第 36 届全国青少年信息学奥林匹克冬令营

CCF NOI WC 2019

正式赛

时间：2019 年 1 月 29 日 08:00 ~ 13:00

题目名称	数树	远古计算机	I 君的商店
题目类型	传统型	提交答案型	交互型
目录	`tree`	`oldcomputer`	`shop`
可执行文件名	`tree`	N/A	`shop`
输入文件名	`tree.in`	`oldcomputer*.in`	N/A
输出文件名	`tree.out`	`oldcomputer*.out`	N/A
每个测试点时限	4.0 秒	N/A	1.0 秒
内存限制	512 MB	N/A	512 MB
测试点/包数目	25	5	6
测试点是否等分	否	否	否

提交源程序文件名

对于 C++ 语言	`tree.cpp`	N/A	`shop.cpp`
对于 C 语言	`tree.c`	N/A	`shop.c`
对于 Pascal 语言	`tree.pas`	N/A	`shop.pas`

编译选项

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

注意事项：

1、提交的源文件必须存放在已建立好的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。

数树（tree）

【题目背景】

白兔喜欢树。

白云喜欢数数。

有 n 只鼠，白兔用 $n-1$ 根蓝色绳子把它们连成了一棵树，每根蓝色绳子连着两只鼠，白云用 $n-1$ 根红色绳子把它们连成了一棵树，每根红色绳子连接着两只鼠。

白云要给予每只鼠一个数。这个数可以是 $[1,y]$ 中的任意一个整数。

白兔给了白云一个要求：对于两只鼠 p,q ，若存在一条连接这两只鼠的路径同时属于这两棵树，则 p 和 q 必须被给予相同的整数。存在一条路径同时属于这两棵树指的是：

- 存在一个序列 $(a_1=p,a_2,\ldots,a_m=q)$ ，使得：对于所有 $i\in[1,m-1]$ ，都有 a_i 和 a_{i+1} 既有一根红色绳子**直接**相连也有一根蓝色绳子**直接**相连。

白云想知道，她有多少种给予数的方案呢？

鼠在不停地挣扎，想要摆脱绳子的束缚。白云还没有思考出来，鼠便把红色绳子都咬断了。

白兔有些气恼，但是他还是想要知道答案。他便问白云：对于所有红色绳子的连接方案，答案的总和（即求所有红色绳子连接方案的给予数方案之和）是多少？

鼠在不停地挣扎，想要摆脱绳子的束缚。白云还没有思考出来，鼠便把蓝色绳子也咬断了。

白兔有些气恼，但是他还是想要知道答案。他便问白云：对于所有红色和蓝色绳子的连接方案，答案的总和（即求所有红色和蓝色绳子连接方案的给予数方案之和）是多少？两个方案不同当且仅当存在至少一对鼠，在两种方案中，这两只鼠之间直接连接的绳子不同（两只鼠之间连接绳子的可能性有 4 种：没有绳子直接连接，只有红色绳子直接连接，只有蓝色绳子直接连接，两种颜色的绳子均直接连接）。

白云哭了。

【题目描述】

本题包含三个问题：

- 问题 0：已知两棵 n 个节点的树的形态（两棵树的节点标号均为 1 至 n），其中第一棵树是红树，第二棵树是蓝树。要给予每个节点一个 $[1,y]$ 中的整数，使得对于任意两个节点 p,q ，如果存在一条路径 $(a_1=p,a_2,\ldots,a_m=q)$ 同时属于这两棵树，则 p,q 必须被给予相同的数。求给予数的方案数。
  - 存在一条路径同时属于这两棵树的定义见【题目背景】。
- 问题 1：已知蓝树，对于红树的所有 n^{n-2} 种选择方案，求问题 0 的答案之和。

- 问题 2：对于蓝树的所有 n^{n-2} 种选择方案，求问题 1 的答案之和。

提示：n 个节点的树一共有 n^{n-2} 种。

在不同的测试点中，你将可能需要回答不同的问题。我们将用 op 来指代你需要回答的问题编号（对应上述 0、1、2）。

由于答案可能很大，因此你只需要输出答案对 $998,244,353$ 取模的结果即可。

【输入格式】

从文件 ***tree.in*** 中读入数据。

第一行三个用空格隔开的整数 n, y, op。

如果 $op = 0$ ，则接下来 $2 \times (n-1)$ 行，前 $(n-1)$ 行每描述一条蓝色绳子，接下来 $(n-1)$ 行每行描述一条红色绳子。

