“科大国创杯”2023 年安徽省青少年信息学科普日活动 ACSP-J 组

2023 年 4 月 8 日 14:00 ~ 18:00

题目名称	评分	数数	行走	石子
题目类型	传统型	传统型	传统型	传统型
目录	score	count	walk	stone
可执行文件名	score	count	walk	stone
输入文件名	score.in	count.in	walk.in	stone.in
输出文件名	score.out	count.out	walk.out	stone.out
测试点时限	1.0 秒	1.0 秒	2.0 秒	1.0 秒
内存限制	1024 MiB	1024 MiB	1024 MiB	1024 MiB

提交源程序文件名

对于 C++ 语言	score.cpp	count.cpp	walk.cpp	stone.cpp

编译选项

对于 C++ 语言	-O2 -std=c++14

注意事项与提醒：

1. 文件名（程序名和输入输出文件名）必须使用英文小写。
2. 函数 main() 的返回值类型必须是 int，程序正常结束时的返回值必须是 0。
3. 选手需要在桌面上建立以选手的参赛号为名的目录，并由选手为每道题再单独建立一个子目录，子目录名与对应的试题英文名相同。选手提交的每道试题的源程序必须存放在相应的子目录下。
4. 因违反以上三点而出现的错误或问题，申诉时一律不予受理。如果现场有不同要求，以现场为准。
5. 若无特殊说明，结果的比较方式为全文比较（过滤行末空格及文末回车）。
6. 程序可使用的栈内存空间限制与题目的内存限制一致。
7. 只提供 Linux 格式附加样例文件。
8. 评测在当前最新公布的 NOI Linux 下进行，各语言的编译器版本以其为准。

评分 (score)

题目描述

小可可在观看跳水比赛。

有 n 名选手来参加跳水比赛，有 m 名评委。在每位选手跳水之后，每位评委会给出他的分数。为了保证尽量公正客观，每位选手的得分是所有评委给出的分数中去掉最大值和最小值（如果有多个最大值/最小值，只去掉一个）之后，剩下的分数的平均值。

最后得分最大的选手获得第一名，得分第二大的选手获得第二名，以此类推。但是可能会出现同分的情况，在这种情况下，小可可会默认编号较小的选手排名更靠前。即，如果 3 号选手和 5 号选手的得分都是 70，那么小可可会认为 3 号选手的排名比 5 号选手更靠前。

现在小可可已经知道了所有选手得到所有评委的分数，他想让你帮他算出来选手的排名表，即对于 $1 \le i \le n$，算出排名第 i 的选手的编号是什么。

输入格式

第一行两个整数 n, m，分别表示选手个数和评委个数。

接下来 n 行每行 m 个整数，第 i 行第 j 个整数 $a_{i,j}$ 表示在第 i 个选手跳水之后，第 j 个评委给出的分数。

输出格式

输出一行 n 个整数，第 i 个整数表示排名为 i 的选手的编号。

样例 1 输入

```
4 4
4 70 69 34
18 43 85 71
100 50 69 80
67 82 90 43
```

样例 1 输出

```
3 4 2 1
```

样例 1 解释

四位选手的去掉最大、最小值之后的平均分分别是：$51.5, 57, 74.5, 74.5$，但由于三号选手编号比四号选手小，所以排名从 1 到 4 的选手分别为：$3, 4, 2, 1$。

样例 2

见 *samples/score/score2.in* 和 *samples/score/score2.ans*。

数据规模与约定

对于 30% 的数据，满足 $n, m \leq 3$；
对于 60% 的数据，满足 $n, m \leq 10$；
对于 100% 的数据，满足 $2 \leq n \leq 100, 3 \leq m \leq 100, 0 \leq a_{i,j} \leq 100$。

数数 (count)

题目描述

小可可和小多在拼木棍。

他们现在拿到了 n 根木棍，第 i 根木棍的长度是 a_i。他们现在想知道，有多少种在里面选三根木棍的方案，使得这三根木棍能组成一个三角形？

三根木棍能组成一个三角形，当且仅当较短的两根木棍长度和大于最长的那根木棍长度。

输入格式

第一行一个正整数 n，表示木棍的个数。

第二行 n 个正整数，第 i 个正整数 a_i 表示第 i 根木棍的长度。

输出格式

一行一个整数，表示有多少种选三根木棍的方案，使得这三根木棍能组成一个三角形。

样例 1 输入

```
5
3 2 5 3 4
```

样例 1 输出

```
8
```

样例 1 解释

可以选择的编号的方案是：$(1,2,4),(1,2,5),(1,3,4),(1,3,5),(1,4,5),(2,3,5),(2,4,5),(3,4,5)$。

样例 2

见 *samples/count/count2.in* 和 *samples/count/count2.ans*。

数据规模与约定

对于 20% 的数据，满足 $n \leq 100$；

对于 40% 的数据，满足 $n \leq 10^3$；

对于另外 20% 的数据，满足 $a_i \leq 5 \times 10^3$；

对于 100% 的数据，满足 $3 \leq n \leq 8 \times 10^3, 1 \leq a_i \leq 10^9$。

行走 (walk)

题目描述

小可可和小多来到了一个网格图上进行最短路训练。

这是一个 $n \times n$ 的网格图，对于点 (x, y)，如果 $y < n$，则它向 $(x, y+1)$ 有一条**有向边**，边权为 $ea_{x,y}$；如果 $x < n$，则它向 $(x+1, y)$ 有一条**有向边**，边权为 $eb_{x,y}$。小可可需要在很短的时间内找到从 $(1, 1)$ 到 (n, n) 的最短路。

