先吐槽几句，这道题很坑，卡换行，这题2014年区域赛的题有一道题有点像。 不过那个给了30秒直接暴力也能过。 正解是CDQ分治+动态凸包 ，我的方法是用CDQ分治做这个凸包。

对于一个区间[l,r]，我首先将存在于[l, mid]中的点集取出来做出一个凸包，对于[l, mid]之间的‘-1’操作，如果删除的是[l, mid]之间的点，那么就正常的删点；如果删的点是l左边的点，那么这个删除操作就忽略。然后利用构建好的这个凸包去更新[mid+1, r]中的所有询问操作。

看似好像没问题，但是实际上这是错的…

因为在区间[mid+1, r]中可能存在删除操作，删掉了之前构建好的凸包上面的点，导致更新后面的询问出现错误，那么我们还需要做的就是在删除点之后，继续维护这个凸包。对于[mid+1, r]中的加点操作，我们是可以忽略的，因为后面更新右区间的时候也会更新的。

删除点的维护操作是这样的，首先现在凸包上面找到这个点，如果没找到说明不会改变凸包，但是也要将这个点标记为已删除的点。如果这个点在凸包上面，那么找到这个点的在凸包上的前驱和后继，将该点删除，然后在最开始构筑凸包的那个数组中，重新构造前驱到后继之间的凸包，最后在加上后面的凸包即可。

做完这一步的话，就可以正常的更新所有询问咯~

链接https://blog.csdn.net/riba2534/article/details/70882402

Ac代码：

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const ll N = 2e5+10;

typedef pair<ll,ll> PII;

const ll INF=0x3f3f3f3f3f3f3f;

ll ss[N];

struct Point

{

ll x, y;

ll ty, opid;

bool operator<(const Point& o)const{

return x < o.x || (x == o.x && y < o.y);

}

}can[N], p1[N], p2[N], pt[N], tmp[N];

ll n , m , q;

ll m1, m2;

map<PII, ll> mp;

ll ans[N];

bool vis[N];

ll cnt;

ll Cross(Point A, Point B){return (ll)A.x \* B.y - (ll)A.y \* B.x;}

Point operator-(Point A, Point B) {return (Point){A.x - B.x, A.y - B.y};}

void getConvex(ll n)

{

m1 = m2 = 0ll;

sort(pt, pt+n);

for(ll i = 0ll; i < n; i++)

{

while(m1 > 1ll && Cross(pt[i] - p1[m1-2], p1[m1-1] - p1[m1-2ll]) <= 0ll) m1--;

p1[m1++] = pt[i];

}

for(ll i = n-1; i >= 0ll; i--)

{

while(m2 > 1ll && Cross(pt[i] - p2[m2-2], p2[m2-1] - p2[m2-2ll]) <= 0ll) m2--;

p2[m2++] = pt[i];

}

}

void updateConvex(Point deltP)

{

ll idx = -1ll;

for(ll i = 0ll; i < m1; i++)

{

if(p1[i].x == deltP.x && p1[i].y == deltP.y){idx = i;break;}

}

if(idx != -1ll)

{

ll l = idx - 1ll, r = idx + 1ll;

ll tail = 0;

for(ll i = r+1; i < m1; i++)

tmp[tail++] = p1[i];

ll lpt, rpt;

if(l == -1) lpt = -1;

if(r == m1) rpt = cnt - 1;

for(ll i = 0; i < cnt; i++){

if(l != -1 && pt[i].x == p1[l].x && pt[i].y == p1[l].y && vis[pt[i].opid]) lpt = i;

if(r != m1 && pt[i].x == p1[r].x && pt[i].y == p1[r].y && vis[pt[i].opid]) rpt = i;

}

///

m1 = l+1ll;

for(ll i = lpt + 1ll; i <= rpt; i++)

{

if(vis[pt[i].opid] == 0) continue;

while(m1 > 1 && Cross(pt[i] - p1[m1-2], p1[m1-1] - p1[m1-2]) <= 0) m1--;

p1[m1++] = pt[i];

}

for(ll i = 0; i < tail; i++)

p1[m1++] = tmp[i];

}

idx = -1;

for(ll i = 0; i < m2; i++)

{

if(p2[i].x == deltP.x && p2[i].y == deltP.y) {idx = i;break;}

}

if(idx != -1)

