树上莫队算法

继续回来写博客……记录点有意思的题目什么的。

貌似写过这个的没多少人……所以我也记录一点。

首先序列上的莫队大家都应该很熟悉了……

那么树上的莫队要怎么搞呢？

先来看个题目……SPOJ COT2：求树上两点间路径上有多少个不同的点权。

序列上的莫队是把询问按照左端点分块了……可是树上没有左端点，怎么办呢？我们把树分块。

按照DFS时间戳顺序，将树分成O(sqrt(n))个大小为O(sqrt(n))的块，那么树上的莫队询问排序的第一关键字就是第一个节点所在的块了！

这样分块以后，任意两个块之间的距离也是O(sqrt(n))级别的，所以时间复杂度是有保证的。

第二个关键字自然就是节点的DFS时间戳了！

但是，还有一个问题。树上的区间要怎么转移呢？要怎么从一个区间变到另一个区间呢？

这就有些难了，因为树上有LCA，貌似不好处理。

Orz了wyfcyx后，找到了vfk的博客看了一下。

“

用S(v, u)代表 v到u的路径上的结点的集合。
用root来代表根结点，用lca(v, u)来代表v、u的最近公共祖先。
那么
S(v, u) = S(root, v) xor S(root, u) xor lca(v, u)
其中xor是集合的对称差。
简单来说就是节点出现两次消掉。
lca很讨厌，于是再定义
T(v, u) = S(root, v) xor S(root, u)

观察将curV移动到targetV前后T(curV, curU)变化：

T(curV, curU) = S(root, curV) xor S(root, curU)

T(targetV, curU) = S(root, targetV) xor S(root, curU)

取对称差：

T(curV, curU) xor T(targetV, curU)= (S(root, curV) xor S(root, curU)) xor (S(root, targetV) xor S(root, curU))

由于对称差的交换律、结合律：

T(curV, curU) xor T(targetV, curU)= S(root, curV) xorS(root, targetV)

两边同时xor T(curV, curU)：

T(targetV, curU)= T(curV, curU) xor S(root, curV) xor S(root, targetV)

发现最后两项很爽……哇哈哈

T(targetV, curU)= T(curV, curU) xor T(curV, targetV)

（有公式恐惧症的不要走啊 T_T）

也就是说，更新的时候，xor T(curV, targetV)就行了。

即，对curV到targetV路径(除开lca(curV, targetV))上的结点，将它们的存在性取反即可。

”

