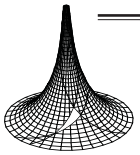


3/1 (2005), 103–119
tmcs@math.klte.hu
http://tmcs.math.klte.hu

Teaching
Mathematics and
Computer Science

An evaluating tool for programming contests

Márk Kósa, János Pánovics and Lénárd Gunda

Abstract. Students of the University of Debrecen majoring in informatics have been participating in regional ACM international collegiate programming contests since 1995. In earlier times arrangement of the local rounds was difficult because we had to check the contestants' submissions by hand. Beyond the discomfort, this hindered the efficient work of the jury and involved a number of possibilities of making mistakes.

The Programming Contest Result Manager (PCRM) program developed in the past two years provides a solution to the above problems. The program automates the evaluation of submissions and provides both the jury and the contestants with a user interface. This application can help the jury not only in ACM type but also in other kinds of practical programming contests.

Key words and phrases: ACM, automatic solution evaluation, online jury, programming contests.

ZDM Subject Classification: B60.

1. About programming contests

Students of the University of Debrecen majoring in informatics have been participating in regional ACM international collegiate programming contests since 1995. In earlier times arrangement of the local rounds was difficult because we had to check the contestants' submissions manually. Beyond the discomfort, this hindered the efficient work of the jury and involved a number of possibilities of making mistakes.

What is a programming contest anyway? During a programming contest, contestants (which can also be teams) compete against one another by solving problems. There are several problems in a competition. For each problem, the contestants create a solution which is a source file written in a programming language. This source file, when compiled, generates a program, which then can read input data from either the standard input or from a file. It writes the generated output either to the standard output or to a file.

Our aim was to develop a software which is able to evaluate large numbers of programs during contest conditions as well as to check solutions submitted by students of the *Programming* course at our university.

In this paper we will introduce the Programming Contest Result Manager program. We will discuss what the program is good for, even if the name may seem obvious.

Programming Contest Result Manager (PCRM) is a program that can be used to help judge a programming contest where the contestants must solve problems on a computer and then send the solutions to the online jury. It can also be used to judge an offline competition as well. The PCRM program can automatically check the sent solutions, i.e. compile them, run them, and check their output. It can work with programs that produce an output file and also with programs that read from the standard input and write to the standard output.

When the jury receives the source file from a contestant for a certain problem, it compiles it with the appropriate compiler and runs the program with test cases. For each problem, there is one or maybe more (but at least one) test case file, which the program must process, and generate a correct output. An output is correct, if it is the same as a pre-generated one, or if a program that verifies outputs, finds that the output is correct.

When using PCRM, there are test case files for each problem. However, each physical file can of course contain more test cases – this is the case in ACM mode, where all test cases must actually be in one file. This usually means that the input is built so that it can contain several test cases following one another.

PCRM helps you in automating the judging process by receiving the solutions (automatically), compiling them, running them, evaluating the result, and keeping a track of events and generating (or showing) the complete result.

PCRM also includes a POP3 client program that can accept incoming solutions from the contestants via email and can fully automate the entire judging process.

1.1. Evaluation modes

What kind of contests can PCRM judge? The program can work in one of three modes: *problem based mode*, *score based mode* and *ACM mode*.

In *problem based mode*, the only measure is the number of problems solved. The individual test cases do not count, i.e. a problem is only solved if all test cases are solved correctly. You can have as many problems as you wish and each problem can have as many test cases as needed.

In *score based mode*, the winner is determined by the number of test case files for which the output of the program is correct. Every test case file that is evaluated is equal to a score of 1 (or more, if a solution checking program is used. In this case, the solution checking program determines the score for a test case file). That is, if we have 5 problems and 3 test case files for each, the maximum score is 15 and the minimum score is 0. You can have as many problems as you wish and each problem can have as many test cases as needed.