如果 $op = 1$ ，则接下来 $n-1$ 行，每行描述一条蓝色绳子。

如果 $op = 2$ ，则接下来没有输入。

描述绳子的各行将包含两个用空格隔开的整数，分别表示被这条绳子连接的两只鼠的编号。鼠的编号是从 1 开始的。

【输出格式】

输出到文件 ***tree.out*** 中。

输出一个整数，表示答案对 $998,244,353$ 取模的结果。

【样例 1 输入】

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

【样例 1 输出】

```
2
```

【样例 1 解释】

两棵树相同，所以任意两个点都必须被给予相同的数，方案数为 2 。

【样例 2 输入】

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

【样例 2 输出】

```
10
```

【样例 2 解释】

红树共有三种可能的情况：

1. 包含绳子 $(1,2)$（表示连接 $1,2$ 号鼠的绳子，下同）、$(2,3)$ ：此时任意两只鼠都必须被给予相同的数，问题 0 的答案为 2 。
2. 包含绳子 $(1,2)$、$(1,3)$：此时 1 号鼠和 2 号鼠必须被给予相同的数，问题 0 的答案为 $2\times2=4$ 。
3. 包含绳子 $(2,3)$ 、$(1,3)$ ：此时 2 号点和 3 号点必须被给予相同的数，问题 0 的答案为 $2\times2=4$ 。

综上，问题 1 的答案为 $2+4+4=10$ 。

【样例 3 输入】

```
3 2 2
```

【样例 3 输出】

```
30
```

【样例 3 解释】

蓝树一共有三种可能的情况。不难发现，对于蓝树的每一种情况，求得的问题 1 的答案都是 10 。

所以答案为 $10\times3=30$ 。

【样例 4】

见选手目录下的 ***tree/tree4.in*** 与 ***tree/tree4.ans***。

【样例 5】

见选手目录下的 ***tree/tree5.in*** 与 ***tree/tree5.ans***。

【样例 6】

见选手目录下的 ***tree/tree6.in*** 与 ***tree/tree6.ans***。

【子任务分值】

在本场比赛中，测试点（或子任务）的分值分布与你是否为集训队选手有关。本题的分值设置在【子任务】的表格中呈现。

【子任务】

<table>
<tr><th>问题类型 (op)=</th><th>测试点编号</th><th>n</th><th>y</th><th>集训队每点分值</th><th>非集训队每点分值</th></tr>
<tr><td rowspan="3">0</td><td>1</td><td>≤ 10</td><td rowspan="9">无特殊限制</td><td rowspan="3">2</td><td>18</td></tr>
<tr><td>2</td><td rowspan="2">$\le 10^5$</td><td rowspan="2">5</td></tr>
<tr><td>3</td></tr>
<tr><td rowspan="9">1</td><td>4</td><td>$= 3$</td><td rowspan="2">1</td><td rowspan="8">4</td></tr>
<tr><td>5</td><td>$= 5$</td></tr>
<tr><td>6</td><td rowspan="2">≤ 500</td><td rowspan="4">6</td></tr>
<tr><td>7</td></tr>
<tr><td>8</td><td rowspan="2">≤ 5000</td></tr>
<tr><td>9</td></tr>
<tr><td>10</td><td rowspan="5">$\le 10^5$</td><td>$= 1$</td><td>1</td></tr>
<tr><td>11</td><td rowspan="4">无特殊限制</td><td rowspan="4">5</td><td rowspan="4">2</td></tr>
<tr><td>12</td></tr>
<tr><td>13</td></tr>
<tr><td>14</td></tr>
<tr><td rowspan="9">2</td><td>15</td><td>$= 3$</td><td rowspan="2">1</td><td rowspan="8">4</td></tr>
<tr><td>16</td><td>$= 10$</td></tr>
<tr><td>17</td><td rowspan="2">≤ 500</td><td rowspan="4">6</td></tr>
<tr><td>18</td></tr>
<tr><td>19</td><td rowspan="2">≤ 5000</td></tr>
<tr><td>20</td></tr>
<tr><td>21</td><td rowspan="5">$\le 10^5$</td><td>$= 1$</td><td>1</td></tr>
<tr><td>22</td><td rowspan="4">无特殊限制</td><td rowspan="4">5</td><td rowspan="4">2</td></tr>
<tr><td>23</td></tr>
<tr><td>24</td></tr>
<tr><td>25</td></tr>
</table>

