NOI 2025 模拟赛

四川省集

测试时间：2025.05.24

题目名称	自卑	dsa	game
题目类型	传统型	传统型	交互型
目录	inferiority	dsa	game
可执行文件名	inferiority	dsa	game
输入文件名	inferiority.in	dsa.in	game.in
输出文件名	inferiority.out	dsa.out	game.out
提交文件名	inferiority.cpp	dsa.cpp	game.cpp
时间限制	4.0 秒	2.0 秒	1.0 秒
内存限制	64 MiB	1024 MiB	512 MiB
子任务数目	4	20	5
测试点是否等分	否	是	否
编译选项	-O2 -std=c++14		

【注意事项（请仔细阅读）】

1. 选手提交的源程序请**直接放在个人目录下**，无需建立子文件夹；
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. C++ 中函数 main() 的返回值类型必须是 int，值必须为 0。
4. **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**。
5. 若无特殊说明，结果比较方式为**忽略行末空格、文末回车后的全文比较**。
6. 程序可使用的栈空间大小与该题内存空间限制一致。
7. 在终端中执行命令 `ulimit -s unlimited` 可将当前终端下的栈空间限制放大，但你使用的栈空间大小不应超过题目限制。
8. 若无特殊说明，每道题的**代码大小限制为 100KB**。
9. 若无特殊说明，输入与输出中同一行的相邻整数、字符串等均使用一个空格分隔。
10. 输入文件中可能存在行末空格，请选手使用更完善的读入方式（例如 scanf 函数）避免出错。
11. 直接复制 PDF 题面中的多行样例，数据将带有行号，建议选手直接使用对应目录下的样例文件进行测试。

12. 使用 std::deque 等 STL 容器时，请注意其内存空间消耗。

13. 请务必使用题面中规定的的编译参数，保证你的程序在本机能够通过编译。此外**不允许在程序中手动开启其他编译选项**，一经发现，本题成绩以 0 分处理。

自卑（inferiority）

【题目背景】

什么是自卑呢？

【题目描述】

有 n 个人排成一列，每个人有一个能力值 a_i 。

老师将会举行 m 次考试，第 i 次考试将只有 $[l_i, r_i]$ 内的人参加。

每次考试中，对于 $[l_i, r_i]$ 中的任两个人 x, y , 若 $a_x > a_y \times 2$, 则 x 会给 y 带来 $a_x \oplus a_y$ 的自卑值, 其中 $\oplus$ 表示异或。

该次考试产生的自卑值为其中所有人产生的所有自卑值之和。

求每次考试产生的自卑值。

【输入格式】

从文件 ***inferiority.in*** 中读入数据。

第一行两个正整数 n, m。

第二行 n 个整数，表示 a_i。

第三行到第 $m + 2$ 行, 每行两个整数 l_i, r_i。

【输出格式】

输出到文件 ***inferiority.out*** 中。

m 行, 每行一个整数表示答案。

【样例 0 输入】

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

【样例 0 输出】

```
24
12
0
8
16
0
24
```

【样例 1】

见选手目录下的 ***inferiority/sample1.in*** 与 ***inferiority/sample1.out***。
该样例满足子任务 1 的限制条件。

【样例 2】

见选手目录下的 ***inferiority/sample2.in*** 与 ***inferiority/sample2.out***。
该样例满足子任务 2 的限制条件。

【样例 3】

见选手目录下的 ***inferiority/sample3.in*** 与 ***inferiority/sample3.out***。
该样例满足子任务 3 的限制条件。

【样例 4】

见选手目录下的 ***inferiority/sample4.in*** 与 ***inferiority/sample4.out***。
该样例满足子任务 4 的限制条件。

【测试点约束】

对于 100% 的数据，满足 $n, m \leq 10^5, 1 \leq a_i \leq n$。

子任务编号	特殊限制	分值
1	$n, m \leq 300$	10
2	$n, m \leq 5000$	10
3	$n, m \leq 2 \times 10^4$	30
4	无	50

dsa（dsa）

【题目背景】

Hanghanghanghanghanghanghanghang...

