

题解

显然，题目直接做十分困难，但我们发现结果具有单调性，就联想到二分，再看到 $1 \leq n \leq 5e4$ ，很容易想到 $check()$ 的复杂度应该是 $O(n \log n)$ ，现在假设二分到阈值 x ，对于其中的一个军队，往上跳肯定比往下跳更优，于是我们维护这个军队能跳到的最高位置（这里的最高位置是 1 的儿子），然后我们简单的 dfs 一遍，便能搜出 1 节点的哪些儿子需要其他儿子的军队去驻扎，哪些儿子自己已经能防止疫情传播。

我们定义 1 节点到 u 节点的距离为 $dist[u]$ ，定义军队 i 的自由值为 $x - dist[u(\text{军队初始化所在节点})]$ 。对于 1 节点的一个能自己防止疫情传播的儿子 v ，如果其子树（不包含 v ）能防止疫情传播，我们把全部能到达 v 的军队的自由值放到集合 A 里，否则我们把能到达 v 的军队按自由值排序，然后取出其最小值（权值 val ）以及 v (id) 放在集合 B 里，把剩下的军队的自由值放到集合 A 。接着我们把集合 A 中的元素排序，对集合 B 中的每个元素，找到集合 A 中第一个大于 $dist[id]$ 的元素 u ，如果 $u < val$ 则将集合 A 中 u 删除，并加入 val 。（集合 A 可以用 set 维护，集合 B 可以用普通数组维护）。

处理完上述步骤，剩下的不能自己防止疫情传播的 v ，把 $dist[v]$ 放到 C 集合，我们就可以用双指针维护集合 A, C 的匹配（大对大，小对小），总复杂度 $O(n \log n \log A)$ 。

代码有点长。

```
#include<bits/stdc++.h>
#define int long long
#define pii pair<int,int>
using namespace std;
int read()
{
    char a=getchar();int res=0,f=1;
    while(!isdigit(a)) {if(a=='-') f=-1;a=getchar();}
    while(isdigit(a)) res=(res<<3)+(res<<1)+a-'0',a=getchar();
    return res*f;
}
const int KK=22;
int sta2[KK],tooot;
void write(long long a,bool l)
{
    if(a<0) putchar('-'),a=-a;
    do{sta2[++tooot]=a%10;a/=10;}while(a);
    while(tooot) {putchar(sta2[tooot]+'0');tooot--;}
    if(l) putchar(10);else putchar(32);
}
const int N=5e4+10,K=21;
int
toot,n,m,cnt,siz,tot,te[N],pre[N],value[N],dist[N],head[N],suf[N],army[N],top[N],
dep[N],fa[N][K];
vector<int> vis[N];
multiset<int> q;
multiset<int>::iterator it;
struct ppp{int to,ne,val;}edge[N<<1];
void dfs(int u,int f,int t)
{
    top[u]=t;
```

```

    dep[u]=dep[f]+1;
    fa[u][0]=f;
    for(int i=1;(1<<i)<=dep[u];i++) fa[u][i]=fa[fa[u][i-1]][i-1];
    for(int i=head[u];i;i=edge[i].ne)
    {
        int v=edge[i].to;
        if(v==f) continue;
        dist[v]=dist[u]+edge[i].val;
        dfs(v,u,t);
    }
}
bool dfs2(int u,int f)
{
    bool l=0,l2=1;
    for(int i=head[u];i;i=edge[i].ne)
    {
        int v=edge[i].to;
        if(v==f) continue;
        l=1;
        if(!dfs2(v,u)) l2=0;
    }
    if(!l||!l2)
    {
        if(vis[u].size())
        {
            if(f==1)
            {
                for(int i=1;i<vis[u].size();i++) q.insert(value[vis[u][i]]-
dist[u]);
                te[u]=value[vis[u][0]]-dist[u];
                //printf("%d %d\n\n",u,te[u]);
            }
            return 1;
        }
        return 0;
    }
    if(f==1) for(int i=0;i<vis[u].size();i++) q.insert(value[vis[u][i]]-
dist[u]);
    return 1;
}
bool cmp(int a,int b)
{
    return value[a]<value[b];
}
bool check(int x)
{
    //printf("%lld\n",x);
    for(int i=1;i<=n;i++) te[i]=0;
    for(int i=1;i<=n;i++) vis[i].clear();
    cnt=tot=0;
    for(int i=1;i<=m;i++)
    {
        int u=army[i],val=x;
        for(int j=20;j>=0;j--) if(dep[u]-(1<<j)>=dep[top[u]]&&dist[u]-dist[fa[u]
[j]]<=val) val-=dist[u]-dist[fa[u][j]],u=fa[u][j];
    }
}

```

```

        //printf("%d %d\n", army[i], u);
        value[i]=val;
        vis[u].push_back(i);
    }
    for(int i=1;i<=n;i++) sort(vis[i].begin(), vis[i].end(), cmp);
    for(int i=head[1]; i; i=edge[i].ne) if(!dfs2(edge[i].to, 1))
suf[++tot]=edge[i].val;
    for(int i=head[1]; i; i=edge[i].ne)
    {
        int v=edge[i].to;
        if(q.size()==0) break;
        it=q.lower_bound(edge[i].val);
        if(it!=q.end() && *it >= te[v])
        {
            //printf("%d\n", *it);
            q.erase(it);
            q.insert(te[v]);
        }
    }
    while(q.size()) pre[++cnt]=*q.begin(), q.erase(q.begin());
    sort(suf+1, suf+tot+1);
    //for(int i=1;i<=cnt;i++) printf("%d ", pre[i]); printf("\n");
    //for(int i=1;i<=tot;i++) printf("%d ", suf[i]); printf("\n");
    for(int i=1, j=1; i<=tot; i++)
    {
        while(suf[i]>pre[j] && j<=cnt) j++;
        if(j==cnt+1) return 0;
        j++;
    }
    return 1;
}
void add(int a, int b, int c)
{
    ++toot;
    edge[toot].ne=head[a];
    edge[toot].to=b;
    edge[toot].val=c;
    head[a]=toot;
}
signed main()
{
    //freopen("control.in", "r", stdin);
    //freopen("control.out", "w", stdout);
    n=read();
    for(int i=1; i<=n; i++)
    {
        int a=read(), b=read(), c=read();
        add(a, b, c);
        add(b, a, c);
    }
    for(int i=head[1]; i; i=edge[i].ne)
siz++, dist[edge[i].to]=edge[i].val, dfs(edge[i].to, 1, edge[i].to);
    //for(int i=1; i<=n; i++) printf("%d ", dist[i]); printf("\n");
    m=read();
    for(int i=1; i<=m; i++) army[i]=read();
}

```

```

    if(siz>m)
    {
        write(-1,1);
        return 0;
    }
    int l=0,r=5e14+10;
    while(l<r)
    {
        int mid=(l+r)>>1;
        if(check(mid)) r=mid;
        else l=mid+1;
    }
    write(l,1);
    return 0;
}
/*
17
1 2 4
2 3 3
2 4 2
4 5 7
4 6 3
4 9 6
6 7 2
6 8 4
1 10 5
10 15 2
15 16 6
15 17 7
1 11 10
11 12 3
11 13 4
13 14 5
3
8 6 17
*/

```