PCRM was originally developed for ACM-style competitions. In *ACM mode*, the contestant with the highest number of correctly solved problems wins. If there are two or more contestants with the same number of solutions, a score is calculated, and the contestant with the least score wins. The score is calculated as follows: For each problem, the contestant gets as many points as the number of minutes passed since the beginning of the competition. For every unsuccessful submission (compile error, runtime error, wrong answer, etc.) they also get penalty points. But penalty is only given if the problem is accepted in the end. So, if I sent in a solution to problem A after 80 minutes, got a wrong answer, sent it in after 84 minutes again (I am fast at finding the error), then again at 92 minutes, which is finally accepted, I will get $92 + 2 \times$ penalty points (assuming penalty is 20, that sums up to 132). If someone else sent in a correct solution after me with no prior incorrect submissions (say at 118 minutes in the competition), that contestant would still beat me. In ACM mode, there can only be one test case file for each problem. This does not mean one test case because these problems usually have a test case format which allows several test cases to be put into a single file. Even if you specify more test cases, PCRM will only work with one.

1.2. Jury responses

During an ACM contest, when a solution is sent to the jury, a result is returned after some time to the contestants. Based on this result the contestant has to decide what to do next.

In practical terms, this usually means that the contestant, after sending in the source file, can expect a result on some webpage, which will contain the time of entry, a symbol of the program he sent in and the response he got.

Here is a list and an explanation of the responses that PCRM can return:

- *Accepted:* This is what all contestants want. It means the program passed all tests and satisfied all restrictions and was accepted.
- *Compile error:* If the compiler returns an error, this message is sent back to the contestant.
- *Runtime error:* This means that the program has not terminated successfully. This probably means a segmentation fault, access violation, exception or something like that. Please make sure the programs return 0 at the end, because otherwise it might be treated as a runtime error.
- *Time limit exceeded:* For every problem, a time limit can be set. This is the maximum amount of time the program is allowed to run. When exceeded, the program will be terminated, and this error is returned.
- *Memory limit exceeded:* For every problem, a memory limit can be set. The PCRM program checks if this has been exceeded, and if it has, then it will return this error, possibly terminating the program. Note: under Windows a check is only made at the end of a successful run (when none of the above errors occurred) because Windows provides a facility for safely determining the peak memory usage at that time. Under Linux, a more manual and error prone process is used – Windows wins out in process information requests.
- *Wrong answer:* The program finished without any errors and produced an output, but the output is wrong.

2. Programming Contest Result Manager

In this section we will discuss the functional schema of the PCRM system and give a short description of some configuration parameters.

2.1. Functional schema

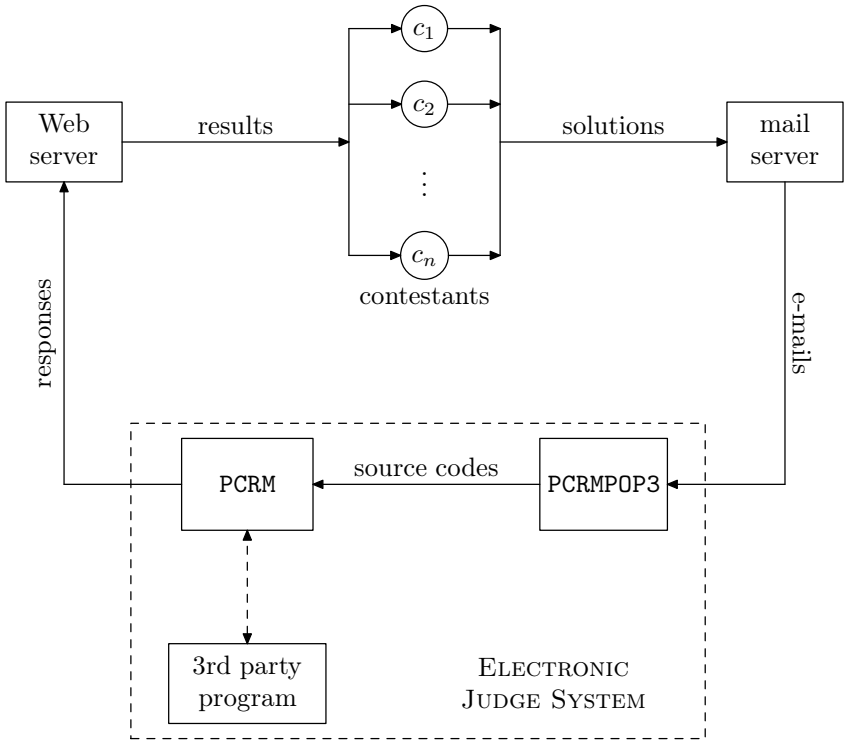


Figure 1. Functional schema of the system

The electronic judge system consists of two main parts: PCRMPOP3 and PCRM. PCRMPOP3 is a simple shell that uses the POP3 protocol to download messages from a server, store them, and then uses a MIME message parser to parse the message and extract the source code contained in it. Afterwards it can optionally invoke PCRM to do the checking.

