全国青少年信息学奥林匹克竞赛

CCF NOI 2024

第一试

时间：2024 年 7 月 18 日 08:00 ～ 13:00

题目名称	集合	百万富翁	树的定向
题目类型	传统型	交互型	传统型
目录	`set`	`richest`	`tree`
可执行文件名	`set`	`richest`	`tree`
输入文件名	`set.in`	`richest.in`	`tree.in`
输出文件名	`set.out`	`richest.out`	`tree.out`
每个测试点时限	1.0 秒	6.0 秒	3.0 秒
内存限制	512 MiB	512 MiB	2048 MiB
测试点数目	20	2	25
测试点是否等分	是	否	是
预测试点数目	20	2	25

提交源程序文件名

对于 C++ 语言	`set.cpp`	`richest.cpp`	`tree.cpp`

编译选项

对于 C++ 语言	`-O2 -std=c++14 -static`

注意事项（请仔细阅读）

1. 文件名（程序名和输入输出文件名）必须使用英文小写。赛后正式测试时将以选手留在题目目录下的源代码为准。
2. `main` 函数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. 因违反以上两点而出现的错误或问题，申诉时一律不予受理。
4. 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）。
5. 选手提交的程序源文件必须不大于 100 KB。
6. 程序可使用的栈空间内存限制与题目的内存限制一致。
7. 禁止在源代码中改变编译器参数（如使用 `#pragma` 命令），禁止使用系统结构相关指令（如内联汇编）和其他可能造成不公平的方法。
8. 选手可使用快捷启动页面中的工具 selfEval 进行自测。在将待测程序（不必是全部题目）放到题目目录下后，即可选择全部或部分题目进行自测。注意：自测有次数限制，且自测结果仅用于选手调试，并不做为最终正式成绩。

集合（set）

【题目描述】

小 Y 和小 S 在玩一个游戏。

给定正整数 m，定义**基本集合**为大小为 3、元素在 $1 \sim m$ 内的集合。例如，给定 $m = 4$，则集合 $\{1, 2, 3\}$ 与集合 $\{2, 3, 4\}$ 都是基本集合。

定义**集合序列**为由基本集合构成的序列。例如，$A = [\{1, 2, 3\}, \{2, 3, 4\}]$ 是一个集合序列，其中 $A[1] = \{1, 2, 3\}, A[2] = \{2, 3, 4\}$ 都是基本集合。

对于一个 $1 \sim m$ 的排列 $p[1], p[2], \ldots, p[m]$ 与集合 $S \subseteq \{1, 2, \ldots, m\}$，定义 $f_p(S)$ 为将 S 内每一个元素 x 置换为 $p[x]$ 后的所得到的集合，即 $f_p(S) = \{p[x] | x \in S\}$。

对于两个长度为 k 的集合序列 A, B，定义 A 和 B **等价**当且仅当存在一个 $1 \sim m$ 的排列 p，使得 A 置换排列 p 后得到 B，即对于所有 $1 \leq i \leq k$，$f_p(A[i]) = B[i]$。

给定两个长度为 n 的集合序列 A, B。有 q 次询问：每次小 S 会询问小 Y，在给定 l, r 的情况下，判断集合序列 $[A[l], A[l+1], \ldots, A[r]]$ 与集合序列 $[B[l], B[l+1], \ldots, B[r]]$ 是否等价？

时光荏苒，小 S 和小 Y 也会散去。而我们和一个人保持连接的方式就是记住，仅此而已。

【输入格式】

从文件 ***set.in*** 中读入数据。

输入的第一行包含三个正整数 n, m, q，分别表示集合序列的长度、元素范围和询问次数。

输入的第二行包含 $3n$ 个正整数。第 $3i-2, 3i-1, 3i$（$1 \leq i \leq n$）个正整数分别表示 $A[i]$ 的三个元素。保证这三个元素均在 $[1, m]$ 范围内且互不相同。

输入的第三行包含 $3n$ 个正整数。第 $3i-2, 3i-1, 3i$（$1 \leq i \leq n$）个正整数分别表示 $B[i]$ 的三个元素。保证这三个元素均在 $[1, m]$ 范围内且互不相同。

