Olympiads in Informatics, 2019, Vol. 13, 209–215

DOI: 10.15388/ioi.2019.14

TPS (Task Preparation System): A Tool for Developing Tasks in Programming Contests

Kian MIRJALALI, Amir Keivan MOHTASHAMI, Mohammad ROGHANI, Hamid ZARRABI-ZADEH

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
e-mail: {mirjalali, mohtashami, roghani}@ce.sharif.edu, zarrabi@sharif.edu

Abstract. The task preparation system (TPS) is a tool developed mainly for preparing IOI tasks. It was originally developed for, and successfully used in IOI 2017, and since then, it has been used in several other nationwide and international programming contests, including IOI 2019. The tool consists of a command-line interface for local (offline) work, and a web interface which integrates with git and provides more features. This article presents the main features of the task preparation system, and briefly describes how it works.

Keywords: competitive programming, task preparation, Olympiad in Informatics, programming contest.

1. Introduction

The host technical and scientific committees of IOI 2017 had comprehensive discussions on how the development of IOI tasks could be eased. The first thing to fix was the directory structure of a task. Based on this convention, a set of scripts began to be developed to simplify the work. This finally resulted in a more mature software which is called TPS (task preparation system). Here are the feedback of two senior members of the IOI International Scientific Committee when presented with TPS, its documentation, and the tasks developed:

> *"That's a great job you made. The system looks really nice and comfortable. My congratulations."*

> *"I must admit I'm also impressed that you've been able to develop a tool like this so quickly. This will surely be very helpful in organizing IOIs."*

Usage of TPS was not limited to IOI 2017. It has been used in many programming contests in Iran, including the national IOI team selection contests. The widespread usage of this tool motivated the authors of this article to share it with other members of the IOI community who are interested in designing and developing problems for programming contests.

TPS has a command-line interface (shortly, tps-cli) that is used locally by the developers to prepare tasks on their own machines. Each task is developed in a directory maintained through a git repository. A secured GitLab server was used in IOI 2017. The secondary part of TPS is the web component (shortly, tps-web) which is deployed besides the git repository in the server and provides additional features such as solutions invocation for more exact timings.

This article is organized as follows. We first describe the directory structure of the tasks in Section 2. We then explain the structure and features of tps-cli in Section 3. Finally, we briefly cover the main features of tps-web in Section 4.

2. Task Directory Structure

The TPS command-line interface acts based on a standardized directory structure which is usually maintained in a git repository for sharing with other task developers. The following files and directories are present in a task directory:

- **`problem.json`:** This is the main json file containing the general information about the task, such as short name, title, task type, memory limit, time limit, and possibly the tps-web URL if tps-web is set up.
- **`solution/`:** This directory contains the solution source codes, whether the solutions are correct or not.
- **`solutions.json`:** A json file containing an entry for each solution in the "`solution/`" directory, specifying the expected verdict of the solution on the subtasks, or whether it is a model solution (the one used for generating test outputs).
- **`subtasks.json`:** A json file specifying all subtasks by their names, scores, and validators assigned for verifying the conformance of their corresponding test inputs with their constraints.
- **`validator/`:** A directory containing codes for validating the format of test input files and checking their conformance with the constraints of the problem.
- **`gen/`:** This directory contains everything related to generating test data. The text file "`data`" in this directory specifies which tests are going to be generated by which generator and with what parameters. Manually created tests are also placed in the "`manual`" subdirectory. An example of "`gen/data`" file is shown in Fig. 1.
- **`grader/`:** For each programming language allowed in the contest, such as C++ and Java, there is a subdirectory here containing the graders for that language. A grader is a program that links with the contestant's solution and provides it with grading interface say for reading the input and writing to the output.

```
@subtask samples
manual 01.in

@subtask n_3
general_graph     100    280    17    6
tree               98      5

@subtask n_2
@include n_3
general_graph   91555  40000   134    6
tree         10000     90

@subtask full
general_graph  100000 800000   543   44
tree         100000    800
```

Fig. 1. A sample "`gen/data`" file.

- **`checker/`:** A directory containing the checker, a program that evaluates the output of the contestant's solution per test case and specifies its score.
- **`public/`:** This directory contains the files provided to the contestants, such as sample tests, compiling scripts, and basic graders for local testing.
- **`statement/`:** This directory contains the files related to the task statement.
- **`scripts/`:** The primary implementation of tps-cli commands is placed in this directory.

