

全国青少年信息学奥林匹克竞赛

CCF NOI 2019

第二试

时间：2019 年 7 月 18 日 08:00 ~ 13:00

题目名称	弹跳	斗地主	I 君的探险
题目类型	传统型	传统型	交互型
目录	jump	landlords	explore
可执行文件名	jump	landlords	explore
输入文件名	jump.in	landlords.in	explore.in
输出文件名	jump.out	landlords.out	explore.out
每个测试点时限	2.0 秒	4.0 秒	2.0 秒
内存限制	128 MB	512 MB	512 MB
子任务数目	25	10	25
测试点是否等分	是	是	是

提交源程序文件名

对于 C++ 语言	jump.cpp	landlords.cpp	explore.cpp
对于 C 语言	jump.c	landlords.c	explore.c
对于 Pascal 语言	jump.pas	landlords.pas	explore.pas

编译选项

对于 C++ 语言	-O2 -lm
对于 C 语言	-O2 -lm
对于 Pascal 语言	-O2

注意事项

1. 选手提交的源文件必须存放在已建立好的带有下发样例的文件夹中（该文件夹与试题同名）。
2. 文件名（包括程序名和输入输出文件名）必须使用英文小写。
3. 结果比较方式为忽略行末空格、文末回车后的全文比较。
4. C/C++ 中函数 main() 的返回值类型必须是 int，值为 0。
5. 对于因未遵守以上规则对成绩造成的影响，相关申诉不予受理。

弹跳 (jump)

【题目描述】

跳蚤国有 n 座城市，分别编号为 $1 \sim n$ ，1 号城市为首都。所有城市分布在一个 $w \times h$ 范围的网格上。每座城市都有一个整数坐标 (x, y) ($1 \leq x \leq w, 1 \leq y \leq h$)，不同城市的坐标不相同。

在跳蚤国中共有 m 个弹跳装置，分别编号为 $1 \sim m$ ，其中 i 号弹跳装置位于 p_i 号城市，并具有参数 t_i, L_i, R_i, D_i, U_i 。利用该弹跳装置，跳蚤可花费 t_i ($t_i > 0$) 个单位时间，从 p_i 号城市跳至坐标满足 $L_i \leq x \leq R_i, D_i \leq y \leq U_i$ ($1 \leq L_i \leq R_i \leq w, 1 \leq D_i \leq U_i \leq h$) 的任意一座城市。需要注意的是，一座城市中可能存在多个弹跳装置，也可能没有弹跳装置。

由于城市间距离较远，跳蚤们必须依靠弹跳装置出行。具体来说，一次出行将经过若干座城市，依次经过的城市的编号可用序列 a_0, a_1, \dots, a_k 表示；在此次出行中，依次利用的弹跳装置的编号可用序列 b_1, b_2, \dots, b_k 表示。其中每座城市可在序列 $\{a_j\}$ 中出现任意次，每个弹跳装置也可在序列 $\{b_j\}$ 中出现任意次，且满足，对于每个 j ($1 \leq j \leq k$)，编号为 b_j 的弹跳装置位于城市 a_{j-1} ，且跳蚤能通过该弹跳装置跳至城市 a_j 。我们称这是一次从城市 a_0 到城市 a_k 的出行，其进行了 k 次弹跳，共花费 $\sum_{i=1}^k t_{b_i}$ 个单位时间。

现在跳蚤国王想知道，对于跳蚤国除首都（1 号城市）外的每座城市，从首都出发，到达该城市最少需要花费的单位时间。跳蚤国王保证，对每座城市，均存在从首都到它的出行方案。

【输入格式】

从文件 `jump.in` 中读入数据。

第一行包含四个整数 n, m, w, h ，变量的具体意义见题目描述。

接下来 n 行，第 i 行包含两个整数 x_i, y_i ，表示 i 号城市的坐标。

接下来 m 行，第 i 行包含六个整数 $p_i, t_i, L_i, R_i, D_i, U_i$ ，分别表示 i 号弹跳装置所在的城市编号、弹跳所需的时间、可到达的矩形范围。这些整数的具体意义见题目描述。

【输出格式】

输出到文件 `jump.out` 中。

输出 $n-1$ 行，第 i 行包含一个整数，表示从跳蚤国首都到 $i+1$ 号城市最少需要花费的单位时间。

【样例 1 输入】

```
5 3 5 5
1 1
3 1
4 1
2 2
3 3
1 123 1 5 1 5
1 50 1 5 1 1
3 10 2 2 2 2
```