【题目描述】

给你一个正整数序列 $a_1, a_2, , a_n$ 。设 $S = n + A_i$.

有 S 张卡片。每张卡片上都写有一个整数。卡片上的整数是 $a_1, a_2, \cdots a_n, -1, \cdots -1$ 。其中，有 $\sum a_i$ 张卡片上写着 -1 。

NIT 现在站在数线上的坐标 0 处。他将执行以下操作 S 次。

假设 x 是 NIT 的当前坐标。选择并丢弃他手中的一张牌。设 v 为弃牌上的数字，并跳转到坐标 $x + v$ 。如果他跳到了坐标 0 ，则获得一枚硬币。

对于每一个 $k = 1, 2, 3 \cdots , n$ ，求 NIT 的下棋顺序中使得他恰好赚到 k 枚硬币模数为 998244353 的个数。

您应该计算走棋顺序。也就是说，如果两张牌的数字相同，弃掉其中一张与弃掉另一张是没有区别的。

【输入格式】

从 ***dsa.in*** 中读入数据。

第一行一个正整数 n。

第二行 n 个正整数 $a_1, a_2, \cdots a_n$。

【输出格式】

输出到文件 ***dsa.out*** 中。

n 行，第 i 行输出 $k = i$ 的答案。

【样例 1 输入】

```
2
1 1
```

【样例 1 输出】

```
2
4
```

【样例 2 输入】

```
3
1 2 3
```

【样例 2 输出】

```
140
220
144
```

【样例 3】

见选手目录下的 ***dsa/dsa3.in*** 与 ***dsa/dsa3.ans***。
该样例满足测试点 5 ∼ 7 的限制条件。

【样例 4】

见选手目录下的 ***dsa/dsa4.in*** 与 ***dsa/dsa4.ans***。
该样例满足测试点 8 ∼ 10 的限制条件。

【样例 5】

见选手目录下的 ***dsa/dsa5.in*** 与 ***dsa/dsa5.ans***。
该样例满足测试点 11 ∼ 13 的限制条件。

【样例 6】

见选手目录下的 ***dsa/dsa6.in*** 与 ***dsa/dsa6.ans***。
该样例满足测试点 14 ∼ 15 的限制条件。

【样例 7】

见选手目录下的 ***dsa/dsa7.in*** 与 ***dsa/dsa7.ans***。
该样例满足测试点 16 ∼ 17 的限制条件。

【样例 8】

见选手目录下的 ***dsa/dsa8.in*** 与 ***dsa/dsa8.ans***。
该样例满足测试点 18 ∼ 20 的限制条件。

【测试点约束】

测试点编号	特殊限制
$1 \sim 2$	$n \leq 5, a_i \leq 2$
$3 \sim 4$	$n \leq 10, a_i \leq 5$
$5 \sim 7$	$n \leq 20$
$8 \sim 10$	$a_i \leq 1$
$11 \sim 13$	$a_i \leq 2$
$14 \sim 15$	$n \leq 50$
$16 \sim 17$	$n \leq 500$
$18 \sim 20$	$n \leq 5000$

对于所有数据 $1 \leq n \leq 5000, 1 \leq a_i \leq 5000$。

game（game）

【题目背景】

你会博弈论吗？

红磷和白磷很喜欢玩游戏，有向无环图上的公平博弈已经满足不了他们了，这次，他们遇到了有环的有向图。

他们发现，每轮会有多个棋子，每次他们可以选择一个棋子，一条出边移动（棋子间不会互相影响），移动不了的人就是输家；

显然的，红磷和白磷都足够聪明。

【题目描述】

这是一道交互题。

交互库会给出一个 n 个点 m 条边的有向无环图，你可以更改 k 次边（添加/删除），不需要保证你操作完成后的图还是一个有向无环图。

然后交互库会询问你 q 次，每次交互库选定一个点 x，你可以给出一个可重集 S，交互库会将 x 加入 S，然后在 S 中的点上放置棋子（个数是其在 S 中的出现次数），再让红磷和白磷开始一轮游戏，并告诉你游戏结果：先手胜利/后手胜利/平局（即会进行无限久）。