PCRM uses the settings in a configuration file to determine its working mode. It compiles the programs and runs them with the test files. Both the compilation and running command lines can be set for each of the languages. This way you can customize the way programs are handled and it is also possible to use other programs to run the compiled program (as in the case of Java).

If the program reads input from standard input, PCRM redirects standard input for the program so that it receives the required test cases. Standard output can be redirected the same way, if output is written to it.

After the program terminates successfully within the given time limit, not exceeding the given memory limit, the output is compared to the pre-determined output, and if they match, the test case is accepted. PCRM is also able to invoke a third party program that evaluates the output and informs PCRM via its exit code whether the solution is correct or not.

2.2. Configuring PCRM

The `pcrm.ini` file is one of the most important parts of PCRM. It contains all the information needed to run a competition. This file can be divided into the following sections: global, HTML, POP3, compiler, run, problem and contestant.

2.2.1. Global section

The `[global]` section contains generic settings for the program and the environment.

- `numcont`: The number of contestants in the competition. This number must be greater than 0, and does not have currently an upper limit.
- `numprob`: The number of problems in the competition. This number must be greater than 0, and currently PCRM can handle at most 64 problems.
- `mode`: Program operating mode. This basically determines how PCRM will calculate the score after evaluating the results. Currently, there are three operating modes available. You can read more about them – especially the third one – in section 1.1.
- `compout`: When PCRM compiles the solutions, you can choose to see the output or to hide it. This setting governs what happens to the output.
- `runout`: Display runtime output for certain programs (should be 0), including the program that is run, and standard output is not redirected (because the program writes output to a file). Should not be used, only for testing and debugging.
- `result`: Result calculation method.
- `retest`: Retesting solution setting. This setting tells PCRM what to do when a second or third or further test is requested.

- `pop3head`: Check for POP3 header. PCRMPOP3 adds a small header to all files, which contain information about the file, like contestant, problem and a time stamp. When enabling this option, PCRM will use this information, which of course makes the judging more accurate. It also verifies the integrity of files this way.
- `acmstart`: Contest start time.
- `acmstop`: Contest end time.
- `acmpenal`: Penalty points for wrong solutions. Default is 20 (ACM standard).

2.2.2. HTML section

PCRM can also generate the output in HTML format. In this case, you can fine tune the settings used to generate it with the `[html]` section.

- `frame`: In case the jury needs to show additional messages, it is possible to insert an `<iframe>` between the results and the contest history. You can also set the file from which the frame content is included with other settings.
- `framesrc`: The name of the file to display. This will be written into the `<iframe>`'s `src` property.
- `framewidth`: The width of the frame in pixels.
- `frameheight`: The height of the frame in pixels.
- `showlang`: If the value of this settings is 1, then a reference for the programming language of the submitted solution will also be included in the results table.
- `destfile`: File where to generate the HTML result. Normally, the result is written to a file named `pcrm_res.html` in the same directory where PCRM runs.
- `autorefresh`: If this value is set, the written HTML file will also contain the setting, so the browser will refresh the HTML file automatically after a period of time.
- `gfxproblem`: If this value is 1, then instead of the problem name the little circle with a letter is displayed. Only used together with `resgfx` mode in the program.
- `historynum`: Adds numbers to the history list, beginning with 1. It can be used to see how many history rows there are.
- `noresultafter`: After the specified time, the global results will not be generated (this is the result file given in destfile).

- `judgeresults`: Sets mode for another type of output generation. If enabled, the program will generate more result files (see `judgefiles` option). These files contain the complete result as if it was generated with normal generation and also individual files for the contestants. This option can be used with `noresultafter`, so after a certain time only judges can view the complete result and contestants can view only their own results.
- `judgefiles`: Specify path and name of files for `judgeresults`.
- `addtime`: Adds time information to accepted programs (how long it ran).
- `addtime2`: Adds detailed time information to accepted programs (how long it ran). The `addtime` option must also be enabled. Currently, this option will not work on Linux systems.
- `addmemory`: Adds memory usage information to accepted programs (how much memory it consumed).

