比赛链接

补题情况

题目	蒋贤蒙	王赵安	王智彪
A	0	0	0
B	0	0	0
C	0	0	0
D	0	0	0
E	0	0	0
G	0	0	0
I	0	0	0
J	2	0	2
K	0	0	0

题解

J. Illuminations

题意

题解

这题可以分成两个部分，一个是求凸包切线部分，一个是求环区间最小覆盖部分。

关于求环区间最小覆盖部分，首先肯定是断环成链，然后枚举 $i=1\sim n$ 区间 $[i,i+n)$ 的最小线段覆盖，但这是 $O\left(n^2\right)$ 的，有以下几种解法。

第一种，本人最初过题思路，设 $\text{dp}(l,r)$ 表示区间 $[l,r]$ 的最小线段覆盖。

固定 l显然 $\text{dp}(l,r)$ 是分段的。然后对给定 l显然策略为找到覆盖他的线段中的最大的右端点 k

于是有

$$ \text{dp}(l,i) \begin{cases} 1, &i\le k\\ \text{dp}(k+1,i)+1, &i\gt k \end{cases} $$

于是可以可持久化线段树维护每个 $\text{dp}(i,\ast)$ 数组的所有分段点，然后 $O(n\log n)$ 查询 $\text{dp}(i,i+n-1)(i=1\sim n)$得到答案。

```
const double eps=1e-11;
const double inf=1e20;
const double pi=acos(-1);
const int maxp=300020;
int n,m;
struct xd {
```

```
    int id,l,r;
} xx[maxp];
int xxs,maxv;
int sgn(double x) {
    if(fabs(x)<eps) return 0;
    if(x<0) return -1;
    return 1;
}

struct Point {
    double x,y;
    Point() {}
    Point(double _x,double _y) {
        x = _x;
        y = _y;
    }
    void input() {
        scanf("%lf%lf",&x,&y);
    }
    void output() {
        printf("%.2f %.2f\n",x,y);
    }
    bool operator == (Point b)const {
        return sgn(x-b.x) == 0 && sgn(y-b.y) == 0;
    }
    bool operator < (Point b)const {
        return sgn(x-b.x)== 0?sgn(y-b.y)<0:x<b.x;
    }
    Point operator -(const Point &b)const {
        return Point(x-b.x,y-b.y);
    }
    //叉积
    double operator ^(const Point &b)const {
        return x*b.y-y*b.x;
    }
    //点积
    double operator *(const Point &b)const {
        return x*b.x + y*b.y;
    }
    //返回长度
    double len() {
        return hypot(x,y);//库函数
    }
    //返回长度的平方
    double len2() {
        return x*x + y*y;
    }
    //返回两点的距离
    double distance(Point p) {
        return hypot(x-p.x,y-p.y);
```

```
    }
    Point operator +(const Point &b)const {
        return Point(x+b.x,y+b.y);
    }
    Point operator *(const double &k)const {
        return Point(x*k,y*k);
    }
    Point operator /(const double &k)const {
        return Point(x/k,y/k);
    }
    //计算 pa 和pb 的夹角
    //就是求这个点看 a,b 所成的夹角
    //测试 LightOJ1203
    double rad(Point a,Point b) {
        Point p = *this;
        return fabs(atan2(fabs((a-p)^(b-p)),(a-p)*(b-p)));
    }
    //计算 pa 和pb 的有向角
    double tmprad(Point a,Point b) {
        Point p = *this;
        return atan2(((a-p)^(b-p)),(a-p)*(b-p));
    }
    //化为长度为 r 的向量
    Point trunc(double r) {
        double l = len();
        if(!sgn(l))return *this;
        r /= l;
        return Point(x*r,y*r);
    }
    //逆时针旋转90度
    Point rotleft() {
        return Point(-y,x);
    }
    //顺时针旋转90度
    Point rotright() {
        return Point(y,-x);
    }
    //绕着 p 点逆时针旋转 angle
    Point rotate(Point p,double angle) {
        Point v = (*this)-p;
        double c = cos(angle), s = sin(angle);
        return Point(p.x + v.x*c-v.y*s,p.y + v.x*s + v.y*c);
    }
};

struct Line {
    Point s,e;
    Line() {

    }
    Line(Point _s,Point _e) {
```

```
        s=_s;
        e=_e;
    }

};
struct polygon {
    int n;
    Point p[maxp];
    Line l[maxp];
    void input(int _n) {
        n = _n;
        for(int i = 0; i < n; i++)
            p[i].input();
    }
    void add(Point q) {
        p[n++] = q;
    }
    void getline() {
        for(int i = 0; i < n; i++) {
            l[i] = Line(p[i],p[(i+1)%n]);
        }
    }
    struct cmp {
        Point p;
        cmp(const Point &p0) {
            p = p0;
        }
        bool operator()(const Point &aa,const Point &bb) {