——vfk博客

这样就很好处理了，只要把LCA扔出去，考虑剩下的部分，转移一下就可以了。查答案的时候再把LCA那个点反过来，就能统计出答案了。

这样，类比序列上的莫队，我们对树上的询问也可以分块了，时间复杂度同样是O(nsqrt(n))。

BZOJ 3757貌似是同一个题，就贴这个代码吧……COT2没写……SPOJ上怕T……

```cpp
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<string>
#include<cstring>
#include<queue>
#include<vector>
#include<cmath>
using namespace std;
inline int getint()
{
    char c=getchar();
    int con=0;
    while(c<'0'||c>'9') c=getchar();
    while(c>='0'&&c<='9') con=con*10+c-'0',c=getchar();
    return con;
}
const int MAXN=100010;
int n,m,K,lca,u,v,c[MAXN],dfn[MAXN],belongn[MAXN];
int tot,root,dfs_clock,remain;
int head[MAXN],to[MAXN],next[MAXN],cnt;
int anc[MAXN][21],dep[MAXN],Log[MAXN];
int Stack[MAXN],top;
int p[MAXN],ans,con[MAXN];
```

```
bool used[MAXN];
struct Query
{
    int u,v,a,b,sub;
    friend bool operator<(const Query &i,const Query &j)
    {
        if(belongn[i.u]==belongn[j.u]) return dfn[i.v]<dfn[j.v];
        else return belongn[i.u]<belongn[j.u];
    }
}Q[MAXN];
inline void adde(int f,int t)
{
    cnt++,to[cnt]=t,next[cnt]=head[f],head[f]=cnt;
    cnt++,to[cnt]=f,next[cnt]=head[t],head[t]=cnt;
}
int DFS(int x)
{
    int size=0;
    dfn[x]=++dfs_clock;
    for(int i=head[x];i;i=next[i])
        if(to[i]!=anc[x][0])
        {
            dep[to[i]]=dep[x]+1,anc[to[i]][0]=x;
            size+=DFS(to[i]);
            if(size>=K)
            {
                tot++;
                for(int i=1;i<=size;i++)
                    belongn[Stack[top--]]=tot;
                size=0;
            }
        }
    Stack[++top]=x;
    return size+1;
}
int LCA(int p,int q)
{
    if(dep[p]<dep[q]) swap(p,q);
    int d=dep[p]-dep[q];
    for(int i=Log[d];i>=0;i--)
        if(d&(1<<i)) p=anc[p][i];
    for(int i=Log[n];i>=0;i--)
        if(anc[p][i]!=anc[q][i]) p=anc[p][i],q=anc[q][i];
    if(p!=q) return anc[p][0];
    else return p;
}
void work(int u,int v,int lca)
{
    while(u!=lca)
    {
        if(!used[u]) {p[c[u]]++,used[u]=true;if(p[c[u]]==1) ans++;}
        else {p[c[u]]--,used[u]=false;if(p[c[u]]==0) ans--;}
        u=anc[u][0];
    }
    while(v!=lca)
    {
```

```
        if(!used[v]) {p[c[v]]++,used[v]=true;if(p[c[v]]==1) ans++;}
        else {p[c[v]]--,used[v]=false;if(p[c[v]]==0) ans--;}
        v=anc[v][0];
    }
}
int main()
{
    //freopen("apple.in","r",stdin);
    //freopen("apple.out","w",stdout);
    n=getint(),m=getint();
    K=(int)sqrt(n);
    for(int i=1;i<=n;i++) c[i]=getint();
    for(int i=1;i<=n;i++)
    {
        u=getint(),v=getint();
        if(u==0) root=v;
        else if(v==0) root=u;
        else adde(u,v);
    }
    for(int i=1;i<=m;i++)
    {
        Q[i].u=getint(),Q[i].v=getint();
        Q[i].a=getint(),Q[i].b=getint();
        Q[i].sub=i;
    }
    remain=DFS(root);
    for(int i=1;i<=remain;i++) belongn[Stack[top--]]=tot;
    sort(Q+1,Q+m+1);
    Log[0]=-1;
    for(int i=1;i<=n;i++) Log[i]=Log[i>>1]+1;
    for(int i=1;i<=Log[n];i++)
        for(int j=1;j<=n;j++)
            anc[j][i]=anc[anc[j][i-1]][i-1];
    work(Q[1].u,Q[1].v,lca=LCA(Q[1].u,Q[1].v));
    if(!used[lca]) {p[c[lca]]++,used[lca]=true;if(p[c[lca]]==1) ans++;}
    else {p[c[lca]]--,used[lca]=false;if(p[c[lca]]==0) ans--;}
    con[Q[1].sub]=ans;
    if(p[Q[1].a]!=0&&p[Q[1].b]!=0) con[Q[1].sub]--;
    if(!used[lca]) {p[c[lca]]++,used[lca]=true;if(p[c[lca]]==1) ans++;}
    else {p[c[lca]]--,used[lca]=false;if(p[c[lca]]==0) ans--;}
    for(int i=2;i<=m;i++)
    {
        work(Q[i-1].u,Q[i].u,LCA(Q[i-1].u,Q[i].u));
        work(Q[i-1].v,Q[i].v,LCA(Q[i-1].v,Q[i].v));
        lca=LCA(Q[i].u,Q[i].v);
        if(!used[lca]) {p[c[lca]]++,used[lca]=true;if(p[c[lca]]==1) ans++;}
        else {p[c[lca]]--,used[lca]=false;if(p[c[lca]]==0) ans--;}
        con[Q[i].sub]=ans;
        if(p[Q[i].a]!=0&&p[Q[i].b]!=0&&Q[i].a!=Q[i].b) con[Q[i].sub]--;
        if(!used[lca]) {p[c[lca]]++,used[lca]=true;if(p[c[lca]]==1) ans++;}
        else {p[c[lca]]--,used[lca]=false;if(p[c[lca]]==0) ans--;}
    }
    for(int i=1;i<=m;i++) printf("%d\n",con[i]);
    return 0;
}
```