2.2.3. POP3 section

The `[pop3]` section contains settings that are used by the PCRMPOP3 program. These are mostly email settings and options how to launch the PCRM program itself.

- `server`: POP3 Server domain name or IP address.
- `user`: Username to send to server.
- `pass`: Password for user name.
- `checkwait`: If listen or full auto-judge mode, wait time between checks.
- `subject`: Mail subject type and verification.
- `savemail`: Save all messages to `pop3/` folder.
- `genresult`: Generate results after POP3 operation? If set to 1 or 2, then after an auto1 or autoall command is executed, a result generation will also be requested from the PCRM program.
- `genalways`: If set to true then results will be generated in autoall mode after every run, not just ones that update the source files and run PCRM. This can be used to display the time left "constantly".
- `pcrmpath`: Sets where the `PCRM.EXE` (or `PCRM` for Linux) executable can be found. This defaults to the current directory, but you can set it to something else like `../debug/windows/pcrm.exe` or something like that.
- `autoend`: Automatically end contest? This can be used with `autoall` and is used to end the checking cycle as soon as the end of the competition is

reached. This option also checks the acmstop time in the `[global]` section to see when it needs to stop. Will always generate a result according to the `genresult` parameter before exiting. The option can be used to terminate the competition automatically, when time is up, and no more results will be generated. (Note: this also exits from listen mode.)

2.2.4. Compiler sections

The `[compiler`*xxx*`]` section contains compile lines, which tell the program how to compile the solution source codes. There are two compiler sections, `[compilerwin]` and `[compilerlnx]`. Under Windows systems the former one is used and under Linux systems the latter one is used.

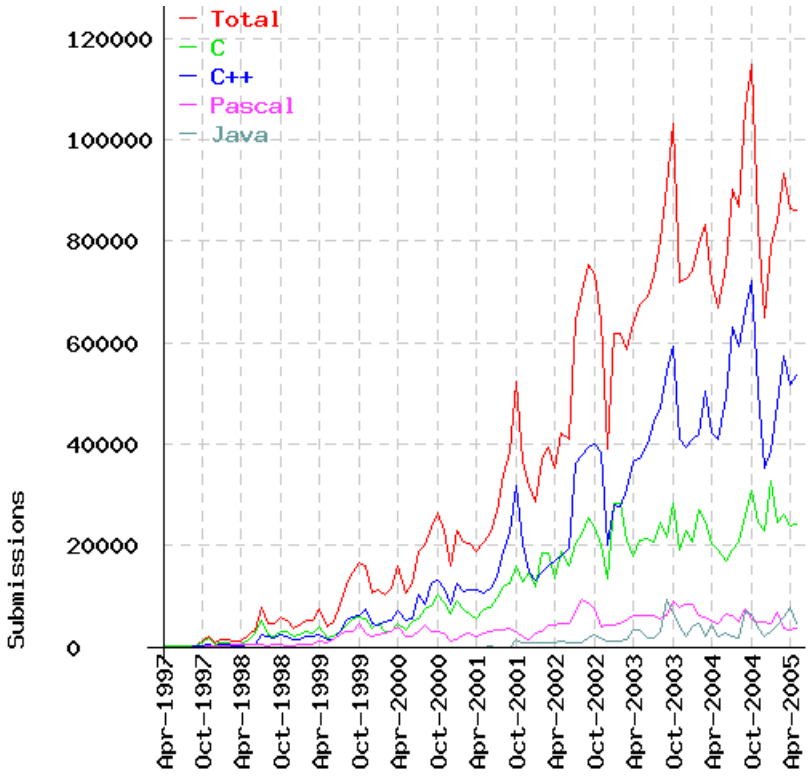


Figure 2. Submission statistics from `online-judge.uva.es`

Regarding the statistics of the international on-line judge systems [1, 2, 3, 4, 5] PCRM has been prepared to support the following programming languages: C, C++, Assembly, Java, Pascal, Basic, C#. To enable any of these languages, there should be a line in this section for each language in the form `lang=compileline`, where `lang` can be `c`, `cpp`, `asm`, `java`, `pas`, `bas`, `cs`.

