树链剖分模板完整版

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=123458;
#define ll long long
vector<int>G[maxn];
ll tr[maxn*4];
ll tag[maxn*4];
int
n,m,a[maxn],cnt,f[maxn],d[maxn],siz[maxn],son[maxn],rk[maxn],t
op[maxn],tid[maxn];
void add_edge(int x,int y)
{
    G[x].push_back(y);
    G[y].push_back(x);
}
void dfs1(int u,int fa,int depth)
{
    f[u]=fa;
    d[u]=depth;
    siz[u]=1;
    for(int i=0; i<G[u].size(); i++)
    {
        int v=G[u][i];
        if(v==fa)
            continue;
        dfs1(v,u,depth+1);
        siz[u]+=siz[v];
        if(siz[v]>siz[son[u]])
            son[u]=v;
    }
}
void dfs2(int u,int t)
{
    top[u]=t;
    tid[u]=++cnt;
```

```
    rk[cnt]=u;
    if(!son[u])
        return;
    dfs2(son[u],t);
    for(int i=0; i<G[u].size(); i++)
    {
        int v=G[u][i];
        if(v!=son[u]&&v!=f[u])
            dfs2(v,v);
    }
}
void build(int o,int l,int r)
{
    if(l==r)
    {
        tr[o]=a[rk[l]];
        return;
    }
    int lson=o<<1,rson=lson|1;
    int m=(l+r)>>1;
    build(lson,l,m);
    build(rson,m+1,r);
    tr[o]=tr[lson]+tr[rson];
}
void update(int o,int l,int r,int ql,int qr,ll k)//区间修改:将
[l,r]区间每个数都加上k
{
    int lson=o<<1,rson=lson|1;
    int m=(l+r)>>1;
    if(ql<=l&&qr>=r)
    {
        tr[o]+=k*(r-l+1);
        tag[o]+=k;
        return;
    }
    if(tag[o])
```

```
73      {
74          tag[lson]+=tag[o];
75          tag[rson]+=tag[o];
76          tr[lson]+=tag[o]*(m-l+1);
77          tr[rson]+=tag[o]*(r-m);
78          tag[o]=0;
79      }
80      if(qr<=m)
81          update(lson,l,m,ql,qr,k);
82      else if(ql>m)
83          update(rson,m+1,r,ql,qr,k);
84      else
85      {
86          update(lson,l,m,ql,qr,k);
87          update(rson,m+1,r,ql,qr,k);
88      }
89      tr[o]=tr[lson]+tr[rson];
90  }
91  ll query_sum(int o,int l,int r,int ql,int qr)//区间查询
92  {
93      int lson=o<<1,rson=lson|1;
94      int m=(l+r)>>1;
95      if(ql<=l&&qr>=r)
96          return tr[o];
97      if(tag[o])
98      {
99          tag[lson]+=tag[o];
100         tag[rson]+=tag[o];
101         tr[lson]+=tag[o]*(m-l+1);
102         tr[rson]+=tag[o]*(r-m);
103         tag[o]=0;
104     }
105     if(qr<=m)
106         return query_sum(lson,l,m,ql,qr);
107     if(ql>m)
108         return query_sum(rson,m+1,r,ql,qr);
```

```
109     return
110 query_sum(lson,l,m,ql,qr)+query_sum(rson,m+1,r,ql,qr);
111 }
112 int LCA(int x,int y)
113 {
114     if(x==y)
115         return x;
116     int fx=top[x],fy=top[y];
117     while(fx!=fy)
118     {
119         if(d[fx]>=d[fy])
120         {
121             x=f[fx];
122         }
123         else
124         {
125             y=f[fy];
126         }
127         fx=top[x];
128         fy=top[y];
129     }
130     if(tid[x]<=tid[y]) return x;
131     else return y;
132 }
133 ll sum(int x,int y)
134 {
135     ll ans=0;
136     int fx=top[x],fy=top[y];
137     while(fx!=fy)
138     {
139         if(d[fx]>=d[fy])
140         {
141             ans+=query_sum(1,1,n,tid[fx],tid[x]);
142             x=f[fx];
143         }
144         else
```

```
145         {
146             ans+=query_sum(1,1,n,tid[fy],tid[y]);
147             y=f[fy];
148         }
149         fx=top[x];
150         fy=top[y];
151     }
152     /*
153     基于边权时:
154     if(tid[x]==tid[y])
155         return ans;
156     if(tid[x]<tid[y])
157         ans+=query_sum(1,1,n,tid[x]+1,tid[y]);
158     else
159         ans+=query_sum(1,1,n,tid[y]+1,tid[x]);
160     */
161
162     //基于点权:
163     if(tid[x]<=tid[y])
164         ans+=query_sum(1,1,n,tid[x],tid[y]);
165     else
166         ans+=query_sum(1,1,n,tid[y],tid[x]);
167     return ans;
168 }
169 void updates(int x,int y,ll c)
170 {
171     int fx=top[x],fy=top[y];
172     while(fx!=fy)
173     {
174         if(d[fx]>=d[fy])
175         {
176             update(1,1,n,tid[fx],tid[x],c);
177             x=f[fx];
178         }
179         else
180         {
```

```
            update(1,1,n,tid[fy],tid[y],c);
            y=f[fy];
        }
        fx=top[x];
        fy=top[y];
    }
    //基于边权时参考 sum 函数
    if(tid[x]<=tid[y])
        update(1,1,n,tid[x],tid[y],c);
    else
        update(1,1,n,tid[y],tid[x],c);
}
int main()
{
    cin>>n;
    for(int i=1; i<=n; i++)//基于点权时直接赋值,基于边权则将边权存
于子节点上
        cin>>a[i];
    for(int i=1; i<n; i++)
    {
        int x,y;
        cin>>x>>y;
        add_edge(x,y);
    }
    cnt=0;
    dfs1(1,0,1);
    dfs2(1,1);
    build(1,1,n);
    cin>>m;
    for(int i=1; i<=m; i++)
    {
        //操作
    }
    return 0;
}

```