疫情控制

一. 考察内容：

树上倍增 二分答案

二. 题目分析：

[写题思路]

这是一道图论综合题，首先考虑几个问题：

1. 如果一个军队从某个点往树根爬，则他原本控制的节点仍会被控制，则所有军队在有限时间内只管向上爬即可。
2. 一个军队只可能控制自己所在的根的一个子树，或者更帮其他根的子树控制疫情。
3. 如果一个军队帮其他子树控制疫情，则最优的方案就是爬到该子树的根节点。

现在我们考虑二分一个答案，然后进行判断；

1. 让所有军队都在时间够用的情况下尽可能向上爬，但是不要爬到首都。
2. 记录一下每个军队的用时，对于不在根的子节点的军队，剩余的用时已经不足以该军队获得更优的位置（因为步骤1时我们已经尽可能的往上爬了）。
3. 对于所有根的子树，我们可以遍历一下他被控制的次数，形象的说就是疫情从首都到叶子城市至少会碰到几次检查站。
4. 现在我们找出没有被控制的所有根节点的子节点，尝试调用其他子节点的军队来帮助该节点控制疫情。
5. 对军队按照已经走的路程+到根节点需要走的路程从小到大排序。
6. 用一个堆维护当前没有被控制疫情的节点，按照该节点到首都的距离从大到小排列，然后按照5的排序顺序遍历当前**在根节点的子节点的军队**，如果该军队在二分出的答案时间内可以到达需要帮助的节点，则让该军队去帮助需要帮助的节点。
7. 如果所有军队都调度完成之后，仍有不能控制疫情的节点，则该答案偏小。

最终输出二分的答案即可。

三. 代码实现：

```
#define _CRT_SECURE_NO_DEPRECATE
/************************
*创建时间：2018 08 24
*文件类型：源代码文件
*题目来源：洛谷
*当前状态：已通过
*备忘录：图论 树 二分答案 倍增法 树上倍增
         理清思路！！！
         记得初始化！！！
*作者：HtBest
************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
#include <queue>
// #include <sys/wait.h>
// #include <sys/types.h>
// #include <unistd.h>
```

```
using namespace std;
#define MAXN 50005
int n,m,a[MAXN],head[MAXN],_edge,dfsx[MAXN],_dfsx,
fa[MAXN],fav[MAXN],son[MAXN],bro[MAXN],f[MAXN][30],cover[MAXN];
long long d[MAXN][30];
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];
struct A
{
    int a;
    long long v;
    A(int a=0,long long v=0):a(a),v(v){}
    bool operator < (A b) const
    {
        return v+fav[a]<b.v+fav[b.a];
    }
}army[MAXN];
/* Variable explain:
n:节点数
m:军队数
a[i]:军队初始位置
head[i]:每个节点的第一条出边
edge[i]:邻接表
_edge:邻接表指针
dfsx[i]:dfs序
_dfsx:dfs标记
fa[i]:节点i的父节点
fav[i]:节点i距离父节点的路径
son[i]:节点i的第一个儿子
bro[i]:节点i的下一个兄弟
f[i][j]:节点i向上跳2^j层所在的节点
cover[i]:子树i被军队覆盖的次数
d[i][j]:节点i到节点f[i][j]之间的距离
army[i]:第i个军队爬树之后的信息
*/
void adde(int a,int b,int v)//建图
{
    edge[++_edge]=EDGE(a,b,v,head[a]);
    head[a]=_edge;
}
void addt(int a,int b,int v)//建树
{
    bro[b]=son[a];
    son[a]=b;
    fa[b]=a;
    fav[b]=v;
}
void read()
{
    int ls1,ls2,ls3;
    freopen("blockade.in","r",stdin);
    freopen("blockade.out","w",stdout);
    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);
    scanf("%d",&m);
    for(int i=1;i<=m;++i)scanf("%d",&a[i]);
    return;
}
void init()//预处理倍增
{
    for(int i=1;i<=n;++i)f[i][0]=fa[i],d[i][0]=fav[i];
    for(int j=1;j<20;++j)
        for(int i=1;i<=n;++i)
```

```
            f[i][j]=f[f[i][j-1]][j-1],d[i][j]=d[i][j-1]+d[f[i][j-1]][j-1];
}
void dfs1(int a)//建树
{
    dfsx[a]=++_dfsx;
    // printf("%d\n",a);
    for(int i=head[a];i;i=edge[i].next)
    {
        int b=edge[i].b,v=edge[i].v;
        if(dfsx[b])continue;
        addt(a,b,v);
        dfs1(b);
    }
}
int dfs2(int a)//找每个子树的覆盖次数
{
    int T=0;
    for(int i=son[a];i;i=bro[i])
    {
        ++T;
        if(cover[i]||dfs2(i))--T;//这里只需要统计一次即可，因为多次覆盖得不到更好的结果
    }
    return cover[a]+(son[a]&&!T);
}
bool judge(long long x)
{
    priority_queue <A> q;
    for(int i=1;i<=n;++i)cover[i]=0;
    for(int i=1;i<=m;++i)//让所有军队尽可能爬树
    {
        army[i].a=a[i];//初始化军队位置
        army[i].v=0;
        for(int j=19;j>=0;--j)
        {
            if(army[i].v+d[army[i].a][j]<=x&&f[army[i].a][j]&&f[army[i].a][j]!=1)//时间允
许的情况下，不能爬到根，也不能爬超过根
            {
                army[i].v+=d[army[i].a][j];
                army[i].a=f[army[i].a][j];//爬树
            }
        }
        ++cover[army[i].a];//覆盖
        // printf("%d %lld\n",army[i].a,army[i].v);
    }
    for(int i=son[1];i;i=bro[i])
    {

        cover[i]=dfs2(i);
        if(!cover[i])q.push(A(i));
    }
    sort(army+1,army+1+m);
    int TT=1;
    while(!q.empty())
    {
        A now=q.top();//找到第一个没有被覆盖的村庄
        q.pop();
        for(;TT<=m;++TT)
        {
            if(fa[army[TT].a]!=1)continue;
            // printf("军队%d试图移动至%d\n移动花费：%d+%d,
%lld\n",army[TT].a,now.a,fav[now.a],fav[army[TT].a],army[TT].v);
            if((A(army[TT].a)<now||cover[army[TT].a]>1)&&fav[now.a]
+army[TT].v+fav[army[TT].a]<=x)
            {
                --cover[army[TT].a],++cover[now.a];
                if(!cover[army[TT].a])q.push(A(army[TT].a));
                ++TT;
                break;
            }
```

```
        }
    }
    for(int i=son[1];i;i=bro[i])if(!cover[i])return false;
    return true;
}
void solve()
{
    long long l=0,r=1e10,ans=-1;
    while(l<=r)
    {
        long long m=(l+r)>>1;
        if(judge(m))ans=m,r=m-1;
        else l=m+1;
    }
    printf("%lld\n",ans);
}
int main()
{
    read();
    dfs1(1);
    init();
    solve();
    // for(int i=son[1];i;i=bro[i])printf("%3d ",i);puts("");
    // for(int i=son[1];i;i=bro[i])printf("%3d ",cover[i]);puts("");
    return 0;
}
```

<题目跳转> <查看代码>