消耗战

一. 考察内容：

虚树 树形DP

二. 题目分析：

[题目大意]

给出一个树，有一些点会有资源，割断一条边有一定花费，求出割断1节点到所有资源点的最小花费，其中有m个询问，每次资源点都会重新分配，总资源点数不超过5e5。

[写题思路]

这道题是虚树的模板题，虚树是一种优化建树的方法，将树上有用的点在较小时间复杂度内独立建出一棵树，从而快速维护信息的一种数据结构。

下面是虚树建树步骤：

1. 遍历题目中给出的树，求出每个节点的dfs序。
2. 对于需要建树的点，将他们按照他们的dfs序排序，很显然，dfs序小的节点一定在dfs序大的节点的左/上方（假设遍历时先遍历靠左边的儿子，再遍历靠右边的儿子）。
3. 用一个栈维护这些顶点，这个栈维护的是当前顶点的最右边一条链，首先将第一个顶点放入栈。
4. 设栈顶元素为a，考虑下一个顶点b，有两种可能：① 在栈顶元素的下方，② 在栈顶元素的右方，如果是情况①，lca(a,b)=a，则跳过这步，否则，我们为了满足“栈维护最右链”的特征，必须将深度大于lca(a,b)的所有节点都弹出，弹出时为弹出的节点和弹出后的栈顶元素连边建树（最后弹出的节点需要与lca(a,b)连边）。
5. 这时我们查看栈顶元素是否为lca(a,b)，如果是，则不执行操作，否则将lca(a,b)放入栈中。
6. 将b放入栈中，完成本次加点操作。
7. 循环4-6，直到遍历完所有有用的点。
8. 把栈中的所有点都弹出，弹出时相邻两个点连边建树。

