overloaded with solutions uploaded by the contestants without any previous tests and checks, the latter is more important didactically, especially on contests where the repeatedly uploaded solutions do not influence the final achievements of the contestants negatively.
- Evaluating the test results of the solutions. The second step of the evaluation process (testing the compiled solution) can easily be separated physically from the first one (compiling the source code) as the jury computers usually evaluate all solutions from all aspects and leave the final judgements for other software components. This is needed because of parallel functioning and the fault tolerance of the system.

  By considering the rules of the contest, the limitations assigned to the different tasks and the answers provided by the solutions the soundness of the solutions can clearly be determined.

  When choosing the system responsible for administering the contest it is the measurable program characteristics that deserve the most attention. These are the followings:

  – *Execution time*—the time needed for the solution to process test data,

  – *Memory use*—the maximum amount of memory allocated by the solution during testing. Up-to-date evaluation software are also capable of determining the maximums of the heap and stack values.

  – *Compilation time*—the time spent by the compilation program on compiling. On programming contest, due to the restricted domain of tasks, numerous tasks can be entirely solved during compilation time, which is fulfilled by a resource wasting state-space generation while compiling and a simple search for input during running.

  - *Maximal length of source code* or the maximal length of the preprocessed file—this is needed because of the above mentioned fact.
  - *Exit code* is needed for verifying the validity of the compiled program. For example it is possible that a program with an inappropriate termination that does not free the memory properly is still capable of producing and adequate output within the time limits; nevertheless such solutions should be rejected on didactical reasons.

- *Exotic parameters* such as the number of comments, the appearance of the source code, the extent of error handling, the expressivity of the variable identifiers, the number of empty lines are all proper indices of a contest. However, unfortunately, most evaluation systems do not operate with any of these parameters.[9]
- *Manual evaluation of the solutions.* Errors may occur in any systems. These can only be detected during the contest if we continuously observe and control its functioning. The tool of controlling is often a computer similar to the machines responsible for evaluation.

  Due to the possibility of errors, faulty test data, wrong sample solutions or verification programs the system should also support reevaluation function. The condition of this is that both the test data and the solutions should be entirely and uniformly available to the organizers.
- *Messaging.* The tool of communication permitted on the contest can be a simple mailing software or a special messaging system that suits the needs of the contest, nevertheless it has to be fitful for electronic controlling. However, before starting installation or implementing it is worth to browse through and compare the functions and capacities of the chosen evaluation system with the following expectations.
  - *Bidirectional, controlled communication* is required. The realized communication system should always provide possibility for communication between the organizers and the contestants, while exchanges between the contestants should be limited to those allowed by the rules.
  - It should be possible to give *public answers to public questions* and *private answers to private questions* or to change the visibility of questions and answers according to the rules of the contest.

[9] The number of contests on which these parameters are evaluated in any form is quite low.

  – It should be capable of handling and forwarding the messages generated by the *evaluation system*.

  This last aspect stresses the need of choosing a messaging service that is integrated into our own system or into the evaluation system instead of using a mail delivery agent like *postfix* or *sendmail*.

- *Printing.* Printing support is an important aspect when choosing the evaluation system because providing clear identifications to the documents is a must. If it does not provide a printing function than we need to find a solution independent from the core system.
- *Publishing the results.* In case of immediate evaluation, publishing the current results for the contestants and the teachers accompanying them is an important responsibility of the organizers. The level of excitement may be aroused if the listing of the results is blocked a certain, previously set time (usually 1 hour) before the end of the competition. However, although this function could easily be realized, only a few evaluation systems actually have it.[10]

Organizing competitions is not only about technical challenges. Organizers also have to provide for the comfort, the catering and in case of longer contests the accommodation of the participants. Nevertheless, in this paper I will not discuss these issues except for one small thing. I believe it is important to notice that the participants of programming contests are people who have great skills with computers and are rather acquainted with this environment. Moreover, their efforts to solve the task are rather exhausting and should not be disturbed or interrupted. Therefore, although most system administrators usually forbid the consumption of food and drinks in order to avoid any accidents, this prohibition should be suspended for the sake of the comfort of the participants during the contests and they should be allowed to take certain foods and drinks, for example cakes and refreshments with them to their computers.

