1011.祖孙问询

一棵n个节点的有根树，有m个询问(x,y)，问x与y的祖孙关系。注意：节点编号不一定是1到n。

输入文件为tree.in

第一行包括一个整数n表示节点个数

接下来n行每行一对整数对a和b表示a和b之间有连边。如果b是-1，那么a就是树的根。第n+2行是一个整数m表示询问个数。

接下来m行，每行两个正整数x和y，保证x和y不同。

说明:

对于30%的数据，n,m≤1000

对于100%的数据,n,m≤40000，每个节点的编号都不超过40000

输出文件为tree.out

对于每一个询问，输出1:如果x是y的祖先，输出2:如果y是x的祖先，否则输出0

输入样例

10

234 -1

12 234

13 234

14 234

15 234

16 234

17 234

18 234

19 234

233 19

5

234 233

233 12

233 13

233 15

233 19

输出样例

1

0

0

0

2

2161.败家子

你家有一本家族族谱，形状像1棵树，共n个节点，编号1到n，根节点是1号。相邻两个节点就是父母和子女的直系关系。节点u的个人资产为x[u]元，这个数字有可能是负数。初始时，每个人资产为x[u]都为0元。

目前有q个事件依次发生，形式如下：

第一种事件：LOSE u p代表u号节点的个人资产损失p元，即x[u]减少p。

第二种事件：CHECK u代表队u号节点为根的子树进行总资产求和的查账。

输入文件prodigal.in 输入第一行为正整数n，n<=100000。接着n-1行为树边的描述，每行两个正整数u,v代表u和v之间有一条边。接着一行输入正整数q，q<=200000。接着共q行，每行为字符串搭配1个正整数u,或2个正整数数字u和p。其中u都在1到n之间，p<=100。

输出文件prodigal.out 输出共一行，为若干个整数，即查账的答案，由空格隔开。

输入样例：

3

1 2

1 3

4

LOSE 3 10

CHECK 1

LOSE 2 20

CHECK 1

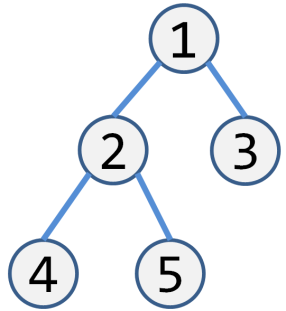
输出样例：

-10 -30

2162.欧拉序列2

欧拉序列一共有两种形态，今天我们介绍第二类欧拉序列。

有1棵树，共n个节点编号1到n，根节点是1号。欧拉序列有2种常见定义，今天学习第1种定义：从根节点开始按照DFS顺序访问节点和边，访问包括回溯的步骤。在DFS时若某节点u有多个子节点，优先挑选编号小的子节点访问。对每个节点u，每一次访问u时，我们会将u按照访问顺序依次记录在欧拉序列（第二类）中。这种欧拉序列总长度为2\*n-1，每个节点至少出现1次，也可能出现多次。



如图，第二类欧拉序列为124252131。

根据欧拉序列，我们顺便整理出以下实用信息：

T[u].L代表节点u第一次访问是欧拉序列里的第几个。

T[u].R代表节点u最后一次访问是欧拉序列里的第几个。

这样的话，节点u为根的子树就对应序列中的一个区间[T[u].L,T[u].R]。

在图中例子中，

T[1].L=1, T[1].R=9,

T[2].L=2, T[2].R=6,

T[3].L=8, T[3].R=8,

T[4].L=3, T[4].R=3,

T[5].L=5, T[5].R=5,

输入文件euler.in 输入第一行为正整数n，n<=1000。接着n-1行为树边的描述，每行两个正整数u,v代表u和v之间有一条边。

输出文件euler.out 输出共三行：第一行为欧拉序列，共2\*n-1个正整数；第二行为T[u].L信息，共n个正整数；第二行为T[u].R信息，共n个正整数。格式要求数字之间由空格隔开，但行末不能有空格。

输入样例：

5

1 2

1 3

2 4

2 5

输出样例：

1 2 4 2 5 2 1 3 1

1 2 8 3 5

9 6 8 3 5

拓展题：1901. 换根子树（模板题）

一棵n个节点的无根树，有q个问询。每个问询为X Y，表示求以X为根时，子树Y的节点数。

本题强制在线回答，也就是每个问询需要直接给出答案，才能够继续处理下一个问询。你需要使用以下代码模板生成输入数据及计算输出

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | int nextInt(int i,int n){      return abs((i\*i+996703)^(i\*i\*i+167317))%n+1;  }    int main(){      cin>>n>>q>>p>>seed;      for (int i=1;i<n;++i) // 生成树结构          add(i+1, i<=p ? i: nextInt(i,i));      int ans=0;      for (int i=1,X,Y,lastAns=seed;i<=q;i++){ // 生成问询          X=nextInt(lastAns+i,n),Y=nextInt(X+i,n);          ans^=(lastAns=solve(X,Y));      }      cout<<ans<<endl;      return 0;  } |

其中add(u,v)表示u,v间有一条边，add函数具体代码由你自己实现。lastAns表示上一个问询的答案（也就是以X为根时子树Y的节点数，solve函数是由你自己实现的算法逻辑）

输入格式:一行四个正整数n,q,p,seed

输出格式:一个整数，表示所有q个问询对应答案的异或。

输入样例

10 10 3 5

输出样例

11

说明

20%的数据n,q<=10

50%的数据n,q<=1000

70%的数据n,q<=100000

100%的数据n<=100000,q<=2000000,1<=p<=n, 1<=seed<=1000000

总共有40%的数据p<=10