The following directories are also part of the TPS directory structure, but are not stored in the git repository:

- **`tests/`:** A directory containing the generated tests.
- **`sandbox/`:** Solutions will be compiled and executed in this directory.
- **`logs/`:** This directory contains the logs of the last execution of test generation or solution invocation.

3. TPS Command-Line Interface

The TPS command-line interface provides an easy way to execute predefined scripts commonly used during the process of task development. The scripts reside in the "`scripts/`" directory, and can be customized per need of a task. The "`tps`" command itself is placed in the PATH, and can be called anywhere in the terminal. For example, when the command "`tps compile` <*arguments*>" is issued, tps-cli runs the script "`scripts/compile.sh` <*arguments*>". The following commands are currently supported in tps-cli:

- **`compile`:** Compiles a given solution with the appropriate grader, and leaves the result it in the `sandbox` directory. It wraps the complexities of compiling solutions with different languages. The public version of the grader (the one given to the contestants) is used if `-p` or `--public` is passed to the command.
- **`run`:** Runs the compiled solution in the sandbox with no restrictions (like time limits). Input/output files can be specified through redirection.

- **`gen:`** Generates the test cases based on file "`gen/data`" and runs the corresponding validators on the inputs. The generated test cases are stored in the "`tests/`" directory. Multiple options are available such as specifying a subset of tests to generate (using wildcards), stopping on the first failure, and using an alternative model solution for generating outputs. An example of executing the command "`tps gen`" is shown in Fig. 2.
- **`invoke:`** Compiles and executes a given solution on the generated test cases considering the task restrictions (e.g. time limits). The score of the solution on each test case is also determined using the checker. Multiple options are available such as specifying a subset of tests to invoke, stopping on the first failure, and setting a different time limit. A sample execution of "`tps invoke`" is shown in Fig. 3.
- **`make-public:`** Updates the contents of the public directory (especially if the public graders are going to be generated through erasing the secret parts of the judge graders) and creates the archive provided to the contestants, based on file "`public/files`".
- **`verify:`** Verifies the validity and integrity of the task directory structure including the json files.
- **`analyze:`** Opens the analysis page of the task in tps-web.

```
> tps gen
generator          compile[OK]
solution           compile[OK]
validator          compile[OK]
0-01               gen[OK]     val[OK]     sol[OK]
1-01               gen[OK]     val[FAIL]   sol[OK]
1-02               gen[OK]     val[OK]     sol[OK]
2-01               gen[OK]     val[OK]     sol[OK]
2-02               gen[OK]     val[OK]     sol[OK]
3-01               gen[OK]     val[OK]     sol[OK]
3-02               gen[OK]     val[OK]     sol[OK]

Finished.
>
```

Fig. 2. An example of executing "`tps gen`".

```
> tps invoke solution/roads-test.java
solution           compile[OK]
checker            compile[OK]
0-01               sol[OK]    0.063     check[OK]       1  [Correct]
1-01               sol[OK]    0.069     check[OK]       1  [Correct]
1-02               sol[FAIL]  0.065     check[SKIP]     0  [Runtime Error]
2-01               sol[OK]    0.172     check[OK]       1  [Correct]
2-02               sol[OK]    0.076     check[OK]       0  [Wrong Answer]
3-01               sol[OK]    0.079     check[OK]   0.666  [Partially Correct]
3-02               sol[FAIL]  3.012     check[SKIP]     0  [Time Limit Exceeded]

Finished.
>
```

Fig. 3. A sample execution of "`tps invoke`".

More information on the available options for the `tps` commands can be obtained by passing argument `-h` or `--help`. The source code and full documentation of tps-cli is available at: `https://github.com/ioi-2017/tps`.

4. TPS Web Interface

The TPS command-line interface is equipped with a web interface called tps-web. The web interface retrieves data directly from a task directory, parses the json files, and visualizes the task state to the users. As a result, users have a vision on all parts of the task and can efficiently apply different actions on it. Some of the main features of tps-web is briefly described below.

Task visualization. The web interface visualizes various components of a task. For example, one can see all the solutions for a task, and the verdict each of which should receive on the subtasks. (See Fig. 4.) As another example, there is a markdown viewer through which one can easily see the current version of the problem statement. (See Fig. 5.)