5. Tasks after the contest

The end of competing does not mean the end of the work for the organizers. There are numerous tasks that have to be managed on a professional level after the participants have finished their work.

[10] Most of the time only if the rules of the type of the competition prescribe this function.

(1) *Dealing with possible remonstrations and complaints that were lodged during or after the contest, mending problems*—most of the time these can not or can just partly be made automatic. This is the part of the work, following the contests, for which it is impossible to fully prepare.

(2) *Archiving the data of the contestants, the tasks and the received solutions*—these data are quite useful for future organizers and are also important for managing complaints that are lodged later.

(3) *Printing the results and the certificates.*

(4) *Publishing statistics, tasks and official solutions.* Publishing the statistics, the tasks and the official solutions is a general custom; nevertheless the test data are usually kept in secret in order to avoid further complaints.

Similarly to the test data, the solutions devised by the contestants are also excluded from publishing, mostly due to copyright restrictions.

6. Conclusion

In the last years I have participated on numerous programming contests both as an organiser and as an adviser accompanying contestants. I sincerely hope that the observations I have included in this paper will be of some help to system administrators managing the instrumental background, young teachers of informatics in their early endeavours and also experienced organizers in organising successful programming contests. I also hope that organizers will be able to spare effort and energy by observing some of my comments and that I managed to throw light on some areas that might pose unexpected difficulties during the process of organising programming contests.

References

[1] Central European Olympiad in Informatics – 2005 Sárospatak (2005), `http://ceoi.inf.elte.hu/ceoi2005/`.

[2] Association for Computing Machinery's International Collegiate Programming Contest – Central European Regional Contest (2006), `http://icpc.cs.bme.hu/`.

[3] Logo Országos Számítástechnikai Tanulmányi Verseny (2007), `http://logo.inf.elte.hu/`.

[4] Nemes Tihamér Országos Középiskolai Számítástechnikai Tanulmányi Verseny (2007), `http://nemes.inf.elte.hu/`.

[5] Central-European Olympiad in Informatics (2007), `http://ceoi.inf.elte.hu/`.

[6] International Olympiad in Informatics (2007), `http://www.ioinformatics.org/`.

[7] The ACM International Collegiate Programming Contest (2008), `http://icpc.baylor.edu/`.

[8] ICFP (International Conference on Functional Programming) Programming Contest (2008), `http://www.icfpcontest.org/`.

[9] Project Hoshimi (2008), `http://www.project-hoshimi.com/`.

[10] Core Wars (2008), `http://www.corewars.org/`.

[11] Topcoder (2008), `http://www.topcoder.com/`.

[12] ImagineCup, 2008, `http://imaginecup.com/`.

[13] <19 Szabadfogású Számítógép Verseny (2008), `http://verseny.c3.hu/2007/`.

[14] G. Struble, *SIGCSE Bull.* **23**, no. 2, Experience hosting a high-school level programming contest (May. 1991), 36–38.

[15] M. P. Conlon, *J. Comput. Small Coll.* **20**, no. 5, RockTest: a programming contest management system (May. 2005), 27–35.

[16] P. Salenieks, J. Naylor, *SIGCSE Bull.* **20**, no. 4, Professional skills assessment in programming competitions (Dec. 1988), 9–14.

[17] Virtualization (2008), `http://en.wikipedia.org/wiki/Virtualization`.

[18] System Internetowy Olimpiady (2008), `http://sio.mimuw.edu.pl/index.phtml`.

[19] Universidad de Valladolid: OnlineJudge (2008), `http://icpcres.ecs.baylor.edu/onlinejudge/`.

[20] V. Khera, O. Astrachan, D. and Kotz, *SIGCSE Bull.* **25**, no. 1, The internet programming contest: a report and philosophy (Mar. 1993), 48–52.

[21] Universal Character Set: ISO10646 (2008), `http://en.wikipedia.org/wiki/ISO/IEC_10646`.

BENCE GOLDA
EÖTVÖS LORÁND UNIVERSITY
FACULTY OF INFORMATICS
PÁZMÁNY PÉTER SÉTÁNY 1/C H–1117 BUDAPEST
HUNGARY

E-mail: `gbence@algernon.hu`

(Received June, 2008)