你需要回答交互库。

【实现细节】

你需要实现两个函数：

```
1 vector<pair<int, int>> init(vector<pair<int, int>> Edges, int n, int m,
     int k, int q, int qlim);
2 int ask();
```

init 函数传入初始的 DAG，点数，边数，修改边的次数限制，询问次数，每次询问的 $\sum |S|$ 的限制，你要返回你操作完后的图，不允许有重边。

ask 函数会在每次交互库询问你时调用，你要返回交互库选定的 x。

交互库会实现一个函数：

```
1 int play(vector<int> S);
```

你可以调用它，传入一个可重集 S，它会返回游戏结果：1(先手胜利)/0(平局)/-1(后手胜利)，请注意满足每次询问中，你调用的 play 函数的 $\sum |S| \leq qlim$

可以保证的是，每次 play 的时间复杂度为：$\mathcal{O}(n|S|\log n)$。

比赛时会下发头文件 game.h 和交互库 grader.o。

你需要在你的代码中引用 game.h。

你需要在编译时使用以下命令：

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

请在 Linux(虚拟机) 下使用，不保证能在 Windows 下使用。

【输入格式】

注意，这是交互库的输入，你不应该，也不能通过读入获取答案。

第一行三个非负整数 n, m, k。

接下来 m 行，每行两个非负整数 u, v，表示一条有向边，保证形成一个 'DAG'。

接下来一行两个非负整数 $q, qlim$。

接下来 q 行，每行一个非负整数，表示每次询问交互库选定的 x。

本题的点编号从 0 开始。

【输出格式】

注意，这是交互库的输出，你不应该，也不能通过输出 token 来通过此题目。

如果你没有违反限制：交互库会记录你的答案，并且输出如下：

```
总询问次数：{}，正确：{}，错误：{}，修改了 {} 条边，最大的 Σ|S| 是 {}，你的得分是：{}
```

如果你违反了限制：交互库会输出你违反的限制，如下：

```
不合法的使用：{}
```

【测试点约束】

子任务编号	$n \leq$	$m \leq$	$q \leq$	$qlim =$	$k =$	特殊性质	测试点数目
1	20	200	100	20	4251	/	2
2	200	19900	500	500	4251	A	4
3	500	0	1000	20	125000	/	2
4	1000	0	1000	20	4251	/	5
5	1000	19900	1000	20	4251	/	7

特殊性质 A：保证给出的图中，$\forall i \neq j$，i 和 j 间有一条边。

评分方式

评分机制如下：1. 正确率 pC：

- 如果 $pC < 96\%$，$A = 0$。
- 如果 $96\% \leq pC < 100\%$，$A = \frac{pC-0.96}{0.04} \times 0.7$。
- 如果 $pC = 100$，$A = 100$。

2. 修改边的限制 ku：

- 如果 $ku > 2k$，$B = 0$
- 如果 $k < ku \leq 2k$，$B = \frac{2k-ku}{k} \times 0.5 + 0.2$。
- 如果 $ku \leq k$，$B = 100$

3. 每次询问的 $\sum |S|$ 的限制 qu：

- 如果 $qu > 2qlim$，$C = 0$。
- 如果 $qlim < qu \leq 2qlim$，$C = \frac{2qlim-qu}{qlim} \times 0.5 + 0.2$。
- 如果 $qu \leq qlim$，$C = 100$。

每个测试点的得分率将是 $A \times B \times C$。

【提示】

你会写这道题的交互库吗？

本着对选手们的信任，本题交互库并未做防攻击机制，请不要对交互库进行攻击，否则会被置为 0 分