Special Judge

本页面主要介绍部分评测工具/OJ 的 spj 编写方法。

简介

Special Judge（简称：spj，别名：checker）是当一道题有多组解时，用来判断答案合法性的程序。

> **Warning**
>
> spj 还应当判断文件尾是否有多余内容，及输出格式是否正确（如题目要求数字间用一个空格隔开，而选手却使用了换行）。但是，目前前者只有 Testlib 可以方便地做到这一点，而后者几乎无人去特意进行这种判断。
>
> 判断浮点数时应注意 NaN。不合理的判断方式会导致输出 NaN 即可 AC 的情况。
>
> 在对选手文件进行读入操作时应该要检查是否正确读入了所需的内容，防止造成 spj 的运行错误。（部分 OJ 会将 spj 的运行错误作为系统错误处理）

> **Note**
>
> 以下均以 C++ 作为编程语言，以「要求标准答案与选手答案差值小于 1e-3，文件名为 num，单个测试点满分为 10 分」为例。

Testlib

参见：Testlib/简介，Testlib/Checker

Testlib 是一个 C++ 的库，用于辅助出题人使用 C++ 编写算法竞赛题。

必须使用 Testlib 作为 spj 的 评测工具/OJ：Codeforces、洛谷、UOJ 等。

可以使用 Testlib 作为 spj 的 评测工具/OJ：LibreOJ (Lyrio)、Lemon、牛客网等。

SYZOJ 2 所需的修改版 Testlib 托管于 pastebin[1]，但此修改版并未修改交互模式。syzoj/testlib 处托管了一份可以在 SYZOJ 2 上使用交互模式的 Testlib。

Lemon 所需的修改版 Testlib 托管于 GitHub - GitPinkRabbit/Testlib-for-Lemons。注意此版本 Testlib 注册 checker 时应使用 `registerLemonChecker()`，而非 `registerTestlibCmd()`。此

版本继承自 matthew99 的旧版，添加了一些 Testlib 的新功能。

DOMJudge 所需的修改版 Testlib 托管于 cn-xcpc-tools/testlib-for-domjudge。此版本 Testlib 同时可作为 Special Judge 的 checker 和交互题的 interactor。

Arbiter 所需的修改版 Testlib 托管于 testlib-for-arbiter。

其他评测工具/OJ 大部分需要按照其 spj 编写格式修改 Testlib，并将 testlib.h 与 spj 一同上传；或将 testlib.h 置于 include 目录。

```
#include "testlib.h"
//
#include <cmath>

int main(int argc, char *argv[]) {
  /*
   * inf: 输入
   * ouf: 选手输出
   * ans: 标准输出
   */
  registerTestlibCmd(argc, argv);

  double pans = ouf.readDouble(), jans = ans.readDouble();

  if (abs(pans - jans) < 1e-3)
    quitf(_ok, "Good job\n");
  else
    quitf(_wa, "Too big or too small, expected %f, found %f\n", jans,
pans);
}
```

Lemon

Note

Lemon 有现成的修改版 Testlib，建议使用 Testlib。

```
#include <cmath>
#include <cstdio>

int main(int argc, char* argv[]) {
  /*
   * argv[1]: 输入
   * argv[2]: 选手输出
   * argv[3]: 标准输出
   * argv[4]: 单个测试点分值
   * argv[5]: 输出最终得分 (0 ~ argv[4])
   * argv[6]: 输出错误报告
   */
  FILE* fin = fopen(argv[1], "r");
```

```
  FILE* fout = fopen(argv[2], "r");
  FILE* fstd = fopen(argv[3], "r");
  FILE* fscore = fopen(argv[5], "w");
  FILE* freport = fopen(argv[6], "w");

  double pans, jans;
  fscanf(fout, "%lf", &pans);
  fscanf(fstd, "%lf", &jans);

  if (abs(pans - jans) < 1e-3) {
    fprintf(fscore, "%s", argv[4]);
    fprintf(freport, "Good job\n");
  } else {
    fprintf(fscore, "%d", 0);
    fprintf(freport, "Too big or too small, expected %f, found %f\n",
jans,
            pans);
  }
}
```

