第34届全国青少年信息学奥林匹克冬令营

CCF NOI 冬令营

竞赛时间：2017年2月8日 8:30 ~ 13:30

题目名称	棋盘	挑战	排序
题目类型	传统型	传统型	提交答案型
目录	chess	challenge	sort
可执行文件名	chess	challenge	N/A
输入文件名	chess.in	challenge.in	sort*.in
输出文件名	chess.out	challenge.out	sort*.out
每个测试点时限	3秒	3秒	N/A
内存限制	1 GB	2 GB	N/A
测试点数目	20	9	8
每个测试点分值	5	N/A	N/A

提交源程序文件名

对于C++ 语言	chess.cpp	challenge.cpp	N/A
对于C 语言	chess.c	challenge.c	N/A
对于Pascal 语言	chess.pas	challenge.pas	N/A

编译选项

对于C++ 语言	-O2 -lm	-O2 -lm	N/A
对于C 语言	-O2 -lm	-O2 -lm	N/A
对于Pascal 语言	-O2	-O2	N/A

棋盘（chess）

【问题描述】

小Z在一块棋盘上玩游戏。这块棋盘是一个由 n 个顶点和 m 条边组成的无向连通图，图上的顶点编号为 $[1,n]$ 中的正整数。游戏开始时，每个顶点上都放有一个棋子，每个棋子上有一个 $[0,n-1]$ 中的整数，表示棋子的编号。不同棋子的编号互不相同。

每次操作时，小Z需要先选择棋盘上的一条边，这条边的一个端点此时放有 0 号棋子。然后，小Z将这条边两个端点上的棋子交换。

现在你已经知道棋盘的模样，以及初始状态下每个顶点上的棋子编号。小Z想请你回答 q 个询问。每个询问指定了棋盘的目标状态，你需要回答能否通过若干次操作，将棋盘由初始状态变为这个目标状态。

【输入格式】

从文件***chess.in*** 中读入数据。

第一行输入三个正整数 n,m,q ，分别表示图中的顶点数、边数，以及询问的个数。

接下来 m 行，每行两个整数 $u,v(1 \le u,v \le n)$ ，表示图中有一条连接 u,v 两顶点的无向边。保证 $u \ne v$ ，即图中不存在自环。保证给出的图是连通图，即任意两个顶点都可以经过若干条边而相互到达。

接下来一行有 n 个空格隔开的整数，其中第 i 个整数表示初始状态下 i 号顶点上的棋子的编号，保证编号都为 $[0,n-1]$ 中的整数，且两两不同。

接下来 q 行，每行有 n 个空格隔开的整数，表示一个询问，其中第 i 个整数表示目标状态中第 i 号顶点上的棋子的编号，保证编号都为 $[0,n-1]$ 中的整数，且两两不同。

【输出格式】

输出到文件***chess.out*** 中。

输出 q 行，每行一个字符串“Yes”或者“No”(不含引号)作为回答。“Yes”表示从棋子的初始状态可以经过若干次操作而达到询问中所指定的目标状态，“No”表示不存在这样的操作步骤。注意：“Yes”和“No”的首字母都是大写的。

【样例1输入】

```
5 6 3
2 1
4 5
3 5
```

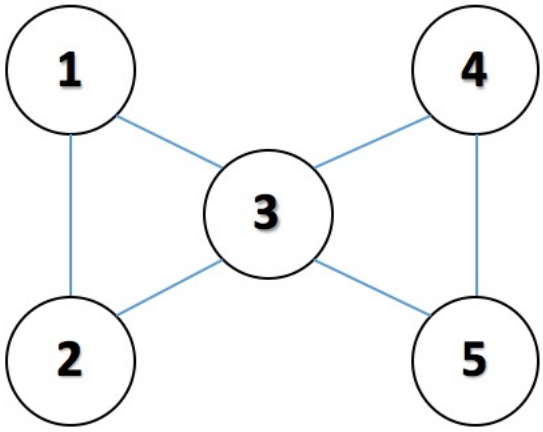
```
3 4
2 3
1 3
1 2 3 4 0
0 1 2 3 4
2 1 0 4 3
4 3 0 1 2
```