为了优化你的阅读体验，我们把**测试点编号**放在了表格的中间，请注意这一点。

所有测试点均满足 $3 \le n \le 10^5$，$1 \le y < 998244353$，$op \in \{0, 1, 2\}$。

远古计算机（oldcomputer）

这是一道提交答案题。

【题目背景】

企鹅大陆是一片神奇的土地，在厚厚的冰层下埋藏着一个巨大的宝藏。探险家企鹅豆豆挖穿冰层到达了宝库，但他发现了一个令人发愁的问题。一共有五座宝库，每个宝库是由某些远古计算机控制的。由于年代久远，这些计算机里的程序已经消失不见了，只有给这些计算机重新填写代码并且顺利运行，输出了正确结果才能触发开门的机关。

【题目描述】

每一个宝库大门由一个计算机集群控制，计算机之间用数据线相连以便传输数据。但是有很多数据线已经损坏了，所以只留下了一部分连线。一开始数据线上没有数据，当一台计算机向数据线上写入时，数据线上就有了一个整数。每条数据线上最多可以同时传输一个整数，当整数被读取后便会消失，数据线就又回到没有数据的状态。

每台远古计算机有两个储存单元，分别名为 `a` 和 `b`，每个储存单元能够储存一个 -2147483648 到 2147483647 之间的整数。

每个时刻，每台远古计算机可以执行一条指令，一共有以下几种指令：

- mov `reg` `val` 将储存单元 `reg` 的值赋值为 `val` 的值
- add `reg` `val` 给储存单元 `reg` 加上 `val` 的值
- dec `reg` `val` 给储存单元 `reg` 减去 `val` 的值
- mul `reg` `val` 给储存单元 `reg` 乘以 `val` 的值
- div `reg` `val` 给储存单元 `reg` 除以 `val` 的值，这里的除法是向零取整的除法，如 $\frac{-5}{2} = -2$
- and `reg` `val` 给储存单元 `reg` 二进制与上 `val` 的值
- or `reg` `val` 给储存单元 `reg` 二进制或上 `val` 的值
- xor `reg` `val` 给储存单元 `reg` 二进制异或上 `val` 的值
- jmp `val` 跳转到整个程序第 `val` 条语句，语句从程序开头开始，用从 `1` 开始的正整数计数
- jz `reg` `val` 如果 `reg` 的值为 0，那么跳转到第 `val` 行
- jnz `reg` `val` 如果 `reg` 的值不为 0，那么跳转到第 `val` 行
- jgz `reg` `val` 如果 `reg` 的值严格大于 0，那么跳转到第 `val` 行
- jsz `reg` `val` 如果 `reg` 的值严格小于 0，那么跳转到第 `val` 行
- read `x` `reg` 从远古计算机 x 读取一个数到储存单元 `reg` 当中，如果数据线上缓存了一个数字，将读取这个数字并返回，否则等待下个周期再次尝试读取。$x = 0$ 时视为从标准输入数据读取一个数。

- write `val` `x` 将 `val` 的数值向远古计算机 x 方向所在数据线写入，当且仅当数据线上没有存有数据才会成功写入，否则等待下个周期再次尝试写入。$x = 0$ 时视为向标准输出数据写出一个数。

