```cpp
const long long string_hash(const char* str){
    long long h1 = 0, h2 = 0;
    for (int i = 0; str[i] != 0; i++){
        h1 = ((h1 * 10007) + str[i] + 97) % 1000000009;
        h2 = ((h2 * 997) + str[i] + 143) % 666666667;
    }
    return (h1 << 31) | h2;
}

/*----------- Pre-Processing ------------*/
void dfs0(int u)
{
    for(auto it=g[u].begin();it!=g[u].end();it++)
        if(*it!=DP[0][u])
        {
            DP[0][*it]=u;
            level[*it]=level[u]+1;
            dfs0(*it);
        }
}
void preprocess()
{
    level[0]=0;
    DP[0][0]=0;
    dfs0(0);
    for(int i=1;i<LOGN;i++)
        for(int j=0;j<n;j++)
            DP[i][j] = DP[i-1][DP[i-1][j]];
}
int lca(int a,int b)
{
    if(level[a]>level[b])swap(a,b);
    int d = level[b]-level[a];
    for(int i=0;i<LOGN;i++)
        if(d&(1<<i))
            b=DP[i][b];
    if(a==b)return a;
    for(int i=LOGN-1;i>=0;i--)
        if(DP[i][a]!=DP[i][b])
            a=DP[i][a],b=DP[i][b];
    return DP[0][a];
}
int dist(int u,int v)
{
    return level[u] + level[v] - 2*level[lca(u,v)];
}
/*-----------------Decomposition Part--------------------------*/
int nn;
void dfs1(int u,int p)
{
    sub[u]=1;
    nn++;
    for(auto it=g[u].begin();it!=g[u].end();it++)
        if(*it!=p)
        {
            dfs1(*it,u);
            sub[u]+=sub[*it];
        }
}
int dfs2(int u,int p)
{
    for(auto it=g[u].begin();it!=g[u].end();it++)
        if(*it!=p && sub[*it]>nn/2)
            return dfs2(*it,u);
    return u;
}
void decompose(int root,int p)
{
    nn=0;
    dfs1(root,root);
    int centroid = dfs2(root,root);
    if(p==-1)p=centroid;
    par[centroid]=p;
    for(auto it=g[centroid].begin();it!=g[centroid].end();it++)
    {
        g[*it].erase(centroid);
        decompose(*it,centroid);
    }
    g[centroid].clear();
}
/*----------------- Handle the Queries -----------------*/

void update(int u)
{
```

```
85        int x = u;
86        while(1)
87        {
88            ans[x] = min(ans[x],dist(x,u));
89            if(x==par[x])
90                break;
91            x = par[x];
92        }
93    }
94    int query(int u)
95    {
96        int x = u;
97        int ret=INF;
98        while(1)
99        {
100           ret = min(ret,dist(u,x) + ans[x]);
101           if(x==par[x])
102               break;
103           x = par[x];
104       }
105       return ret;
106   }
107   int main()
108   {
109       scanf("%d %d",&n,&m);
110       for(int i=0;i<n-1;i++)
111       {
112           int u,v;
113           scanf("%d %d",&u,&v);
114           g[u-1].insert(v-1);
115           g[v-1].insert(u-1);
116       }
117       preprocess();
118       decompose(0,-1);
119       for(int i=0;i<n;i++)
120           ans[i]=INF;
121       update(0);//first node is initially painted red
122       while(m--)
123       {
124           int t,v;
125           scanf("%d %d",&t,&v);v--;
126           if(t==1)
127               update(v);
128           else
129               dout(query(v));
130       }
131       return 0;
132   }
133
```