            Point a = aa, b = bb;
            int d = sgn((a-p)^(b-p));
            if(d == 0) {
                return sgn(a.distance(p)-b.distance(p)) < 0;
            }
            return d > 0;
        }
    };
    //进行极角排序
    //首先需要找到最左下角的点
    //需要重载号好 Point 的<操作符 (min 函数要用)
    void norm() {
        Point mi = p[0];
        for(int i = 1; i < n; i++)mi = min(mi,p[i]);
        sort(p,p+n,cmp(mi));
    }
    //得到凸包的另外一种方法
    //测试 LightOJ1203 LightOJ1239
    void Graham(polygon &convex) {
        norm();
        int &top = convex.n;
        top = 0;
```

```
        if(n == 1) {
            top = 1;
            convex.p[0] = p[0];
            return;
        }
        if(n == 2) {
            top = 2;
            convex.p[0] = p[0];
            convex.p[1] = p[1];
            if(convex.p[0] == convex.p[1])top--;
            return;
        }
        convex.p[0] = p[0];
        convex.p[1] = p[1];
        top = 2;
        for(int i = 2; i < n; i++) {
            while( top > 1 && sgn((convex.p[top-1]-convex.p[top-2])^(p[i]-
convex.p[top-2])) <= 0 )top--;
            convex.p[top++] = p[i];
        }
        if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//特判
    }

    //下面是过凸包外一点求凸包左右切点的板子
    int getl(int l,int r,Point po) {
        int ans=l,mid;
        l++;
        while(l<=r) {
            mid=l+r>>1;
            if(sgn((p[mid]-po)^(p[(mid-1+n)%n]-p[mid]))<=0) {
                ans=mid;
                l=mid+1;
            } else r=mid-1;
        }
        return ans;
    }

    int getr(int l,int r,Point po) {
        int ans=r,mid;
        r--;
        while(l<=r) {
            mid=l+r>>1;
            if(sgn((p[mid]-po)^(p[(mid+1)%n]-p[mid]))>=0) {
                ans=mid;
                r=mid-1;
            } else l=mid+1;
        }
        return ans;
    }

    void work(Point po,int id) {
```

```
        int pos=0;
        if(sgn(po.x)>0) {
            int l=1,r=n-1,mid;
            while(l<=r) {
                mid=l+r>>1;
                if(sgn(p[mid]^po)>=0) {
                    pos=mid;
                    l=mid+1;
                } else {
                    r=mid-1;
                }
            }
        } else if(sgn(po.y)>0) {
            pos=n-1;
        }
        int l,r;
        if(sgn(po.x)>0) {
            l=getl(0,pos,po);
            r=getr(pos,n,po);
        } else {
            l=getl(maxv,n,po);
            r=getr(0,maxv,po);
        }
        if(r==0) r=n;
        xx[++xxs]= {id,l+1,r};
    }
    //到此结束
} PO,po,po1,PPO;

namespace Tree {
    const int MAXN=4e5+5;
    int lef[MAXN<<2],rig[MAXN<<2];
    pair<int,int> s[MAXN<<2],lazy[MAXN<<2];
    void build(int k,int L,int R) {
        lef[k]=L,rig[k]=R;
        if(lef[k]==rig[k])
            return;
        int M=L+R>>1;
        build(k<<1,L,M);
        build(k<<1|1,M+1,R);
    }
    void push_tag(int k,pair<int,int> v) {
        s[k]=max(s[k],v);
        lazy[k]=max(lazy[k],v);
    }
    void push_up(int k) {
        s[k]=max(s[k<<1],s[k<<1|1]);
    }
    void push_down(int k) {
        if(lazy[k].first) {
```

```
            push_tag(k<<1,lazy[k]);
            push_tag(k<<1|1,lazy[k]);
            lazy[k]=make_pair(0,0);
        }
    }
    void update(int k,int L,int R,pair<int,int> v) {
        if(L<=lef[k]&&rig[k]<=R) {
            push_tag(k,v);
            return;
        }
        push_down(k);
        int mid=lef[k]+rig[k]>>1;
        if(mid>=L)
            update(k<<1,L,R,v);
        if(mid<R)
            update(k<<1|1,L,R,v);