【样例1输出】

```
Yes
Yes
No
```

【样例1解释】

棋盘如下图所示：



对于第一组询问，只要将 0 号点沿着 5 – 4 – 3 – 2 – 1 的路径移动。对于第二组询问，将 0 号点沿着 5 – 3 – 1 – 2 – 3 移动。对于第三组询问是无解的。

【样例2】

见选手目录下的***chess/chess2.in*** 与***chess/chess2.ans***。

【数据范围和约定】

本题共有20个测试点。对于每个测试点，只有当你的输出与标准输出完全相同时，才能得到该测试点的满分。下表为各个测试点的数据范围和约定。

测试点编号	n	m	数据特性
1	≤ 8	≤ 15	无
2			
3	≤ 50	$= n - 1$	
4			
5		$= n$	特性1
6			
7			无
8			
9		≤ 100	特性2
10			
11			特性3
12			
13			
14			无
15			
16			
17			
18			
19			
20			

$1 \leq q \leq 1000$

特性1：所有顶点的度数均为 2 。

特性2：这个棋盘可以绘制在一个 $k \times k$ 的矩形网格图上。其中 $k \times k = n$ 且 $2k \times (k-1) = m$ 。

特性3：每条边至多属于一个简单环。

挑战（challenge）

【问题描述】

你和同学们找了三道题目用来练习。

这次练习的目标是写出能在时间限制里通过尽量大规模数据的代码。

同学们纷纷写出了优秀的代码。现在，他们向你发起了挑战，他们对每个问题都设置了若干个测试数据，这是他们能通过的最大规模的测试数据。现在，他们想看一看你写的代码究竟能超过多少同学的代码，通过多大规模的测试数据。

本题分为3个任务，每个任务对应一道题和相应的若干个测试点，你需要对于每个任务，设计一个能通过尽量多测试点的程序。

任务1

给定 n 个32 位无符号整数，将它们从小到大排序。

任务2

有 $2n$ 个人在玩“石头剪刀布”游戏。他们排成两排，每排 n 个人。每个人在每一局游戏都使用固定策略，即对于第 $i(i \in \{1,2\})$ 排的第 $j(0 \le j < n)$ 个人，用一个整数 a_{ij} 表示他的策略，其中0 表示只出石头，1 表示只出剪刀，2 表示只出布。

现在有 q 个询问，每个询问给定三个整数 $x, y, l(0 \le x, y < n, 1 \le l \le n - \max(x, y))$，问将第一排的第 $x \sim x + l - 1$ 个人和第二排的第 $y \sim y + l - 1$ 个人比赛之后，第一排有多少个人会赢。

上文中“比赛”的意思是，对于所有整数 i 满足 $0 \le i < l$，让第一排的第 $x + i$ 个人和第二排的第 $y + i$ 个人进行“石头剪刀布”游戏。

任务3

我们称一个合法的括号串为：只由左括号和右括号构成，两种括号的数量相等，且任意一个前缀的左括号数量不少于右括号数量的串。现在给定一个由“(”、“)”和“?”构成的串，问有多少种不同的方案，使得将每个“?”都替换成一个括号之后，该串变成一个合法的括号串。两种方案不同，当且仅当至少有一个位置的“?”被替换成了不同的括号。