Solution invocations. The tps-web allows users to select a set of solutions, and invoke them on a subset of test cases. The invocation results are then presented in a table to-

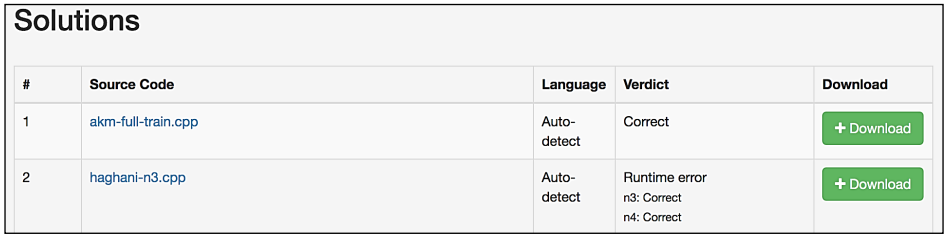


Fig. 4. A sample solutions page in tps-web.

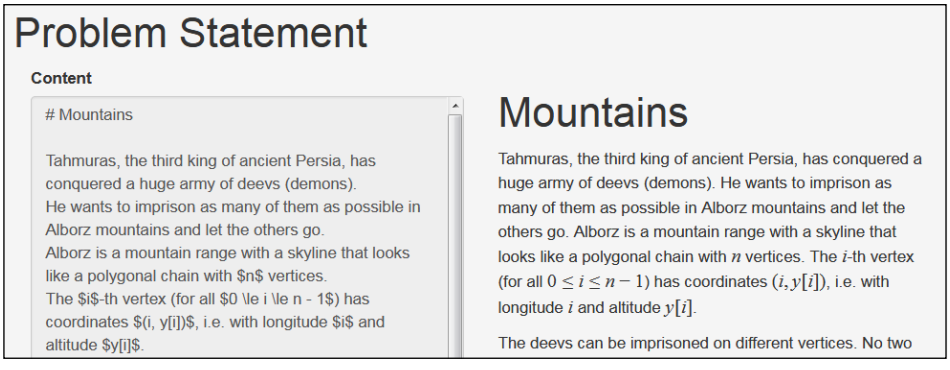
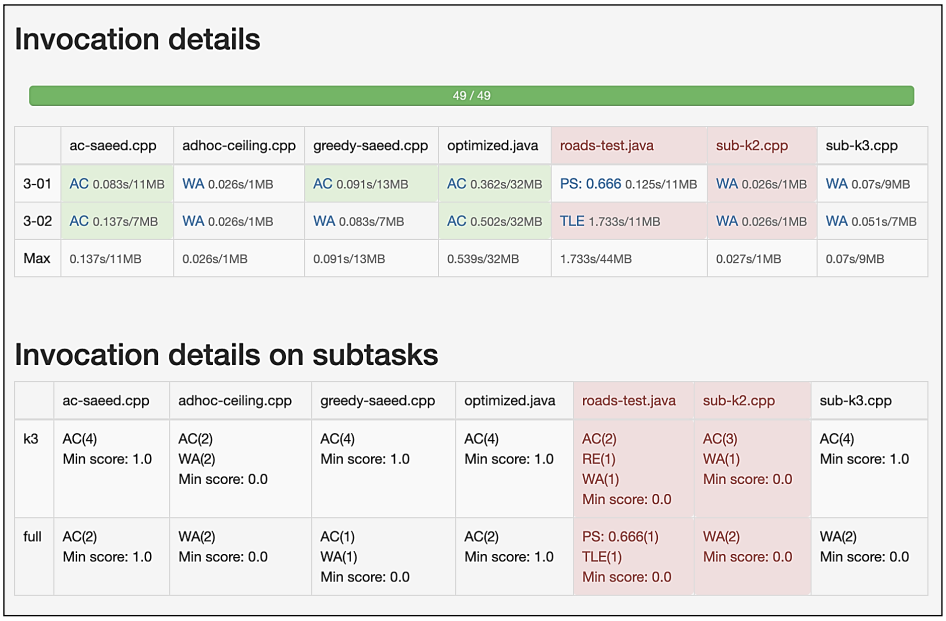


Fig. 5. A sample task statement page in tps-web.