接下来 q 行，每行包含两个正整数 l, r，表示一次询问。

【输出格式】

输出到文件 ***set.out*** 中。

输出 q 行，每行包含一个字符串 `Yes` 或 `No`，表示对应询问的两个序列是否等价。

【样例 1 输入】

```
4 4 10
1 2 3 1 2 3 1 2 4 1 2 3
```

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

【样例 1 输出】

```
Yes
No
No
No
Yes
Yes
Yes
Yes
Yes
Yes
```

【样例 1 解释】

以下用 (l,r) 表示对 l,r 的询问。

- 对于询问 $(1,1)$，令排列 $p=[1,2,4,3]$，则 $f_p(A_1)=\{p[1],p[2],p[3]\}=\{1,2,4\}=B[1]$，因此该询问对应的两个序列等价。
- 对于询问 $(1,2),(1,3),(1,4)$，由于 $A[1]=A[2]$ 但 $B_1\neq B_2$，因此这些询问对应的两个序列都不等价。
- 对于询问 $(2,2),(2,3),(2,4),(3,3),(3,4),(4,4)$，令排列 $p=[2,3,4,1]$，则 $f_p(A_2)=\{p[1],p[2],p[3]\}=\{2,3,4\}=B_2$，$f_p(A_3)=\{p[1],p[2],p[4]\}=\{1,2,3\}=B_3$，$f_p(A_4)=\{p[1],p[2],p[3]\}=\{2,3,4\}=B_4$，因此这些询问对应的两个序列都等价。

【样例 2】

见选手目录下的 ***set/set2.in*** 与 ***set/set2.ans***。
这个样例满足测试点 1 ～ 3 的约束条件。

【样例 3】

见选手目录下的 ***set/set3.in*** 与 ***set/set3.ans***。
这个样例满足测试点 8 的约束条件。

【样例 4】

见选手目录下的 ***set/set4.in*** 与 ***set/set4.ans***。
这个样例满足测试点 15, 16 的约束条件。

【数据范围】

对于所有测试数据保证：$1 \leq n \leq 2 \times 10^5$，$3 \leq m \leq 6 \times 10^5$，$1 \leq q \leq 10^6$，$1 \leq l \leq r \leq n$。

测试点编号	$n \leq$	$m \leq$	$q \leq$
$1 \sim 3$	50	4	50
$4 \sim 6$	50	5	50
7	200	4	200
8	200	5	200
9	200	4	2×10^5
10	200	5	2×10^5
11	2×10^5	4	2×10^5
12	2×10^5	5	2×10^5
13, 14	2,000	6,000	10^3
15, 16	2,000	6,000	10^6
17, 18	2×10^4	6×10^4	10^2
19, 20	2×10^5	6×10^5	10^6

百万富翁（richest）

这是一道交互题。

【题目描述】

小 Y 的银行有 N 个客户，编号为 0 到 $N-1$。客户 i 有 W_i 元存款，且**客户之间的存款金额互不相同**。

小 P 是小 Y 的深度合作伙伴，他希望知道哪个客户的存款最多。小 P 无法直接获取客户的存款金额，但他可以依次发送若干次**请求**，每次**请求**包含若干个**查询**，每个**查询**是一个二元组 (i, j)，表示小 P 想知道客户 i 和客户 j 的存款金额哪个更多。如果 $W_i > W_j$，小 Y 会回答 i，否则回答 j。

小 P 的**请求**数 t 和所有请求的**查询**次数总和 s 有上限，他希望你帮他写一个程序来找到存款最多的客户。

【实现细节】

选手不需要，也不应该实现 `main` 函数。

选手应确保提交的程序包含头文件 `richest.h`，可在程序开头加入以下代码实现：

```
#include "richest.h"
```

选手需要实现以下函数：

```
int richest(int N, int T, int S);
```

- N 表示客户的数量；
- T 表示对于当前函数调用，**请求**数 t 不应超过此值；
- S 表示对于当前函数调用，所有请求的**查询**次数总和 s 不应超过此值；
- 该函数需要返回存款最多的客户的编号；
- 对于每个测试点，该函数**会被交互库调用恰好 10 次**。

