Template

Geo2D

November 1, 2018

目录

1 Geometry

1.1 DelaunayTriangulation

```
typedef __int128 T;
const int N = 1e4 + 10 , NODE = N * 20;
const T eps = 0 , inf = 1e8;// be careful with inf
int sgn(T x){return (x>eps)-(x<-eps);}
struct P{
  T x,y;int id;P(){} P(T x,T y,int id=0):x(x),y(y),id(id){}
  P operator - (const P&b) const {return P(x-b.x,y-b.y);}
  T operator * (const P&b) const {return x*b.x+y*b.y;}
  T operator / (const P&b) const {return x*b.y-y*b.x;}
};
T norm(P a){return a*a;}
T cross(P a,P b,P p){return (b-a)/(p-a);}
// be careful with integer limitation
bool inCir(P a,P b,P c,P p){
  T A = (b - p) / (c - p) * (norm(a) - norm(p));
  T B = (c - p) / (a - p) * (norm(b) - norm(p));
  T C = (a - p) / (b - p) * (norm(c) - norm(p));
  return sgn(A + B + C) > 0;
}
struct Tri;
struct Edge{
  Tri* tri;int side;
  Edge(Tri* tri=NULL,int side=0):tri(tri),side(side){}
};
struct Tri{
  P p[3];Edge edge[3];Tri*son[3];Tri(){}
  Tri(P p0,P p1,P p2){p[0]=p0,p[1]=p1,p[2]=p2;rep(i,0,3) son[i]=NULL;}
  bool has_son() const {return son[0];}
  bool contains(P q) const {
    rep(i,0,3) if(sgn(cross(p[i],p[(i+1)%3],q))<0) return false;
    return true;
  }
}tri_pool[NODE],*pt;
void set_edge(Edge a,Edge b){
  if(a.tri) a.tri->edge[a.side]=b;
  if(b.tri) b.tri->edge[b.side]=a;
}
Tri* root;
// use bfs to handle in-line case
Tri* find(P p){
  Tri*c=root;
  while(c->has_son()) rep(i,0,3)
    if(c->son[i]->contains(p))
    {c=c->son[i];break;}
  return c;
}
void flip(Tri*x,int px){
  Tri* y=x->edge[px].tri;int py=x->edge[px].side;
  if(!y||!inCir(x->p[0],x->p[1],x->p[2],y->p[py])) return;
  Tri*s[2];
  s[0]=new(pt++) Tri(x->p[(px+1)%3],y->p[py],x->p[px]);
  s[1]=new(pt++) Tri(y->p[(py+1)%3],x->p[px],y->p[py]);
  set_edge(Edge(s[0],0),Edge(s[1],0));
  set_edge(Edge(s[0],1),x->edge[(px+2)%3]),set_edge(Edge(s[0],2),y->edge[(py+1)%3]);
  set_edge(Edge(s[1],1),y->edge[(py+2)%3]),set_edge(Edge(s[1],2),x->edge[(px+1)%3]);
  rep(i,0,2) x->son[i]=y->son[i]=s[i];
  x->son[2]=y->son[2]=0;
  rep(i,0,2) flip(s[i],1),flip(s[i],2);
}
void add(P p){
  Tri*c=find(p),*s[3];
  rep(i,0,3) c->son[i]=s[i]=new(pt++) Tri(c->p[i],c->p[(i+1)%3],p);
  rep(i,0,3) set_edge(Edge(s[i],0),Edge(s[(i+1)%3],1));
  rep(i,0,3) set_edge(Edge(s[i],2),c->edge[(i+2)%3]);
  rep(i,0,3) flip(s[i],2);
}
void init(P*p,int n){
  pt=tri_pool;
  root=new(pt++) Tri(P(-inf,-inf),P(inf,-inf),P(0,inf));
  random_shuffle(p,p+n);
  rep(i,0,n) add(p[i]);
}
```