        push_up(k);
    }
    pair<int,int> query(int k,int pos) {
        if(lef[k]==rig[k])
            return s[k];
        push_down(k);
        int mid=lef[k]+rig[k]>>1;
        if(mid>=pos)
            return query(k<<1,pos);
        else
            return query(k<<1|1,pos);
    }
}
namespace JXM {
    const int MAXN=4e5+5;
    struct Node {
        int s,v,lch,rch;
    } node[MAXN*40];
    int root[MAXN],node_cnt;
    void update(int &k,int p,int vl,int vr,int pos,int v) {
        node[k=++node_cnt]=node[p];
        node[k].s++;
        if(vl==vr) {
            node[k].v=v;
            return;
        }
        int vm=vl+vr>>1;
        if(vm>=pos)
            update(node[k].lch,node[p].lch,vl,vm,pos,v);
        else
            update(node[k].rch,node[p].rch,vm+1,vr,pos,v);
    }
    int query_max(int k,int vl,int vr) {
        if(vl==vr)return vl;
        int vm=vl+vr>>1;
```

```
        if(node[k].rch)
            return query_max(node[k].rch,vm+1,vr);
        else
            return query_max(node[k].lch,vl,vm);
    }
    int query(int k,int vl,int vr,int pos) {
        if(!k)return 0;
        if(vl==vr)return 1;
        int vm=vl+vr>>1;
        if(vm>=pos)
            return query(node[k].lch,vl,vm,pos);
        else
            return node[node[k].lch].s+query(node[k].rch,vm+1,vr,pos);
    }
    vector<pair<int,int> > res;
    void dfs(int k,int vl,int vr) {
        if(!k)return;
        if(vl==vr) {
            res.push_back(make_pair(vl,node[k].v));
            return;
        }
        int vm=vl+vr>>1;
        dfs(node[k].lch,vl,vm);
        dfs(node[k].rch,vm+1,vr);
    }
    void pt(int v) {
        enter(v);
        dfs(root[v],1,6);
        for(pair<int,int> p:res) {
            space(p.second);
            enter(p.first);
        }
        puts("");
        res.clear();
    }
    void solve() {
        int m=xxs;
        int n=PO.n,n2=PO.n<<1;
        Tree::build(1,1,n2);
        _rep(i,1,m) {
            if(xx[i].r<xx[i].l)xx[i].r+=n;
            Tree::update(1,xx[i].l,xx[i].r,make_pair(xx[i].r,xx[i].id));
        }
        int ans=MAXN;
        for(int i=n2; i; i--) {
            pair<int,int> t=Tree::query(1,i);
            if(t.first!=0) {
                update(root[i],root[t.first+1],1,n2,t.first,t.second);
                if(i<=n&&query_max(root[i],1,n2)>=i+n-1)
                    ans=min(ans,query(root[i],1,n2,i+n-2)+1);
```

```
            }
        }
        if(ans==MAXN) {
            puts("-1");
            return;
        }
        enter(ans);
        _rep(i,1,n) {
            if(root[i]&&query_max(root[i],1,n2)>=i+n-1) {
                if(ans==query(root[i],1,n2,i+n-2)+1) {
                    dfs(root[i],1,n2);
                    for(pair<int,int> p:res) {
                        space(p.second);
                        if(p.first>=i+n-1)
                            break;
                    }
                    return;
                }
            }
        }
    }
}

int main() {
    cin>>n>>m;
    PPO.n=PO.n=n,po.n=m;
    for(int i=0; i<n; i++) {
        scanf("%lf %lf",&PPO.p[i].x,&PPO.p[i].y);
    }
    PPO.Graham(PO);
    Point pp=PO.p[0];
    for(int i=0; i<n; i++) PO.p[i]=PO.p[i]-pp;
    for(int i=0; i<n; i++) if(sgn(PO.p[i].x-PO.p[maxv].x)>=0) maxv=i;
    for(int i=0; i<m; i++) {
        scanf("%lf %lf",&po.p[i].x,&po.p[i].y);
        po.p[i]=po.p[i]-pp;
    }
    for(int i=0; i<m; i++) PO.work(po.p[i],i+1);
    JXM::solve();
    return 0;
}
```

