树上邻域数点（count）

这是一道交互题。

【题目描述】

给出五元组 (T, I, S_V, S_E, ι)，其中：

- T 是一棵 n 个点的有根树 $T = (V, E)$，其中 V 为 T 的点集，E 为 T 的边集。树的节点被编号为 $1, 2, \ldots, n$，其中根节点编号为 1。
- I 是一个集合，集合中的元素称作**信息**。其中有两个不同的特殊元素：单位元 ϵ 和不合法信息 $\perp$。

对于一般的信息，其都具有**点集合**和**边集合**两个属性。特别的，对于单位元，其只有边集合的属性，而对于不合法信息，其没有以上两种属性。

- 对于信息 $o \in I \setminus \{\epsilon, \perp\}$，$o$ 的**点集合**是 V 的一个二元子集，记作 $S_V(o)$，满足 $S_V(o) \subseteq V$ 且 $|S_V(o)| = 2$。其中，两个集合 A, B 的差 $A \setminus B$ 被定义为 $A \setminus B = \{x \in A | x \notin B\}$。
- 对于信息 $o \in I \setminus \{\perp\}$，$o$ 的**边集合**是 E 的一个子集，记作 $S_E(o)$，满足 $S_E(o) \subseteq E$。规定单位元的边集合为空，也即 $S_E(\epsilon) = \varnothing$。
- 对于树上的任何一条边 $e \in E$，记 $e = (u, v)$，存在一个关于 e 的信息 $\iota(e) \in I$，它以其端点为点集合、自身为边集合，即 $S_V(\iota(e)) = \{u, v\}$，$S_E(\iota(e)) = \{e\}$。

信息有两种合并的方式，分别记作 R 和 C。对于 $\forall a, b \in I$，记 $r = R(a, b), c = C(a, b)$，满足 $r, c \in I$，则：

- 单位元和任何信息合并都得到对方。也即，如果 $a = \epsilon$，那么 $r = c = b$；如果 $b = \epsilon$，那么 $r = c = a$。
- 不合法信息和任何信息合并都得到不合法信息。也即，如果 $a = \perp$ 或者 $b = \perp$，那么 $r = c = \perp$。
- 对于剩下的情况，如果两个信息的**边集合**的交集非空，或者**点集合**的交集的大小不为 1，则合并得到不合法信息。也即，如果 $S_E(a) \cap S_E(b) \neq \varnothing$ 或 $|S_V(a) \cap S_V(b)| \neq 1$，则 $r = c = \perp$。
- 否则，有

$$S_E(r) = S_E(c) = S_E(a) \cup S_E(b),$$
$$S_V(r) = S_V(a),$$
$$S_V(c) = S_V(a) \oplus S_V(b).$$

其中 $\oplus$ 表示集合的对称差运算，也即 $A \oplus B = (A \cup B) \setminus (A \cap B)$。

定义 T 中两个点的树上距离为树上以两个点为端点的唯一简单路径经过的边数。

给出评分参数 M 和 q 次询问，每次询问给出树上的一个点 u 和一个非负整数 d。记点集 X 为 T 中所有与 u 的树上距离不超过 d 的点构成的集合，又记边集 $Y = \{(a,b) \in E \mid a,b \in X\}$ 为 X 内部的边集。可以证明，从 ϵ 和所有 $\iota(e)$ $(e \in E)$ 出发，总是能通过有限次 R, C 的调用得到信息 $o \neq \perp$ 满足 $S_E(o) = Y$。

每组询问中，你需要在 R 和 C 的调用次数总和不超过 M 的限制下构造出一个满足这样的要求的信息 o。特别地，如果 $d = 0$，则直接返回单位元 ϵ 即可。

【实现细节】

请确保你的程序开头有 `#include "count.h"`。

头文件 `count.h` 中实现了如下内容：

1. 定义了信息对应的数据类型 `info`；
2. 定义了 ϵ 所对应的 `info` 类型常量 `emptyinfo`，你可以在程序中直接使用。
3. 定义并实现了以下两个信息合并函数，你可以在程序中直接调用：

```
info MR(info a,info b);
info MC(info a,info b);
```

