Kruskal 重构树

在运行$Kruskal$算法求最小生成树时，对于两个可以合并的节点(x,y),断开其中的连边，并新建一个结点T,把T向(x,y)连边并做它们的父亲，同时把(x,y)之间的边权当作T的点权。

性质

- 是一棵二叉树
- 按最小生成树建立的话是一个大根堆
- 任意两个点路径上边权的最大值为它们的LCA的点权

题目

2021ICPC 上海H. Life is a Game

给定一个带点权和边权的无向图$G(n,m)$和q次询问(x,k)，表示询问的起点和初始能力值，规定只能通过小于等于能力值的边，能力值为当前询问的k加上走过的点权值和。

启发式并查集，在合并过程中提前处理k值小的询问。

```
#include<bits/stdc++.h>
#define mk make_pair
#define pii pair<int,int>
#define int long long
#define F first
#define S second
using namespace std;
const int N=1e5+7;

struct Query{
    int x,k,id;
    bool operator <(const Query &b)const{
        return k>b.k;
    }
}s[N];

int fa[N];

int find(int x){
    return fa[x]==x?x:fa[x]=find(fa[x]);
}
```

```cpp
struct Edge{
    int u,v,w,nxt;
    bool operator <(const Edge &b)const{
        return w>b.w;
    }
}e[N<<1];
int cnt=0,head[N];
inline void addedge(int u,int v,int w){
    e[++cnt]=(Edge){u,v,w,head[u]};head[u]=cnt;
}
int n,m,q,a[N],vis[N],ans[N];
priority_queue<Query>qq[N]; //询问列表
bool cmp(Edge A,Edge B){return A.w<B.w;}

void kruskal(){
    stable_sort(e+1,e+1+cnt,cmp);
    int idx=1;
    for(int j=1;j<=cnt;++j){
        int u=e[j].u,v=e[j].v,w=e[j].w;
        int fx=find(u),fy=find(v);
        if(fx==fy) continue;
        while(qq[fx].size()){ //处理询问
            Query p=qq[fx].top();
            if(a[fx]+p.k<w) ans[p.id]=a[fx]+p.k; //不能继续更新答案了
            else break;
            qq[fx].pop();
        }
        while(qq[fy].size()){ //处理询问
            Query p=qq[fy].top();
            if(a[fy]+p.k<w) ans[p.id]=a[fy]+p.k; //不能继续更新答案了
            else break;
            qq[fy].pop();
        }
        if(qq[fx].size()>qq[fy].size()) swap(fx,fy);
        fa[fx]=fy;a[fy]+=a[fx];
        while(qq[fx].size()){
            qq[fy].emplace(qq[fx].top());
            qq[fx].pop();
        }
    }
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>m>>q;
    for(int i=1;i<=n;++i) fa[i]=i;
    for(int i=1;i<=n;++i) cin>>a[i];
    for(int i=1,u,v,w;i<=m;++i){
        cin>>u>>v>>w;
        addedge(u,v,w);
        addedge(v,u,w);
    }
    for(int i=1;i<=q;++i) cin>>s[i].x>>s[i].k,s[i].id=i;
    for(int i=1;i<=q;++i) qq[s[i].x].emplace(s[i]);
    kruskal();
    int ff=find(1);
```

```
    while(qq[ff].size()){ //处理询问
        Query p=qq[ff].top();
        qq[ff].pop();
        ans[p.id]=a[ff]+p.k;
    }
    for(int i=1;i<=q;++i) cout<<ans[i]<<"\n";
    return 0;
}
/*
5 4 5
1 2 3 4 5
1 2 3
2 4 4
2 3 7
3 5 15
1 1
1 2
1 7
5 3
5 15
*/
```

Kruskal重构树：根据性质可以直接建树倍增求每个询问的答案

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5+5;
const int M = 21;
typedef long long ll;
const ll inf = 1e18;
vector<int>G[N];

int n,m,q,fa[N],tot=0,b[N],fa2[N][M];
ll a[N],mi[N][M];
struct Node{
    int u,v,w;
    bool operator <(const Node a)const{
        return w<a.w;
    }
}e[N<<1];

int find(int rt){
    return rt==fa[rt]?rt:(fa[rt]=find(fa[rt]));
}

void dfs1(int u,int f){ //处理父亲和子树和信息
    fa2[u][0]=f;
    for(int i=1;i<M;++i) fa2[u][i]=fa2[fa2[u][i-1]][i-1];
    for(auto v:G[u]){
        dfs1(v,u);
        a[u]+=a[v];
    }
}
void dfs2(int u,int f){  //处理u向上跳每一倍的最少能力值
    if(f){
```