第二种，由于对给定 l 最优策略为找到覆盖他的线段中的最大的右端点 r 然后跳到 $r+1$ 于是每个状态的后继都是唯一的。

因此如果对每个 l 建边 $l\to r+1$ 可以得到一棵树。

考虑树上倍增，可以 $O(n\log n)$ 查询每个结点 $i=1\sim n$ 跳到不小于 $i+n-1$ 的祖先的最小步数得到答案。

另外这也可以通过 dfs 维护根到当前结点的路径然后二分来实现。

第三种，首先淘汰掉被另一条线段完全包含的线段，然后选取余下的线段中最短的线段的每个位置作为起点暴力跳查询答案。

设最短的线段长度为 L，则每次跳的长度不小于 L，于是最多跳 $O(\frac nL)$ 次，于是总复杂度 $O(L\times \frac nL)\sim O(n)$。

关于淘汰被另一条线段完全包含的线段，也可以利用桶排等技巧 $O(n)$ 实现。

关于为什么选最短的线段的每个位置作为起点可以保证得到最优解，本人无法证明。

```
const double eps=1e-11;
const double inf=1e20;
const double pi=acos(-1);
const int maxp=300020;
int n,m;
struct xd {
    int id,l,r;
} xx[maxp];
int xxs,maxv;
int sgn(double x) {
    if(fabs(x)<eps) return 0;
    if(x<0) return -1;
    return 1;
}

struct Point {
    double x,y;
    Point() {}
    Point(double _x,double _y) {
        x = _x;
        y = _y;
    }
    void input() {
        scanf("%lf%lf",&x,&y);
    }
    void output() {
        printf("%.2f %.2f\n",x,y);
    }
    bool operator == (Point b)const {
        return sgn(x-b.x) == 0 && sgn(y-b.y) == 0;
    }
    bool operator < (Point b)const {
        return sgn(x-b.x)== 0?sgn(y-b.y)<0:x<b.x;
    }
    Point operator -(const Point &b)const {
        return Point(x-b.x,y-b.y);
    }
    //叉积
    double operator ^(const Point &b)const {
        return x*b.y-y*b.x;
```

```
    }
    //点积
    double operator *(const Point &b)const {
        return x*b.x + y*b.y;
    }
    //返回长度
    double len() {
        return hypot(x,y);//库函数
    }
    //返回长度的平方
    double len2() {
        return x*x + y*y;
    }
    //返回两点的距离
    double distance(Point p) {
        return hypot(x-p.x,y-p.y);
    }
    Point operator +(const Point &b)const {
        return Point(x+b.x,y+b.y);
    }
    Point operator *(const double &k)const {
        return Point(x*k,y*k);
    }
    Point operator /(const double &k)const {
        return Point(x/k,y/k);
    }
    //计算 pa 和pb 的夹角
    //就是求这个点看 a,b 所成的夹角
    //测试 LightOJ1203
    double rad(Point a,Point b) {
        Point p = *this;
        return fabs(atan2(fabs((a-p)^(b-p)),(a-p)*(b-p)));
    }
    //计算 pa 和pb 的有向角
    double tmprad(Point a,Point b) {
        Point p = *this;
        return atan2(((a-p)^(b-p)),(a-p)*(b-p));
    }
    //化为长度为 r 的向量
    Point trunc(double r) {
        double l = len();
        if(!sgn(l))return *this;
        r /= l;
        return Point(x*r,y*r);
    }
    //逆时针旋转 90 度
    Point rotleft() {
        return Point(-y,x);
    }
    //顺时针旋转 90 度
    Point rotright() {
```

```
        return Point(y,-x);
    }
    //绕着 p 点逆时针旋转 angle
    Point rotate(Point p,double angle) {
        Point v = (*this)-p;
        double c = cos(angle), s = sin(angle);
        return Point(p.x + v.x*c-v.y*s,p.y + v.x*s + v.y*c);
    }
};

struct Line {
