https://nanti.jisuanke.com/t/T3252

给定一树，并标记若干(u,v)之间的简单路径，多次询问(x,y)之间的简单路径完整经过了多少标记的路径

可以发现一定是询问两端点同时满足某条件，一选定路径才能被覆盖，于是可以转化为对二维平面上点对(x,y)所在位置的限制：

讨论选定路径，不放设$dfsn[u]<dfsn[v],dfsn[x]<dfsn[y]$：

如果$dfsn[u]+siz[u]<=dfsn[v]$，即二者不构成子树包含关系，则对于
$dfsn[x]\in[dfsn[u],dfsn[u]+siz[u])\ \&\ dfsn[y]\in[dfsn[v],dfsn[v]+siz[v])$的点对都会贡献1；

否则设u,v链上u的儿子为d，对于
$dfsn[x]\in[1,dfsn[d])\ \&\ dfsn[y]\in[dfsn[v],dfsn[v]+siz[v])\cup dfsn[x]\in[dfsn[v],dfsn[v]+siz[v])\ \&\ dfsn[y]\in[dfsn[d]+siz[d]$
的点对都会贡献1

接下来的问题就转化为二维区间加和单点求和，可以做二维差分，答案就是其左下角的差分值之和，将询问和修改都按x排序，这样顺着扫过来的时候用一个一维树状数组就可以维护y坐标上的前缀和，于是树状数组上求前缀和就是答案

```
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
const int MAX=1e5+5;
struct node{
  int dx,dy,id;
  bool operator <(const node &x)const{
    return x.dx>dx;
  }
}h[MAX<<3],Q[MAX];
int cc;
int nxt[MAX<<1],to[MAX<<1],head[MAX],cnt;
inline void add(int x,int y){
  ++cnt;
  nxt[cnt]=head[x],to[cnt]=y,head[x]=cnt;
  ++cnt;
  nxt[cnt]=head[y],to[cnt]=x,head[y]=cnt;
}
int son[MAX],siz[MAX],fa[MAX],top[MAX],depth[MAX],dfsn[MAX],cnt_dfsn;
void dfs1(int now,int father){
  siz[now]=1;
  fa[now]=father;
  depth[now]=depth[father]+1;
  for(int aim,i=head[now];i;i=nxt[i]){
    aim=to[i];
    if(aim!=father){
      dfs1(aim,now);
      siz[now]+=siz[aim];
      if(siz[son[now]]<siz[aim])son[now]=aim;
    }
  }
}
void dfs2(int now,int start){
  top[now]=start,dfsn[now]=++cnt_dfsn;
  if(son[now])dfs2(son[now],start);
  for(int aim,i=head[now];i;i=nxt[i]){
    aim=to[i];
    if(aim!=fa[now] && aim!=son[now]){
```

```
      dfs2(aim,aim);
    }
  }
}
inline int get(int x,int y){
  int ret;
  while(top[x]!=top[y]){
    if(depth[top[x]]<depth[top[y]])swap(x,y);
    ret=top[x];
    x=fa[top[x]];
  }
  if(depth[x]<depth[y])swap(x,y);
  if(x!=y)ret=son[y];
  return ret;
}
int n,m,q,ans[MAX];
int tree[MAX];
inline int lowbit(int x){return x&(-x);}
inline void modify(int x,int y){for(;x<=n;x+=lowbit(x))tree[x]+=y;}
inline int query(int x){
  int ret=0;
  for(;x;x-=lowbit(x))ret+=tree[x];
  return ret;
}
int main(){
  scanf("%d %d %d",&n,&m,&q);
  for(int x,y,i=1;i<n;i++)scanf("%d %d",&x,&y),add(x,y);
  dfs1(1,1),dfs2(1,1);
  for(int u,v,i=1;i<=m;i++){
    scanf("%d %d",&u,&v);
    if(dfsn[u]>dfsn[v])swap(u,v);
    if(dfsn[u]+siz[u]<=dfsn[v]){
      h[++cc]=node{dfsn[u],dfsn[v],1};
      h[++cc]=node{dfsn[u],dfsn[v]+siz[v],-1};
      h[++cc]=node{dfsn[u]+siz[u],dfsn[v],-1};
      h[++cc]=node{dfsn[u]+siz[u],dfsn[v]+siz[v],1};
    }else{
      int d=get(u,v);
      h[++cc]=node{1,dfsn[v],1};
      h[++cc]=node{1,dfsn[v]+siz[v],-1};
      h[++cc]=node{dfsn[d],dfsn[v],-1};
      h[++cc]=node{dfsn[d],dfsn[v]+siz[v],1};

      h[++cc]=node{dfsn[v],dfsn[d]+siz[d],1};
      h[++cc]=node{dfsn[v]+siz[v],dfsn[d]+siz[d],-1};
    }
  }
  sort(h+1,h+cc+1);
  for(int i=1;i<=q;i++){
    scanf("%d %d",&Q[i].dx,&Q[i].dy),Q[i].id=i;
    if(dfsn[Q[i].dx]>dfsn[Q[i].dy])swap(Q[i].dx,Q[i].dy);
    Q[i].dx=dfsn[Q[i].dx],Q[i].dy=dfsn[Q[i].dy];
  }
  sort(Q+1,Q+q+1);
  int now=0;
  for(int i=1;i<=q;i++){
    while(now+1<=cc && h[now+1].dx<=Q[i].dx)now++,modify(h[now].dy,h[now].id);
    ans[Q[i].id]=query(Q[i].dy);
```

```
  }
  for(int i=1;i<=q;i++)printf("%d\n",ans[i]);
  return 0;
}
```