Contents

- 1 Overview
- 2 Problem Metadata
- 3 Problem Statements
- 4 Test data
- 5 Example Submissions
    - 5.1 Input/Output Methods
- 6 Validators
    - 6.1 Input Format Validators
    - 6.2 Output Validators
    - 6.3 File Conventions For Validators
    - 6.4 Default Diff Capabilities
- 7 Verification
- 8 Appendix
    - 8.1 Configuration file format
    - 8.2 limits
        - 8.2.1 Sample problem.yaml
    - 8.3 Directory structure
        - 8.3.1 Sample Directory / Filenames
- 9 Miscellaneous

Overview

This document describes a standard for distributing and sharing problems for algorithmic programming contests. It is primarily designed for the ACM ICPC but is applicable also to other contests.

All problems must have a "short name" consisting solely of lower case letters a-z and digits 0-9. All files related to a given problem are provided in a directory named after the short name of the problem.

Problem Metadata

Metadata about the problem (e.g., name, source, license, limits) are provided in a file `<short-name>/problem.yaml`. The format of this file is defined in the Appendix.

Problem Statements

The problem statement of the problem is provided in the directory `<short-name>/problem_statement/`.

This directory must contain one LaTeX file per language, named `problem.<language>.tex`, that contains the problem text itself, including input and output specifications, but not sample input and output. Language must be given as an ISO 639-1 alpha-2 language code. Optionally, the language code can be left out, the default is then English.

A template will be provided that \imports this file as well as the sample input and output. Files needed by this file must all be in `<problem>/problem_statement/` , problem.tex should reference auxiliary files as if the working directory is `<problem>/problem_statement/`.

Test data

The test data are provided in subdirectories of `<short_name>/data/`. The sample data in `<short_name>/data/sample/` and the secret data in `<short_name>/data/secret/`.

All input and answer files have the filename extension .in and .ans respectively. Optionally a text file (with filename extension .txt) describing the purpose of an input file may be present. Input, answer and description files are matched by the base name.

Example Submissions

Correct and incorrect solutions to the problem are provided in a directory `<short_name>/submissions/`. The submissions/ directory has subdirectories corresponding to the judgement that the programs are supposed to receive:

1. `submissions/accepted/`: Solutions correctly solving the problem
2. `submissions/wrong_answer/`: Submissions producing incorrect answers
3. `submissions/time_limit_exceeded/`: Submissions that are supposed to be too slow
4. `submissions/run_time_error/`: Submissions that are supposed to crash

Every file or directory in these directories represents a separate solution. Same requirements as for submissions with regards to filenames. It is mandatory to provide at least one accepted solution. Everything else is optional, and empty subdirectories can be omitted.

Input/Output Methods

Submissions must operate in such a way that they receive problem input data on their standard input stream, and write to their standard output stream.

Validators

Input Format Validators

Input Format Validators, for verifying the correctness of the input files, are provided in `<short_name>/input_format_validators/`. They must adhere to the Input Format Validator standard.

Output Validators

Output Validators are used if the problem requires more complicated output validation than what is provided by the default diff variant described below. They are provided in `<short_name>/output_validators/`, and must adhere to the Output Validator standard.

File Conventions For Validators

A validator is either a file or a directory. A validator in the form of a directory may include two scripts "build" and "run".

If the "build" script is present, the validator must be compiled by executing the build script. Otherwise, the validator will be compiled as if it was a submission.

If the "run" script is present, the validator must be run by executing the run script. Otherwise, the validator will be run as if it was a submission (except that it is given the command-line arguments specified in problem.yaml, if there are any).

Default Diff Capabilities

TODO: describe the default diff capabilities.

Verification

Solutions or validators in languages that is not supported by the CCS should be ignored and a warning to that effect shown.

Verification Checks (in order)

1. Check files (all files present as required + check problem.yaml)
2. Check compile (check that all programs compile)
3. Check input (run input validators)
4. Check solutions (run all solutions check that they get the expected verdicts)

```
Warn if:
      there is no problem.tex
      there are no *.in in data/sample/

Error if:
      there is no problem.yaml
      any value in problem.yaml is invalid
      there are no *.in in data/secret/
      there are .in files without corresponding .ans files in data/*/
      there are .ans files without corresponding .in files in data/*/
      there are no solutions in submissions/accepted/
      there are no validators in input_format_validators/
      validator begins with "custom" and there are no validators in output_validators/
      there are validators in output_validators/ and validator does not begin with "custom"
      any validator (input format or output) does not compile

For each *.in in public_data and judge_data:
      For each validator in input_format_validators/:
              If the validator does not accept the input file: Error!

For each solution in test_submissions/accepted/:
      For each *.in in data/*/:
              Run the solution on the input
              For the built-in validator if corrector is "diff" or each validator in output_valida
                      If the validator does not accept the output of the solution: Error!
Let t be the longest time any of the solutions ran on any of the inputs.

For each solution in submissions/time_limit_exceeded/:
      For each *.in in data/*/:
```

```
              Run the solution on the input for at least t * time_limit_safety_margin seconds.
Let t_slow be the shortest time any of the solutions ran on any of the inputs.

If t_slow is less than t * time_limit_safety_margin: Error!

The execution time limit should be at least t * time_limit_multiplier and at most 1 second more.

For each solution in submissions/wrong_answer/:
      For each *.in in data/*/:
              Run the solution on the input
              For the built-in validator if corrector = "diff" or each validator in output_validat
                      If the validator accepts the output of the solution: Error!

For each solution in submissions/run_time_error/:
      For each *.in in data/*/:
              Run the solution on the input
              If the solution is not judged Run-Time Error: Error!
```