    Point s,e;
    Line() {

    }
    Line(Point _s,Point _e) {
        s=_s;
        e=_e;
    }

};
struct polygon {
    int n;
    Point p[maxp];
    Line l[maxp];
    void input(int _n) {
        n = _n;
        for(int i = 0; i < n; i++)
            p[i].input();
    }
    void add(Point q) {
        p[n++] = q;
    }
    void getline() {
        for(int i = 0; i < n; i++) {
            l[i] = Line(p[i],p[(i+1)%n]);
        }
    }
    struct cmp {
        Point p;
        cmp(const Point &p0) {
            p = p0;
        }
        bool operator()(const Point &aa,const Point &bb) {
            Point a = aa, b = bb;
            int d = sgn((a-p)^(b-p));
            if(d == 0) {
                return sgn(a.distance(p)-b.distance(p)) < 0;
            }
            return d > 0;
```

```
        }
    };
    //进行极角排序
    //首先需要找到最左下角的点
    //需要重载号好 Point 的<操作符 (min 函数要用)
    void norm() {
        Point mi = p[0];
        for(int i = 1; i < n; i++)mi = min(mi,p[i]);
        sort(p,p+n,cmp(mi));
    }
    //得到凸包的另外一种方法
    //测试 LightOJ1203 LightOJ1239
    void Graham(polygon &convex) {
        norm();
        int &top = convex.n;
        top = 0;
        if(n == 1) {
            top = 1;
            convex.p[0] = p[0];
            return;
        }
        if(n == 2) {
            top = 2;
            convex.p[0] = p[0];
            convex.p[1] = p[1];
            if(convex.p[0] == convex.p[1])top--;
            return;
        }
        convex.p[0] = p[0];
        convex.p[1] = p[1];
        top = 2;
        for(int i = 2; i < n; i++) {
            while( top > 1 && sgn((convex.p[top-1]-convex.p[top-2])^(p[i]-
convex.p[top-2])) <= 0 )top--;
            convex.p[top++] = p[i];
        }
        if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//特判
    }

    //下面是过凸包外一点求凸包左右切点的板子
    int getl(int l,int r,Point po) {
        int ans=l,mid;
        l++;
        while(l<=r) {
            mid=l+r>>1;
            if(sgn((p[mid]-po)^(p[(mid-1+n)%n]-p[mid]))<=0) {
                ans=mid;
                l=mid+1;
            } else r=mid-1;
        }
        return ans;
```

```
    }

    int getr(int l,int r,Point po) {
        int ans=r,mid;
        r--;
        while(l<=r) {
            mid=l+r>>1;
            if(sgn((p[mid]-po)^(p[(mid+1)%n]-p[mid]))>=0) {
                ans=mid;
                r=mid-1;
            } else l=mid+1;
        }
        return ans;
    }

    void work(Point po,int id) {
        int pos=0;
        if(sgn(po.x)>0) {
            int l=1,r=n-1,mid;
            while(l<=r) {
                mid=l+r>>1;
                if(sgn(p[mid]^po)>=0) {
                    pos=mid;
                    l=mid+1;
                } else {
