modal

目录

MTT

```cpp
#include <algorithm>
#include <iostream>
#include <cstdio>
#include <cstring>
// #define int long long
const int mod = 1e9 + 7;
namespace Math {
  inline int pw(int base, int p, const int mod) {
    static int res;
    for (res = 1; p; p >>= 1, base = static_cast<long long> (base) * base % mod) if (p & 1)
        res = static_cast<long long> (res) * base % mod;
    return res;
  }
  inline int inv(int x, const int mod) { return pw(x, mod - 2, mod); }
}

const int mod1 = 469762049, mod2 = 998244353, mod3 = 1004535809, G = 3;
const long long mod_1_2 = static_cast<long long> (mod1) * mod2;
const int inv_1 = Math::inv(mod1, mod2), inv_2 = Math::inv(mod_1_2 % mod3, mod3);
struct Int {
  int A, B, C;
  explicit inline Int() { }
  explicit inline Int(int __num) : A(__num), B(__num), C(__num) { }
  explicit inline Int(int __A, int __B, int __C) : A(__A), B(__B), C(__C) { }
  static inline Int reduce(const Int &x) {
    return Int(x.A + (x.A >> 31 & mod1), x.B + (x.B >> 31 & mod2), x.C + (x.C >> 31 & mod3));
  }
  inline friend Int operator + (const Int &lhs, const Int &rhs) {
    return reduce(Int(lhs.A + rhs.A - mod1, lhs.B + rhs.B - mod2, lhs.C + rhs.C - mod3));
  }
  inline friend Int operator - (const Int &lhs, const Int &rhs) {
    return reduce(Int(lhs.A - rhs.A, lhs.B - rhs.B, lhs.C - rhs.C));
  }
  inline friend Int operator * (const Int &lhs, const Int &rhs) {
    return Int(
    static_cast<long long> (lhs.A) * rhs.A % mod1,
    static_cast<long long> (lhs.B) * rhs.B % mod2,
    static_cast<long long> (lhs.C) * rhs.C % mod3);
  }
  inline int get() {
    long long x = static_cast<long long> (B - A + mod2) % mod2 * inv_1 % mod2 * mod1 + A;
    return (static_cast<long long> (C - x % mod3 + mod3) % mod3 * inv_2 % mod3 * (mod_1_2 %
        mod) % mod + x) % mod;
  }
```

```
};

#define maxn 131072
namespace Poly {
  #define N (maxn << 1)
  int lim, s, rev[N];
  Int Wn[N | 1];
  inline void init(int n) {
    s = -1, lim = 1; while (lim < n) lim <<= 1, ++s;
    for (register int i = 1; i < lim; ++i) rev[i] = rev[i >> 1] >> 1 | (i & 1) << s;
    const Int t(Math::pw(G, (mod1 - 1) / lim, mod1), Math::pw(G, (mod2 - 1) / lim, mod2),
        Math::pw(G, (mod3 - 1) / lim, mod3));
    *Wn = Int(1); for (register Int *i = Wn; i != Wn + lim; ++i) *(i + 1) = *i * t;
  }
  inline void NTT(Int *A, const int op = 1) {
    for (register int i = 1; i < lim; ++i) if (i < rev[i]) std::swap(A[i], A[rev[i]]);
    for (register int mid = 1; mid < lim; mid <<= 1) {
      const int t = lim / mid >> 1;
      for (register int i = 0; i < lim; i += mid << 1) {
        for (register int j = 0; j < mid; ++j) {
          const Int W = op ? Wn[t * j] : Wn[lim - t * j];
          const Int X = A[i + j], Y = A[i + j + mid] * W;
          A[i + j] = X + Y, A[i + j + mid] = X - Y;
        }
      }
    }
    if (!op) {
      const Int ilim(Math::inv(lim, mod1), Math::inv(lim, mod2), Math::inv(lim, mod3));
      for (register Int *i = A; i != A + lim; ++i) *i = (*i) * ilim;
    }
  }
  #undef N
}
Int A[maxn << 1], B[maxn << 1], C[maxn << 1], D[maxn << 1], res[maxn << 1];
signed main() {
  int n, m;
  std::cin >> n >> m;
  Poly::init(n + n);
  for (int i = 0; i < Poly::lim; i ++) {
    A[i] = Int(0);
    B[i] = Int(0);
  }
  A[1] = Int(1), A[n - 1] = Int(1);
  B[0] = Int(1);
  while (m) {
    if (m & 1) {
      for (int i = 0; i < n; i ++) {
```

```
      C[i] = B[i];
      D[i] = A[i];
    }
    for (int i = n; i < Poly::lim; i ++) {
      C[i] = Int(0);
      D[i] = Int(0);
    }
    Poly::NTT(C), Poly::NTT(D);
    for (int i = 0; i < Poly::lim; i ++) {
      res[i] = C[i] * D[i];
    }
    Poly::NTT(res, 0);
    for (int i = 0; i < n + n; i ++) {
      B[i] = Int(((res[i].get() + res[i + n].get()) % mod + mod) % mod);
    }
  }

  for (int i = 0; i < n; i ++) {
    C[i] = A[i];
  }
  for (int i = n; i < Poly::lim; i ++) {
    C[i] = Int(0);
  }
  Poly::NTT(C);
  for (int i = 0; i < Poly::lim; i ++) {
    res[i] = C[i] * C[i];
  }
  Poly::NTT(res, 0);
  for (int i = 0; i < n + n; i ++) {
    A[i] = Int(((res[i].get() + res[i + n].get()) % mod + mod) % mod);
  }
  m >>= 1;
 }
 std::cout << B[0].get() << '\n';
 return 0;
}
// 10 100 the last time
```