- 两个函数分别返回 $R(a,b)$ 与 $C(a,b)$ 对应的信息。

你需要保证调用 $R(a,b)$ 或 $C(a,b)$ 时结果不为 $\perp$，否则程序可能会出现异常行为。

4. 定义并实现了判定一个信息是否为单位元的函数，你可以在程序中直接调用：

```
bool isempty(info a);
```

- 这个函数返回真当且仅当 a 为单位元。

可以查看参考交互库了解更多实现细节。

你不需要，也不应该实现主函数。你需要实现如下几个函数：

```
void init(int T, int n, int q, vector<int> fa, vector<info> e, int
  M);
```

- `T` 表示测试点编号，`n` 表示树的点数，`q` 表示询问数，`M` 表示该测试点的评分参数。
- `fa` 和 `e` 的长度均为 $n-1$。对于 $0 \leq i < n-1$，$fa[i]$ 和 $i+2$ 为第 i 条边 e_i 的两个端点，$e[i]$ 为题目描述中提到的 $\iota(e_i)$ 所对应的 `info` 类型元素。数据保证 $fa[i]$ 小于 $i+2$。

```
info ask(int u, int d);
```

- 给出一个询问，参数的意义见题目描述。你需要在函数结束时返回一个满足题设条件的信息。

最终测试时，在每个测试点，交互库会恰好调用一次 `init` 函数，随后调用 q 次 `ask` 函数。交互库会使用特殊的实现方式，单个 `info` 类型的变量会恒定消耗 12 字节内存，这与下发的参考交互库不同。为保证程序运行时内存使用在题目限制内，你需要保证运行过程中没有过多的 `info` 类型变量同时存在。

保证在满足调用次数限制且不进行 `isempty` 函数调用的情况下，最终测试的交互库运行所需的时间不超过 0.6 秒，交互库本身所消耗的内存不超过 16 MiB。保证在只执行 10^8 次 `isempty` 函数调用的情况下，最终测试的交互库运行的时间不超过 0.25 秒。

在下发文件中包含一个名为 `count.cpp` 的文件，作为示例程序，选手可以在此基础上继续实现本题。在下发文件中还额外包含一个名为 `count_backup.h` 的备份文件，我们保证其与 `count.h` 文件完全相同。

【测试程序方式】

本题目录下提供了两个交互库的参考实现 `grader.o` , `checker.o`, 其为两个不同的交互库编译产生的可链接文件。最终测试时所用的交互库实现与该实现有不同，因此选手的解法不应依赖交互库的具体实现，同时也不应该依赖 `count.h` 中 `info` 类型的具体实现。

你需要修改下发的 `count.h` 来帮助进行链接。具体的，在将源代码 `count.cpp` 和程序 `grader.o` 进行链接的时候，你需要注释掉 `count.h` 代码的第 5 行，并保留第 4 行的代码。链接 `checker.o` 方法类似，需要注释掉 `count.h` 代码的第 4 行，并保留第 5 行的代码. 选手可以对 `count.h` 的实现自行修改来实现不同程序的编译。