1.2 Geo2D

```
/*
 * 平面图欧拉定理：V + F - E = 2
 */
typedef db T;
const db eps = 1e-9 , pi = acosl(-1.);
int sgn(T x){return (x>eps)-(x<-eps);}
struct P{
   T x,y; P(){} P(T x,T y):x(x),y(y){}
   P operator - (const P&b) const {return P(x-b.x,y-b.y);}
   P operator + (const P&b) const {return P(x+b.x,y+b.y);}
   T operator * (const P&b) const {return x*b.x+y*b.y;}
   T operator / (const P&b) const {return x*b.y-y*b.x;}
   P operator * (const T&k) const {return P(x*k,y*k);}
   P operator / (const T&k) const {return P(x/k,y/k);}
   bool operator < (const P&b) const {return sgn(x-b.x)?x<b.x:y<b.y;}
   bool operator == (const P&b) const{return !sgn(x-b.x)&&!sgn(y-b.y);}
   bool operator != (const P&b) const{return !(*this == b);}
   P rot90(){return P(-y,x);}
   // 向量与 x 轴的夹角，取值范围（ -, pi pi]
   db arg() const {return atan2(y,x);}
};
T norm(P a){return a*a;}
T abs(P a) {return sqrtl(norm(a));}
// For given three points a,b,p, find the projection point x of p onto ab.
P proj(P p,P a,P b){return (b-a)*((p-a)*(b-a)/norm(b-a))+a;}
// For given three points a,b,p, find the reflection point x of p onto ab.
P reflect(P p,P a,P b){return proj(p,a,b)*2-p;}
T cross(P o,P a,P b){return (a-o)/(b-o);}
int crossOp(P o,P a,P b){return sgn(cross(o,a,b));}
// 向量夹角
```

```
db rad(P p1,P p2){return atan2l(p1/p2,p1*p2);}
bool onPS(P p,P s,P t){return sgn((t-s)/(p-s))==0&&sgn((p-s)*(p-t))<=0;}
bool order(const P&a,const P&b){ return a.arg() < b.arg();}
// 向量逆时针旋转 rad （弧度，精度可能不太够）
P rot(P a, T rad) {
    return P(a.x * cos(rad) - a.y * sin(rad), a.x * sin(rad) + a.y * cos(rad));
}
P rot(P a, P o, T rad) {
    return rot(a - o, rad) + o;
}
struct L{ P s,t;L(){} L(P s,P t):s(s),t(t){}};
P insLL(L a,L b){ // line x line
   P s = a.s - b.s , v = a.t - a.s , w = b.t - b.s;
   db k1 = s / w , k2 = w / v;
   if(sgn(k2) == 0) return abs(b.s - a.s) < abs(b.t - a.s) ? b.s : b.t;
   return a.s + v * (k1 / k2);
}
// 判断点是否在线段上（不包括端点）
bool onS0(P p, P a, P b) {
    return sgn((p - a) / (b - a)) == 0 && sgn((p - a) * (p - b)) < 0;
}
// 判断点是否在线段上（包括端点）
bool onS1(P p, P a, P b) {
    return sgn((p - a) / (b - a)) == 0 && sgn((p - a) * (p - b)) <= 0;
}
bool isSSr(const L&a,const L&b){ // seg x seg restrict
   T c1=(a.t-a.s)/(b.s-a.s) , c2=(a.t-a.s)/(b.t-a.s),
     c3=(b.t-b.s)/(a.s-b.s) , c4=(b.t-b.s)/(a.t-b.s);
   return sgn(c1) * sgn(c2) < 0 && sgn(c3) * sgn(c4) < 0;
}
bool isSS(L a,L b){ // seg x seg , replace x->y to accelerate
   T c1=(a.t-a.s)/(b.s-a.s),c2=(a.t-a.s)/(b.t-a.s);
   T c3=(b.t-b.s)/(a.s-b.s),c4=(b.t-b.s)/(a.t-b.s);
   return sgn(c1) * sgn(c2) <= 0 && sgn(c3) * sgn(c4) <= 0 &&
      sgn(max(a.s.x,a.t.x) - min(b.s.x,b.t.x)) >= 0 &&
      sgn(max(b.s.x,b.t.x) - min(a.s.x,a.t.x)) >= 0 &&
      sgn(max(a.s.y,a.t.y) - min(b.s.y,b.t.y)) >= 0 &&
      sgn(max(b.s.y,b.t.y) - min(a.s.y,a.t.y)) >= 0;
}
// 判断直线线段是否相交（端点也算）
bool isLS(P a1, P a2, P b1, P b2) {
    T c1 = (a2 - a1) / (b1 - a1), c2 = (a2 - a1) / (b2 - a1);
    return sgn(c1) * sgn(c2) <= 0;
}