{

ll l = idx - 1, r = idx + 1;

ll tail = 0;

for(ll i = r+1; i < m2; i++)

tmp[tail++] = p2[i];

ll lpt, rpt;

if(l == -1) lpt = cnt;

if(r == m2) rpt = 0;

for(ll i = 0; i < cnt; i++){

if(l != -1 && pt[i].x == p2[l].x && pt[i].y == p2[l].y && vis[pt[i].opid]) lpt = i;

if(r != m2 && pt[i].x == p2[r].x && pt[i].y == p2[r].y && vis[pt[i].opid]) rpt = i;

}

///

m2 = l+1;

for(ll i = lpt - 1; i >= rpt; i--)

{

if(vis[pt[i].opid] == 0) continue;

while(m2 > 1 && Cross(pt[i] - p2[m2-2], p2[m2-1] - p2[m2-2]) <= 0) m2--;

p2[m2++] = pt[i];

}

for(ll i = 0; i < tail; i++)

p2[m2++] = tmp[i];

}

}

void updateans(ll x, ll y, ll idx)

{

Point o =(Point){x, -y};

if(m1 <= 2){

for(ll i = 0; i < m1; i++)

ans[idx] = max(ans[idx], (ll)p1[i].x \* y + (ll)p1[i].y \* x);

}else{

ll lb = -1, ub = m1-1;

while(lb + 1 < ub)

{

ll mid = lb + ub >> 1;

if(Cross(p1[mid+1] - p1[mid], o) <= 0) lb = mid;

else ub = mid;

}

lb = max(0ll, lb);

ans[idx] = max(ans[idx], (ll)p1[lb].x \* y + (ll)p1[lb].y \* x);

if(ub < m1) ans[idx] = max(ans[idx], (ll)p1[ub].x \* y + (ll)p1[ub].y \* x);

}

if(m2 <= 2) {

for(ll i = 0; i < m2; i++)

ans[idx] = max(ans[idx], (ll)p2[i].x \* y + (ll)p2[i].y \* x);

}else{

ll lb = -1, ub = m2 - 1;

while(lb + 1 < ub){

ll mid = lb + ub >> 1;

if(Cross(p2[mid+1] - p2[mid], o) <= 0) lb = mid;

else ub = mid;

}

lb = max(0ll, lb);

ans[idx] = max(ans[idx],(ll)p2[lb].x \* y + (ll)p2[lb].y \* x);

if(ub < m2) ans[idx] = max(ans[idx], (ll)p2[ub].x \* y + (ll)p2[ub].y \* x);

}

}

void divide(ll l, ll r)

{

if(l >= r) return ;

ll mid = l + r >> 1;

for(ll i = l; i <= mid; i++) vis[i] = 0;

for(ll i = l; i <= mid; i++)

{

if(can[i].ty == 1) vis[i] = 1;

if(can[i].ty == -1 && can[i].opid >= l && can[i].opid <= mid) vis[can[i].opid] = 0;

}

cnt = 0;

for(ll i = l; i <= mid; i++)

if(vis[i]) pt[cnt++] = can[i];

getConvex(cnt);

for(ll i = mid+1; i <= r; i++){

if(can[i].ty == 0) updateans(can[i].x, can[i].y, i);

if(can[i].ty == -1 && can[i].opid >= l && can[i].opid <= mid && vis[can[i].opid]){

vis[can[i].opid] = 0;

updateConvex(can[can[i].opid]);

}

}

divide(l, mid);

divide(mid+1, r);

}

int main()

{

// Open();

ll t;

scanf("%lld",&t);

for(int I = 1; I <= t; I++)

{

scanf("%lld %lld", &m , &q) ;

n = m+q;

mp.clear();

for(ll i = 0; i < n; i++)

{

ll x, y;

scanf("%lld %lld", &x, &y);

ll ty ;

if(i < m )

ty = 1 ;

else

ty = 0 ;

can[i] = (Point){y, x, ty, i};

if(ty == 0) swap(can[i].x, can[i].y), ans[i] = -INF;

if(ty == 1) mp[PII(y, x)] = i;

if(ty == -1) can[i].opid = mp[PII(y, x)];

}

divide(0, n-1);

for(ll i = 0; i < n; i++)

{

if(can[i].ty == 0)

//printf("%I64d\n", ans[i]);

ss[++cnt] = ans[i];

}

for(ll i = 1; i <= cnt; i++){

if(i == cnt) cout<<ss[i];

else cout<<ss[i]<<endl;

}

if(I != t) cout<<endl;

}

return 0;

}