

NOIP2024 模拟赛 1

第一试

时间：2024 年 10 月 1 日 08:30 ~ 13:00

题目名称	硬币	狼群	比赛	半完全幂
题目类型	传统型	传统型	传统型	传统型
目录	coin	wolf	match	power
可执行文件名	coin	wolf	match	power
输入文件名	coin.in	wolf.in	match.in	power.in
输出文件名	coin.out	wolf.out	match.out	power.out
每个测试点时限	1.0 秒	1.0 秒	2.0 秒	1.0 秒
内存限制	256 MiB	256 MiB	512 MiB	512 MiB
测试点数目	10	10	25	10
测试点是否等分	是	是	是	是

提交源程序文件名

对于 C++ 语言	coin.cpp	wolf.cpp	match.cpp	power.cpp
-----------	----------	----------	-----------	-----------

编译选项

对于 C++ 语言	-O2 -std=c++14
-----------	----------------

硬币 (coin)

【题目描述】

在 Floating Island 的货币系统中，有各种类型的硬币：

- 面值是不同的正整数。有一个面值为 1 的硬币类型。
- 对于每一对不同的硬币类型，其中**一个硬币类型的面值能整除另一个硬币类型的面值**。

不同面值的硬币看起来完全相同，唯一的区别是它们有不同的颜色。Floating Island 中的每枚硬币都只有一种颜色：

- 同一类型的硬币都是同一种颜色。
- 不同类型的硬币都有不同的颜色。

你知道 Floating Island 中使用的所有硬币面值，但你不知道它们的颜色。你甚至不知道硬币上使用的颜色集合。

对于每种面值，你想知道该面值的硬币的颜色。为了实现这一点，你有一张信用卡，可以从一个可以提供无限金额的 ATM 机上进行查询。每次查询你讲提供一个正整数 X ，ATM 机将以**最少的硬币**来支付你所请求的金额。不难发现最少硬币的支付方案是唯一的。

请求出你确定每种面值对应的颜色所需的最少查询次数。

【输入格式】

从文件 *coin.in* 中读入数据。

第一行一个正整数 n ，表示硬币面值的数量。

第二行 n 个递增的正整数 $a_1, \dots, a_n$ ，表示硬币的面值。

【输出格式】

输出到文件 *coin.out* 中。

输出一行一个整数，表示最少查询次数。

【样例 1 输入】

1 1

2 1

【样例 1 输出】

1 1

【样例 1 解释】

只有一种硬币面值，因此查询一次即可。

【样例 2 输入】

```
1 2
2 1 3
```

【样例 2 输出】

```
1 1
```

【样例 2 解释】

查询一次 $X = 5$ ，数量为 2 的颜色对应的面值为 1，数量为 1 的颜色对应的面值为 3。

【样例 3 输入】

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

【样例 3 输出】

```
1 2
```

【样例 4 输入】

```
1 5
2 1 2 4 8 16
```

【样例 4 输出】

```
1 3
```

【样例 5】

见选手目录下的 *coin/coin5.in* 与 *coin/coin5.ans*。

【子任务】

保证对于所有测试点满足： $1 \leq n \leq 60$ ， $1 = a_1 < \dots < a_n \leq 10^{18}$ 。

测试点编号	$n \leq$
1, 2	2
3, 4	3
5 ~ 7	15
8 ~ 10	60

狼群 (wolf)

【题目描述】

Floating Island 中存在一个狼群。狼群中的 n 只狼被编号为 $1 \sim n$ 。

对于第 i 只狼，它在初始时位于网格的顶点 (x_i, y_i) 处。在每一轮移动中，它都会选择移动到相邻的 4 个网格顶点之一，即 $(x - 1, y)$ 、 $(x + 1, y)$ 、 $(x, y - 1)$ 和 $(x, y + 1)$ 之一。

在 m 轮移动结束后，狼群中的 n 只狼都移动到了同一个网格顶点。求满足条件的狼的移动路径方案数，对 $10^9 + 7$ 取模。

两种移动路径方案被认为不同，当且仅当存在某一轮移动中，某一只狼的移动方向不同。

【输入格式】

从文件 *wolf.in* 中读入数据。

第一行两个正整数 n 和 m 。

接下来的 n 行，每行两个整数 x_i, y_i ，表示第 i 只狼的初始位置。

【输出格式】

输出到文件 *wolf.out* 中。

一行一个整数，表示答案。

【样例 1 输入】

```
1 2 1
2 3 0
3 5 0
```

【样例 1 输出】

```
1 1
```

【样例 1 解释】

两只狼必须在 $(4, 0)$ 处相遇。

【样例 2 输入】

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

【样例 2 输出】

```
1 0
```

【样例 2 解释】

两只狼无法在最后相遇。

【样例 3 输入】

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

【样例 3 输出】

```
1 4
```

【样例 3 解释】

两只狼最后必须在 $(2, 0)$ 相遇。第 1, 3 只狼只有 1 种方法到达 $(2, 0)$ ，第 2 只狼可以先往任意方向走一步，最后再走回到 $(2, 0)$ 。

【样例 4 输入】

```
1 2 1
2 7 8
3 8 7
```

【样例 4 输出】

12

【样例 4 解释】

有 (7, 7) 和 (8, 8) 两种可能的相遇地点。

【样例 5】

见选手目录下的 *wolf/wolf5.in* 与 *wolf/wolf5.ans*。

【子任务】

保证对于所有测试点满足： $1 \leq n \leq 50$ ， $1 \leq m \leq 10^5$ ， $|x_i|, |y_i| \leq 10^5$ 。