Invocation details

49 / 49

	ac-saeed.cpp	adhoc-ceiling.cpp	greedy-saeed.cpp	optimized.java	roads-test.java	sub-k2.cpp	sub-k3.cpp
3-01	AC 0.083s/11MB	WA 0.026s/1MB	AC 0.091s/13MB	AC 0.362s/32MB	PS: 0.666 0.125s/11MB	WA 0.026s/1MB	WA 0.07s/9MB
3-02	AC 0.137s/7MB	WA 0.026s/1MB	WA 0.083s/7MB	AC 0.502s/32MB	TLE 1.733s/11MB	WA 0.026s/1MB	WA 0.051s/7MB
Max	0.137s/11MB	0.026s/1MB	0.091s/13MB	0.539s/32MB	1.733s/44MB	0.027s/1MB	0.07s/9MB

Invocation details on subtasks

	ac-saeed.cpp	adhoc-ceiling.cpp	greedy-saeed.cpp	optimized.java	roads-test.java	sub-k2.cpp	sub-k3.cpp
k3	AC(4) Min score: 1.0	AC(2) WA(2) Min score: 0.0	AC(4) Min score: 1.0	AC(4) Min score: 1.0	AC(2) RE(1) WA(1) Min score: 0.0	AC(3) WA(1) Min score: 0.0	AC(4) Min score: 1.0
full	AC(2) Min score: 1.0	WA(2) Min score: 0.0	AC(1) WA(1) Min score: 0.0	AC(2) Min score: 1.0	PS: 0.666(1) TLE(1) Min score: 0.0	WA(2) Min score: 0.0	WA(2) Min score: 0.0

Fig. 6. An example of invocation in tps-web.

gether with their running time and memory used. An important feature here is the highlighting of the results which do not meet the expected verdict. An example of tps-web invocation is illustrated in Fig. 6.

Integrating with judging systems. Besides having an internal judging system for invocations, tps-web is also capable of integrating with other judging systems, such as CMS, in order to obtain an exact timing. It is in particular useful in programming contests such as IOI where the execution time of solutions are important up to milliseconds, and any small difference in hardware and software environment (such as sandboxing) can considerably affect the timing.

Discussion forum. In the time of task development, there are various topics that need to be discussed such as designing subtasks, reporting a bug, or an issue in the problem statement. In tps-web, every user can open a discussion about an issue and other users can reply to the thread.

Secure file sharing. During task development, there are numerous situations where developers need an easy way to share a file that is used temporarily during development. To address this need, tps-web provides a facility to store temporary files during development.

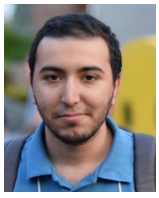
Export final packages. Since each judging system has its own directory format for presenting a task, tps-web provides the facility to automatically generate a package from the TPS directory structure. In particular, it has a built-in exporter for the CMS.

The source code and full documentation of tps-web is available at:
`https://github.com/ioi-2017/tps-web`.

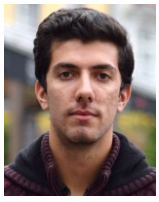
Acknowledgments

We would like to acknowledge the complete list of people who were involved in developing TPS (in alphabetical order): Amirmohsen Ahanchi, Soroush Ebadian, Kiarash Golezardi, Ali Haghani, Keyvan Khademi, Hamed Saleh, Hamed Valizadeh, Mohammad Reza Maleki, Kian Mirjalali, Amir Keivan Mohtashami, Mohammad Roghani, and Hamid Zarrabi-Zadeh.

K. Mirjalali is a PhD candidate in the Computer Engineering Department at Sharif University of Technology. He was a member of the International Technical Committee (ITC) in IOI 2015 and also a member of the Host Technical and Scientific Committees (HTC, HSC) in IOI 2017. He won a silver medal in CEOI 2003 and was a world-finalist in ICPC 2007. He has been a scientific committee member of Iranian National Olympiad in Informatics (INOI) since 2003.



A.K. Mohtashami is a B.Sc student in the Computer Engineering Department at Sharif University of Technology. He was a silver medalist in IOI 2015. He was also a member of the IOI 2017 Host Technical Committee. He also has been awarded Asia West Championship in ICPC 2018 and was co-coach of Sharif's bronze medalist team in ICPC 2019.



M. Roghani is a B.Sc student in the Computer Engineering Department at Sharif University of Technology. He was a member of the Host Scientific Committee (HSC) in IOI 2017. He has received a Gold Medal in Iranian National Olympiad in Informatics (INOI) 2014 and a Silver Medal in Asia-Pacific Informatics Olympiad (APIO) 2015.

H. Zarrabi-Zadeh is a faculty member of the Computer Engineering Department at Sharif University of Technology. He was a member of the International Scientific Committee (ISC) from 2014 to 2015, and a member of the International Technical Committee (ITC) from 2015 to 2018. He was also the chair of IOI 2017 Host Technical Committee, the director of ICPC in the west Asia region Tehran site since 2012, and a bronze medalist as coach in ICPC 2019 world finals.