感觉好神啊……vfk的那个方法确实好用，这个算法也好有趣。

（未完待续……）

WC2013 糖果公园 待填坑

```cpp
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<string>
#include<cstring>
#include<queue>
#include<vector>
#include<cmath>
using namespace std;
typedef long long LL;
const int MAXN=300010;
int n,m,u,v,Q,K,Type,C[MAXN],Ctmp[MAXN];
LL V[MAXN],W[MAXN],ans,con[MAXN];
int head[MAXN],to[MAXN],next[MAXN],cnt;
int belongn[MAXN],tot,remain;
int anc[MAXN][20],dep[MAXN],Log[MAXN];
int Stack[MAXN],top;
int Qtot,Ctot,Query_clock=1;
int p[MAXN],lca;
bool used[MAXN];
struct QQ
{
    int x,y,t,id;
    friend bool operator<(const QQ &i,const QQ &j)
    {
        if(belongn[i.x]<belongn[j.x]) return true;
        else if(belongn[i.x]==belongn[j.x]&&belongn[i.y]<belongn[j.y]) return true;
        else if(belongn[i.x]==belongn[j.x]&&belongn[i.y]==belongn[j.y]&&i.t<j.t) return true;
        return false;
    }
}Query[MAXN];
struct CC{int pos,x,y;}Change[MAXN];
inline void adde(int f,int t)
{
    cnt++,to[cnt]=t,next[cnt]=head[f],head[f]=cnt;
    cnt++,to[cnt]=f,next[cnt]=head[t],head[t]=cnt;
}
int DFS(int x)
{
    int size=0;
    for(int i=head[x];i;i=next[i])
        if(to[i]!=anc[x][0])
        {
            anc[to[i]][0]=x;
            dep[to[i]]=dep[x]+1;
            size+=DFS(to[i]);
            if(size>=K)
            {
```

```
                tot++;
                for(int i=1;i<=size;i++) belongn[Stack[top--]]=tot;
                size=0;
            }
        }
    Stack[++top]=x;
    return size+1;
}
int LCA(int p,int q)
{
    if(dep[p]<dep[q]) swap(p,q);
    int d=dep[p]-dep[q];
    for(int i=Log[d];i>=0;i--)
        if(d&(1<<i)) p=anc[p][i];
    for(int i=Log[n];i>=0;i--)
        if(anc[p][i]!=anc[q][i]) p=anc[p][i],q=anc[q][i];
    if(p!=q) return anc[p][0];
    return p;
}
void Reverse(int x)
{
    if(!used[x]) p[C[x]]++,ans+=V[C[x]]*W[p[C[x]]];
    else ans-=V[C[x]]*W[p[C[x]]],p[C[x]]--;
    used[x]^=1;
}
void ChangeVer(int x,int d)
{
    int pos=Change[x].pos;
    if(d==1)
    {
        if(used[pos]) Reverse(pos),C[pos]=Change[x].y,Reverse(pos);
        else C[pos]=Change[x].y;
    }
    else
    {
        if(used[pos]) Reverse(pos),C[pos]=Change[x].x,Reverse(pos);
        else C[pos]=Change[x].x;
    }
}
void work(int x,int y,int lca)
{
    while(x!=lca) Reverse(x),x=anc[x][0];
    while(y!=lca) Reverse(y),y=anc[y][0];
}
int main()
{
    scanf("%d%d%d",&n,&m,&Q);
    K=(int)pow((double)n,2.0/3);
    for(int i=1;i<=m;i++) scanf("%lld",&V[i]);
    for(int i=1;i<=n;i++) scanf("%lld",&W[i]);
    for(int i=1;i<n;i++) scanf("%d%d",&u,&v),adde(u,v);
    for(int i=1;i<=n;i++) scanf("%d",&C[i]),Ctmp[i]=C[i];
    for(int i=1;i<=Q;i++)
    {
        scanf("%d%d%d",&Type,&u,&v);
        if(Type==0)
```

```
        {
            Ctot++,Query_clock++;
            Change[Ctot].pos=u,Change[Ctot].x=Ctmp[u],Change[Ctot].y=v;
            Ctmp[u]=v;
        }
        else
        {
            Qtot++;
            Query[Qtot].x=u,Query[Qtot].y=v,Query[Qtot].t=Query_clock;
            Query[Qtot].id=Qtot;
        }
    }
    remain=DFS(1);
    for(int i=1;i<=remain;i++) belongn[Stack[top--]]=tot;
    sort(Query+1,Query+Qtot+1);
    Log[0]=-1;
    for(int i=1;i<=n;i++) Log[i]=Log[i>>1]+1;
    for(int i=1;i<=Log[n];i++)
        for(int j=1;j<=n;j++)
            anc[j][i]=anc[anc[j][i-1]][i-1];
    work(Query[1].x,Query[1].y,lca=LCA(Query[1].x,Query[1].y));
    for(int i=1;i<Query[1].t;i++) ChangeVer(i,1);
    Reverse(lca),con[Query[1].id]=ans,Reverse(lca);
    for(int i=2;i<=Qtot;i++)
    {
        work(Query[i-1].x,Query[i].x,LCA(Query[i-1].x,Query[i].x));
        work(Query[i-1].y,Query[i].y,LCA(Query[i-1].y,Query[i].y));
        for(int j=Query[i-1].t;j<Query[i].t;j++) ChangeVer(j,1);
        for(int j=Query[i-1].t-1;j>=Query[i].t;j--) ChangeVer(j,-1);
        lca=LCA(Query[i].x,Query[i].y);
        Reverse(lca),con[Query[i].id]=ans,Reverse(lca);
    }
    for(int i=1;i<=Qtot;i++) printf("%lld\n",con[i]);
    return 0;
}
```