选手可以通过调用以下函数向交互库发送一次**请求**：

```
std::vector<int> ask(std::vector<int> a, std::vector<int> b);
```

- 在调用 `ask` 函数时需要保证传入参数 a 和 b 的长度相同，且其中的每个元素都必须是小于 N 的非负整数，表示该**请求**中的所有**查询**；
- 该函数会返回一个类型为 `std::vector<int>` 且长度与 a 和 b 相同的变量，设为 c。其中 $c[i]$ 表示在客户 $a[i]$ 和 $b[i]$ 中存款金额较多的客户的编号。

题目保证在规定的**请求**与**查询**次数限制下，交互库运行所需的时间不超过 3 秒；交互库使用的内存大小固定，且不超过 256 MiB。

【测试程序方式】

试题目录下的 grader.cpp 是提供的交互库参考实现，最终测试时所用的交互库实现与该参考实现有所不同，因此选手的解法不应该依赖交互库实现。

选手可以在本题目录下使用如下命令编译得到可执行程序：

```
g++ grader.cpp richest.cpp -o richest -O2 -std=c++14 -static
```

对于编译得到的可执行程序：

- 可执行文件将从**标准输入**读入以下格式的数据：
  - 输入的第一行包含四个非负整数 N, T, S, R，其中 R 是交互库生成测试数据的随机种子。
- 输入完成后，交互库将调用 10 次函数 `richest`，用输入的参数生成的测试数据进行测试。`richest` 函数返回后，交互库会输出以下信息：
  - 输出的前 10 行中，每行首先包含三个整数 r, t, s，表示该次执行的结果，其中 r 是 `richest` 函数的返回值，t 和 s 的含义如题目描述中所示，然后包含该次运行的正确性等信息。
  - 输出的第 11 行包含 10 次运行的总信息。

【交互示例】

假设可执行文件生成的测试数据为 $N = 4$，$W = [101, 103, 102, 100]$，$T = 100$，$S = 100$。

下面是一个正确的交互过程：

选手程序	交互库	说明
	调用 richest(4, 100, 100)	开始测试
调用 ask([0, 2], [1, 3])	返回 [1, 2]	$W_0 < W_1$，$W_2 > W_3$
调用 ask([0, 2, 3], [1, 1, 1])	返回 [1, 1, 1]	$W_0 < W_1$，$W_2 < W_1$，$W_3 < W_1$
运行结束并返回 1	向屏幕打印交互结果	交互结束，结果正确

在这个例子中，$r = 1$，$t = 2$，$s = 5$，满足请求与查询次数的限制。

【下发文件说明】

在本试题目录下：

1. `grader.cpp` 是提供的交互库参考实现。
2. `richest.h` 是头文件，选手不用关心具体内容。
3. `template_richest.cpp` 是提供的示例代码，选手可在此代码的基础上实现。

选手注意对所有下发文件做好备份。最终评测时只测试本试题目录下的 **richest.cpp**，对该程序以外文件的修改不会影响评测结果。

【数据范围】

对于所有测试数据保证：所有 W_i 两两不同。

本题共 2 个测试点，每个测试点的分值和数据范围见下表。

测试点编号	分值	$N=$	$T=$	$S=$
1	15	1 000	1	499 500
2	85	1 000 000	20	2 000 000

【评分方式】

注意：

- 选手不应当通过非法方式获取交互库的内部信息，如试图直接读取数组 W 的值，或直接与标准输入、输出流进行交互。此类行为将被视为作弊；
- **最终的评测交互库与样例交互库的实现不同，且可能是适应性的：在不与 ask 此前返回的结果相矛盾的前提下，最终的评测交互库可能会动态调整 W 的值。**

本题首先会受到和传统题相同的限制，例如编译错误会导致整道题目得 0 分，运行时错误、超过时间限制、超过空间限制等会导致相应测试点得 0 分等。选手只能在程序中访问自己定义的和交互库给出的变量及其对应的内存空间，尝试访问其他空间将可能导致编译错误或运行错误。