完成虚树建树之后，在虚树上进行树形DP，对于每个节点，维护将该节点的子树与1号节点切断的最小花费，最后统计答案即可。

三. 代码实现：

```
#define _CRT_SECURE_NO_DEPRECATE
/*************************
*创建时间：2018 09 23
*文件类型：源代码文件
*题目来源：BZOJ
*当前状态：已通过
*备忘录：虚树 树形DP
*作者：HtBest
*************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
#include <queue>
#include <bitset>
// #include <sys/wait.h>
```

```
// #include <sys/types.h>
// #include <unistd.h>
using namespace std;
#define MAXN 250001
int n,m,head[MAXN],_edge,deep[MAXN],dfsx[MAXN],_dfsx,d[MAXN][20],minv[MAXN][20],
fa[MAXN],fav[MAXN],son[MAXN],bro[MAXN],node[MAXN],
sta[MAXN],top,num,last[MAXN],_last,nowlast,flag[MAXN];
long long f[MAXN];
struct EDGE
{
    int a,b,v,next;
    EDGE(int a=0,int b=0,int v=0,int next=0):a(a),b(b),v(v),next(next){}
}edge[2*MAXN];
/* Variable explain:
n:节点数
m:边数
head[i]:i的第一条出边
edge[i]:邻接表
_edge:邻接表标记
deep[i]:i在树上的深度
dfsx[i]:i的dfs序
_dfsx:dfs序标记
d[i][j]:i向上跳j层所在的位置
minv[i][j]:i到d[i][j]的路径中权值最小的边
fa[i]:i节点的父亲
fav[i]:i节点到父亲的权值
son[i]:i节点的第一个儿子
bro[i]:i节点的下一个兄弟
node[i]:对当前有资源的节点进行排序
f[i]:阻断i节点的子树资源的最小花费
sta[i]:虚树建树用栈
top:sta的栈顶
num:当前有资源的节点个数
last[i]:i节点最后一次初始化的时间
_last:时间轴
nowlast:当前询问的初始时间
flag[i]:i节点最后一次有资源的询问时间
*/
void clear(int i)
{
    fa[i]=son[i]=bro[i]=fav[i]=0;
    last[i]=++_last;
}
void adde(int a,int b,int v)
{
    edge[++_edge]=EDGE(a,b,v,head[a]);
    head[a]=_edge;
}
void addt(int a,int b,int v)
{
//  printf("%d---%d %d\n",a,b,v);
    if(last[a]<=nowlast)clear(a);
    if(last[b]<=nowlast)clear(b);
    bro[b]=son[a];
    son[a]=b;
    fa[b]=a;
    fav[b]=v;
}
bool cmp(int a,int b)
{
    return dfsx[a]<dfsx[b];
}
void read()
{
    int ls1,ls2,ls3;
    scanf("%d",&n);
```

```
    for(int i=1;i<n;+
+i)scanf("%d%d%d",&ls1,&ls2,&ls3),adde(ls1,ls2,ls3),adde(ls2,ls1,ls3);
    return;
}
void dfs(int a,int c)
{
    deep[a]=c;
    dfsx[a]=++_dfsx;
    for(int i=head[a];i;i=edge[i].next)
    {
        int b=edge[i].b,v=edge[i].v;
        if(fa[a]==b)continue;
        addt(a,b,v);
        dfs(b,c+1);
    }
}
void init()
{
    for(int i=1;i<=n;++i)d[i][0]=fa[i],minv[i][0]=fav[i];
    d[1][0]=1;
    for(int j=1;j<=18;++j)
        for(int i=1;i<=n;++i)
            d[i][j]=d[d[i][j-1]][j-1],minv[i][j]=min(minv[i][j-1],minv[d[i][j-1]][j-1]);
}
int LCA(int a,int b,int &v)
{
    v=1e9;
    if(deep[a]>deep[b])swap(a,b);
    int h=deep[b]-deep[a];
    for(int i=0;h;++i)
    {
        if(h&1)v=min(v,minv[b][i]),b=d[b][i];
        h>>=1;
    }
    if(a==b)return a;
    for(int i=18;i>=0;--i)
    {
        if(d[a][i]!=d[b][i])
            v=min(v,min(minv[a][i],minv[b][i])),a=d[a][i],b=d[b][i];
    }
    return d[a][0];
}
void build()
{

    sort(node+1,node+1+num,cmp);
    sta[++top]=node[1];
    for(int i=2;i<=num;++i)
    {
        int ls1;
        int lca=LCA(sta[top],node[i],ls1);
        while(top>1&&deep[sta[top-1]]>deep[lca])
        {
            LCA(sta[top-1],sta[top],ls1);
            addt(sta[top-1],sta[top],ls1);
            --top;
        }
        if(sta[top]!=lca)
        {
            LCA(lca,sta[top],ls1);
            addt(lca,sta[top],ls1);
            --top;
        }
        if(sta[top]!=lca)sta[++top]=lca;
        sta[++top]=node[i];
    }
    int now=sta[top--];
    while(top)
    {
        int ls1;
        LCA(sta[top],now,ls1);
        addt(sta[top],now,ls1);
```

```
        now=sta[top--];
    }
}
void dfs2(int a)
{
    f[a]=fav[a];
    if(flag[a]==m+1)return;
    long long ans=0;
    for(int b=son[a];b;b=bro[b])
    {
        dfs2(b);
        ans+=f[b];
    }
    f[a]=min(f[a],ans);
}
int main()
{
    // freopen(".in","r",stdin);
    // freopen(".out","w",stdout);
    read();
    dfs(1,0);
    init();
//  for(int i=1;i<=n;++i)
//  {
//      for(int j=0;j<=4;++j)printf("%d",d[i][j]);
//      puts("");
//  }
    scanf("%d",&m);
    while(m--)
    {
        long long ans=0;
        nowlast=_last;
        scanf("%d",&num);
        for(int i=1;i<=num;++i)scanf("%d",&node[i]),flag[node[i]]=m+1;
        node[++num]=1;
        build();
        dfs2(1);
        for(int i=son[1];i;i=bro[i])ans+=f[i];
        printf("%lld\n",ans);
    }
    return 0;
}
```

<题目跳转> <查看代码>