CSP-S 模拟赛

2019 年 11 月 13 日　08:00 - 11:30

题目概况

题目名称	地主斗	无聊	地斗主
源程序文件名	`landlord.c/cpp/pas`	`boring.c/cpp/pas`	`landlords.c/cpp/pas`
每个测试点时限	1 秒	1.5 秒	3 秒
测试点数目	捆绑测试	捆绑测试	20
每个测试点分值			5
运行内存上限	512 MB	512 MB	512 MB
题目类型	传统题	传统题	传统题
结果比较方式	全文比较	全文比较	Special Judge

※ 除特殊说明外，输入、输出文件的数字之间均以恰一个空格分隔，行末字符为 Line Feed（`'\n'`，ASCII 10），且该字符也存在于输入文件的最后一行末尾。全文比较会过滤行末回车和文末空格。

编译命令（以第一题为例）

C	`gcc -o landlord landlord.c -O2 -lm`
C++	`g++ -o landlord landlord.cpp -std=c++11 -O2 -lm`
Pascal	`fpc landlord.pas -O2`

注意事项

评测时，栈空间大小无特殊限制。

Problem A. 地主斗

输入文件: landlord.in
输出文件: landlord.out
时间限制: 1 秒
内存限制: 512 MB

这道题目的主角，不是小 U，不是小 Z，而是 CZL。

至于这道题目的名称为什么叫“地主斗”，是为了呼应 T3 的题目名称和背景。

CZL 在小 U 的班上，喜欢给班上的同学表演扑克牌魔术。作为一名合格的魔术表演者，他必须熟练、快速地洗牌。

CZL 的一轮洗牌的规则如下：假设从上往下一共有 n 张牌，从上往下的第 i 张牌在洗牌过后，变成了从上往下的第 p_i 张牌。

CZL 觉得一次洗牌不足以把牌洗乱，因此，他会以同样的规则洗 m 次牌。

假设从上往下第一张牌的编号是 1，第二张牌的编号是 2，...，第 i 张牌的编号是 i，请你求出来，洗完 m 次牌后，从上往下第 i 张牌的编号是多少 $(1 \le i \le n)$。

输入格式

第一行包含两个正整数 n, m，表示牌的个数，洗牌的次数。

第二行包含 n 个正整数 $p_1, p_2, ..., p_n$，保证 $p_1, p_2, ..., p_n$ 是 $1, 2, ...,$ 的一个全排列。

输出格式

输出一行 n 个整数，其中第 i 个正整数表示最终从上往下第 i 张牌的编号。

样例 1

landlord.in	landlord.out
5 1 2 3 4 5 1	5 1 2 3 4

样例 2

landlord.in	landlord.out
5 5 2 3 4 5 1	1 2 3 4 5

样例 3

见选手文件夹中的 landlord3.in/ans。

数据范围

对于所有的测试数据，满足 $1 \leq n \leq 100000, 1 \leq m \leq 10^9$。

子任务 1 (分值: 50)

保证 $n, m \leq 2000$。

子任务 2 (分值: 50)

无特殊限制。

Problem B. 无聊

输入文件:	boring.in
输出文件:	boring.out
时间限制:	1.5 秒
内存限制:	512 MB

在学农基地的寝室里，小 U 躺在自己的床上，无所事事。他盯着天花板，回想起自己跌宕起伏的 OM 和 IO 的竞赛生涯，顿时一阵怅然若失的情绪涌上心头。

为了之后更好地适应文化课的学习，小 U 必须阻止自己无止境的回忆。他要给自己找点事情做。

于是，他给自己编了很多小学生的计算题。这样的计算题由长度为 n 的字符串组成，并且这个字符串只有加法和数字。小 U 厌倦了数学里面那些繁杂的规则，所以他规定：任何一个只包含加号和数字的串，都有它唯一的运算结果。具体计算方法如下：

