2017-2018 ACM-ICPC Northern Eurasia Contest (NEERC 17)

比赛链接

A. Archery Tournament

题意

按时间顺序给定 n 个操作。

1. 操作 1 表示在坐标 (x,y) 上放置一个半径为 y 的靶子。
2. 操作 2 表示在对坐标 (x,y) 进行一次射击，如果命中某个靶子(命中边界不算命中)，则将这个靶子移去。

输出每次射击命中的靶子的编号，保证任何时刻图中现有的靶子不重叠。

题解

不难发现，对每次射击询问 (x,y)靶子被命中的必要条件是与直线 $x=x_i$ 相交。

又已知任何时刻图中现有的靶子不重叠，于是不难得出结论任何时刻与直线 $x=x_i$ 相交的圆的个数仅有 $\log v$ 个。

于是线段树维护 x 轴上的所有圆的投影即可，时间复杂度 $O(n\log^2 v)$

```
struct cyc{
    int x,y,id;
    bool operator < (const cyc &b)const{
        return id<b.id;
    }
    bool check_in(const pair<int,int> &q)const{
        return 1LL*(x-q.first)*(x-q.first)+1LL*(y-q.second)*(y-
q.second)<1LL*y*y;
    }
};
const int MAXN=2e5+5,MAXM=MAXN*30,MAXV=1e9;
int rt,ls[MAXM],rs[MAXM],node_cnt;
set<cyc> s[MAXM];
cyc cycs[MAXN];
void update(int &k,int nl,int nr,int ql,int qr,cyc c,bool del){
    if(!k)k=++node_cnt;
    if(ql<=nl&&nr<=qr){
        if(del)
        s[k].erase(c);
```

```
        else
        s[k].insert(c);
        return;
    }
    int nm=nl+nr>>1;
    if(nm>=ql)
    update(ls[k],nl,nm,ql,qr,c,del);
    if(nm<qr)
    update(rs[k],nm+1,nr,ql,qr,c,del);
}
int query(int k,int nl,int nr,int pos,pair<int,int> q){
    if(!k)return -1;
    for(set<cyc>::iterator it=s[k].begin();it!=s[k].end();it++){
        if(it->check_in(q))
        return it->id;
    }
    if(nl==nr)return -1;
    int nm=nl+nr>>1;
    if(nm>=pos)
    return query(ls[k],nl,nm,pos,q);
    else
    return query(rs[k],nm+1,nr,pos,q);
}
int main()
{
    int n=read_int();
    _rep(i,1,n){
        int t=read_int(),x=read_int(),y=read_int();
        if(t==1){
            cycs[i]=cyc{x,y,i};
            update(rt,-MAXV,MAXV,max(-MAXV,x-
y),min(MAXV,x+y),cycs[i],false);
        }
        else{
            int ans=query(rt,-MAXV,MAXV,x,make_pair(x,y));
            enter(ans);
            if(ans!=-1)
            update(rt,-MAXV,MAXV,max(-MAXV,cycs[ans].x-
cycs[ans].y),min(MAXV,cycs[ans].x+cycs[ans].y),cycs[ans],true);
        }
    }
    return 0;
}
```

J. Journey from Petersburg to Moscow

题意

给定边权图，求从点 1 到点 n 的最短路。其中如果路径边数超过 k 则仅最大的 k 条边对路径长度产生贡献。

题解

设路径边权从大到小依次为 $c_1,c_2\cdots c_k,c_{k+1}\cdots c_t$，设新路径长度函数为 $f(x)=kx+\sum_{i=1}^t \max(c_i-x,0)$。

不难发现该函数在 $x\in [c_{k+1},c_k]$ 时取到最小值，且此时函数值恰好等于最大的 k 条边对路径长度产生贡献。

于是不妨枚举所有 c_k，对边权做变换 $w\to w-c_k$，然后跑最短路，求所有结果的最小值。

另外需要考虑路径长度小于 k 的情况，直接最短路算法即可，为了减少分类讨论也可以令此时 $c_k=0$，总时间复杂度 $O(m^2\log m)$。

```
const int MAXN=3005,MAXM=3005;
const LL inf=1e15;
struct Edge{
    int to,w,next;
}edge[MAXM<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v,int w){
    edge[++edge_cnt]=Edge{v,w,head[u]};
    head[u]=edge_cnt;
}
struct{
    int u,v,w;
}edge2[MAXM];
int ww[MAXM];
struct Node{
    LL dis;
    int u;
    bool operator < (const Node &b)const{
        return dis > b.dis;
    }
};
LL dis[MAXN];
bool vis[MAXN];
LL solve(int n,int m,int k,int minw){
    _rep(i,1,n)head[i]=0,dis[i]=inf,vis[i]=false;
    edge_cnt=0;
    _for(i,0,m){
        Insert(edge2[i].u,edge2[i].v,max(edge2[i].w-minw,0));
        Insert(edge2[i].v,edge2[i].u,max(edge2[i].w-minw,0));
    }
    priority_queue<Node> q;
```

```
    dis[1]=0;
    q.push(Node{dis[1],1});
    while(!q.empty()){
        int u=q.top().u;q.pop();
        if(vis[u])continue;
        vis[u]=true;
        for(int i=head[u];i;i=edge[i].next){
            int v=edge[i].to;
            if(dis[u]+edge[i].w<dis[v]){
                dis[v]=dis[u]+edge[i].w;
                q.push(Node{dis[v],v});
            }
        }
    }
    return dis[n]+1LL*k*minw;
}
int main()
{
    int n=read_int(),m=read_int(),k=read_int();
    _for(i,0,m){
        edge2[i].u=read_int();
        edge2[i].v=read_int();
        ww[i]=edge2[i].w=read_int();
    }
    sort(ww,ww+m);
    int m2=unique(ww,ww+m)-ww;
    LL ans=inf;
    _for(i,0,m2)
    ans=min(ans,solve(n,m,k,ww[i]));
    enter(min(ans,solve(n,m,k,0)));
    return 0;
}
```