单流 EK

```
struct FlowGraph {
 int s, t, vtot, etot, head[V], dis[V], cur[V], pre[V], pre_edge[V];
 struct edge {
  int v, nxt;
  T f;
```

```
} e[E * 2];
void addedge(int u, int v, T f) {
   e[etot] = {v, head[u], f}; head[u] = etot ++;
   e[etot] = {u, head[v], 0}; head[v] = etot ++;
}
bool bfs() {
   for (int i = 1; i <= vtot; i ++) {
      dis[i] = 0;
      cur[i] = head[i];
   }
   std::queue<int> q;
   q.push(s); dis[s] = 1;
   while (! q.empty()) {
      int u = q.front(); q.pop();
      for (int i = head[u]; ~i; i = e[i].nxt) {
         if (e[i].f && ! dis[e[i].v]) {
            int v = e[i].v;
            dis[v] = dis[u] + 1;
            pre[v] = u;
            pre_edge[v] = i;
            if (v == t) return true;
            q.push(v);
         }
      }
   }
   return false;
}
T dicnic() {
   T flow = 0;
   while (bfs()) {
      flow ++;
      int x = t;
      while (x != s) {
         int i = pre_edge[x];
         e[i].f -= 1;
         e[i ^ 1].f += 1;
         x = pre[x];
      }
   }
   return flow;
}
void init(int _s, int _t, int _vtot) {
   s = _s;
   t = _t;
   vtot = _vtot;
   for (int i = 1; i <= vtot; i ++) {
      head[i] = -1;
```

```
    }
  }
};
```