修改后，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++ count.cpp -c -O2 -std=c++14 -lm && g++ count.o grader.o -o
   count
g++ count.cpp -c -O2 -std=c++14 -lm && g++ count.o checker.o -o
   count
```

其中第一行命令会编译当前 `count.cpp` 后与 `grader.o` 链接起来，生成可执行文件 `count`，第二行命令则会编译当前 `count.cpp` 后与 `checker.o` 链接起来，生成可执行文件 `count`。

按上述方法编译得到的可执行文件 `count`，其运行方式如下：

- 可执行文件将从标准输入读入以下格式的数据：
  - 第一行四个整数 id, n, q, M，分别表示测试点编号、树的点数、询问数和评分参数；
  - 第二行 $n-1$ 个整数 $p_2, p_3, \cdots, p_n$，分别表示 2 至 n 的父亲节点编号，在本地调试时你需要保证 $\forall i \in [2, n], p_i < i$；
  - 接下来 q 行每行两个整数 u, d，描述一次询问。

- 读入之后，交互库会进行测试。如果你的程序不满足交互库限制，其会在输出中返回对应的错误信息。否则，对于链接的可执行文件，其输出如下:
  - 总共一行三个整数 C_1, C_2, C_3，其中：
    - C_1 表示程序在 `init` 函数中调用交互库函数的总次数；
    - C_2 表示程序在运行过程中调用交互库函数的总次数；
    - C_3 表示程序在 q 次 `ask` 函数中调用交互库函数的次数的最大值。
    - 对于上述三个统计量，我们只会计入 `MR`、`MC` 函数的调用次数，而不会计入 `isempty` 函数的调用次数。
- 在链接不同文件的时候，其能够进行的检查也不同，具体的:
  - `grader.o`：其在运行时不会检查 `ask` 函数返回的信息是否正确，但可以帮助选手判断交互操作是否符合要求。这份程序运行时间最接近评测时的交互库，因此选手可以利用该程序测试运行速度，但不保证程序正确性。
  - `checker.o`：其在运行时会检查 `ask` 函数返回的信息是否正确，也可以帮助选手判断交互操作是否符合要求。同时其会检查 `ask` 函数返回的信息是否正确。这份程序可以进行答案正确性的检查。

选手在调试时需要保证输入可执行文件 `count` 的数据满足上述输入格式，否则不保证输出结果正确。

【样例 1】

见选手目录下的 ***count/count1.in*** 与 ***count/count1.ans***。

【样例 2】

见选手目录下的 ***count/count2.in*** 与 ***count/count2.ans***。
该组样例满足数据范围中的特殊性质 A。

【样例 3】

见选手目录下的 ***count/count3.in*** 与 ***count/count3.ans***。
该组样例满足数据范围中的特殊性质 B。

【样例 4】

见选手目录下的 ***count/count4.in*** 与 ***count/count4.ans***。

【评分方式】

最终评测只会收取 `count.cpp`，修改选手目录下其他文件不会对评测结果产生影响。注意：

- 未初始化的 **`info`** 类型的变量不保证是 **`emptyinfo`**。
- 请不要尝试访问或修改 **`info`** 类型的成员变量，否则将被视为攻击交互库。
- 请不要在 **`init`** 函数调用之前调用 **`MR`** 和 **`MC`** 函数，否则可能回发生未定义行为。
- 你只能访问自己定义的变量和交互库返回的 **`info`** 类型变量，尝试访问其他空间将可能导致编译错误或运行时错误。

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。

在上述条件以外，在一个测试点中，若程序执行了非法的函数调用或询问操作中给出了错误回答，该测试点将会获得 0 分。否则，记 C_1, C_3 分别表示你的程序在 `init` 函数中调用交互库函数的次数，和你的程序在所有 q 次 `ask` 函数中调用交互库函数的次数的最大值。如果 $C_1 \leq 3 \cdot 10^7$ 且 C_3 不超过该测试点的评分参数 M，你将获得该测试点的分数，否则你无法获得该测试点的分数。注意：计算 C_1, C_3 时只会计入 `MR`、`MC` 函数的调用次数，而不会计入 `isempty` 函数的调用次数。

【数据范围】

对于所有测试点，$1 \leq n \leq 2 \cdot 10^5$，$1 \leq q \leq 10^6$；每组询问中，有 $1 \leq u \leq n$，$1 \leq d \leq n-1$。

测试点	$n=$	$q=$	特殊性质	$M=$
1	10^3	10^4		500
2	$2,000$			
3,4	10^5	10^6	A	5
5,6	6×10^4	6×10^4	B	50
7				5
8	10^5	10^5		
9	$7,500$	5×10^4	C	500
10	10^4			
11	$15,000$			
12	2×10^4			50
13	$25,000$			5
14	3×10^4	10^5		
15	6×10^4	10^6	D	
16				
17	8×10^4			
18	10^5			
19	1.5×10^5			
20	2×10^5			1

特殊性质 A：保证 $\forall i\in[1,n-1]$，编号为 $i+1$ 的点的父节点为 i.

特殊性质 B：保证所有询问均满足 $u=1$。

特殊性质 C：保证所有询问均满足 $d\le100$。

特殊性质 D：保证所有询问均满足 $d\ge1000$。