【样例 1 输出】

```
50
50
60
123
```

【样例 2】

见选手目录下的 *jump/jump2.in* 与 *jump/jump2.ans*。

这组样例的数据范围为 $n = 10000$, $m = 20000$, $w = 10000$, $h = 1$ 。

【样例 3】

见选手目录下的 *jump/jump3.in* 与 *jump/jump3.ans*。

这组样例的数据范围为 $n = 10000$, $m = 20000$, $w = 10000$, $h = 10000$ 。

【数据范围与提示】

对于所有测试点和样例满足：

$1 \leq n \leq 70000$, $1 \leq m \leq 150000$, $1 \leq w, h \leq n$, $1 \leq t_i \leq 10000$ 。

每个测试点的具体限制见下表。

测试点编号	$1 \leq n \leq$	$1 \leq m \leq$	特殊限制
1 ~ 8	100	100	无
9 ~ 13	50000	100000	每个弹跳装置恰好可达一座城市, 且 $L_i = R_i, D_i = U_i$
14 ~ 18	50000	100000	$h = 1$
19 ~ 22	25000	50000	无
23 ~ 25	70000	150000	无

请注意, 本题的内存限制为 128MB。

斗地主 (landlords)

【题目描述】

小 S 在和 小 F 玩一个叫“斗地主”的游戏。

可怜的小 S 发现自己打牌并打不过小 F，所以他想要在洗牌环节动动手脚。

一副牌一共有 n 张牌，从上到下依次标号为 $1 \sim n$ 。标号为 i 的牌分数是 $f(i)$ 。在本题， $f(i)$ 有且仅有两种可能： $f(i) = i$ 或 $f(i) = i^2$ 。

洗牌的方式和我们日常生活中的比较类似，以下我们用形式化的语言来定义：

洗牌环节一共分 m 轮，这 m 轮洗牌依次进行。第 i 轮洗牌时：

1. 小 S 会拿出从最上面往下数的前 A_i 张牌。这样这副牌就被分成了两堆：第一堆是最上面的 A_i 张牌，第二堆是剩下的 $n - A_i$ 张牌，且这两堆牌内相对顺序不变。特别地，当 $A_i = n$ 或 $A_i = 0$ 时，有一堆牌是空的。
2. 接下来对两堆牌进行合并，从而产生新的第三堆牌。当第一堆牌还剩下 X 张，第二堆牌还剩下 Y 张的时候，以 $\frac{X}{X+Y}$ 的概率取出第一堆牌的最下面的牌，并将它放入新的第三堆牌的最上面，以 $\frac{Y}{X+Y}$ 的概率取出第二堆牌的最下面的牌，并将它放入新的第三堆牌的最上面。
3. 重复操作 2，一直取到两堆牌都为空为止。这样我们就完成了一轮洗牌。

因为洗牌过程是随机的，所以小 S 发现自己没法知道某个位置上具体是哪张牌。但小 S 想问你在经历了这 m 轮洗牌后，某个位置上的牌的期望分数是多少。小 S 一共会问你 Q 个这样的问题。

【输入格式】

从文件 `landlords.in` 中读入数据。

输入的第一行包含三个正整数 $n, m, type$ ，分别表示牌的数量，洗牌的轮数与 $f(i)$ 的类型。当 $type = 1$ 时， $f(i) = i$ 。当 $type = 2$ 时， $f(i) = i^2$ 。

接下来一行，一共 m 个整数，表示 $A_1 \sim A_m$ 。

接下来一行一个正整数 Q ，表示小 S 的询问个数。

接下来 Q 行，每行一个正整数 c_i ，表示小 S 想要知道从上往下第 c_i 个位置上的牌的期望分数。保证 $1 \leq c_i \leq n$ 。

【输出格式】

输出到文件 `landlords.out` 中。

输出一共 Q 行，每行一个整数，表示答案在模 998244353 意义下的取值。

即设答案化为最简分式后的形式为 $\frac{a}{b}$ ，其中 a 和 b 互质。输出整数 x 使得 $bx \equiv a \pmod{998244353}$ 且 $0 \leq x < 998244353$ 。可以证明这样的整数 x 是唯一的。

