树上莫队

引入

树上莫队看名字就知道，其实是把莫队搬到了树上。一般来说，这种问题有几个特征：

询问可以离线

答案并不好用树形DP或者数据结构维护，往往只能暴力跑到所有的点求。

我们拿一道例题：

SP10707 COT2

给定一个n个节点的树，每个节点表示一个整数，问u到v的路径上有多少个不同的整数。

这样的话，我们很容易就想到需要用莫队。可是莫队算法是基于一个序列的，我们怎么在树上跑莫队呢？

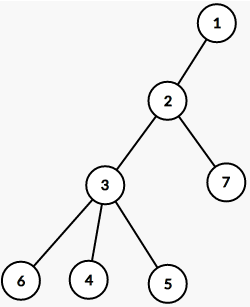
把树整成一个序列不就好了。一般来说，有dfs序和欧拉序两种情况。

什么？你问我什么是欧拉序？

欧拉序

操作：当访问到一个点的时候，把它加进序列。当离开这个点的时候，把它加进序列。

举个栗子：



这棵树的欧拉序就是：1,2,3,6,6,4,4,5,5,3,7,7,2,1

树上莫队

有了这个序列有什么用？

就拿刚才那道题来说。题目是给你m组询问，每个询问给定一个x，y，问在x~y的这条路径上，所有经过的点的权值中，本质不同的数有多少个。

那么假如我们给了一个x，y，该怎么办？

分类讨论：

如果lca等于x，那么路径上的点肯定在x的子树里，并且y的子树里的点一定没有用。所以我们要找的点在s【x】~s【y】之间（s【x】表示x节点在序列里第一次出现的位置，t【x】即第二次出现的位置）。

如果lca不等于x，那么路径上的点肯定不在x和y的子树里。所以我们要找的点在t【x】~s【y】之间。当然别忘了特判lca。

但是还有一个问题就是，这个区间序列内仍会有无用点，不过无用点一定出现了两次，所以差分一下就好了。

至此，我们已经解决了问题。

Coding

#include<bits/stdc++.h>

#define rep(i,a,b) for(int i=a;i<=b;++i)

#define per(i,a,b) for(int i=a;i>=b;--i)

using namespace std;

const int N=4e5+10;

struct node{

int l,r,id,add,ans;

}q[N];

int n,m,tot,num,blo,ans,cnt[N],dep[N],v[N],s[N\*2],t[N\*2],pot[N\*2],fa[N][25],ver[N],Next[N],lin[N],a[N],b[N];

void discrete(){

sort(b+1,b+n+1);

for(int i=1;i<=n;++i) a[i]=lower\_bound(b+1,b+n+1,a[i])-b;

}

void dfs(int x,int f){

fa[x][0]=f,s[x]=++num,pot[num]=x;

rep(i,1,20)if((1<<i)<=dep[x])fa[x][i]=fa[fa[x][i-1]][i-1];

for(int i=lin[x];i;i=Next[i]){

int y=ver[i];

if(y==f)continue;

dep[y]=dep[x]+1;

dfs(y,x);

}t[x]=++num,pot[num]=x;

}

int lca(int x,int y){

if(dep[x]<dep[y])swap(x,y);

per(i,20,0)if((1<<i)<=dep[x]-dep[y])x=fa[x][i];

if(x==y)return x;

per(i,20,0)if(fa[x][i]!=fa[y][i]){

x=fa[x][i],y=fa[y][i];

}

return fa[x][0];

}

void add(int x,int y){ver[++tot]=y;Next[tot]=lin[x];lin[x]=tot;}

bool cmp(node a,node b){return (a.l/blo==b.l/blo&&a.r<b.r)||(a.l/blo<b.l/blo);}

bool CMP(node a,node b){return a.id<b.id;}

void upd(int x){x=a[x];++cnt[x];if(cnt[x]==1) ans++;}

void del(int x){x=a[x];--cnt[x];if(cnt[x]==0) ans--;}

void update(int x){(v[x]) ? del(x):upd(x);v[x]^=1;}

int main(){

scanf("%d%d",&n,&m);blo=sqrt(n);

rep(i,1,n)scanf("%d",&a[i]),b[i]=a[i];

sort(b+1,b+n+1);

rep(i,1,n)a[i]=lower\_bound(b+1,b+n+1,a[i])-b;

int x,y;

rep(i,2,n){

scanf("%d%d",&x,&y);

add(x,y),add(y,x);

}

dfs(1,1);

rep(i,1,m){

scanf("%d%d",&x,&y);

int top=lca(x,y);

if(s[x]>s[y])swap(x,y);

if(x==top)q[i].l=s[x],q[i].r=s[y],q[i].id=i;

else q[i].l=t[x],q[i].r=s[y],q[i].id=i,q[i].add=top;

}

sort(q+1,q+m+1,cmp);

int l=1,r=0;

for(int i=1;i<=m;++i){

while(l<q[i].l) update(pot[l++]);

while(l>q[i].l) update(pot[--l]);

while(r<q[i].r) update(pot[++r]);

while(r>q[i].r) update(pot[r--]);

if(q[i].add) update(q[i].add);

q[i].ans=ans;

if(q[i].add) update(q[i].add);

}

sort(q+1,q+m+1,CMP);

for(int i=1;i<=m;++i) printf("%d\n",q[i].ans);

return 0;

}