Appendix

Configuration file format

problem.yaml is a YAML file consisting of a mapping with the following keys:

Key	Comments	Example
name	mandatory	Hello World
source	optional	ICPC World Finals 2011
author	optional, defaults to "Unknown"	
license	optional, defaults to "cc by-sa"	
rights_owner	mandatory	ICPC
keywords	optional	
difficulty	optional	
limits	Sequence of mappings with keys as defined below	
validator	optional, set of values from the following (delimited by spaces): • "case_sensitive" - upper/lower case differences are significant. If this parameter is not specified, any changes in case are to be ignored. • "space_change_sensitive" - whitespace differences are significant. If this parameter is not specified, any non-zero amounts of whitespace are considered identical. • "float_relative_tolerance X" - accepts token if it is a floating point number and the relative error is <= X • "float_absolute_tolerance X" - accepts token if it is a floating point number and the absolute error is <= X • "float_tolerance X" - accepts token if either "float_relative_tolerance X" or "float_absolute_tolerance X" would accept • "custom" - use a custom output validator. Must be the first value. All following values will be passed as command-line arguments to each of the output validators	

limits

A sequence of mappings with the following keys:

Key	Comments	Example
time_multiplier	optional, defaults to 5	3.5
time_safety_margin	optional, defaults to 2	1.5
memory	optional, in Mb, defaults to 2048	3072
output	optional, in Mb, defaults to 8	4
compilation_time	optional, in seconds, defaults to 60	120
validation_time	optional, in seconds, defaults to 60	120
validation_memory	optional, in Mb, defaults to 2048	3072
validation_output	optional, in Mb, defaults to 8	4

Sample problem.yaml

Typical problem.yaml:

```
# Problem configuration
name: Squares to Circles
source: ICPC Mid-Atlantic Regional Contest
author: John von Judge
rights_owner: ICPC
```

Maximal problem.yaml:

```
# Problem configuration
name: Squares to Circles
source: ICPC Mid-Atlantic Regional Contest
author: John von Judge
license: cc by-sa
rights_owner: ICPC

limits:
   time_multiplier: 5
   time_safety_margin: 2
   memory: 4096
   output: 16
   compilation_time: 240
   validation_time: 240
   validation_memory: 3072
   validation_output: 4

validator: space_change_sensitive float_absolute_tolerance 1e-6
```

Directory structure

```
<short_name>/
      problem.yaml - problem configuration file
      problem_statement/
              problem.tex - problem statement
              - any files that problem.tex needs to include, e.g. images
      data/
              sample/
                      *.in - sample input files
                      *.ans - sample answer files
              secret/
                      *.in - input files
```

```
                *.ans - answer files
                *.txt - optional data file description
    submissions/
            accepted/
                    - single file or directory per solution
            wrong_answer/
                    - single file or directory per solution
            time_limit_exceeded/
                    - single file or directory per solution
            run_time_error/
                    - single file or directory per solution
    input_format_validators/
            - single file or directory per validator
    output_validators/
            - single file or directory per validator
```

Sample Directory / Filenames

This is a sample list of directories/files for a problem named *squares*

```
squares/problem.yaml
squares/problem_statement/problem.en.tex
squares/problem_statement/problem.sv.tex
squares/problem_statement/square1.png
squares/problem_statement/square2.png
squares/data/sample/squares_sample1.in
squares/data/sample/squares_sample1.ans
squares/data/sample/squares_sample2.in
squares/data/sample/squares_sample2.ans
squares/data/secret/squares1.in
squares/data/secret/squares1.ans
squares/data/secret/squares1.txt
squares/data/secret/squares2_cornercases.in
squares/data/secret/squares2_cornercases.ans
squares/data/secret/squares3_bigcases.in
squares/data/secret/squares3_bigcases.ans
squares/submissions/accepted/squares.cpp
squares/submissions/accepted/Squares.java
squares/submissions/accepted/squares.c
squares/submissions/wrong_answer/wrong.cpp
squares/submissions/time_limit_exceeded/tle.c
squares/submissions/run_time_error/rte.c
squares/input_format_validators/squares_input_checker1.py
squares/input_format_validators/squares_input_checker2/check.c
squares/input_format_validators/squares_input_checker2/data.h
squares/output_validators/squares_validator/validator.f
squares/output_validators/squares_validator/build
squares/output_validators/squares_validator/run
```

Miscellaneous

The following should be added somewhere above:

- Textfiles should be UTF-8 encoded.