                    r=mid-1;
                }
            }
        } else if(sgn(po.y)>0) {
            pos=n-1;
        }
        int l,r;
        if(sgn(po.x)>0) {
            l=getl(0,pos,po);
            r=getr(pos,n,po);
        } else {
            l=getl(maxv,n,po);
            r=getr(0,maxv,po);
        }
        if(r==0) r=n;
        xx[++xxs]= {id,l+1,r};
    }
    //到此结束
} PO,po,po1,PPO;
namespace JXM {
    const int MAXN=4e5+5;
    vector<xd> c[MAXN];
    pair<int,int> dp[MAXN];
    void solve() {
        int n=PO.n,n2=PO.n<<1;
```

```
        int m=xxs;
        _rep(i,1,m){
            if(xx[i].r<xx[i].l)xx[i].r+=n;
            c[xx[i].r-xx[i].l].push_back(xx[i]);
        }
        m=0;
        _for(i,0,MAXN){
            for(xd t:c[i])
            xx[++m]=t;
        }
        _rep(i,1,m){
//              space(i);space(xx[i].l);enter(xx[i].r);
            dp[xx[i].l]=make_pair(xx[i].r,i);
        }
        _rep(i,1,n2)
        dp[i]=max(dp[i],dp[i-1]);
        _rep(i,1,n){
            pair<int,int> t1=dp[i],t2=dp[i+n];
            dp[i]=max(t1,make_pair(t2.first-n,t2.second));
            dp[i+n]=max(make_pair(t1.first+n,t1.second),t2);
        }
        int pos=0;
        _rep(i,1,n){
            if(dp[i].first<i){
                puts("-1");
                return;
            }
            if(!pos)
            pos=dp[i].second;
            else if(xx[pos].r-xx[pos].l>xx[dp[i].second].r-
xx[dp[i].second].l)
            pos=dp[i].second;
        }
        int ans=MAXN;
        _rep(i,xx[pos].l,xx[pos].r){
            int st=(i-1)%n+1,pos2=st,cnt=0;
            while(pos2<st+n){
                pos2=dp[pos2].first+1;
                cnt++;
            }
            ans=min(ans,cnt);
        }
        enter(ans);
        _rep(i,xx[pos].l,xx[pos].r){
            int st=(i-1)%n+1,pos2=st,cnt=0;
            while(pos2<st+n){
                pos2=dp[pos2].first+1;
                cnt++;
            }
            if(ans==cnt){
                int pos2=st;
```

```
                while(pos2<st+n){
                    space(xx[dp[pos2].second].id);
                    pos2=dp[pos2].first+1;
                }
                return;
            }
        }
    }
}

int main() {
    cin>>n>>m;
    PPO.n=PO.n=n,po.n=m;
    for(int i=0; i<n; i++) {
        scanf("%lf %lf",&PPO.p[i].x,&PPO.p[i].y);
    }
    PPO.Graham(PO);
    Point pp=PO.p[0];
    for(int i=0; i<n; i++) PO.p[i]=PO.p[i]-pp;
    for(int i=0; i<n; i++) if(sgn(PO.p[i].x-PO.p[maxv].x)>=0) maxv=i;
    for(int i=0; i<m; i++) {
        scanf("%lf %lf",&po.p[i].x,&po.p[i].y);
        po.p[i]=po.p[i]-pp;
    }
    for(int i=0; i<m; i++) PO.work(po.p[i],i+1);
    JXM::solve();
    return 0;
}
```