前言

整理了一些常见问题和技巧。

作业遇到问题可以先按照以下流程尝试自己找出错误，实在找不出来**可以问助教，助教可能会根据情况给一些提示。**在询问助教的时候**请告诉助教你做了哪些尝试。**

错误自查

如果 OpenJudge 上提交了你觉得正确的结果但是发现错了，可以按照以下方式先自查。

这里需要**强调**，一般来说，输入输出样例的数据都比较弱，没有覆盖各种可能的情况，**样例数据能正确输出并不代表你的程序是对的。**

Presentation Error

输出格式错误，比如空格数量和换行没有按照要求来。这个应该较好查出错误。但是可能还有一些特殊情况：

- 某些题目可能会出现虽然你的答案理论上是错误答案（Wrong Answer），但是因为这个答案跟正确答案也只有换行和空格不一样，所以最后报了 PE。一个例子是打印月历这个题，有些同学就遇到了这个情况。这样的题目应该不多，了解即可。
- 可能是某个特殊情况下你的输出格式不对，请多试一些数据。
- 有些题目可能输出的描述不够清楚，甚至和样例输出看起来不太一样，请以样例输出的格式为准。

Runtime Error

一般是运行时遇到了异常，比如字典访问一个不存在的 key 或是数组越界访问，都会导致这个结果。

在遇到这个情况时，理论上只要你能检查出你的代码中哪里可能触发这个错误，然后去检查那一部分的逻辑即可，但实际上盯着代码看可能很难看出具体哪里出错了，这里可以提供一些技巧：

1. 使用 `try ... except ...` 语句，如果不知道这个语句可以先去学习一下。这个语句可以在出错时捕获异常并执行异常处理逻辑。下面是一个示例：

   ```
   N = int(input()) # 假设有 N 组数据
   for _ in range(N):
       # 这里是你认为没问题的代码
       try:
           # 这里是你认为有问题的代码
       except:
           print("!") # 随便输入一个错误答案
           continue # 记得跳过之后的逻辑，防止又出了其他的错误
       # 这里是你认为没问题的代码
   ```

   如果你确实将出问题的代码用 `try ... except ...` 包起来了，那你提交之后应该会从 `Runtime Error` 变成 `Wrong Answer`，如果不是的话说明出错的不是你包起来的语句。你可以用二分查找的思想来包裹不同的语句，最后定位到真正出问题的地方，再找到问题出在哪一句代码之后可能就比较好修改了。

   这个方式比较暴力，但是对做作业来说也算是一种方法。

2. 随机生成数据。这是一个较为通用的方法。随机生成大量数据并运行，可能就能碰见某次随机生成的数据触发了这个错误。

3. 小黄鸭调试法。这也是一个较为通用的方法，可以搜索这个关键词学习。

Compile Error

语法都错了，一般会提示你错哪了。

Time Limit Exceeded

首先查看是否出现了死循环。

可以按照老师上课讲的方法尝试分析一下复杂度，一般按 1s 时间里 1e8 次操作估算（以题目中给的时间限制为准，因为 python 比较慢，实际评测时的时间限制是题目中写的限制的10倍）。如果数量级确实差的很大，说明你的方法有问题。如果你感觉算法复杂度没什么问题，那可能是某些操作时间有点慢，可以尝试优化一下。如果还是不行，可以来问助教，告诉助教为什么你觉得你的方法没问题（即你对你算法的复杂度分析）。

Memory Limit Exceeded

我们的作业不会卡内存（除非你的代码用的实在太多了），一般大概遇不到。

Wrong Answer

这个大概是同学们做题最常遇到的问题，也是助教经常遇到有人来问的问题。在很多情况下可能是没有考虑一些特殊情况。在遇到这个问题时，请首先对照题目要求，思考有没有什么比较特殊的输入输出情况。如果仍然没找到错误，可以尝试构造一些随机输入输出观察一下。如果始终找不到错误，可以来问助教，但是需要描述你想了哪些情况，构造了哪些数据。

一些特殊情况

有时候会遇到有些同学 Wrong Answer 的原因其实是因为多组数据时上一组的输出没有换行，但是因为本地测试时都是在 terminal 里面手动输入的，输入输出是混在一起的，可能没有看清楚，这个可以注意一下。本地写代码时也可以使用以下后面的文件输入输出技巧，不用每次都手动输入。

技巧

输入输出重定向（从文件读入并输出到文件）

在本地执行文件时，因为代码里用的都是 input() 方法，从标准输入流读取数据，可能很多同学都是手动输入或者将数据复制进来输入，这里介绍一个输入输出重定向的方法。

可以在代码的开头加以下几行：

```
import sys
sys.stdin = open("data.in", "r", encoding="utf-8")
sys.stdout = open("data.out", "w", encoding="utf-8")
```

这些代码的作用是，从 `data.in` 文件中读取输入，输出到 `data.out`。注意，这里用的是相对路径，`data.in`和`data.out`应当在默认工作路径下。你也可以换成你喜欢的文件路径。注意，提交时应当把这几行代码删掉或者注释掉。

以下面这个处理 `a + b` 问题的代码为例。

```
N = int(input())
for _ in range(N):
    x, y = map(int, input().split())
    print(x + y)
```

可以将代码修改为：

```
import sys
sys.stdin = open("data.in", "r", encoding="utf-8")
sys.stdout = open("data.out", "w", encoding="utf-8")

N = int(input())
for _ in range(N):
    x, y = map(int, input().split())
    print(x + y)
```

然后建一个 `data.in` 文件：

```
4
1 2
3 4
2 3
1 0
```

运行代码，你就能在 `data.out` 文件里查看结果了。这种方式不仅方便，而且输出不会跟你的输入混在一起，方便你查看结果。

随机生成数据

前面介绍了有一种 debug 的方式是随机生成一些数据来测试你的代码，可以跟文件输入输出的技巧结合。写一个用于生成数据的代码，将生成的数据写入到 `data.in` 文件中，然后运行代码后查看结果是否符合预期即可。当然你也可以写一个循环，一直生成数据并运行代码，查看是否会遇到 Runtime Error 的情况。