wqs二分

一种用于解决恰好选 a 个物品的最优方案的算法。

其中设 $F(x)$ 表示 $a=x$ 的最佳收益函数，则 $F(x)$ 必须是凸函数。

算法实现

由于 $F(x)$ 为凸函数，所以对固定的斜率 k 求出斜线与 $F(x)$ 构成的凸包的切点，当斜率单调变化时切点位置也是单调变化的。

于是可以二分找到切点位于 $x=a$ 的斜率然后计算该点答案。接下来考虑对固定的 k 如何计算切点 x 以及切点对应的 $F(x)$。

设切线为 $y=kx+b$，于是有 $b=y-kx$。

求切点过程可以认为是对每个物品作一个大小为 k 的偏移，然后求解此时的最佳方案，同时记录最优方案中选中的物品个数。

最优方案对应最大的 b，这个方案对应的物品个数就是切点横坐标 x，然后再反过来利用 $y=b+kx$ 即可得到原始答案。

注意有些时候会出现多个最佳方案的情况，这个时候要强制一下偏序，比如强制取物品最大的方案。

算法例题

例题一

洛谷p2619

题意

给定一个图，每条边一个边权且有一种颜色(黑/白)。要求构造一棵生成树，满足恰好有 a 条白边，在此基础上边权和最小。

题解

设 $F(x)$ 表示恰好选 x 条白边时的最小生成树边权和，不难发现 $F(x)$ 是下凸的。

二分斜率，然后每次对每条白边减去等于斜率的偏移量，然后跑一遍最小生成树同时记录最优方案选中的白边数量。

黑白边边权相同时优先考虑白边。得到最优斜率 k 后答案为白边作 k 偏移量后的最小生成树 $+ka$。

时间复杂度 $O\left(m\log m\log V\right)$，注意这种生成树的题一般可以黑白边分开排序双指针处理，好

像常数可以大幅减小。

```
const int MAXN=5e4+5,MAXM=1e5+5,MAXW=105;
struct Edge{
    int u,v,w,c;
    bool operator < (const Edge &b)const{
        return w<b.w||(w==b.w&&c<b.c);
    }
}edge[MAXM];
int p[MAXN];
int Find(int x){
    return x==p[x]?x:p[x]=Find(p[x]);
}
pair<int,int> solve(int n,int m,int w){
    _rep(i,1,n)p[i]=i;
    _for(i,0,m){
        if(!edge[i].c)
        edge[i].w-=w;
    }
    sort(edge,edge+m);
    int s1=0,s2=0;
    _for(i,0,m){
        int x=Find(edge[i].u),y=Find(edge[i].v);
        if(x!=y){
            p[x]=y;
            s1+=edge[i].w;
            s2+=edge[i].c==0;
        }
    }
    _for(i,0,m){
        if(!edge[i].c)
        edge[i].w+=w;
    }
    return make_pair(s1,s2);
}
int main(){
    int n=read_int(),m=read_int(),k=read_int();
    _for(i,0,m){
        edge[i].u=read_int()+1;
        edge[i].v=read_int()+1;
        edge[i].w=read_int();
        edge[i].c=read_int();
    }
    int lef=-MAXW,rig=MAXW,ans;
    while(lef<=rig){
        int mid=lef+rig>>1;
        if(solve(n,m,mid).second>=k){
            ans=mid;
            rig=mid-1;
```

```
        }
        else
        lef=mid+1;
    }
    enter(solve(n,m,ans).first+ans*k);
    return 0;
}
```