```
        mi[u][0]=b[f];
        for(int i=1;i<M;++i){
            int f=fa2[u][i-1];
            if(!f){
                mi[u][i]=inf;
                continue;
            }
            ll now=a[f]-a[u]+mi[u][i-1];
            if(now<mi[fa2[u][i-1]][i-1]){
                mi[u][i]=mi[u][i-1]+mi[fa2[u][i-1]][i-1]-now;
            }else mi[u][i]=mi[u][i-1];
        }
    }else for(int i=0;i<M;++i) mi[u][i]=inf;
    for(auto v:G[u]) dfs2(v,u);
}

int gettop(int u,ll val){  //倍增向上处理询问
    for(int i=M-1;i>=0;--i){
        if(mi[u][i]<=val){
            int v=fa2[u][i];
            val+=a[v]-a[u];
            u=v;
        }
    }
    return u;
}

void Kruskal(){ //重构树
    sort(e,e+m);
    for(int i=0;i<m;++i){
        int u=e[i].u,v=e[i].v,w=e[i].w;
        int fu=find(u),fv=find(v);
        if(fu==fv) continue;
        fa[fu]=fa[fv]=++tot;
        fa[tot]=tot;
        b[tot]=w;
        G[tot].push_back(fu);
        G[tot].push_back(fv);
    }
}

int main(){
    scanf("%d%d%d",&n,&m,&q);
    for(int i=1;i<=n;++i) fa[i]=i,scanf("%lld",&a[i]);
    tot=n;
    for(int i=0;i<m;++i) scanf("%d%d%d",&e[i].u,&e[i].v,&e[i].w);
    Kruskal();
    dfs1(tot,0);
    dfs2(tot,0);
    while(q--){
        int u,k;
        scanf("%d%d",&u,&k);
        int v=gettop(u,k+a[u]);
        printf("%lld\n",a[v]+k);
    }
}
```

[NOI2018] 归程

有N座山峰，每座山峰有他的高度h_i。有些山峰之间有双向道路相连，共M条路径，每条路径有长度和海拔，海拔越大表示越难走.现在有Q组询问，每组询问从点v开始只经过海拔小于等于x的路径所能到达的山峰中第k远的山峰，如果无解输出-1。

```
#include<bits/stdc++.h>
#define mk make_pair
#define pii pair<int,int>
#define F first
#define S second
using namespace std;
const int N=1e6+7;
const int inf=0x3f3f3f3f,M=20;

int n,m,q,k,s,tot;

struct Edge{
    int u,v,l,a,nxt;
    bool operator <(const Edge &B)const{
        return a>B.a;
    }
}e[N<<1];
int head[N],cnt=0;
inline void addedge(int u,int v,int w,int f){
    e[++cnt]=(Edge){u,v,w,f,head[u]};head[u]=cnt;
}

int dis[N],vis[N];
void dij(int ss){  //dijkstra堆优化求最短路
    for(int i=1;i<=n;++i) vis[i]=0,dis[i]=inf;
    priority_queue<pii>q;
    dis[ss]=0;q.emplace(mk(0,ss));
    while(q.size()){
        int fro=q.top().S;
        q.pop();
        if(vis[fro]) continue;
        vis[fro]=1;
        for(int i=head[fro];i;i=e[i].nxt){
            int to=e[i].v,w=e[i].l;
            if(dis[to]>dis[fro]+w){
                dis[to]=dis[fro]+w;
                q.emplace(mk(-dis[to],to));
            }
        }
    }
}

int fa[N],f[N][25],g[N][25],ch[N][2],mi[N];
inline int find(int x){
    return (fa[x]==x)?x:fa[x]=find(fa[x]);
}

void Kruskal(){
    tot=n;
```

```
    for(int i=0;i<=2*n+1;++i) fa[i]=i;
    stable_sort(e+1,e+1+cnt);
    for(int i=1;i<=cnt;++i){
        int fx=find(e[i].u),fy=find(e[i].v);
        if(fx==fy) continue;
        ++tot;
        ch[tot][0]=fx;ch[tot][1]=fy;
        fa[fx]=fa[fy]=f[fx][0]=f[fy][0]=tot;
        g[fx][0]=g[fy][0]=e[i].a;
    }
}

int dfs(int x){
    mi[x]=inf;
    if(!ch[x][0]||!ch[x][1]) return mi[x]=dis[x];
    mi[x]=min(mi[x],dfs(ch[x][0]));
    mi[x]=min(mi[x],dfs(ch[x][1]));
    return mi[x];
}

int gettop(int bg,int val){
    for(int i=M;i>=0;--i){
        if(g[bg][i]>val&&g[bg][i]<=inf) bg=f[bg][i];
    }
    return bg;
}

void clear(){
    memset(g,0x3f,sizeof(g));
    memset(head,0,sizeof(head));
    memset(ch,0,sizeof(ch));
    memset(f,0,sizeof(f));
    cnt=0;
}

void solve(){
    cin>>n>>m;
    for(int i=1,u,v,l,a;i<=m;++i){
        cin>>u>>v>>l>>a;
        addedge(u,v,l,a);
        addedge(v,u,l,a);
    }
    dij(1);
    Kruskal();
    for(int j=1;j<=M;++j){
        for(int i=1;i<=2*n;++i){
            f[i][j]=f[f[i][j-1]][j-1];
            g[i][j]=min(g[i][j-1],g[f[i][j-1]][j-1]);
        }
    }
    dfs(tot);
    cin>>q>>k>>s;
    for(int i=1,lst=0,v,p;i<=q;++i){
        cin>>v>>p;
        v=(v+k*lst-1)%n+1;p=(p+k*lst)%(s+1);
        int pos=gettop(v,p);
        cout<<(lst=mi[pos])<<"\n";
    }
```

```
    clear();
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    int T=1;
    cin>>T;
    while(T--){
        solve();
    }
    return 0;
}
```