由于本题部分任务数据规模比较大，我们用一些特殊的方式进行输入输出。先定义一些将要用于输入/输出的函数（以下代码中除法向下取整，数组下标从0开始，异或为按位异或，右移为逻辑右移）。

```
定义函数 next_integer(32位无符号整数 x):
    将 x 赋值为 x 异或 (x 左移 13位)
```

```
    将 x 赋值为 x 异或 (x 右移 17位)
    将 x 赋值为 x 异或 (x 左移 5位)
    返回 x

定义函数 output_arr(指针 a, 32位无符号整数 size):
    如果 (size 模 4) 不等于 0:
        输出 -1
        返回 假
    令 32位无符号整数 count 等于 size 除以 4
    令 32位无符号整数指针 b 等于 a 强制转换为 32位无符号整数指针
    令 32位无符号整数 ret 等于 size
    令 32位无符号整数 x 等于 23333333
    令 32位无符号整数 i 从 0 循环至 count - 1:
        将 ret 赋值为 ret 异或 (b[i] 加 x)
        将 x 赋值为 next_integer(x)
    输出 ret
    返回 真
```

【输入格式】

从文件***challenge.in*** 中读入数据。

第一行一个整数 $task\_id(1 \le task\_id \le 3)$，表示任务编号。接下来是每个具体任务的输入内容。

在输入的同一行中，相邻的两个整数会被一个空格隔开。

对于任务1： 一行，两个整数 n, s。令 $a_0 = \text{next\_integer(s)}, a_i = \text{next\_integer}(a_{i-1}), 1 \le i < n$，则 $a_0, a_1, \cdots, a_{n-1}$ 即为需要排序的 n 个整数。

对于任务2： 第一行两个整数 n, q。第二行一个仅包含“0”, “1”, “2” 的长度为 n 的字符串，第 i 个字符所代表的整数表示第一排第 i 个人的策略（即 a_{1i} ）。第三行格式同第二行，表示第二排各个人的策略。

对于任务3： 第一行一个整数 n，表示给定的串的长度。第二行一个字符串，即为给定的串。

【输出格式】

输出到文件***challenge.out*** 中。

对于任务1： 令 $b_0, b_1, \cdots, b_{n-1}$ 为排好序的数组，调用output_arr(b, n * 4) 即可。

对于任务2： 将每个询问的答案依次存入**32位无符号整数**数组 b 中（即，存入 $b_0, b_1, \cdots, b_{q-1}$ 中），然后调用output_arr(b, q * 4) 即可。

对于任务3： 输出一个整数，表示不同的方案数除以 2^{32} 得到的**余数**。

【数据范围和约定】

本题共有9 个测试点，不同测试点的分值可能不同。对于每个测试点，只有当你的输出与标准输出完全相同时，才能得到该测试点的满分。下表为各个测试点的数据范围和约定。为了方便阅读，“测试点编号”一列被放到了表格的中间而不是左边。

任务编号	分值	测试点编号	数据范围和约定
1	5	1	$n = 100000$
	19	2	$n = 10^8$
	11	3	$n = 2 \times 10^8$
2	7	4	$n = q = 1000$
	23	5	$n = q = 300000$
3	9	6	$n = 1000$
	5	7	$n = 120000$
	7	8	$n = 225000$
	14	9	$n = 266666$

【提示和说明】

对于所有语言都打开-O2 编译开关。

所有测试点的时间限制均为3s 。所有测试点的空间限制均为2GB （对于32 位计算机，该限制即为单个程序能使用的内存的上界）。

评测在本机进行。请注意，所有机器的配置与型号都是相同的。

对于每种编程语言，我们都提供了一个**只实现了输入输出功能的模板程序**，名为challenge_model.pas/c/cpp 。你可以在这个模板程序的基础上开始解题，也可以不使用这个模板程序，这与你的得分无关。

我们提供了一些**样例输入输出文件**，名为challenge1.in/ans ~ challenge9.in/ans ，其中第 i 组样例输入的数据范围与第 i 个测试点的数据范围相同。各测试点的数据范围详见【数据范围与约定】。