在每次 `richest` 函数调用中，选手程序使用的请求次数 t 和所有请求的查询次数总和 s 需在对应限制下，否则将会获得 0 分。

在上述条件基础上：

- 在测试点 1 中，程序得到满分当且仅当 `ask` 函数调用合法且 `richest` 函数返回的答案正确；
- 在测试点 2 中，程序得到的分数将按照以下方式计算：
  - 若 `ask` 函数调用不合法，则获得 0 分；
  - 若 `ask` 函数调用均合法，设 $\max t$ 表示多次调用 `richest` 函数时所得的 t 的最大值，$\max s$ 表示 s 的最大值，则程序将获得 $\lfloor 85 \cdot f(\max t) \cdot g(\max s) \rfloor$ 分，其中 f 与 g 的计算方式如下表所示：

$\max t$	$f(\max t)$
$\max t \le 8$	1
$9 \le \max t \le 20$	$1 - \frac{1}{4}\sqrt{\max t - 8}$

$\max s$	$g(\max s)$
$\max s \leq 1\,099\,944$	1
$1\,099\,945 \leq \max s \leq 1\,100\,043$	$1 - \frac{1}{6}\log_{10}(\max s - 1\,099\,943)$
$1\,100\,044 \leq \max s \leq 2\,000\,000$	$\frac{2}{3} - \frac{1}{1\,500}\sqrt{\max s - 1\,100\,043}$

以下是测试点 2 中，不同的 t 和 s 对得分影响的示例：

$\max t$	$\max s$	测试点 2 的得分
$= 20$	$\leq 1\,099\,944$	11
$= 19$	$\leq 1\,099\,944$	14
$= 18$	$\leq 1\,099\,944$	17
$= 17$	$\leq 1\,099\,944$	21
$= 16$	$\leq 1\,099\,944$	24
$= 15$	$\leq 1\,099\,944$	28
$= 14$	$\leq 1\,099\,944$	32
$= 13$	$\leq 1\,099\,944$	37
$= 12$	$\leq 1\,099\,944$	42
$= 11$	$\leq 1\,099\,944$	48
$= 10$	$\leq 1\,099\,944$	54
$= 9$	$\leq 1\,099\,944$	63
≤ 8	$\in [1\,099\,974, 1\,099\,978]$	63
≤ 8	$\in [1\,099\,969, 1\,099\,973]$	64
≤ 8	$\in [1\,099\,965, 1\,099\,968]$	65
≤ 8	$\in [1\,099\,962, 1\,099\,964]$	66
≤ 8	$\in [1\,099\,959, 1\,099\,961]$	67
≤ 8	$\in [1\,099\,957, 1\,099\,958]$	68
≤ 8	$\in [1\,099\,955, 1\,099\,956]$	69
≤ 8	$\in [1\,099\,953, 1\,099\,954]$	70
≤ 8	$= 1\,099\,952$	71
≤ 8	$= 1\,099\,951$	72
≤ 8	$\in [1\,099\,949, 1\,099\,950]$	73
≤ 8	$= 1\,099\,948$	75
≤ 8	$= 1\,099\,947$	76
≤ 8	$= 1\,099\,946$	78
≤ 8	$= 1\,099\,945$	80
≤ 8	$\leq 1\,099\,944$	85

树的定向（tree）

【题目描述】

给定一棵含有 n 个顶点的树，顶点从 1 到 n 编号。树上第 i（$1 \le i \le n-1$）条边连接顶点 u_i 和 v_i。

现在，我们想要给树的每条边一个定向。任何一个定向都可以用一个长度为 $n-1$ 的字符串 $S = s_1 s_2 \cdots s_{n-1}$ 来描述。其中，$s_i = 0$ 代表第 i 条边定向为 $u_i \to v_i$，否则 $s_i = 1$ 代表第 i 条边定向为 $v_i \to u_i$。

给定 m 个顶点对 (a_i, b_i)，其中 $1 \le a_i, b_i \le n$ 且 $a_i \ne b_i$。

一个**完美定向**定义为：在此定向下，对于任意 $1 \le i \le m$，a_i **不能到达** b_i。

试求在所有完美定向中，所对应的字符串字典序最小的定向。**数据保证存在至少一个完美定向。**

定义字符串 $S = s_1 \cdots s_{n-1}$ 的字典序小于 $T = t_1 \cdots t_{n-1}$：若存在一个下标 k，使得 $s_1 = t_1, \cdots, s_{k-1} = t_{k-1}, s_k < t_k$。

