数据结构编程作业说明

清华大学计算机系

数据结构教学团队

2014.9

目录

一、 概览

1. 编程作业在线提交，网址 http://dsa.cs.tsinghua.edu.cn/oj/。
2. 每学期 4 次作业，每次作业 3 道题。
3. 如无特殊说明，所有题目统一使用标准输入和标准输出。
4. 输入通常都指定了取值范围，你的程序不必考虑范围之外的情况。
5. 有的题目限制了你能使用的方法。
6. 程序输出应与标准答案一致，请留意多余的空格、制表符、回车和换行。
7. 应对算法的时间和空间复杂度做充分和准确的估计，正确但低效的程序无法获得高分，甚至可能无法得分。

二、 作业提交

1. 在截止时间前，提交代码至 OJ 并“标记为最终版本”（Mark it as final version）；截止时间前可以多次标记，以最后标记的为准；通过其它渠道（电子邮件、网络学堂等）提交无效。
2. 每道题若查阅资料参考了代码，应当额外在 OJ 上通过 Honor Code 给予声明。
3. 代码如果含有多个文件，可以将所有代码置于顶层目录直接打包（zip、tar、gz 等格式）提交。
4. 每次提交的文件不超过 200KB。

三、 评测标准

5. 无效的提交记 0 分，包括（但不限于）以下方面：
   a) 未在作业截止时间前标记最终版本。
   b) 标记的最终版本在 OJ 上无法通过编译。
   c) 未在题面规定的范围内选用数据结构和算法。但题目的提示只是提供了一些方法，不具有约束力。
6. 每题得分由黑盒测试结果所决定。

四、 黑盒测试

1. 每次作业有一个 Deadline。每个作业 deadline 后的提交不会记入成绩，请务必留意。
2. 在 DDL 前，可以进行自由提交，全部 20 个测试点将会参与测试。

五、 晋级课堂

1. 对于学有余力的同学，可以在 MOOC 课堂任意一次 PA 达到 90 分之后在讨论区递交申请在 OJ 上加入 THU 课堂（清华大学全校通选课数据结构）。
2. 进一步地，在 THU 课堂的任意两次 PA 达到 90 之后可以继续在讨论区递交申请在 OJ 上加入 CST 课堂（清华大学计算机系专业基础课数据结构）。
3. THU 和 CST 课堂的黑盒测试策略与 MOOC 课堂有所区别，请务必留意。

附：FAQ

1. 杂项

a) 助教的联系方式？
请使用 xuetangX 或 edX 的讨论区与助教取得联系。

b) 课程讲义在哪下载？
讲义统一在 MOOC 学堂在线的数据结构课堂发布。
http://xuetangx.com/courses/TsinghuaX/30240184X/2014_T2/info

c) 如何提问，并更加高效地得到有帮助的解答？
描述问题时请不要过于概括，比如这样两个问题：
“我本地运行正确怎么在 OJ 上就错了？”
“我本地运行正确，为什么 OJ 上编译错误，并提示 main should return int？”
显然，后者更能得到有帮助的回答。

d) 请问总评成绩的额外加分是什么意思？
针对于 MOOC 课堂，我们会针对各位同学在讨论区内的活跃情况（例如一些纠错、回应其他同学的疑惑）酌情予以加分。（灌水虽好，可不要贪杯哦）