2.2.5. Run sections

The `[run`*xxx*`]` section contains run lines, which tell the program how the compiled solution programs should be run. There are two run sections, `[runwin]` and `[runlnx]`. Under Windows systems the former one is used and under Linux systems the latter one is used.

They work basically the same way as the compile lines, except this time you specify the run command.

2.2.6. Problem sections

Every problem has its own `[problem#]` section, where the `#` sign is replaced with the problem number, counting from 1. Every problem can have some test cases (at least 1). Some parameters:

- `numtest`: Number of test cases.
- `maxtime`: Time limit for each test case, in milliseconds (1/1000 s).
- `maxmemory`: Memory limit for the program, in kilobytes. Default value is 0, which disables the memory size checking. Only available under Linux and Windows NT4/2000/XP.

2.2.7. Contestant section

This is where contestant information is stored. There are two types of entries. The first one lists the name of the contestant or team, and the second one gives a passphrase, which is used with the PCRMPOP3 program when handling email messages.

- `#=name`
- `P#=passphrase`

The `#` sign is replaced with the contestant number, starting at 1.

3. PCRMPOP3

PCRMPOP3 can monitor incoming email, and is able to fully automate the judging process by saving problems as they arrive and launching PCRM to check these problems at once, and also by automatically regenerating the output HTML file. This simply means, you just start it and you don't need to worry about anything else, just sit back and relax until the contest ends.

For the messages received, there is no restriction on the email address that is used. The email must always contain one and only one attachment which is a source code file. If the email does not conform to these restrictions, PCRMPOP3 simply ignores it. The source code should not be included in the message body, the file has to be attached to the email. Also, the source code file usually has to be named strictly as required by PCRM! PCRMPOP3 can also handle base64 encoded attachments.

3.1. Running modes

You can specify the following commands in the command line. If you run loop commands (`listen`, `autoall`), the normal way to exit the program is by using Ctrl-C (under Windows, you can also use Ctrl-Break).

- `check`: Run one email check.
- `listen`: Run in a loop and check incoming email periodically.
- `auto1`: Simple auto judging. Check email and then run a check on each of the solutions that were downloaded (if any). Also generate results into HTML if requested.
- `autoall`: Full auto judging. Run mail checking in a loop, and if any new solutions arrive, also start the PCRM program to check those solutions. After checking, if not set otherwise, generate an output (with the `resgfx` command of PCRM).

Here is a sample output of running PCRMPOP3 with `autoall` command which is the typical running mode during an ACM-like contest:

```
acmjury@it:~$ ./pcrmpop3 autoall
PCRM (Programming Contest Result Manager) POP3 Client, version 1.9.1
Linux build, Mar 12 2004 14:41:57
Copyright (C) 2003-2004 by Lenard Gunda

Running in a loop - waiting time is 60 seconds
```

```
Email check! (connecting to server localhost)
Connected and logged in to POP3 Server
There is 1 message in mailbox (size: 514 bytes)
Disconnecting from POP3 Server
Result updated
Wait ...
Email check! (connecting to server localhost)
Connected and logged in to POP3 Server
There are 4 messages in mailbox (size: 9474 bytes)
New message detected (index:2 / uid:aaed734d9560913c89828c13db395885)
 + Attachment: p1c1.c
 + Storing attachment for: Problem 1 - Contestant 1
 + Writing file: "contestant/1/p1c1.c.1079488974.175127"
 + Updated "problem1/p1c1.c"
New message detected (index:3 / uid:8cec411abc2df1611e7bce2a3d3e27ab)
 + Attachment: p1c2.c
 + Storing attachment for: Problem 1 - Contestant 2
 + Writing file: "contestant/2/p1c2.c.1600234907.175140"
 + Updated "problem1/p1c2.c"
New message detected (index:4 / uid:5fde5a57d456b6bcd624f2769ff8c206)
 + Attachment: p1c3.c
 + Storing attachment for: Problem 1 - Contestant 3
 + Writing file: "contestant/3/p1c3.c.372261666.175156"
 + Updated "problem1/p1c3.c"
Disconnecting from POP3 Server
Checking received results with PCRM executable
======================================================================
Checking problem 1 for contestant #1 (Gunda Lenard)
 * Executing compile: gcc -lm -w -O2 -o p1c1 p1c1.c
 + Compiled successfully!
Contestant #1 / Problem 1 / Testcase 1
 * Executing: sudo /usr/sbin/chroot . ./p1c1
 - Failed: Runtime Error (RE)!
 - Skipping further tests! (break-break)
Checking problem 1 for contestant #2 (Panovics Janos)
 * Executing compile: gcc -lm -w -O2 -o p1c2 p1c2.c
 + Compiled successfully!
Contestant #2 / Problem 1 / Testcase 1
 * Executing: sudo /usr/sbin/chroot . ./p1c2
 - Failed: Wrong Answer (WA)!
 - Skipping further tests! (break-break)
Checking problem 1 for contestant #3 (Kosa Mark)
 * Executing compile: gcc -lm -w -O2 -o p1c3 p1c3.c
 + Compiled successfully!
Contestant #3 / Problem 1 / Testcase 1
```

```
 * Executing: sudo /usr/sbin/chroot . ./p1c3
 + Accepted! +(1)+
======================================================================
PCRM checks complete
Result updated
Wait ...
```