测试点编号	$n \leq$	$m \leq$	$ x_i , y_i \leq$
1	1	10^5	10^5
2, 3	50	10^2	10^2
4 ~ 6		8	10^5
7 ~ 10		10^5	

比赛 (match)

【题目描述】

Floating Island 正在举办一年一度的魔法比赛，共计有 n 个人参加，有三种不同的场地，每一名选手在不同的场地上有不同的排名。

作为比赛的裁判，小 I 每次可以任意选择两名没有被淘汰的选手进行比赛，并且指定一种比赛场地，然后在这种场地上较弱的选手会被淘汰。

由于魔法的世界在不断地变化，现在有若干的询问和修改的事件，每一次修改会交换两名选手在某一种场地上的排名。

每次询问，小 I 想知道 x 号选手有没有可能赢得比赛。请你帮助她回答这些问题。

【输入格式】

从文件 *match.in* 中读入数据。

第一行包含两个整数 n 和 q ，表示选手数量和询问以及修改的事件数量。

接下来三行，每行一个 1 到 n 的全排列，第 i 位表示排名为 i 的选手编号。从左往右，战斗力逐渐变弱。

接下来 q 行有以下两种类型：

- 1 x ：表示询问 x 能否赢得比赛。
- 2 $p\ a\ b$ ：表示交换了第 p 种场地中选手 a 和 b 的排名。

【输出格式】

输出到文件 *match.out* 中。

对于每个询问，输出一行 Yes 或 No。

【样例 1 输入】

```
1 4 4
2 1 2 3 4
3 2 1 3 4
4 2 4 3 1
5 1 1
6 1 4
7 2 3 1 4
8 1 4
```

【样例 1 输出】

```
1 Yes
2 Yes
3 No
```

【样例 2 输入】

```
1 6 7
2 4 6 1 5 3 2
3 5 1 4 2 6 3
4 4 6 1 5 2 3
5 1 2
6 2 2 4 5
7 1 1
8 2 2 4 5
9 2 2 5 6
10 1 2
11 1 1
```

【样例 2 输出】

```
1 Yes
2 No
3 No
4 Yes
```

【样例 3】

见选手目录下的 *match/match3.in* 与 *match/match3.ans*。

【样例 4】

见选手目录下的 *match/match4.in* 与 *match/match4.ans*。

【样例 5】

见选手目录下的 *match/match5.in* 与 *match/match5.ans*。

【样例 6】

见选手目录下的 *match/match6.in* 与 *match/match6.ans*。

【子任务】

保证对于所有测试点满足： $1 \leq n, q \leq 10^5$ ， $1 \leq p \leq 3$ ， $1 \leq a, b, x \leq n$ ， $a \neq b$ 。

测试点编号	$n \leq$	$q \leq$	特殊性质
1 ~ 3	15	10	无
4 ~ 7	10^3		
8 ~ 13	10^5		
14 ~ 18		10^5	不存在修改操作
19 ~ 25			无

半完全幂 (power)

【题目描述】

Floating Island 的魔法少女小 I 喜欢完全幂：
定义一个正整数 n 是完全幂，当且仅当存在正整数 $b > 1$ 和 $c > 1$ ，使得 $b^c = n$ 。
例如， $8 = 2^3$ 和 $243 = 3^5$ 是完全幂，而 1 和 54 则不是。
一天，小 I 发现完全幂的数量很少。
为了避免失望，她迅速发明了半完全幂：
定义一个正整数 n 是半完全幂，当且仅当存在正整数 $a \geq 1$ ， $b > 1$ 和 $c > 1$ ，使得 $a < b$ 且 $a \cdot (b^c) = n$ 。
例如， $243 = 1 \cdot 3^5$ 和 $54 = 2 \cdot 3^3$ 是半完全幂，而 1 和 24 则不是。
请注意，对于一些半完全数，可能存在多个对应的三元组 (a, b, c) 。
例如，432 可以表示为 $2 \cdot 6^3$ ，也可以表示为 $3 \cdot 12^2$ 。
给定正整数 L 和 R ，求区间 $[L, R]$ 中的半完全幂数量。

【输入格式】

从文件 `power.in` 中读入数据。
输入共两行，每行一个正整数，分别为 L 和 R 。

【输出格式】

输出到文件 `power.out` 中。
输出一行一个整数，表示答案。

【样例 1 输入】

```
1 1
2 10
```

【样例 1 输出】

```
1 3
```

【样例 1 解释】

注意 1 并不被认为是半完全幂。

【样例 2 输入】

```
1 18
2 58
```

【样例 2 输出】

```
1 9
```

【样例 2 解释】

18 和 58 之间共有 9 个半完全幂，分别为 18, 25, 27, 32, 36, 48, 49, 50, 54。

【样例 3 输入】

```
1 60
2 70
```

【样例 3 输出】

```
1 1
```

【样例 3 解释】

60 和 70 之间只有 64 一个半完全幂。

【样例 4】

见选手目录下的 *power/power4.in* 与 *power/power4.ans*。

【样例 5】

见选手目录下的 *power/power5.in* 与 *power/power5.ans*。

【子任务】

保证对于所有测试点满足： $1 \leq L \leq R \leq 8 \times 10^{16}$ 。

测试点编号	$L, R \leq$
1, 2	10^3
3, 4	10^6
5	10^8
6, 7	10^{12}
8 ~ 10	8×10^{16}