洛谷U19464 山村游历(Wander)（LCT）

洛谷题目传送门
LCT维护子树信息常见套路详见我的总结

闲话

题目摘自WC模拟试题(by Philipsweng)，原题目名Wander，~~“山村游历”是自己搞出来的中文名。~~
数据自测，如有问题欢迎反馈
~~对耐心的人来说，这道题是个裸题（当我什么也没说）~~

题面

题目描述

在一个偏远的小镇上，有一些落后的山村。山村之间通过一些道路来连接。当然有的山村可能不连通。
一年当中会发生很多大事，比如说有人提议要在山村i与j之间修建一条道路，也有人觉得山村i和j之间的道路需要被拆掉。
由于小镇的落后，镇长不会允许存在两个山村i,j，他们存在超过一条路径到达对方。也就是说，假如在修建山村i,j之间的道路之前，它们已经连通了，那么这条道路就不会被修建。
但拆除道路就不一样了。假如有人建议拆除连接i,j的道路，并且i,j的确有道路相连的话，镇长就会把它拆掉。
除了道路的修建与拆迁外，热情的山村人也会到处拜访其他人。有的时候来自山村i的人会想到山村j玩。
但山村人都是不识路的，那怎么办？他们有一种奇怪的遍历方式。
设一次旅行的起点为S，终点为T，点u的边集为V(i)，那么这个走路过程可以用下面的伪代码来表示。

```
function DFS(u)
    if u==T then
        finish search
    flag[u]<-true
    random shuffle the vertices order in V(u)
        //here all permutations have equal probability to be chosen
    for i in V(u) do
        if flag[i]==false then
            count++;
            DFS(i);
    count++;
```

最后count就是这次旅行所花时间。
很显然对于一次旅行，count可能有多种取值，那么对于这次旅行时间的评估，就是count的期望。
对于每次旅行，你都要告诉山村人他这次旅行时间的评估是多少。
一开始所有的山村之间都是没有道路连接的。

输入输出格式

输入格式：

第一行两个整数N,Q，表示小镇上总共有N个山村，一年中发生了Q件大事。
接下来Q行，每行包括三个整数$type,u,v$。

- 若$type=0$，表示有人建议在u,v之间修建一条道路。

- 若$type = 1$表示有人建议拆除u, v之间的道路。
- 若$type = 2$，表示山村人要进行一次u出发到v结束的旅行。

输出格式：

输出共Q行。
对于第i件大事，若$type = 0$或1，假如这件大事能完成，则输出OK，否则输出ILLEGAL。若$type = 2$，假如这次旅行能到达终点，则输出对应的时间评估，否则输出ILLEGAL。
对于每个时间评估，输出保留4位小数。

输入输出样例

输入样例#1：

```
4 9
0 1 2
0 2 4
0 4 1
2 1 4
0 2 3
2 1 4
1 4 1
1 3 2
2 1 3
```

输出样例#1：

```
OK
OK
ILLEGAL
2.0000
OK
3.0000
ILLEGAL
OK
ILLEGAL
```

说明

对于100%的数据，$N \leq 100000, Q \leq 300000, 1 \leq u, v \leq N$

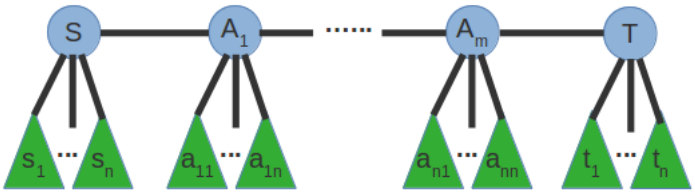
思路分析

这是LCT题目很明显吧，~~赤裸裸地道出了这个小镇的穷，连多修一条路都舍不得~~
我们只要弄明白答案是要维护什么东西就好了。
我费了好大劲，终于搞懂了这个奇葩的游历方式（或许用C++描述会更舒服一些）

```
int count=0;
void dfs(int u)
{
    if(u==T)cout<<count,exit(0);
    flag[u]=true;
    random_shuffle(V[u],V[u]+len[u]);
    for(i=0;i<len[u];++i)
        if(!flag[V[i]])count++,dfs(V[i]);
    count++;
}
```

简单的说，就是只要找不到终点，就会选一条没走过的边，一直走下去，直到碰到死路才回来。
而count就好比他走一步的时间加上走不通退回来的时间。

那么把起点和终点所在的树搞出来，那么每个点都会有若干个子树，就好像下面这样（绿色三角形表示子树，蓝色点表示$S->T$的路径，为了使路径突出，这里搞得不像一棵树了）