As you can see at the end of the output, the program is not terminated at this point, it is waiting for further solutions...

4. PCRM in education

During the 2004 Spring semester the PCRM program was applied in practice at Debrecen University. This time PCRM did not work under contest conditions but only evaluated submissions and generated output off-line once the deadline for sending in the submissions was reached. Students pursuing the Informatics major had to submit solutions to 33 different programming tasks to meet the requirements of the *Programming* course.

Table 1. Programming course statistics

Occasion #	Topic	# of problems	# of evaluations (accepted / total)
1	simple arithmetic problems	7	1504 / 2937
2	string processing	6	1486 / 2138
3	basic algorithms	5	869 / 2855
4	user-defined functions	10	1807 / 2318
5	file processing	5	339 / 3621

The students had to send in solutions to 5–6 tasks on every occasion. As we had about 400 students at that time, PCRM evaluated at least 2000 solutions per occasion. These tasks covered the following topics:

- *Simple arithmetic problems:* The program should generate its output based on numbers appearing in the input. For example:

  > *Write a program that reads two positive integers separated by a comma, and outputs the word* `yes` *or* `no` *depending on whether the two numbers are relative primes.*
  > *The test input and output for this task may look like the following:*

Input	*Output*
`12,121`	`yes`
`16,72`	`no`
`26,49`	`yes`

- *String processing:* The program should process the input character stream. The output should contain exactly one line for every line of the input. For example:

  > *Write a program that reads a line which contains two words separated by a single space, and determines whether the second word is a border of the first one. A word is a border of another one if it is both a beginning part and an ending part of the other word. You may assume that each line consists of at most 64 characters.*
  > *The test input and output for this task may look like the following:*

Input	*Output*
`abababa ababa`	`yes`
`apple apple`	`yes`
`test t`	`yes`
`notation no`	`no`

- *Basic algorithms on different data structures:* For this kind of problems the program has to process a large number of data elements. The two most common tasks are searching and sorting. For example:

  > *Write a program that reads lines. Each line consists of exactly one English word. The output should contain the same words sorted lexicographically, one word per line.*
  > *The test input and output for this task may look like the following:*

Input	*Output*
`lion`	`dolphin`
`elephant`	`elephant`
`dolphin`	`gorilla`
`turkey`	`lion`
`gorilla`	`turkey`

- *Text file processing:* In this case the program should read the input and/or write the output from/to a given text file.

PCRM, of course, can evaluate solutions of any topic, not only the topics mentioned above. To evaluate a task with more than one good solution, PCRM is to use a third party program which compares the output of the submitted code with all the possible correct solutions.