(1) 如果字符串末尾是一个加号，那么把加号去掉。

(2) 如果有连续的两个加号相邻，就去除其中的一个加号，直到没有两个加号相邻了为止。

(3) 如果存在一个加号后面跟着一个 '0'，那么去除这个加号后面的 '0'，直到所有的加号后面都跟着 1-9 为止。

(4) 如果最后该串是空串，那么结果是 0。否则使用我们熟知的顺序运算的规则做计算。

小 U 先给自己出了一道运算题，然后他不断地修改他的题目，不断地重新计算。修改方法是修改字符串上恰好一个位置的字符。奇怪的是，他不止关心整个字符串的运算结果，还关心如果我们把这个字符串（从 0 编号）的第 l 位到第 r 位提取出来，那么这个子串的运算结果是多少。

小 U 算着算着就睡着了，在睡梦中，他仍然想着那些算式的结果，可是却无法进行计算。于是，他把这些算式的结果交给了你来计算。

输入格式

第一行包含一个从 0 编号的只包含加号和数字的字符串 S, 表示一开始小 U 的算式。第二行包含一个正整数 q，表示修改和询问的个数。接下来 q 行，先输入一个字符串。保证这个字符串是 "change" 或 "query"。

如果该字符串是 "change"，则后面跟着一个非负整数 $p(0 \le p < |S|)$，表示修改的位置。再输入一个字符 $c(c \in \{'0','1','2','3','4','5','6','7','8','9','+'\}$，表示修改之后的字符。

如果该字符串是 "query"，则后面跟着两个非负整数 $l, r(0 \le l \le r < |S|$ 表示小 U 询问了这个算式的第 l 位到第 r 位的子串的结果。

输出格式

对于每个 "query" 询问，请你输出一行整数，表示这个结果模 998244353 的最小非负剩余。

样例 1

boring.in	boring.out
+++++++++ 12 query 0 8 change 0 9 change 2 8 change 4 4 change 6 3 change 8 3 query 0 8 change 1 9 change 3 2 change 5 4 change 7 5 query 0 8	0 27 0

样例解释 1

一开始算式是 $+++++++++$，结果为 0。之后算式变成了 $9+8+4+3+3$，结果为 27。最后算式是 998244353，在模 998244353 意义下为 0。

样例 2

见选手文件夹中的 boring2.in/ans。

数据范围

对于所有测试数据，满足 $|S| \leq 10^5, q \leq 10^5$。

子任务 1 (分值: 23)

$|S| \leq 10^3, q \leq 10^3$

子任务 2 (分值: 19)

保证在任何时候，该字符串没有加号，且 $l = 0, r = |S| - 1$。

子任务 3 (分值: 26)

保证没有 “change” 操作。

子任务 4 (分值: 17)

保证询问的 $l = 0, r = |S| - 1$。

子任务 5 (分值: 15)

无特殊限制。

Problem C. 地斗主

输入文件: landlords.in
输出文件: langlords.out
时间限制: 3 秒
内存限制: 512 MB

突然，一阵声音划破了小 U 寝室里原有的寂静和无聊，使小 U 的睡梦戛然而止。原来是 CZL 同学来小 U 的寝室表演扑克牌魔术。

他先让某个人把排列整齐的扑克牌任意打乱。具体打乱的方式是将扑克牌按某个位置从上往下分为两叠，再用随机的方式将两叠归并起来。然后他能够猜出某个位置的扑克牌的花色和大小。

小 U 不喜欢混乱，喜欢探寻混乱中那隐藏的规律。他关心按某种规则打乱过后，第 i 张牌的大小的数学期望是多少。小 U 苦死冥想，却无法得到答案。（当然他也不会那么无聊了）。

小 U 讨厌这无解的混乱。他正在脑子里设计一种将扑克牌按大小排序的方法。首先，他认为 n 张扑克牌的大小互不相同，且从上到下第 i 张扑克牌的大小是 p_i。小 U 的目标是让扑克牌位置越往上，大小越小。他可以取第 i 张扑克牌，然后把扑克牌拿出来，将它往上或往下移动，再把它插入到牌堆中。