2. OJ 使用说明

a) OJ 上需要提交哪些文件？
只需要提交源代码文件(*.cpp/*.c/*.h)和 readme.txt，其余文件尽量不要一同打包。

b) OJ 上的编译环境与本地有何不同，需要注意哪些问题？
OJ 上使用的是 gcc 编译器，与 visual studio 中的 MSVC 编译器有一些微小的区别。例如 main 函数的返回值必须为 int，默认不会包含任何头文件等等。一般认为 gcc 编译器是更加符合 C/C++标准的，所以如果遇到了本地编译成功 OJ 编译失败的例子，请根据提示进一步修改源程序。

c) OJ 上是否禁止使用某些库？
是的，在 OJ 上移除了某些头文件，例如 vector、algorithm。原则上，只要你的作业能

够通过编译，且不违反题目的要求，就没有任何问题。不过不包括手动把这些头文件与你的作业其他代码一同打包提交上来:)

d) 输入输出应该采用哪些函数？

请使用 scanf 和 printf 来代替 cin 和 cout，某些情况下后者效率远远远远低于前者。更高效的读入是用 fread，然后手动解析文本。

e) Runtime Error 是怎么回事？

Runtime Error 是指程序在运行过程中出现了问题，通常是内存访问的问题，比如数组下标越界。一般这些问题在小规模测试的时候不会发现，而在 OJ 上大规模数据测试时候就容易暴露出来，所以请自行构造一些数据来调试程序。此外，需要注意的是，main 函数必须以 return 0;结束，如果返回值非 0，也会被认为是 Runtime Error。如果保证返回值为 0 并且希望知道 OJ 返回错误代码的含义，可以参考 http://unixhelp.ed.ac.uk/CGI/man-cgi?signal+7 。常见的是 8 除 0 错和 11 内存访问错误。

3. 作业说明

a) 必须采用题目中【提示】所指出的方法吗？

不是的，题目中的【提示】只是一种可行思路，不排除有更好的思路，可以任意选择方法。但是如果题目正文和题目的【限制】中对方法有限制，那么必须遵守。

b) 每个作业只能标记一次 Final Version 吗？

Deadline 前可以多次 mark final version，评测时以最后一次 mark 的为准。

c) 程序调试了很久没有调通，可以使用助教来进行调试吗？

由于调试工作量实在太大，所以助教不会来帮助大家进行调试:)比较建议大家在同学中寻找一些伙伴来互相帮助调试，这样双方都会有明显的进步，助教自己的程序就是这样调试的。

d) 题面上的“输入样例”就是第一个测试点吗？

不一定，输入样例与测试点没有直接联系。

e) 教材上的示例代码可以直接在作业中使用吗？

1. 支持借鉴、学习教材中的代码，完善自己作业程序的效率；
2. 如果直接使用教材中的代码，需要进行声明；
3. 原则上，并不鼓励使用教材和课件中的代码，鼓励同学们自己实现、完善相应的数据结构；
4. 教材中的代码，更多的是一种原理的示范，并不保证正确性，如果因此出现 Bug，请同学们自负后果。

f) 如果作业抄袭后果如何？

这会给被抄袭同学带来极大的困扰，因为会导致记-100 分，太惨了:)

提前说明的是本课程对作业雷同的判定比较严格，所以请大家独立完成作业。

需要强调的是，在签署的 Honor Code 中，你已承诺“我未曾也不会向同一课程（包括此后各届）的同学复制或公开我这份程序的代码，我有义务妥善保管好它们”。

附：OJ 使用技巧

1. OJ 使用手册 http://dsa.cs.tsinghua.edu.cn/oj/guide.shtml
   视频教程 https://d2f1egay8yehza.cloudfront.net/tsg-ds/TSGDATAST114-V052400_DTH.mp4
2. 几种主要的读入数据方法，大多数情况下性能是 fread > getchar > scanf > cin 的顺序。
3. OJ 上的编译器采用的是 gcc（g++），对于 Linux 和 Mac 用户可以无缝衔接，而如果使用 Windows 的同学希望能够在本地编译与 OJ 编译之间较快衔接的话，可以考虑使用 MinGW。
4. Runtime Error 的 exit code 同样反映了某种信息，可以查阅相关资料了解 RE 各个 exit code 的意义。参考 http://unixhelp.ed.ac.uk/CGI/man-cgi?signal+7。
5. 评测所依赖的输出流是 stdout，而使用 stderr 的输出不会被纳入最终评测——但如果你使用 stderr 进行调试却没有在提交中移除，这将显著地拖累你程序的性能。
6. 常用的调试工具包括 gdb、MSVC debugger、室友、小黄鸭。
7. 如果不确定某一特性是否为 OJ 所支持，不妨动手试一试。