5. PCRM during a contest

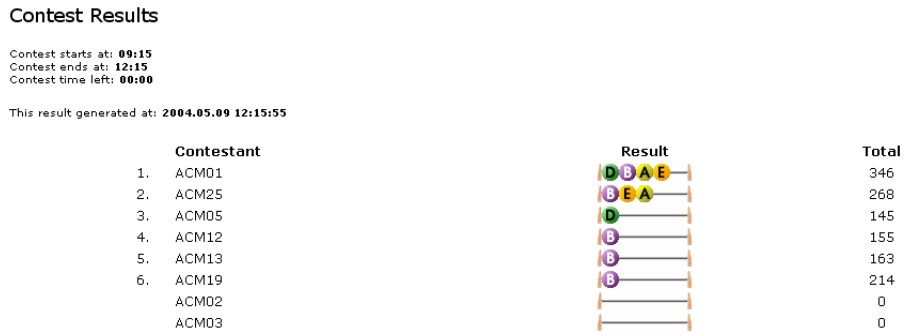
As opposed to using PCRM in education, the main goal of PCRM under contest conditions is to generate feedback about the evaluated solutions submitted by the contestants and to produce the current ranking of the contest in real time. Additionally, the topics covered during a contest are a little different from those in education:

- string processing, pattern matching
- sorting
- arithmetics and algebra
- combinatorics
- number theory
- backtracking
- graph algorithms
- dynamic programming
- geometry

A large number of problems of these topics can be found on programming contest sites [1, 2, 3, 4, 5] and in books [6, 7, 8].

In Figure 3 you can see the final result of the programming contest for our students in 2004 Spring semester at Debrecen University. The figure shows the start time and end time of the contest, the contestants' positions and names as well as the problems they solved correctly in order of submission time, and finally their total points including penalties.

To intensify the excitement of the contest, the program displays the results of the evaluations for the contestants on a public home page which allows contestants to see the other contestants' attempts (see Figure 4). The jury, however, is able to monitor information regarding only one contestant.



Contest Results

Contest starts at: **09:15**
Contest ends at: **12:15**
Contest time left: **00:00**

This result generated at: **2004.05.09 12:15:55**

	Contestant	Result	Total
1.	ACM01	D B A E	346
2.	ACM25	B E A	268
3.	ACM05	D	145
4.	ACM12	B	155
5.	ACM13	B	163
6.	ACM19	B	214
	ACM02		0
	ACM03		0

Figure 3. Contest result

66.	11:18:14	ACM11	D	wrong answer	C
67.	11:18:14	ACM11	D	wrong answer	C
68.	11:18:13	ACM13	B execution time: **0.963 sec**	**accepted**	C
69.	11:16:31	ACM25	A	wrong answer	C
70.	11:16:31	ACM25	A	wrong answer	C
71.	11:12:51	ACM09	E	wrong answer	C
72.	11:12:42	ACM11	E	wrong answer	C
73.	11:10:55	ACM12	B execution time: **0.354 sec**	**accepted**	C++
74.	11:09:58	ACM11	E	wrong answer	C
75.	11:06:14	ACM15	B	time limit exceeded	C
76.	11:06:14	ACM15	B	time limit exceeded	C

Figure 4. List of submissions with PCRM's responses

References

[1] Universidad de Valladolid Problem Set Archive, `http://online-judge.uva.es`.

[2] The 2000's ACM-ICPC Live Archive Around the World, `http://acmicpc-live-archive.uva.es`.

[3] Sphere Online Judge, `http://spoj.sphere.pl`.

[4] Zhejiang University Online Judge, `http://acm.zju.edu.cn`.

[5] Ural State University Problem Set Archive with Online Judge System, `http://acm.timus.ru`.

[6] Steven S. Skiena, *The Algorithm Design Manual*, Springer-Verlag, New York, 1998.

[7] Steven S. Skiena, Miguel A. Revilla, *The Algorithm Design Manual*, Springer-Verlag, New York, 2003.

[8] Juhász István, Kósa Márk, Pánovics János, *C példatár*, Panem, Budapest, 2005.

MÁRK KÓSA, JÁNOS PÁNOVICS AND LÉNÁRD GUNDA
FACULTY OF INFORMATICS
UNIVERSITY OF DEBRECEN
H-4010 DEBRECEN, P.O. BOX 12
HUNGARY

E-mail: mkosa@inf.unideb.hu

E-mail: panovics@inf.unideb.hu

E-mail: lenard.gunda@frenzy.hu

(Received May, 2005)