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;
}
```

例题二

gym102059 M

题意

给定一棵边权树，要求从树上选 k 条边，所有边无公共点且最大化边权和。

题解

设 $F(x)$ 表示选 x 条边的最大边权和，不难发现 $F(x)$ 为凸函数。斜率最大值为 V(无脑加一条边)，最小值为 $-nV$(强行加一条边的最坏影响)。

利用 wqs 二分套树形 dp 求解即可。

```
const int MAXN=2.5e5+5,MAXV=1e6+5;
const LL inf=1e18;
struct Edge{
    int to,next;
    LL w;
    Edge(int to=0,LL w=0,int next=0){
        this->to=to;
        this->w=w;
        this->next=next;
    }
}edge[MAXN<<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 Node{
    LL s;
    int cnt;
    Node(LL s=0,int cnt=0){
        this->s=s;
        this->cnt=cnt;
    }
    bool operator < (const Node &b)const{
```

```
        return s<b.s||(s==b.s&&cnt<b.cnt);
    }
    Node operator + (const Node &b)const{
        return Node(s+b.s,cnt+b.cnt);
    }
    void operator += (const Node &b){
        s+=b.s;
        cnt+=b.cnt;
    }
    Node operator - (const Node &b)const{
        return Node(s-b.s,cnt-b.cnt);
    }
};
Node dp[MAXN][2];
void dfs(int u,int fa){
    dp[u][0]=Node(0,0);
    dp[u][1]=Node(-inf,0);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        dfs(v,u);
        dp[u][0]+=max(dp[v][0],dp[v][1]);
    }
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        dp[u][1]=max(dp[u][1],dp[u][0]-
max(dp[v][0],dp[v][1])+dp[v][0]+Node(edge[i].w,1));
    }
}
Node solve(int n,LL k){
    _rep(i,1,edge_cnt)
    edge[i].w-=k;
    dfs(1,0);
    Node ans=max(dp[1][0],dp[1][1]);
    _rep(i,1,edge_cnt)
    edge[i].w+=k;
    return ans;
}
int main(){
    int n=read_int(),k=read_int();
    _for(i,1,n){
        int u=read_int(),v=read_int(),w=read_int();
        Insert(u,v,w);
        Insert(v,u,w);
    }
    LL lef=-1LL*MAXV*n,rig=MAXV,ans;
    if(solve(n,lef).cnt<k){
        puts("Impossible");
        return 0;
```

```
    }
    while(lef<=rig){
        LL mid=lef+rig>>1;
        if(solve(n,mid).cnt>=k){
            ans=mid;
            lef=mid+1;
        }
        else
        rig=mid-1;
    }
    enter(solve(n,ans).s+ans*k);
    return 0;
}
```

例题三

CF739E

题意

一共有 n 只宝可梦，有 a 个普通球和 b 个高级球。每个宝可梦在一次捕捉失败后就会逃跑。

对第 i 只宝可梦，用普通球的捕获率是 p_i，用高级球的捕获率是 q_i，同时用普通球和高级球的捕获率是 $p_i+q_i-p_iq_i$。

求最优策略下能捕捉宝可梦的期望值。

题解

设 $F(x,y)$ 表示用 x 个普通球和 y 个高级球的期望捕捉数。

不难发现对固定的 x，$F(x,y)$ 是凸函数，于是利用 wqs 二分可以 $O(n\log v)$ 计算出 $F(x,b)$。

然后显然 $F(x,b)$ 也是凸函数，于是再套一层 wqs 二分可以 $O(n\log^2 v)$ 计算出 $F(a,b)$。

```
const int MAXN=2e3+5;
const double eps=1e-8;
struct Node{
    double s;
    int cnt1,cnt2;
    Node(double s=0.0,int cnt1=0,int cnt2=0){
        this->s=s;
        this->cnt1=cnt1;
        this->cnt2=cnt2;
    }
    void operator += (const Node &b){
```

```
        s+=b.s;
        cnt1+=b.cnt1;
        cnt2+=b.cnt2;
    }
    bool operator < (const Node &b)const{
        return s<b.s;
    }
};
int n,a,b;
double p[MAXN],q[MAXN],r[MAXN];
Node dp[MAXN];
Node solve2(double k1,double k2){
    Node ans=Node(0,0,0);
    _rep(i,1,n)
    ans+=max({Node(0,0,0),Node(p[i]-k1,1,0),Node(q[i]-k2,0,1),Node(r[i]-k1-
k2,1,1)});
    return ans;
}
Node solve(double k){
    double lef=0,rig=1;
    while(rig-lef>eps){
        double mid=(lef+rig)/2;
        if(solve2(k,mid).cnt2>=b)
        lef=mid;
        else
        rig=mid;
    }
    Node t=solve2(k,lef);
    t.s+=lef*b;
    return t;
}
int main(){
    n=read_int(),a=read_int(),b=read_int();
    _rep(i,1,n)
    scanf("%lf",&p[i]);
    _rep(i,1,n)
    scanf("%lf",&q[i]);
    _rep(i,1,n)
    r[i]=p[i]+q[i]-p[i]*q[i];
    double lef=0,rig=1;
    while(rig-lef>eps){
        double mid=(lef+rig)/2;
        if(solve(mid).cnt1>=a)
        lef=mid;
        else
        rig=mid;
    }
    printf("%.6lf",solve(lef).s+lef*a);
    return 0;
```

```
}
```