`reg` 表示一个储存单元，只能为 `a` 或者 `b` 之一。

`val` 表示一个储存单元或者一个数字的值，比如填入 `a` 表示 `a` 中储存的值或者填入 `233` 表示 `233` 这个数字。

一台远古计算机读写指令中 `x` 只有与当前远古计算机直接有数据线相连，或者 `x=0` 才被视为合法指令。

每台远古计算机的标准输入输出独立，远古计算机之间互不影响，即每台远古计算机都有独立的一个标准输入端和一个标准输出端。

每个周期计算时，所有需要执行 `write` 指令的远古计算机先计算，然后需要执行 `read` 指令的远古计算机再计算，需要执行其余指令的远古计算机最后计算。

在读取时，如果标准输入数据没有任何可以继续读取的数据，该远古计算机将进入永远等待状态。

一台远古计算机如果执行完了最后一条指令，将会重新从第一条指令开始执行。

如果一台远古计算机没有任何指令，该计算机将永远处于等待状态。

指令计数是从 `1` 开始的正整数。

一条数据线上最多只能暂存一个数据，两台计算机之间只有一条数据线，即可能读取自己上一轮写入的数据，如果两端的远古计算机同时读取或写入同一条数据线上的数据，结果将不可预知。

不存在写入标准输入的方法或是从标准输出当中读取数据的方法。

比如如下样例是从 `1` 号计算机的标准输入读入两个数，并从 `2` 号计算机的标准输出输出两个数之和。

```
node 1
read 0 a
read 0 b
write a 1
write b 1
node 2
read 1 a
read 1 b
add a b
write a 0
```

而以下写法是错误的

```
node 1
read 0 a
```

```
read 0 b
add a b
write a 0
node 2
mov a a
```

因为正确答案中，一号远古计算机的标准输出为空，而二号远古计算机的标准输出才是两个数之和。

【输入格式】

这是一道提交答案题，共有 5 组输入数据，这些数据命名为 ***oldcomputer1.in*** ~ ***oldcomputer5.in***。

这些文件描述了远古计算机之间的连线状态。

文件的第一行是三个非负整数 x、n 和 m，表示测试点编号、远古计算机的个数与他们之间的连线条数，远古计算机是从 1 到 n 标号的。

接下来 m 行，每行两个正整数 x, y，表示计算机 x 和计算机 y 之间通过数据线直接相连。

【输出格式】

对于每组输入数据，你需要提交相应的输出文件 ***oldcomputer1.out*** ~ ***oldcomputer5.out***。

这些文件描述了每台远古计算机的代码内容。

文件由多个代码块组成，每个代码块的具体格式如下：

第一行为一个字符串 `node` 与一个 `1` 到 `n` 之间的整数 `a`，由一个空格隔开，表示接下来是计算机 a 的指令。

接下来多行为该计算机的具体指令内容。

每台计算机的指令应当最多出现一次，否则将会出现未知错误。

所有计算机的指令总条数应当不超过 10^6 行。

【子任务】

子任务 1：

> 1 号远古计算机的标准输入将会有不超过 100 个非负整数，按照原顺序输出到 1 号远古计算机的输出当中。

子任务 2：

1 号远古计算机一个非负整数 k ，按照原输入顺序将斐波那契数列第 k 项输出到 1 号点的标准输出当中，输入数据保证第 k 项不超过 10^9。斐波那契数列通项公式为 $F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2} (2 \le n)$。

子任务 3：

1 号远古计算机的标准输入将会有不超过 100 个非负整数，按照原顺序输出到 n 号远古计算机的输出当中。

子任务 4：

第 1 到 50 号远古计算机分别输入 1 个数，将这 50 个数从 51 到 100 号远古计算机输出，输出顺序任意，每个远古计算机输出数字个数任意。

子任务 5：

第 1 到 10 号远古计算机各输入 1 个数，将这些数对应从 100 到 91 号远古计算机输出，即 i 号点输入的数需要从 $101 - i$ 号点输出。

【评分方式】

对于每个测试点，有三个参数 $a1, a2, a3$。

如果选手提交的代码在 $a1$ 个周期内正确输出了答案，将会得到该测试点 100% 的分。

如果选手提交的代码在 $a2$ 个周期内正确输出了答案，将会得到该测试点 60% 的分。

如果选手提交的代码在 $a3$ 个周期内正确输出了答案，将会得到该测试点 30% 的分。

即在每个周期结束时，会分别进行一次答案正确性判断，以最早正确的一次为准。

【提示】

以下为 `checker` 的正确使用方式。

- `./checker graph.in input.in code.out x -detail`
  - 其中 `graph.in` 为连线关系描述文件。
  - `input.in` 为输入数据。
  - `code.out` 为选手提交的文件。
  - `x` 为一个非负整数表示执行周期数。
  - `-detail` 为一个可选字段，附加上这个字段后将会输出每个周期每个远古计算机执行的代码内容到 `detail.out`。