【输入格式】

从文件 ***tree.in*** 中读入数据。

输入的第一行包含三个非负整数 c, n, m，分别表示测试点编号，树的点数，顶点对的个数。$c = 0$ 表示该测试点为样例。

接下来 $n-1$ 行，每行包含两个正整数 u_i, v_i 表示树的一条边。保证 $1 \le u_i, v_i \le n$ 且这 $n-1$ 条边构成了一棵树。

接下来 m 行，每行包含两个正整数 a_i, b_i。保证 $1 \le a_i, b_i \le n$ 且 $a_i \ne b_i$。

【输出格式】

输出到文件 ***tree.out*** 中。

输出一行包含一个字符串 $S = s_1 \cdots s_{n-1}$，表示字典序最小的完美定向所对应的 01 字符串。

【样例 1 输入】

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

【样例 1 输出】

```
001
```

【样例 1 解释】

在该样例中，若 $S = 000$，则该定向中 1 能到达 4 (存在路径 $1 \to 2 \to 3 \to 4$)，因而不是完美定向。若 $S = 001$，则该定向中 3 不能到达 2，1 不能到达 4，因而是完美定向。故答案为 001。

【样例 2 输入】

```
0 6 8
5 1
2 3
1 2
5 6
4 3
4 3
5 1
6 3
5 4
1 4
5 2
3 6
6 2
```

【样例 2 输出】

```
10101
```

【样例 2 解释】

在该样例中，一组完美定向必定满足 4 不能到达 3，5 不能到达 1，故 $s_1 = s_5 = 1$。若 $s_2 = s_3 = 0$，则存在路径 $1 \to 2 \to 3 \to 4$，故 1 可到达 4，故其不是完美定向。因此，所有完美定向必定满足 S 的字典序不小于 10101。且容易验证 $S = 10101$ 时，对应的定向是完美定向。

【样例 3】

见选手目录下的 ***tree/tree3.in*** 与 ***tree/tree3.ans***。
这个样例满足测试点 $1 \sim 3$ 的约束条件。

【样例 4】

见选手目录下的 ***tree/tree4.in*** 与 ***tree/tree4.ans***。
这个样例满足测试点 $4 \sim 6$ 的约束条件。

【样例 5】

见选手目录下的 ***tree/tree5.in*** 与 ***tree/tree5.ans***。
这个样例满足测试点 $7, 8$ 的约束条件。

【样例 6】

见选手目录下的 ***tree/tree6.in*** 与 ***tree/tree6.ans***。
这个样例满足测试点 $9, 10$ 的约束条件。

【数据范围】

对于所有测试数据保证：$2 \leq n \leq 5 \times 10^5$，$1 \leq m \leq 5 \times 10^5$，$1 \leq u_i, v_i \leq n$ 且所有的边构成了一棵树，$1 \leq a_i, b_i \leq n$ 且 $a_i \neq b_i$。
数据保证存在至少一个完美定向。

测试点编号	n	m	特殊性质
$1 \sim 3$	≤ 15	≤ 50	无
$4 \sim 6$	≤ 300	≤ 300	无
$7, 8$	≤ 400	$= (n-1)(n-2)$	A
$9, 10$	$\leq 2,000$	$\leq 2,000$	B
$11 \sim 14$	$\leq 2,000$	$\leq 2,000$	无
$15, 16$	$\leq 10^5$	$\leq 10^5$	B
$17, 18$	$\leq 10^5$	$\leq 10^5$	无
$19 \sim 21$	$\leq 2 \times 10^5$	$\leq 2 \times 10^5$	无
$22 \sim 25$	$\leq 5 \times 10^5$	$\leq 5 \times 10^5$	无

特殊性质 A: 保证 (a, b) 出现在 (a_i, b_i) 中当且仅当 $a \neq b$ 且 a, b 在树上不相邻。
特殊性质 B: 保证树上编号为 1 的顶点与其他每个顶点均相邻。