请注意，根据NOI 系列比赛的有关规定，你提交的代码长度**不能超过100KB**。

【任务详细说明】

我们在这里对一些任务中的一些可能引起疑问的地方进行了较为详细的说明。

对于任务1：

32位无符号整数即占用内存空间32个二进制位（或4字节）的无符号整数类型，在C/C++中为`unsigned int`，在Pascal中为`dword`。

从小到大排序是指，将 n 个整数 $a_0, a_1, \cdots, a_{n-1}$ **重新排列**，设排列后这些整数为 $b_0, b_1, \cdots, b_{n-1}$，则满足对于任意 $1 \le i < n$，都有 $b_{i-1} \le b_i$。

对于任务2：

“石头剪刀布”游戏的规则是石头赢剪刀，剪刀赢布，布赢石头，除这三种情况以外都不算赢。

输入数据和询问中，每一排人的下标都是从0 开始的，即“第 $x(0 \le x < n)$ 个人”表示的是这一排从左到右数的第 $x+1$ 个人。

对于任务3：

我们给出一些括号串的例子："(())()"是合法的，"((((()))))"也是合法的；"(((("是不合法的，"())("也是不合法的。

对于输入数据"(???"，答案为2，有"(())"和"()()"两种方案。对于输入数据")???"，答案为0，因为不可能有合法的括号串以右括号开头。

排序（sort）

【问题描述】

牛牛最近学习了排序的相关知识，他对排序算法的运行效率产生了浓厚的兴趣并对此展开了一系列的研究。为了优化排序的运行时间，牛牛找到了来参加冬令营的你，希望你帮助他设计一台计算机进行排序。

这台计算机的设计方法如下：

- 需要进行排序的序列存储在大小为 n 的数组 Q 中。
- 进行计算的最小单元是**比较器**，一个比较器可以用一个三元组 (i, j, t) 表示，其中，i, j, t 均为**正整数**，其中 i，j 均在 $[1, n]$ 内且 $i < j$。它的功能是在时刻 t 比较 Q_i 与 Q_j 的大小，若 $Q_i > Q_j$，就交换 Q_i 与 Q_j 中的值，否则什么也不做。
- 若要让计算机正常运行，则需要任意两个比较器之间不能发生冲突。两个比较器 (A, B, T_1) 与 (C, D, T_2) 会发生冲突**当且仅当** A、B、C、D 中有两个相等的数且 $T_1 = T_2$。例如：比较器 $(1, 3, 1)$ 和 $(2, 4, 1)$ 不会发生冲突，而比较器 $(1, 2, 3)$ 和比较器 $(2, 3, 3)$ 会发生冲突。

在运行时，这台计算机中的每个比较器都会按照预先设定好的参数在指定的时间进行相应的操作。这台计算机运行时需要消耗的时间为所有的比较器中参数 t 的最大值 M，该值越小意味着运行时间越短。

牛牛发现，现实中的许多数据具有自己的特性。在设计计算机时往往需要考虑到这些特性，并针对性地进行设计才能达到好的效果。

为了检验你设计的计算机，牛牛提供了 8 个**测试点**共 100 组**测试数据**，每个测试数据包含 K 组测试数据，即 K 个长度为 n 的排列 P。请你为每组**测试数据**设计一台运行时间不超过 m 的计算机，使得对于任意 $i \in [1, K]$，这台计算机能对 P_i 排序。

【输入格式】

输入文件***sort1.in~sort8.in*** 已在选手目录下。

每个输入文件第一行一个整数 $T(1 \le T \le 100)$ 表示这个测试点中的测试数据组数，同时也表示这个测试点的分值。在所有数据中，$\sum T = 100$。

每组数据第一行三个整数 $K, n, m(1 \le n \le 10^5, 1 \le K, m \le 10^4)$，意义在上文中已经说明。

接下来 K 行每行 n 个整数，描述一个 1 到 n 的排列。

数据保证有解。

【输出格式】

针对给定的8个输入文件***sort1.in~sort8.in***，你需要分别提交你的输出文件***sort1.out~sort8.out***。

对于每组测试数据，第一行输出一个整数 $num(1 \le num \le 10^6)$ 表示你设计的计算机中比较器个数。

接下来 num 行，每行 3 个整数 $u, v, t(1 \le u < v \le n, 1 \le t \le 150)$，描述你的计算机的一个比较器 (u, v, t)。