【样例 1 输入】

4 1 1
3
1
1

【样例 1 输出】

249561090

【样例 1 解释】

有 $\frac{1}{4}$ 的概率从上到下的最终结果是 {1, 2, 3, 4}。

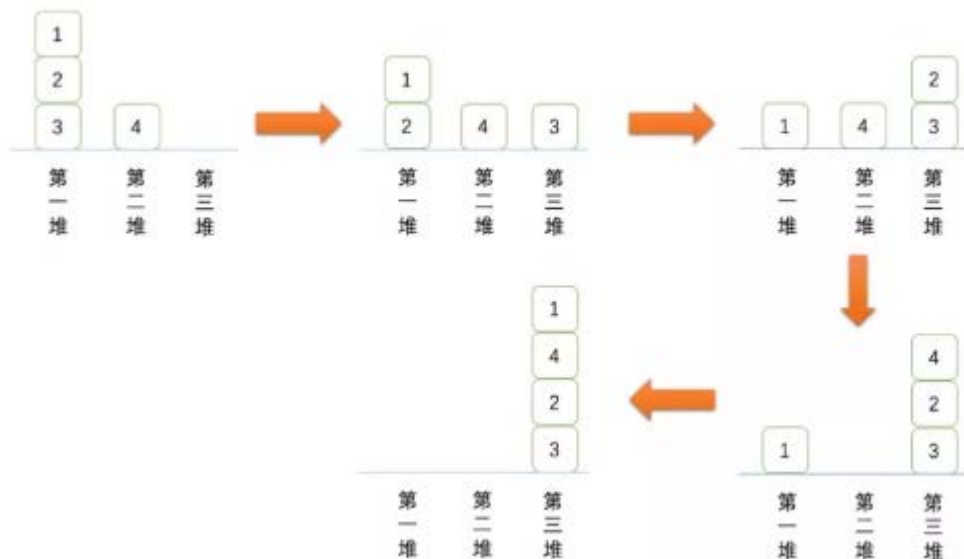
有 $\frac{1}{4}$ 的概率从上到下的最终结果是 {1, 2, 4, 3}。

有 $\frac{1}{4}$ 的概率从上到下的最终结果是 {1, 4, 2, 3}。

有 $\frac{1}{4}$ 的概率从上到下的最终结果是 {4, 1, 2, 3}。

所以最终有 $\frac{1}{4}$ 的概率第一个位置是 4，有 $\frac{3}{4}$ 的概率第一个位置是 1，所以第一个位置的期望分数是 $\frac{7}{4}$ 。

为了帮助你们更直观地了解洗牌的过程，我们在下面画出了结果是 {1, 4, 2, 3} 的过程。



【样例 2】

见选手目录下的 *landlords/landlords2.in* 与 *landlords/landlords2.ans*。

【样例 3】

见选手目录下的 *landlords/landlords3.in* 与 *landlords/landlords3.ans*。

【数据范围与提示】

对于所有的测试点： $3 \leq n \leq 10^7$ ， $1 \leq m, Q \leq 5 \times 10^5$ ， $0 \leq A_i \leq n$ ， $type \in \{1, 2\}$ 。
每个测试点的具体限制见下表：

测试点	n	m	$type =$	其他性质
1	≤ 10	≤ 1	1	无
2	≤ 80	≤ 80		
3			2	
4	≤ 100	$\leq 5 \times 10^5$		1
5	$\leq 10^7$		2	
6				1
7			2	
8				1
9			2	
10				

请注意我们并没有保证 $Q \leq n$ 。

这里我们给出离散型随机变量 X 的期望 $\mathbb{E}[x]$ 的定义：

设离散随机变量 X 的可能值是 X_1, X_2, \dots, X_k ， $\Pr[X_1], \Pr[X_2], \dots, \Pr[X_k]$ 为 X 取对应值的概率。则 X 的期望为

$$\mathbb{E}[x] = \sum_{i=1}^k X_i \Pr[X_i]$$

I 君的探险 (explore)

这是一道交互题。

【题目背景】

时隔半年，I 君的商店终于开不下去了，他决定转让商店，做一名探险家去探索未知的广阔世界。

根据古书记载，他在一个大荒漠的腹地找到了未知文明创造的地下宫殿，宫殿由 N 个大型洞穴和 M 条连接这些洞穴的双向通路构成。I 君能借助古书分辨所处的洞穴，但书中并没有记录 M 条通路的连接结构，因此他难以搜寻传说中藏在宫殿里的无尽财宝。

不过现在 I 君发现了一个神秘机关，通过它可以获知宫殿的信息，I 君决定利用这个机关来得到宫殿的连接结构，请你来协助他。

【题目描述】

地下宫殿可以抽象成一张 N 个点、 M 条边的无向简单图（简单图满足任意两点之间至多存在一条直接相连的边），洞穴从 $0 \sim n-1$ 编号。目前你并不知道边有哪些。