Cena

```
#include <cmath>
#include <cstdio>

int main(int argc, char* argv[]) {
  /*
   * FILENAME.in: 输入
   * FILENAME.out: 选手输出
   * argv[1]: 单个测试点分值
   * argv[2]: 标准输出
   * score.log: 输出最终得分 (0 ~ argv[1])
   * report.log: 输出错误报告
   */
  FILE* fin = fopen("num.in", "r");
  FILE* fout = fopen("num.out", "r");
  FILE* fstd = fopen(argv[2], "r");
  FILE* fscore = fopen("score.log", "w");
  FILE* freport = fopen("report.log", "w");

  double pans, jans;
  fscanf(fout, "%lf", &pans);
  fscanf(fstd, "%lf", &jans);

  if (abs(pans - jans) < 1e-3) {
    fprintf(fscore, "%s", argv[1]);
    fprintf(freport, "Good job\n");
  } else {
    fprintf(fscore, "%d", 0);
    fprintf(freport, "Too big or too small, expected %f, found %f\n",
jans,
            pans);
  }
}
```

CCR

```cpp
#include <cmath>
#include <cstdio>

int main(int argc, char* argv[]) {
  /*
   * stdin: 输入
   * argv[2]: 标准输出
   * argv[3]: 选手输出
   * stdout:L1: 输出最终得分比率 (0 ~ 1)
   * stdout:L2: 输出错误报告
   */
  FILE* fout = fopen(argv[3], "r");
  FILE* fstd = fopen(argv[2], "r");

  double pans, jans;
  fscanf(fout, "%lf", &pans);
  fscanf(fstd, "%lf", &jans);

  if (abs(pans - jans) < 1e-3) {
    printf("%d\n", 1);
    printf("Good job\n");
  } else {
    printf("%d\n", 0);
    printf("Too big or too small, expected %f, found %f\n", jans, pans);
  }
}
```

Arbiter

```cpp
#include <cmath>
#include <cstdio>

int main(int argc, char* argv[]) {
  /*
   * argv[1]: 输入
   * argv[2]: 选手输出
   * argv[3]: 标准输出
   * /tmp/_eval.score:L1: 输出错误报告
   * /tmp/_eval.score:L2: 输出最终得分
   */
  FILE* fout = fopen(argv[2], "r");
  FILE* fstd = fopen(argv[3], "r");
  FILE* fscore = fopen("/tmp/_eval.score", "w");

  double pans, jans;
  fscanf(fout, "%lf", &pans);
  fscanf(fstd, "%lf", &jans);

  if (abs(pans - jans) < 1e-3) {
    fprintf(fscore, "Good job\n");
    fprintf(fscore, "%d", 10);
```

```
  } else {
    fprintf(fscore, "Too big or too small, expected %f, found %f\n",
jans,
            pans);
    fprintf(fscore, "%d", 0);
  }
}
```

HUSTOJ

```
#include <cmath>
#include <cstdio>

#define AC 0
#define WA 1

int main(int argc, char* argv[]) {
  /*
   * argv[1]: 输入
   * argv[2]: 标准输出
   * argv[3]: 选手输出
   * exit code: 返回判断结果
   */
  FILE* fin = fopen(argv[1], "r");
  FILE* fout = fopen(argv[3], "r");
  FILE* fstd = fopen(argv[2], "r");

  double pans, jans;
  fscanf(fout, "%lf", &pans);
  fscanf(fstd, "%lf", &jans);

  if (abs(pans - jans) < 1e-3)
    return AC;
  else
    return WA;
}
```

QDUOJ

相较之下，QDUOJ 略为麻烦。它带 spj 的题目没有标准输出，只能把 std 写进 spj，待跑出标准输出后再判断。

```
#include <cmath>
#include <cstdio>

#define AC 0
#define WA 1
#define ERROR -1

double solve(...) {
  // std
```

```
}

int main(int argc, char* argv[]) {
  /*
   * argv[1]: 输入
   * argv[2]: 选手输出
   * exit code: 返回判断结果
   */
  FILE* fin = fopen(argv[1], "r");
  FILE* fout = fopen(argv[2], "r");

  double pans, jans;
  fscanf(fout, "%lf", &pans);

  jans = solve(...);
  if (abs(pans - jans) < 1e-3)
    return AC;
  else
    return WA;
}
```

HDOJ

HDOJ 和 QDUOJ 的情况基本一致，也需要在 spj 中实现 std 后与选手输出比较。但与 QDUOJ 不同的是，HDOJ 会比较答案与 spj 输出在标准输出的内容后给出最终结果。因此，上传输出时仅需上传 spj 在正确时的输出即可。

