Problem setting specification

IOI system team

June 12, 2014

Contents

Directory structure

每一個題目都會是一個資料夾，並命名為 該題題目名 。所有該題相關的檔案都放置其中，詳細結構後述。

- `config.yaml`: yaml 檔。內有該題的所有設定。
- `attachment`: 資料夾。內含所有要在比賽時給選手的檔案。
- `description`: 資料夾。內含所有語系的題目 pdf 檔。(翻譯組負責)
- `judge`: 資料夾。內含所有 cms 系統評分所需的檔案。
- `testdata`: 資料夾。內含所有測試資料。
- 上述資料夾中請勿放入無關的檔案。例如詳解或是 solution code，測試用 code，測試資料產生器等等。

yaml format

yaml 是一種簡單的資料表達語言。語法上與 python 十分類似。

- 支援巢狀結構，並利用 indent 控制 block 的範圍。
- 註解則是以 # 開頭。

config.yaml

- `name: string`
  - 題目名。
  - 僅含小寫英文及數字和底線 (_)。
  - 需同該題的資料夾名稱。
- `title: string`
  - 題目原始標題，可以是個句子。
- `time_limit: float`
  - 執行時間限制。根據 IOI2013 rule，限制會是 generous，例如為出題者解答的 2 倍。
  - **每一筆測試資料**的限制皆相同。
  - 單位是秒。建議使用整數。
  - 所有測試資料的總執行時間 (所有 subtask 測試資料筆數和 × 時間限制) 不超過 3 分鐘。

config.yaml (cont.)

- `memory_limit: integer`
  - 記憶體使用量限制。根據 IOI2013 rule，限制會是 generous，例如為出題者解答的 2 倍。
  - **每一筆測試資料**的限制皆相同。
  - 單位是 MB。
- `subtask: list of dict`
  - 一個 list ，每個 element 是一個 subtask 的設定。每個 element 都包含以下兩個設定:
  - `score: integer`
    - 該 subtask 的配分。
  - `testdata: list of string`
    - 一個 list，每個 element 是一筆測試資料檔名 (不含附檔名)。
    - 測試資料並**不會**包含其他 subtask。如果需要共用，僅需在各別的 subtask 的設定中加入同樣的測試資料檔名即可。

config.yaml (cont.)

- 未來匯入 cms 後的 `subtask` 順序和 `testdata` 順序，都與本設定檔中的順序相同。

testdata

`testdata` 資料夾中僅放置所有的測試資料。

- subtask 的設定是寫在設定檔 `config.yaml` 中。
- 測試資料分為
  - 輸入檔。附檔名為 `.in`，換行需為 unix 格式。
  - 輸出檔。附檔名為 `.out`，換行需為 unix 格式。

Overview

judge 資料夾僅放置 cms 自動編譯和評分所需的檔案。

選手們僅需撰寫函式的實作，所以出題者需提供以下的檔案:

- grader: main 函式和 API 的實作
- checker: special judge (optional)
- header: 標頭檔

grader

grader 內有該語言 main 函式，以及所有 API 的實作。

- 檔名: `grader.{ext}` ，其中 `ext` 為對應語言的附檔名。
  - C 附檔名: `c`
  - C++ 附檔名: `cpp`
  - Pascal 附檔名: `pas`
- 評分時會與選手的程式編譯成一個執行檔，並從 `stdin` 讀入測試資料，把結果從 `stdout` 輸出。
- 編譯時不能產生任何 warning。
  - 最常見的是 `scanf` 的 return value 沒有存起來，被直接扔掉，這會產生 warning。
- Pascal 可以 link c/c++ 的 `.o` 檔。這讓不熟 Pascal 的出題者可以簡單的實作 grader 。(請參考 IOI2013 的 cave)
- C / C++ 需提供標頭檔，附檔名為: `.h`。

checker

輸出結果預設會使用 `diff` 來判斷正確性。如果題目較為複雜可以提供檢查正確性的程式。

- 檔名: checker
- 可以在評分系統上直接執行的執行檔。
  - 編譯時需 `-static`
  - 題目組僅需提供原始碼
- checker 執行的時候會被提供 3 個參數 (argv) ，分別為
  1. 輸入的檔名。(`.in` file in `testdata/`)
  2. 答案的檔名。(`.out` file in `testdata/`)
  3. 儲存選手程式輸出的檔名。(也就是 grader 的輸出)
- 讀入時需檢查格式是否正確。

checker (cont.)

- 評分的結果需要輸出到 `stdout` ，為一個 0.0 ~ 1.0 的浮點數，表示得分**比例**。
  - subtask 的分數會是該 subtask 的所有測資中得分最小的。
  - ex. Correct: 1.0, Incorrect: 0.0
- 評分的 feedback message 則輸出到 `stderr`。
  - message 僅能有 1 行，並不超過 80 字元。

Overview

`attachment` 資料夾中放有比賽時要提供給選手的檔案。包含

- 讓選手填入程式碼的 template。
- 含有 API 及選手需實做函式的 prototype 之標頭檔。
- 可以與 template 放在一起編譯，讓選手可以在本機編譯並測試的 grader。(下稱為 sample grader)
- 可以在本地編譯指定語言的 script。
- Sample input / output。

最後這些檔案會用 tar 和 gz 封裝後，放上 cms 提供給選手下載，並也會放一份選手的筆電中。

Template

對於 C, C++ 和 Pascal 都要分別提供一份 template，內含要讓選手實做的函式們。

- 函式的實作留空。舉例來說，有一個 return type 為 `int` 的函式，可以簡單地 `return 0;` 。
- 檔名: `{task_name}.{ext}`。其中 `task_name` 是題目名，`ext` 是對應語言的附檔名。

Sample grader and sample testdata

為了讓選手可以在本機測試自己的 solution，我們需要提供一份 grader。基本上大致同提供給 cms 的 grader，唯一的差別是 input / output 的格式可能會需要更動。

- 檔名: `grader.{ext}`。`ext` 是對應語言的附檔名。
- sample grader 僅需支援 sample testdata 的格式。
- sample testdata 的格式和使用方法需要在題目敘述中詳細定義。
- 需提供 `compile_{lang}.sh` 的 script 檔，可以讓選手使用這個 script 直接編譯出可測試用的執行檔。其中 `lang` 為 `c`, `cpp`, `pas`。
  - 支援 C11 和 C++11 標準。
  - C 編譯指令: `/usr/bin/gcc -DEVAL -static -O2 -std=c11 -o {task_name} grader.c {task_name}.c -lm`
  - C++ 編譯指令: `/usr/bin/g++ -DEVAL -static -O2 -std=c++11 -o {task_name} grader.cpp {task_name}.cpp`

Sample grader and sample testdata (cont.)

- Pascal 編譯指令: `/usr/bin/fpc -dEVAL -XS -O2 -o{task_name} grader.pas`
- 提供的 sample grader, template, script 必須能夠在沒有任何 warning 下成功編譯。

example

提供兩個簡單的 a + b problem 當做範例。

1. `add` 是給兩個數字，傳回其總和。
2. `reverse_add` 是給一數字，傳回兩整數使其和為該數。

細節請參考 `http://bitbucket.org/ioi2014/cms-example`