那么，假如有概率选择了某个子树的话，那人一定会把整个子树走完并且回来对吧。假如选择了路径上的边继续走下去，那么就再也不会回去了，以前没走完的子树也一定不会走了。
于是问题简化了。设每个子树在选择沿着$S->T$的路径继续走下去之前被选择的概率为p，那么我们要求的期望大概可以表达成
$\sum_{j=1}^{n}2p_{s_j}size_{s_j}+\sum_{i=1}^{m}\sum_{j=1}^{n}2p_{s_{ij}}size_{a_{ij}}$
仍然不能用LCT维护，我们还需要知道p。
注意这是随机的排列。对于每个排列，有且仅有另一个排列与其顺序相反。如果有一个排列，某一子树排在了路径边的前面（需要计算$size$），那么必定有且仅有另一个对应的排列使得该子树排在路径边的后面（不需要计算$size$）。由于这种等概率的对应关系，$p=\frac{1}{2}$得证。
$\frac{1}{2}$乘上系数2，不就变成了$\sum_{j=1}^{n}size_{s_j}+\sum_{i=1}^{m}\sum_{j=1}^{n}size_{a_{ij}}$么。~~哈哈哈哈全是整数！？别被保留四位小数吓到以为要搞什么概率期望DP高斯消元啦（其实我什么都不会）~~
用LCT维护虚子树size和原树总size，那么这个式子还不如直接变成该原树的总大小减去T的虚子树总大小再减1（T的大小），或者$split(T,S)$（以S为根），变成原树的总大小减去以T为根的子树总大小。
代码在此（看懂了题目，代码真的不需要什么注释了。。。。。。）

```
#include<cstdio>
#include<cstdlib>
#define R register int
#define I inline
const int N=100009;
int f[N],c[N][2],si[N],s[N];
bool r[N];
#define lc c[x][0]
#define rc c[x][1]
I bool nroot(R x){return c[f[x]][0]==x||c[f[x]][1]==x;}
I void pushup(R x){
    s[x]=s[lc]+s[rc]+si[x]+1;
}
I void pushdown(R x){
    if(r[x]){
        R t=lc;lc=rc;rc=t;
        r[lc]^=1;r[rc]^=1;r[x]=0;
    }
}
I void pushall(R x){
    if(nroot(x))pushall(f[x]);
    pushdown(x);
}
I void rotate(R x){
    R y=f[x],z=f[y],k=c[y][1]==x,w=c[x][!k];
    if(nroot(y))c[z][c[z][1]==y]=x;c[x][!k]=y;c[y][k]=w;
    f[w]=y;f[y]=x;f[x]=z;
    pushup(y);
}
/*
I void splay(R x){
    R y;
    pushall(x);
    while(nroot(x)){
        y=f[x];
        if(nroot(y=f[x]))
            rotate((c[y][0]==x)^(c[f[y]][0]==y)?y:x);
        rotate(x);
    }
    pushup(x);
```

```
}
*/
I void splay(R x){
    pushall(x);
    while(nroot(x))rotate(x);
    pushup(x);
}
I void access(R x){
    for(R y=0;x;x=f[y=x]){
        splay(x);
        si[x]+=s[rc];
        si[x]-=s[rc=y];
        pushup(x);
    }
}
I void makeroot(R x){
    access(x);splay(x);
    r[x]^=1;
}
I int findroot(R x){
    access(x);splay(x);
    while(lc)x=lc;
    return x;
}
I void split(R x,R y){
    makeroot(x);
    access(y);splay(y);
}
I bool link(R x,R y){
    makeroot(x);
    if(findroot(y)==x)return 0;
    si[f[x]=y]+=s[x];
    pushup(y);
    return 1;
}
I bool cut(R x,R y){
    makeroot(x);
    if(findroot(y)!=x||f[x]!=y||c[x][1])return 0;
    f[x]=c[y][0]=0;
    return 1;
}
#define G ch=getchar()
#define in(z) G;\
    while(ch<'-')G;\
    z=ch&15;G;\
    while(ch>'-')z*=10,z+=ch&15,G;
int main(){
    register char ch;
    R n,q,type,u,v;
    in(n);in(q);
    for(R i=1;i<=n;++i)s[i]=1;
    while(q--){
        in(type);in(u);in(v);
        if(type<2)puts((type?cut(u,v):link(u,v))?"OK":"ILLEGAL");
        else{
            split(u,v);
            if(findroot(v)!=u)puts("ILLEGAL");
            else printf("%d.0000\n",s[v]-si[v]-1);
        }
    }
    return 0;
}
```

分类: OI——题解 , 数据结构——链剖——LCT