每个洞穴都拥有一个光源，光源有开启、关闭两种状态，只有当光源处于开启状态时它所在的洞穴才会被照亮。初始时所有的光源都处于关闭状态，而光源的状态只能用 I 君发现的神秘机关改变。更具体的，使用神秘机关可以进行如下四种操作：

1. 向机关给定一个编号 x ，机关将会改变 x 号洞穴，以及与 x 号洞穴有通路直接相连的洞穴的光源状态。即原来开启的光源将会关闭；原来关闭的光源将会开启。
2. 向机关给定一个编号 x ，机关将会显示当前 x 号洞穴光源的状态。
3. 向机关给定两个编号 x, y ，表示你确定有一条连接 x 号洞穴与 y 号洞穴的通路，并让机关记录。
4. 向机关给定一个编号 x ，机关将会判断与 x 号洞穴相连的通路是否都已被记录。

机关在完成上一次操作后才能进行下一次操作。机关不能随意使用，因此每种操作的使用次数都有限制，分别为 L_m, L_q, M, L_c 。你的任务是，编写一个程序，帮助 I 君决定如何合理利用神秘机关，从而正确地找到这 M 条通路。

【实现细节】

你不需要，也不应该实现主函数，你只需要实现函数 `explore(N, M)`，这里的 N 和 M 分别表示洞穴和通路的个数。你可以通过调用如下四个函数来和交互库进行交互：

1. `modify(x)`

- 这个函数可以令机关执行操作 1，给定的编号为 x 。
- 你需要保证 $0 \leq x < N$ ，这个函数没有返回值。

2. `query(x)`

- 这个函数可以令机关执行操作 2，给定的编号为 x 。
- 你需要保证 $0 \leq x < N$ ，这个函数返回 0 或 1，表示目前 x 号洞穴的光源为关闭（0 表示）或开启（1 表示）状态。

3. report(x, y)

- 这个函数可以令机关执行操作 3，给定的编号为 x, y 。
- 你需要保证 $0 \leq x, y < N$ 且 $x \neq y$ ，这个函数没有返回值。

4. check(x)

- 这个函数可以令机关执行操作 4，给定的编号为 x 。
- 你需要保证 $0 \leq x < N$ ，这个函数返回 0 或 1，其中返回 1 当且仅当与 x 号洞穴相连的所有通路都已通过操作 3 被记录。

评测时，交互库会恰好调用 explore 一次。

本题保证所使用的图在交互开始之前已经完全确定，不会根据你的程序的交互过程动态构造，因此题目中的交互操作都是确定性的，你不需要关心这些操作在交互库中的具体实现。

数据保证在调用次数限制下，交互库运行所需的时间不超过 1s；交互库使用的内存大小固定，且不超过 128MB。

【实现方法】

选手工作目录下已经提供了一个 template_explore.cpp/c/pas，请将这个文件拷贝一份，重命名为 explore.cpp/c/pas，然后在其基础上答题。

1. 对 C++ / C 语言选手

- 请确保你的程序开头有 #include "explore.h"。
- 你需要实现的函数 explore 的接口信息如下：

```
void explore(int N, int M);
```

- 你可以调用的交互函数的接口如下：

```
void modify(int x);  
int query(int x);  
void report(int x, int y);  
int check(int x);
```

2. 对 Pascal 语言选手

- 注意：Pascal 的代码中实现接口的语法较为复杂，请选手直接在下发的 template_explore.pas 的基础上进行答题，而不是自己从头实现代码。
- 你需要实现的函数 explore 的接口信息如下：

```
procedure explore(N, M : longint);
```

- 注意：这里的函数名称是 _explore 而非 explore，如果使用 explore 将导致编译失败。

- 你可以调用的交互函数的接口如下：

```
procedure modify(x : longint);  
function query(x : longint) : longint;  
procedure report(x : longint; y : longint);  
function check(x : longint) : longint;
```

【测试程序方式】

试题目录下的 `grader.cpp/c` 以及 `graderhelperlib.pas` 是我们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。

1. 对 C/C++ 语言的选手：

- 你需要在本题目录下使用如下命令编译得到可执行程序：
- 对于 C 语言：`gcc grader.c explore.c -o explore -O2 -lm`
- 对于 C++ 语言：`g++ grader.cpp explore.cpp -o explore -O2 -lm`

2. 对于 Pascal 语言的选手：

- 你需要在本题目录下使用如下命令编译得到可执行程序：

```
fpc grader.pas -o"explore" -O2
```

3. 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：
第一行包含三个整数 L_m, L_q, L_c ，第二行包含两个整数 N, M ，意义如题面描述。
接下来 M 行，每行两个整数 x, y ，描述一条连接 x 号洞穴与 y 号洞穴的通路。
- 读入完成之后，交互库将调用恰好一次函数 `explore`，用输入的数据测试你的函数。你的函数正确返回后，交互库会判断你的计算是否正确，若正确则会输出 `Correct` 和交互函数调用次数相关信息，否则会输出相应的错误信息。