附：输入输出技巧

1. 判断输入结束

   有些编程作业题并未指明测试数据的组数，此时需要自己判断输入结束。其实，根据题意正确处理输入数据也是同学们在这门课中需要练习的编程能力之一。

   处理输入的方法很简单，使用 C++风格的 cin，可以这样写

   ```
   string a, b; char c;
   while (cin >> a >> b >> c)
   { /* blablabla */ }
   ```

   如果使用 C 风格的 scanf()函数，则可根据其返回值做出判断，具体地可以这样写：

   ```
   while (scanf("%s\n%s\n%c\n\n", &a, &b, &c) != EOF)
   { /* blablabla */ }
   ```

   这样当格式输入流读到文件末尾时会返回 EOF，于是 while 退出。

2. 重定向

   为便于反复测试及再现运行过程，可采用输出、输入重定向的方法。

   你只需事先将输入数据存成文件，运行时系统会自动从中获取输入。其效果完全等同于你从（作为默认输入流的）键盘逐项输入。

   类似地，你也可以指定另一文件，并使运行的结果自动存入其中。其效果完全等同于从（作为默认输出流的）屏幕截取输出结果。

   重定向的好处很多：可以避免手工输入的出错，忠实可靠地重复测试；可以实现大规模数据的输入；可以完整精确地记录程序的输出，以便事后的对比分析；可以省去默认输入、输出流占用的大量时间，更加准确地测量程序的执行效率。

   d) 方法一：修改源文件，指定重定向的输入、输出文件

   例如，若希望从文件 input.txt 中获取输入，将输出保存到文件 output.txt 中，

则可在主程序开头增加如下语句：

```
#ifndef _OJ_
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
#endif
```

注意：如果用 c++风格的 cin/cout 的话，还要在前面引用头文件的部分加入 #include <cstdio>。

OJ 在编译程序的时候会提供一个_OJ_的符号，所以上面这段语句会在 OJ 运行的时候被跳过。

e) 方法二：在 IDE 中通过设置命令行，重定向输入、输出文件

以 Visual Studio 为例，可打开对应工程的“属性页”，在“配置属性”下的“调试”页，设置“命令行参数”。

输入参数不多时，可直接键入。例如 ADD 一题，键入“100 200”即可。若其中包含特殊字符，则需以'^'引导，或者使用一对半角括号消除歧义。

若输入参数多，且不止一行，则可将其存成一个文件。比如，可在“命令行参数”中键入:

```
< D:\test\input.txt
```

（注意起始字符"<"不能省略）

为将程序的输出保存至指定文件，可在“命令行参数”中继续键入：

```
> D:\result\output.txt
```

（同样地，起始字符"<"也不能省略）

若不希望覆盖文件原有的内容，只需用">>"替换以上的">"，即可将每次运行的输出追加至 D:\result\output.txt。

输入、输出的重定向可同时采用并生效。比如可在“命令行参数”中键入：

```
< D:\test\input.txt >> D:\result\output.txt
```

重定向文件的具体路径与文件名可自行选择，但若包含空格，则需使用一对半角引号消除歧义，比如：

```
< "D:\my test\input.txt" >> "D:\my result\output.txt"
```

3. 帮助资料

关于输入输出的进一步问题，可以自己查阅相关手册或资料。

也可参考标准手册，以上输入输出方法都是 C/C++标准输入输出，在 manual 中都有详细介绍。

cin：http://www.cplusplus.com/reference/iostream/cin/

scanf：http://linux.die.net/man/3/printf

对比：https://www.byvoid.com/blog/fast-readfile/