树哈希

```
#include<bits/stdc++.h>
using i64 = long long;
#define u64 unsigned long long
#define int u64
const u64 b = 1331;
u64 h(u64 x) {
  return x * x * x * 1237123 + 19260817;
}
u64 f(u64 x) {
  u64 cur = h(x & ((1ll << 31) - 1)) + h(x >> 31);
  return cur;
}
void solve() {
  int n, m;
  std::cin >> n >> m;
  std::vector<std::vector<int> > e(n + 1);
  for (int i = 1; i <= m; i ++) {
    int x, y;
    std::cin >> x >> y;
    e[x].push_back(y);
    e[y].push_back(x);
  }

  if (m == n - 1) {
    std::cout << "YES\n";
    return ;
  }
  if (m > n) {
    std::cout << "NO\n";
    return ;
  }

  i64 ans = 0;
  std::vector<int> dfn(n + 1), onloop(n + 1), loop(n * 2 + 1), fa(n + 1);
  int timer = 0, cnt;

  auto dfs = [&](auto self, int u, int p) -> void {
    dfn[u] = ++ timer;
    for (auto v : e[u]) {
```

```
      if (v == p) continue;
      if (dfn[v]) {
        if (dfn[v] < dfn[u]) continue;
        loop[++ cnt] = v;
        onloop[v] = 1;
        while (v != u) {
          loop[++ cnt] = fa[v];
          onloop[fa[v]] = 1;
          v = fa[v];
        }
      } else {
        fa[v] = u;
        self(self, v, u);
      }
    }
  };
  std::vector<u64> ff(n + 1);
  auto getsubTree = [&](auto self, int u, int p) -> void {
    ff[u] = 1;
    for (auto v : e[u]) {
      if (v == p || onloop[v]) continue;
      self(self, v, u);
      ff[u] = ff[u] + f(ff[v]);
    }
  };
  cnt = 0;
  dfs(dfs, 1, 0);
  std::vector<u64> LOOP(cnt * 2 + 1);
  for (int i = 1; i <= cnt; i ++) {
    getsubTree(getsubTree, loop[i], 0);
    LOOP[i] = ff[loop[i]];
    LOOP[i + cnt] = LOOP[i];
  }
  for (int i = 1; i <= cnt; i ++) {
    if (LOOP[i] != LOOP[i + 2]) {
      std::cout << "NO\n";
      return ;
    }
  }
  std::cout << "YES\n";
}
signed main() {
  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);
  int t = 1;
  std::cin >> t;
  while (t --) {
```

```
    solve();
  }
  return 0;
}
```

dfs 树

```
#include<bits/stdc++.h>
using i64 = long long;
using pii = std::array<int, 2>;
void solve() {
  int n, m;
  std::cin >> n >> m;
  std::vector<bool> vis_U(n + 1), vis_E(m + 1);
  std::vector<pii> ans(m + 1); // 0/1, 0/1
  std::vector<std::vector<pii> > e(n + 1);
  std::vector<int> fornow(n + 1);
  std::vector<pii> E(m + 1);
  for (int i = 1; i <= m; i ++) {
    int u, v;
    std::cin >> u >> v;
    E[i] = {u, v};
    e[u].push_back({v, i});
    e[v].push_back({u, i});
  }
  auto check = [&](int y, int id1, int id2) {
    ans[id1] = {(E[id1][0] == y ? 0 : 1), (id1 > id2)};
    ans[id2] = {(E[id2][0] == y ? 0 : 1), (id1 < id2)};
  };
  auto dfs = [&](auto&& self, int u) -> pii { // I konw i konw...
    vis_U[u] = 1;
    pii now = {0, 0};
    // for (int i = fornow[u]; i < e[u].size(); i = fornow[u] +) {
      // int v = e[u][i][0], id = e[u][i][1];
      // for (auto it ; e[u]) {
        //   int v = it[0], id = it[1];
    while (fornow[u] < e[u].size()) {
      int v = e[u][fornow[u]][0], id = e[u][fornow[u]][1];
      fornow[u] ++;
      if (vis_E[id]) continue;
      vis_E[id] = 1;
      pii down = self(self, v); // w, id
      if (down[0]) {
        // u, v, w
        check(v, id, down[1]);
```

```
        } else {
          if (now[0]) {
            check(u, now[1], id);
            now = {0, 0};
          } else {
            now = {v, id};
          }
        }
      }
      return now;
    };
    for (int i = 1; i <= n; i ++) {
      if (! vis_U[i]) {
        pii it = dfs(dfs, i); // 不需要去特判它的,
      }
    }
    for (int i = 1; i <= m; i ++) {
      std::cout << (ans[i][0] ? 'y' : 'x') << (ans[i][1] ? '-' : '+') << '\n';
    }
}
signed main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int t = 1;
    // std::cin >> t;
    while (t --) {
      solve();
    }
    return 0;
}
```