【交互示例】

假设可执行文件读入的数据为：

```
100 200 300  
3 2  
0 1  
1 2
```

数据第一行的三个整数分别表示三种操作的调用次数限制，即 `modify(x)` 调用次数不能超过 100，`query(x)` 调用次数不能超过 200，`check(x)` 调用次数不能超过 300。数据第二行的两个整数分别表示洞穴数和通路条数，即 $N = 3, M = 2$ 。

`report(x, y)` 调用次数不能超过 M ，该例子中即不超过 2 次。

下面是一个正确的交互过程：

选手程序	交互库	说明
	调用 <code>explore(3, 2)</code>	开始测试
调用 <code>modify(0)</code>		对 0 号洞穴做操作 1
调用 <code>query(2)</code>	返回 0	目前 2 号洞穴的光源状态是关闭
调用 <code>report(0, 1)</code>		发现了通路 (0, 1) 并记录
调用 <code>check(0)</code>	返回 1	与 0 号洞穴相关的通路都已被记录
调用 <code>report(2, 1)</code>		发现了通路 (2, 1) 并记录
运行结束并返回	向屏幕打印 <code>Correct</code>	交互结束，结果正确

【下发文件说明】

在本试题目录下：

1. `grader.cpp/c` 以及 `graderhelperlib.pas` 是我们提供的交互库参考实现。
2. `explore.h` 和 `grader.pas` 是头文件，选手不用关心具体内容。
3. `template_explore.cpp/c/pas` 是我们提供的样例解题源代码。
4. `explore1.in`、`explore2.in`、`explore3.in` 是样例输入，可供测试。

选手注意对所有下发文件做好备份。评测只收取本试题目录下的 `explore.c/cpp/pas`，并且对该程序以外的文件的修改无效。

【评分方式】

最终评测只会收取 `explore.cpp/c/pas`，修改选手目录下其他文件对评测无效。

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。你只能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。

在上述条件基础上，在一个测试点中，你得到满分，当且仅当：

1. 你的每次函数调用均合法，且调用 `modify`、`query` 和 `check` 的次数分别不超过 L_m, L_q, L_c 。
2. 由于 `report` 的调用次数限制为 M ，你的每次调用都必须记录一条新的且存在的边：即每次调用 `report(x, y)` 时，应满足：有一条连接 x 号洞穴和 y 号洞穴的通路，且在这次调用之前从未调用过 `report(x, y)` 或 `report(y, x)`。
3. 你实现的函数 `explore` 正常返回。
4. 在 `explore` 函数返回时，你已经通过调用 `report` 记录了全部 M 条通路。

【数据范围与提示】

本题共 25 个测试点，每个测试点 4 分。每个测试点的数据规模和相关限制见下表。

测试点编号	$N =$	$M =$	$L_m =$	$L_q =$	$L_c =$	特殊性质
1	3	2	100	100	100	无
2	100	10N	200	10^4	2M	
3	200			4×10^4		
4	300		299	9×10^4		
5	500		499	1.5×10^5		
6	59998	N/2	17N	17N	0	A
7	99998		18N	18N		
8	199998		19N	19N		
9						
10	99997		18N	18N		B
11	199997		19N	19N		
12	99996	N - 1	10^7	10^7	2M	C
13	199996					
14						
15	99995					D
16						
17	199995					
18	1004	2000	10^7	5×10^4	2M	无
19		3000				
20						
21	50000	2N	10^7			
22	100000					
23	150000	200000				
24	200000	250000				
25		300000				

再次提醒，题目保证测试所使用的图在交互开始之前已经完全确定，而不会根据你的程序的交互动态构造。

表中特殊性质栏中变量的含义如下：

A：保证每个点的度数恰好为 1。

B：保证对于每个 $x > 0$ ，存在恰好一个 $y < x$ 的 y 使得 x 号洞穴与 y 号洞穴有通路直接相连。

C: 存在 $0 \sim N - 1$ 的一个排列 p_0, p_1, \dots, p_{N-1} , 使得对任意 $1 \leq i < N$, 存在一条连接洞穴编号分别为 p_{i-1} 与 p_i 的通路。

D: 保证图连通。

- 提示: 你的程序可以通过判断传入的 N 的个位来区分上述不同的数据类型。