在每次操作前，小 U 要找到第 i 张扑克牌的位置，这会使他排序的 “查找代价” 增加 A。

如果小 U 拿出的那张扑克牌越过了某张在牌堆中的扑克牌（即与那张扑克牌的位置关系发生改变），那么他排序的“移动代价”增加了 B。

帮小 U 得到这个数学期望，并不是你的任务。你的任务是，以最小的总代价（即“查找代价”加上“移动代价”）使小 U 给洗过的扑克牌排好序。当然，你需要构造一种方案来告诉小 U 最小的代价确实可以达到。

输入格式

第一行包含一个正整数和两个非负整数 n, A, B。

第二行包含 n 个正整数，第 i 个正整数表示从上往下第 i 张牌的大小 p_i。保证 $\{p_1, p_2, ..., p_n\} = \{1, 2, ..., n\}$。

输出格式

第一行一个非负整数，表示排序的最小代价。第二行一个非负整数 k，表示你排序使用的操作次数。接下来 k 行，每行一个正整数 p，一个字符 c，和一个正整数 d。p 表示移动的是第 p 张牌。$c \in 'U', 'D'$，若 $c =' U'$，则移动的方向向上，否则移动的方向向下。d 表示移动的这张扑克牌越过的其它扑克牌的个数。你需要保证若 $c =' U'$，则 $d \le p$，否则 $d \le n - p$。

样例 1

landlords.in	langlords.out
3 1 1 1 3 2	2 1 2 D 1

样例解释 1

将第 2 张牌向下移动一个位置，排列变为 $1, 2, 3$。且花费代价为 2。

样例 2

见选手文件夹中的 landlords2.in/ans。

评分标准

如果你试图绕过 SPJ 得分，你的得分将被手动修改为 0 分。

如果你的输出不符合题目中要求的输出格式导致 SPJ 无法正确读取，你将获得 0 分。

如果你输出的答案的第一行正确，那么你可以获得该测试点 40% 的分数。在第一行正确的前提下，如果你之后的构造正确，你将获得该测试点所有分数。

请注意，如果你只会第一问，或者排列 p 已经有序，第二行请务必输出 0，否则 SPJ 会认为你的输出格式错误。对于 $A = B = 0$ 的那些测试点，请你不要使你的 k 过大，这样会导致 SPJ TLE。

如何测试你的程序

由于某种神秘的原因，我们将不会下发 SPJ。如果你想测试你的程序，请你自行实现一个简易的 SPJ。

数据范围

对于所有测试数据，满足 $n \leq 10^5, A, B \leq 10^6$。

测试点编号	n	A	B	特殊性质 1	特殊性质 2
1	≤ 3	≤ 1000000	≤ 1000000	×	×
2	≤ 5	≤ 1000000	≤ 1000000	×	×
4	≤ 5	≤ 1000000	≤ 1000000	×	×
5	≤ 300	$= 0$	$= 0$	×	×
6	≤ 3000	$= 0$	$= 0$	×	×
7	≤ 100000	$= 0$	$= 0$	×	×
8	≤ 100000	$= 0$	≤ 1000000	×	×
9	≤ 100000	$= 0$	≤ 1000000	×	×
10	≤ 100000	≤ 1000000	$= 0$	×	×
11	≤ 100000	≤ 1000000	$= 0$	×	×
12	≤ 100000	≤ 1000000	$= 0$	✓	×
13	≤ 100000	≤ 1000000	≤ 1000000	✓	×
14	≤ 100000	≤ 1000000	≤ 1000000	×	✓
15	≤ 100000	≤ 1000000	≤ 1000000	×	✓
16	≤ 50	≤ 1000000	≤ 1000000	×	×
17	≤ 50	≤ 1000000	≤ 1000000	×	×
18	≤ 1000	≤ 1000000	≤ 1000000	×	×
19	≤ 1000	≤ 1000000	≤ 1000000	×	×
20	≤ 100000	≤ 1000000	≤ 1000000	×	×

特殊性质 1：只存在最多两个 i 使得 $p_i \neq i$。

特殊性质 2：存在一个正整数 i 使得 $p_1, p_2, \ldots, p_i$ 单调递增，$p_i, \ldots, p_n$ 单调递减。