【输入样例】

```
1
3 5 4
1 2 5 3 4
3 4 5 1 2
3 1 4 2 5
```

【输出样例】

```
7
1 2 1
3 4 1
2 3 2
4 5 2
1 2 3
3 4 3
2 3 4
```

【评分方式】

每个测试点单独评分，同时每个测试点你还有可能获得部分分。

每一个测试点总分值为 T 分，每个测试点得分为其中所有测试数据的得分之和。

在每组测试数据中，若 $M \le m$， $num \le 10^6$，比较器之间不会发生冲突且可以对这 K 个排列正确地排序，则该测试点得 1 分，否则得 0 分。

如果你输出了不足 T 个计算机，那么我们将默认你输出的第 i 个计算机对应第 i 组测试数据。

如果你的输出不符合格式要求，我们不保证你能得到该测试点的分数。

【如何测试你的输出】

在终端中先切换到该试题的目录下。

```
cd sort
```

我们提供***checker*** 这个工具来测试你的输出文件是否是可接受的。使用这个工具的方法是，在终端中运行

./checker <input> <output>

其中<input> 是给出的输入文件，<output> 是你的输出文件，例如

./checker sort1.in sort1.out

将测试sort1.out 在以sort1.in 为输入时是否可以接受。

在你调用这个程序后，checker 将根据你给出的输出文件给出每一个测试数据的测试结果，其中包括（如果你输出的计算机同时出现了多种错误，将会返回其中一种）：

1. 非法退出：未知错误。
2. 输入文件错误：输入文件非法，在不修改输入文件的情况下不会触发。
3. The number of the comparators is invalid!：输入的比较器个数不在 $[1, 10^6]$ 范围内，这时checker 将会直接退出。
4. Unexpected EOF：输出文件中给出的计算机不完整。
5. The running time of the comparator should be in [1, 150]：你给出的比较器的运行时间不在 $[1, 150]$ 范围内。
6. Invalid sorting network!：排序网络不合法，包括 $u \geq v$， $u < 0$，比较器间产生冲突等。
7. Invalid! m=a but M=b：计算机的运行时间超过限制。
8. The answer is incorrect：排序网络合法，但是并没有将输入的排列正确排序。
9. Correct! m=a and M=b：排序网络合法且对输入的排列正确排序，此时将得到该组测试数据的分数。
10. Total points: a：如果checker 正常运行到了最后，将会额外输出一行表示你的总得分。其中 a 是你在这组数据中得到的分数。

【部分数据特性】

这里给出**部分**数据的**部分**特性：

1. 测试点 4 中，对于输入的每一个排列 P，它每一个循环的大小都互不相同。循环可以理解为，将一个长度为 n 的排列 P 看成一张 n 个点 n 条边的无向图，其中 i 和 P_i 之间有一条边，这张图中的每一个联通块都是一个循环。
2. 测试点 7 中，对于输入的每一个排列 P，都存在若干个互不相交的区间 $[l_i, r_i]$，使得在对每一个区间 $[l_i, r_i]$ 循环左移一次后，数组有序。循环左移的例子为：$(1, 2, 3, 4)$ 循环左移一步为 $(2, 3, 4, 1)$。同时，该测试点的前 8 组数据，满足对于输入的每一个排列 P，都存在整数 i，使得对区间 $[1, i]$ 循环左移一次后，数组有序。

【提示】

请妥善保存输入文件 **sort*.in**和你的输出文件 **sort*.out**，及时备份，以免误删。
通过自行修改输入文件而获得的得分是无效的。