HDOJ 需上传 Windows 下编译后的二进制文件，而非源代码。

```
#include <cmath>
#include <cstdio>

double solve(FILE* fin) {
  // std, read input from fin
}

int main(int argc, char* argv[]) {
  /*
   * argv[1]: 输入
   * stdin: 选手输出
   */
  FILE* fin = fopen(argv[1], "r");

  double pans, jans;
  if (scanf("%lf", &pans) != 1) {
    printf("WA\n");
    goto finish;
  }

  jans = solve(fin);
  if (abs(pans - jans) < 1e-3)
    printf("AC\n");
  else
```

```
    printf("WA\n");

finish:
  fclose(fin);
  return 0;
}
```

对应的答案文件为：

```
AC
```

SYZOJ 2

> **Note**
>
> SYZOJ 2 有现成的修改版 Testlib，建议使用 Testlib。
>
> LibreOJ 的最新版本已不再基于 SYZOJ，而是基于 Lyrio。Lyrio 支持使用原版 Testlib 编写评测器，这也是更加通用且推荐的做法。

```cpp
#include <cmath>
#include <cstdio>

int main(int argc, char* argv[]) {
  /*
   * in: 输入
   * user_out: 选手输出
   * answer: 标准输出
   * code: 选手代码
   * stdout: 输出最终得分 (0 ~ 100)
   * stderr: 输出错误报告
   */
  FILE* fin = fopen("input", "r");
  FILE* fout = fopen("user_out", "r");
  FILE* fstd = fopen("answer", "r");
  FILE* fcode = fopen("code", "r");

  double pans, jans;
  fscanf(fout, "%lf", &pans);
  fscanf(fstd, "%lf", &jans);

  if (abs(pans - jans) < 1e-3) {
    printf("%d", 100);
    fprintf(stderr, "Good job\n");
  } else {
    printf("%d", 0);
    fprintf(stderr, "Too big or too small, expected %f, found %f\n",
jans,
            pans);

```

```
    }
  }
```

牛客网

> **Note**
>
> 牛客网有现成的修改版 Testlib，建议使用 Testlib。

参见：如何在牛客网出 Special Judge 的编程题

```cpp
#include <cmath>
#include <cstdio>

#define AC 0
#define WA 1

int main(int argc, char* argv[]) {
  /*
   * input: 输入
   * user_output: 选手输出
   * output: 标准输出
   * exit code: 返回判断结果
   */
  FILE* fin = fopen("input", "r");
  FILE* fout = fopen("user_output", "r");
  FILE* fstd = fopen("output", "r");

  double pans, jans;
  fscanf(fout, "%lf", &pans);
  fscanf(fstd, "%lf", &jans);

  if (abs(pans - jans) < 1e-3)
    return AC;
  else
    return WA;
}
```

DOMJudge

> **Note**
>
> DOMJudge 支持任何语言编写的 spj，参见：problemarchive.org output validator 格式。
>
> DOMJudge 有现成的修改版 Testlib，建议使用 Testlib。

DOMJudge 使用的 Testlib 及导入 Polygon 题目包方式的文档：https://github.com/cn-xcpc-tools/testlib-for-domjudge

DOMJudge 的 默认比较器 自带了浮点数带精度比较，只需要在题目配置的 `validator_flags` 中添加 `float_tolerance 1e-3` 即可。

```cpp
#include <cmath>
#include <cstdio>

#define AC 42
#define WA 43
char reportfile[50];

int main(int argc, char* argv[]) {
  /*
   * argv[1]: 输入
   * argv[2]: 标准输出
   * argv[3]: 评测信息输出的文件夹
   * stdin: 选手输出
   */
  FILE* fin = fopen(argv[1], "r");
  FILE* fstd = fopen(argv[2], "r");
  sprintf(reportfile, "%s/judgemessage.txt", argv[3]);
  FILE* freport = fopen(reportfile, "w");

  double pans, jans;
  scanf("%lf", &pans);
  fscanf(fstd, "%lf", &jans);

  if (abs(pans - jans) < 1e-3) {
    fprintf(freport, "Good job\n");
    return AC;
  } else {
    fprintf(freport, "Too big or too small, expected %f, found %f\n",
jans,
            pans);
    return WA;
  }
}
```

也可以使用 Kattis Problem Tools 提供的头文件 validate.h 编写，以实现更加复杂的功能。

参考资料

1. LibreOJ 支持 testlib 检查器啦！ ↩