  - checker 将会运行选手的代码，输出 x 个周期结束时的运行结果到 result.out 中。

- 输入数据 input.in 格式要求如下：
  - 一共 n 行，每行多个整数，表示对应的计算机的标准输入内容。
- 输出数据 result.out 格式如下：
  - 第一行一个字符串为返回的信息。
  - 如果正确运行，还会有 n 行，每行多个整数，表示对应的计算机的标准输出内容。

下发文件一共包括 1.in,2.in,3.in,4.in,5.in,checker。

【子任务分值】

在本场比赛中，测试点（或子任务）的分值分布与你是否为集训队选手有关。本题的分值设置如下：

子任务编号	1	2	3	4	5
集训队分值	10	15	15	30	30
非集训队分值	20	20	20	20	20

I 君的商店（shop）

这是一道交互题。

【题目背景】

V 君、I 君和 Y 君是好朋友。

I 君最近开了一家商店，商店里准备了 N 种物品（编号为 $0 \sim N-1$ 中的整数），每种物品均有无限个可供出售，每种物品的单价是 0 或者 1。

V 君想知道每个物品的价格，他已经通过某种超自然力量知道，这 N 个物品里，价格是 1 的物品恰好有奇数/偶数个，且**至少存在一个物品的价格是 1**。

然而，V 君并不想自己去问 I 君，他选择了这样一种方法：他准备了 $+\infty$ 的钱给 Y 君。然后让 Y 君帮他跑腿：每一次，他会给 Y 君指定两个非空物品**集合** S, T（同一个集合内的物品必须两两不同，即每种物品在每个集合类最多包含一个），Y 君会跑到商店，分别买下这两个集合的物品，把他们送回来，并告诉 V 君哪个集合的物品价格之和更高。但是，当**两集合价格之和相等**的时候，Y 君会按照 I 君的指示来回答 V 君。

带着很多物品跑腿是一个很累的事情，因此，我们定义一次跑腿的体力消耗是 $\mathtt{S}+\mathtt{T}$。其中，$\mathtt{S}$ 表示集合 S 包含的物品个数。

你的任务是：写一个程序，帮助 V 君决定如何合理地让 Y 君跑腿，从而推算出每种物品的价值。Y 君的体力有限，你当然不能让他过于劳 累，也即，你不能让他的总体力消耗超过某个预设的阈值 M。

【实现细节】

你不需要，也不应该实现主函数，你只需要实现下列函数：

- `find_price(task_id, N, K, ans)`
- 其中 task_id 表示子任务编号（见限制与约定）。N 表示物品个数，K 的意义为：
  - 若 $K=0$，表示有偶数个物品价值为 1 .
  - 若 $K=1$，表示有奇数个物品价值为 1 .
- 你需要将计算出的物品价格放在数组 ans[] 中，其中 ans[i] 表示编号为 i 的物品的价格。

你可以通过调用如下函数来向交互库发出询问：

- `query(S, nS, T, nT)`
- 这里 $\mathrm{nS}=\mathtt{S}, \mathrm{nT}=\mathtt{T}$, 数组 $S[0...(\mathrm{nS}-1)]$ 和数组 $T[0...(\mathrm{nT}-1)]$ 分别描述两个集合，你需要保证：
  - $\mathrm{nS}, \mathrm{nT} > 0$.
  - $\forall 0 \le i < \mathrm{nS}$, $0 \le S[i] < N$;
  - $\forall 0 \le i < \mathrm{nT}$, $0 \le T[i] < N$;