长链剖分

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
using namespace std;
#define ll long long
#define MAXN 100010

int n;
struct edge{
    int v,nxt;
} e[MAXN*2];
```

```
int head[MAXN],cnt=0;
void ad(int u,int v){
   e[++cnt].v=v;
   e[cnt].nxt=head[u];
   head[u]=cnt;
}

ll ans=0;
int son[MAXN],maxdep[MAXN];
void dfs1(int u,int fa){
   maxdep[u]=0;
   for(int i=head[u];i;i=e[i].nxt){
      int v=e[i].v;
      if(v==fa) continue;
      dfs1(v,u);
      if(!son[u] || maxdep[v]>maxdep[son[u]])
      son[u]=v;
   }
   maxdep[u]=maxdep[son[u]]+1;
}
//void dfs2(int u,int fa){
   // f[u][0]=1;
   // for(int i=head[u];i;i=e[i].nxt){
      //    int v=e[i].v;
      //    if(v==fa) continue;
      //    dfs(v,u);
      //    for(int j=1;j<=n;j++){
         //       ans+=f[u][j-1]*g[v][j]+g[u][j]*f[v][j-1];
         //       g[u][j]+=1ll*f[u][j]*f[v][j-1];
         //    }
      //    for(int j=0;j<n;j++)
      //       g[u][j]+=g[v][j+1];
      //    for(int j=1;j<=n;j++)
      //       f[u][j]+=f[v][j-1];
      // }
   //// ans+=g[u][0];
   //}
ll *f[MAXN],*g[MAXN],tmp[MAXN*4],*id=tmp;
void dfs2(int u,int fa){
   if(son[u]){
      f[son[u]]=f[u]+1,g[son[u]]=g[u]-1;
      dfs2(son[u],u);
   }
   f[u][0]=1;ans+=g[u][0];
   for(int i=head[u];i;i=e[i].nxt){
      int v=e[i].v;
      if(v==fa || v==son[u]) continue;
```

```
      f[v]=id;id+=maxdep[v]*2;
      g[v]=id;id+=maxdep[v]*2;
      dfs2(v,u);
      for(int j=1;j<=maxdep[v];j++)
      ans+=f[u][j-1]*g[v][j]+g[u][j]*f[v][j-1];
      for(int j=0;j<=maxdep[v];j++)
      g[u][j+1]+=f[u][j+1]*f[v][j];
      for(int j=0;j<maxdep[v];j++)
      g[u][j]+=g[v][j+1];
      for(int j=0;j<=maxdep[v];j++)
      f[u][j+1]+=f[v][j];
   }
}

int main()
{
   // freopen("three.in","r",stdin);
   // freopen("three.out","w",stdout);
   // while(scanf("%d",&n)!=EOF){
      //   if(n==0) break;
      scanf("%d",&n);
      cnt=ans=0;
      for(int i=1;i<n;i++){
         int u,v;
         scanf("%d%d",&u,&v);
         ad(u,v);ad(v,u);
      }
      dfs1(1,0);
      f[1]=id;id+=maxdep[1]*2;
      g[1]=id;id+=maxdep[1]*2;
      dfs2(1,0);
      printf("%lld\n",ans);
      for(int i=1;i<=n;i++)
      head[i]=maxdep[i]=son[i]=0;
      for(int i=1;i<=n*4;i++)
      tmp[i]=0;
      id=tmp;
      // }
   return 0;
}
```

2 个不相交凸多边形的最近距离

```
#include<cstdio>
#include<vector>
```

```cpp
#include<cmath>
#include<string>
#include<string.h>
#include<iostream>
#include<algorithm>
#define PI acos(-1.0)
#define pb push_back
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
const int N=1e6+6;
const int MOD=1e9+7;
template <class T>
bool sf(T &ret){ //Faster Input
  char c; int sgn; T bit=0.1;
  if(c=getchar(),c==EOF) return 0;
  while(c!='-'&&c!='.'&&(c<'0'||c>'9')) c=getchar();
  sgn=(c=='-')?-1:1;
  ret=(c=='-')?0:(c-'0');
  while(c=getchar(),c>='0'&&c<='9') ret=ret*10+(c-'0');
  if(c==' '||c=='\n'){ ret*=sgn; return 1; }
  while(c=getchar(),c>='0'&&c<='9') ret+=(c-'0')*bit,bit/=10;
  ret*=sgn;
  return 1;
}
int sign(double x){
  return abs(x)<1e-7?0:x<0?-1:1;
}
struct Point{
  double x,y;
  Point(double x=0, double y=0) : x(x), y(y) {}
  Point operator - (const Point &rhs) const{
    return Point(x-rhs.x,y-rhs.y);
  }
  bool operator == (const Point &rhs) const{
    return sign(x-rhs.x)==0&&sign(y-rhs.y)==0;
  }
  bool operator < (const Point &rhs)const{
    if(x==rhs.x)  return y<rhs.y;
    else   return x<rhs.x;
  }
}p[N];
typedef Point Vector;
double cross(Vector A,Vector B){
  return A.x*B.y-A.y*B.x;
```