然而，小多会捣乱 q 次：小雪会删去图中的一条边，然后小可可就需要重新计算 $(1, 1)$ 到 (n, n) 的最短路。当小可可计算完成后，小多就会恢复这条边。即：每次小多删掉的边只会影响到这一次小可可的计算。

小可可坚持尝试不借助外力，自己每次计算出答案。可惜小可可不是机器人，没过一会儿他就晕倒了。于是，计算最短路的任务就落到了你的头上。

输入格式

为避免过大的输入量，网格图边的边权将在程序内生成。

第一行两个正整数 n, q，表示网格的大小和小多捣乱的次数。

第二行两个整数 $seed$ 和 B，为辅助数据生成的变量。**请保证它们为全局变量且 $seed$ 是 64 位无符号整形变量。**

接下来按照从第一行到第 n 行，第一列到第 $n-1$ 列的顺序生成 $ea_{i,j}$：每次生成时先调用一次 `xorshift64()`，再将 $ea_{i,j}$ 赋值为 $seed\&(2^B - 1)$。其中 & 表示二进制按位且运算，`xorshift64()` 在选手目录下的 *samples/walk/xorshift* 中已经给出。

再接下来按照从第一行到第 $n-1$ 行，第一列到第 n 列的顺序生成 $eb_{i,j}$：每次生成时先调用一次 `xorshift64()`，再将 $eb_{i,j}$ 赋值为 $seed\&(2^B - 1)$。

请严格按照上述过程生成数据，中途不能自行改变 $seed$ 和 B 的值，否则数据将错误生成。

接下来 q 行，每行四个整数 x, y, x', y'，表示小多捣乱删掉的一条边。保证 $x' = x, y' = y + 1$ 或 $y' = y, x' = x + 1$。

如果你仍不会生成数据，选手目录下的 *samples/walk/sample.cpp* 已经实现好了所有的数据读入/生成，你可以直接使用在你的代码中。**再次提醒，请不要自行改动 $seed$，B，`xorshift64()` 函数和程序读入/生成数据的顺序。**

此数据生成方式仅是为了减少输入量大小，标准算法不依赖于该生成方式。

输出格式

输出 q 行，每行一个整数，表示删掉给定边后 $(1, 1)$ 到 (n, n) 的最短路。

样例 1 输入

```
4 4
10583998785722269293 2
2 2 2 3
2 3 3 3
1 2 2 2
1 1 1 2
```

样例 1 输出

```
5
4
7
9
```

样例 2

见 *samples/walk/walk2.in* 和 *samples/walk/walk2.ans*。

数据规模与约定

对于 20% 的数据，满足 $n, q \leq 5, B = 1$；
对于 40% 的数据，满足 $n, q \leq 300$；
对于 70% 的数据，满足 $n \leq 300$；
对于 100% 的数据，有 $2 \leq n \leq 5000, 1 \leq q \leq 10^5, 1 \leq B \leq 30, 1 \leq x, y, x', y' \leq n$。

石子 (stone)

题目描述

小可可面前有 n 堆石子，第 i 堆石子有 a_i 个石子。小可可想要在开始选择一堆石子，然后从它开始，每次合并这堆石子左边的那堆石子或者右边的那堆石子。合并两堆石子个数为 x, y 的石子堆需要花 $x + y$ 的力气，并且会合并成一堆 $x + y$ 个石子的石子堆。

小可可想花费最小的力气从最初选择的那堆石子开始，将所有石子都合并完。小可可想知道，如果他选择编号在 $[l, r]$ 里面的每一堆石子作为最初的石子，那么他将 n 堆石子合并成一堆花的最小力气是多少。

小可可不想太为难你，所以他保证所有的 a_i 是**随机**的。

输入格式

第一行输入一行三个整数 n, l, r，石子堆数和最开始选择的那堆石子的编号区间。

第二行输入 n 个整数 $a_1, a_2, \cdots, a_n$，表示每堆石子的石子个数。

输出格式

输出一行 $r - l + 1$ 个整数，第 i 个整数表示小可可选择编号为 $l - 1 + i$ 的石子堆作为最开始的那堆石子时，将所有石子都合并完花的最少力气。

样例 1 输入

```
4 1 4
5 1 3 1
```

样例 1 输出

```
25 19 19 19
```

样例 1 解释

对于第 1 堆石子作为初始的石子堆，最优（也是唯一）的合并策略是先合并第 2 堆，再合并第 3 堆，随后合并第 4 堆，花费力气为 25；

对于第 2 堆石子作为初始的石子堆，最优的合并策略是先合并第 3 堆，再合并第 4 堆，随后合并第 1 堆，花费力气为 19；

对于第 3 堆石子作为初始的石子堆，最优的合并策略是先合并第 2 堆，再合并第 4 堆，随后合并第 1 堆，花费力气为 19；

对于第 4 堆石子作为初始的石子堆，最优（也是唯一）的合并策略是先合并第 3 堆，再合并第 2 堆，随后合并第 1 堆，花费力气为 19。

样例 2/3/4

见 *samples/walk/stone*.in* 和 *samples/walk/stone*.ans*。

数据规模与约定

对于 20% 的数据，满足 $n \leq 10$；

对于 40% 的数据，满足 $n \leq 300$；

对于 60% 的数据，满足 $n \leq 5000$；

对于另外 20% 的数据，满足 $n \leq 10^5, r - l + 1 \leq 50$；

对于 100% 的数据，有 $1 \leq l \leq r \leq n \leq 10^5, 1 \leq a_i \leq 10^8$。保证 a_i 随机。随机方式为：先选择一个所有 a_i 的上界 v，对于每个 a_i，它在 $[1, v]$ 中的所有整数中等概率随机选取一个。