  - $\forall 0 \le i < j < \mathrm{nS}$, $S[i] \ne S[j]$ ；
  - $\forall 0 \le i < j < \mathrm{nT}$, $T[i] \ne T[j]$ 。

  $\forall$ 的意思是："对于任意的"。例如：$\forall 0 \le i < \mathrm{nS}$, $0 \le S[i] < N$ 的意思是："对于任意的在 $[0, nS)$ 内的 i ，$S[i]$ 在 $[0, N)$ 内"。

- 调用此函数一次的时间复杂度为 $\Theta(\mathrm{nS} + \mathrm{nT})$ 。它的返回值为 0 或 1，返回值的意义为：
  - 若集合 S 的物品价格和更大，返回 0 ;
  - 若集合 T 的物品价格和更大，返回 1 ;
  - 否则，按照某种未知规则返回 0 或 1 .
- 如题面所述，我们定义这样一次调用的代价为 $\mathrm{nS} + \mathrm{nT}$ 。

评测时，交互库可能会调用 find_price 多次（不超过 10 次），每次调用代表一次新的猜价格游戏，所有的物品的价格都会被重新设定。

【实现方法】

选手工作目录下已经提供了一个 sample_shop.cpp/c/pas ，我们推荐你在其基础上进行答题，但这不是必要的。

对 C++ / C 语言选手

你需要在你的程序开头加上 `#include "shop.h"` 。

你需要实现的函数 find_price ：

```
void find_price(int task_id, int N, int K, int ans[]);
```

函数 query 的接口如下：

```
int query(int S[], int nS, int T[], int nT);
```

对 Pascal 语言选手

你需要包含单元 graderhelperlib 。

你需要实现的函数 find_price：

```
procedure find_price(task_id, N, K: longint; var ans: array[0..N-1] of longint);
```

函数 query 的接口如下：

```
function query(var S: array of longint; nS: longint; var T: array of longint; nT: longint): longint;
```

【如何评测你的程序】

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 shop.c -o shop -O2
```

对于 C++ 语言：

```
g++ grader.cpp shop.cpp -o shop -O2
```

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"shop" -O2
```

可执行文件将从**标准输入**读入以下格式的数据：
第一行一个整数 T（$T \le 100$），表示数据组数。对每组数据：

- 第一行包含两个整数 $task_id, N$；
- 接下来一行一个长度为 N 的 01 串 s，其中 s_i 表示物品 i 的价格。你需要保证至少有一个物品价格为 1 。

读入完成之后，交互库将调用 T 次 `find_price` 函数。

接下来交互库会判断你的函数每次计算是否正确，若全部正确则会输出"Correct"，否则会输出相应的错误信息。

【评分标准】

最终评测时只会收取 shop.cpp/c/pas，修改选手目录中的其他文件对评测无效。

题目首先会受到和非交互式程序题相同的限制。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：

- **每一次** `find_price` 的调用中：
- 你的每一个 `query` 的调用符合规则，且调用代价之和不超过 M；
- 你的函数正确计算了 ans[] 数组。

在一个测试点中，如果你没有获得满分，那么你获得 0 分。

【子任务】

我们令代价之和的上界为 M，记答案数组为 ans[]：

- 子任务 1：$N \le 5$，$M = 100$。
- 子任务 2：$N \le 10^3$，$M = 10^6$。
- 子任务 3：$N \le 10^5$，$M = 100$，保证 $\forall i < j < k$，若 $\text{ans}[i] = \text{ans}[k]$ 则必有 $\text{ans}[j] = \text{ans}[i]$。

- 子任务 4：$N \le 10^4$，$M = 2 \times 10^5$ 。
- 子任务 5：$N \le 5 \times 10^4$，$M = 350100$。
- 子任务 6：$N \le 10^5$，$M = 500100$ 。

【提示】

I 君可能并不愿意让 V 君知道每件物品的价格，在物品价格相等时，他会按照他自己的某种方式来回答问题。

【子任务分值】

在本场比赛中，测试点（或子任务）的分值分布与你是否为集训队选手有关。本题的分值设置如下：

子任务编号	1	2	3	4	5	6
集训队分值	20	11	9	12	17	31
非集训队分值	31	21	13	9	11	15