```
}
int n;
typedef vector<Point> Polygon;
Polygon convex_hull(Polygon P){
   sort(P.begin(),P.end());
   P.erase(unique(P.begin(), P.end()), P.end());
   int n=P.size(),k=0;
   Polygon Q(n*2);
   for(int i=0;i<n;++i){
      while(k>1&&cross(Q[k-2]-Q[k-1],Q[k-2]-P[i])<=0) k--;
      Q[k++]=P[i];
   }
   int t=k;
   for(int i=n-2;i>=0;--i){
      while(k>t && cross(Q[k-2]-Q[k-1],Q[k-2]-P[i])<=0) k--;
      Q[k++]=P[i];
   }
   Q.resize(k-1);
   return Q;
}

double dis(Point a,Point b){
   return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
double dot(Point a,Point b){
   return a.x*b.x+a.y*b.y;
}

double point_to_seg(Point p, Point a, Point b) { //点到线段的距离，两点式
   if (a == b) return dis(p , a);
   Vector V1 = b - a, V2 = p - a, V3 = p - b; //点p到线段ab的距离
   if (sign(dot(V1, V2)) < 0) return dis(p,a); //|pa|
   else if (sign(dot(V1, V3)) > 0) return dis(p,b); //|pb|
   else return fabs(cross(V1, V2)) / dis(b,a);
}

double rc(Polygon p,Polygon q){
   int n=(int)p.size();
   int m=(int)q.size();
   p.pb(p[0]);
   q.pb(q[0]);
   int idp=0,idq=0;
   double miny=p[idp].y;
   for(int i=1;i<n;i++)
   if(sign(p[i].y-miny)<0) miny=p[i].y,idp=i;
   double maxy=p[idq].y;
   for(int i=1;i<m;i++)
```

```
  if(sign(q[i].y-maxy)>0) maxy=q[i].y,idq=i;

  double res=1e20;
  for(int i=0;i<n;i++){
    while( sign(cross(q[idq]-p[idp],p[idp+1]-p[idp]) -
        cross(q[idq+1]-p[idp],p[idp+1]-p[idp])) >0 ) idq++,idq%=m;;
    res=min(res, point_to_seg(p[idp],q[idq],q[idq+1]));
    res=min(res, point_to_seg(p[idp+1],q[idq],q[idq+1]));
    res=min(res, point_to_seg(q[idq],p[idp],p[idp+1]));
    res=min(res, point_to_seg(q[idq+1],p[idp],p[idp+1]));
    idp++,idp%=n;
  }
  return res;
}

int main(void){
  int n,m;
  while(scanf("%d%d",&n,&m)==2 && (n+m)!=0){
    Polygon p;
    for(int i=1;i<=n;i++){
      double x,y;
      scanf("%lf%lf",&x,&y);
      p.pb({x,y});
    }
    p=convex_hull(p);
    Polygon q;
    for(int i=1;i<=m;i++){
      double x,y;
      scanf("%lf%lf",&x,&y);
      q.pb({x,y});
    }
    q=convex_hull(q);
    printf("%.5f\n",rc(p,q));
  }
}
```

1.求出凸包P中y最小的序号idp, 凸包Q中y最大的序号idq

2.P和Q按着逆时针的顺序,枚举凸包P的所有边.当枚举边e时,找到距离该直线最近的点(叉积)

3.维护最小值,分别是4个点到对面直线的最短距离