bool inRegion(T a,T p,T b) {return sgn(a-p)==0||sgn(b-p)==0||(a<p!=b<p);}
bool inRec(P p,L a){ // p in Rectangle
   return inRegion(a.s.x,p.x,a.t.x) && inRegion(a.s.y,p.y,a.t.y);
}
db disPL(P p,L a){return fabs((a.t-a.s)/(p-a.s)) / abs(a.t-a.s);}
db disPS(P p,L a){ // p x seg dis
   if(sgn((a.t-a.s)*(p-a.s)) == -1) return abs(p-a.s);
   if(sgn((a.s-a.t)*(p-a.t)) == -1) return abs(p-a.t);
   return disPL(p,a);
}
db disSS(L a,L b){ // seg x seg dis
   if(isSS(a,b)) return 0;
   return min(min(disPS(a.s,b),disPS(a.t,b)),min(disPS(b.s,a),disPS(b.t,a)));
}
// 直线两点式转一般式
// 直线的一般式： Ax+By+C=0
void getLABC(P a, P b, T &A, T &B, T &C) {
    A = a.y - b.y;
   B = b.x - a.x;
   C = a / b;
}
typedef vector<P> polygon;
polygon convex(polygon A){ // counter-clockwise , < : <=180 , <= : <180
   int n=sz(A),m=0;
   polygon B;B.resize(n<<1);
   sort(all(A));
   rep(i,0,n){
      while(m > 1 && sgn((B[m-1]-B[m-2])/(A[i]-B[m-2]))<=0) --m;
      B[m++]=A[i];
   }
   int k = m;
   per(i,0,n-1){
      while(m > k && sgn((B[m-1]-B[m-2])/(A[i]-B[m-2]))<=0) --m;
      B[m++]=A[i];
   }
   B.resize(m);
   if(sz(B) > 1) B.pop_back();
   return B;
}
T area(polygon A) { // multiple 2 with integer type
   T res=0;
   rep(i,0,sz(A)) res+=A[i]/(A[(i+1)%sz(A)]);
   return fabs(res) / 2;
}
bool isconvex(polygon A){ // counter-clockwise
   bool ok=1;int n=sz(A);
   rep(i,0,2) A.pb(A[i]);
   rep(i,0,n) ok&=((A[i+1]-A[i])/(A[i+2]-A[i]))>=0;
   return ok;
}
int inPpolygon(P p,polygon A){ // -1 : on , 0 : out , 1 : in
   int res=0;
   rep(i,0,sz(A)){
      P u=A[i],v=A[(i+1)%sz(A)];
      if(onPS(p,u,v)) return -1;
      T cross = sgn((v-u)/(p-u)) , d1 = sgn(u.y-p.y) , d2 = sgn(v.y-p.y);
      if(cross > 0 && d1 <= 0 && d2 > 0) ++res;
      if(cross < 0 && d2 <= 0 && d1 > 0) --res;
   }
   return res != 0;
}
T diameter(polygon A) { // longest distance
   int n=sz(A);if(n <= 1) return 0;
   int l=0,r=0;rep(i,1,n) (A[i]<A[l])&&(l=i),(A[r]<A[i])&&(r=i);
   db res=abs(A[l]-A[r]);int i=l,j=r;
   do (++((A[(i+1)%n]-A[i])/(A[(j+1)%n]-A[j])>=0?j:i))%=n,
```

```
      res=max(res,abs(A[i]-A[j]));
   while(i!=l||j!=r);
   return res;
}
polygon convexCut(polygon A,P s,P t){ // counter-clockwise , left hand of st
   int n=sz(A);
   polygon B;
   rep(i,0,n){
      P u=A[i],v=A[(i+1)%n];
      int d1 = sgn((t-s)/(u-s)) , d2 = sgn((t-s)/(v-s));
      if(d1 >= 0) B.pb(u);
      if(d1 * d2 < 0) B.pb(insLL(L(u,v),L(s,t)));
   }
   return B;
}
// sz(A) <= 100,000
namespace NearestPoints{
   T solve(int l,int r,vector<P>&p){
      if(l == r) return 1e100;
      int m=(l+r)>>1;
      T Xm = p[m].x , lim = min(solve(l,m,p) , solve(m+1,r,p));
      inplace_merge(p.begin()+l,p.begin()+m+1,p.begin()+r+1,[&](P a,P b){return a.y<b.y
      ;});
      vector<P> V;
      rep(i,l,r+1) if(fabs(p[i].x - Xm) <= lim) V.pb(p[i]);
      rep(i,0,sz(V)) rep(j,i+1,sz(V)){
         if(fabs(V[j].y - V[i].y) >= lim) break;
         T dis = abs(V[i]-V[j]);
         lim = min(lim,dis);
      }
      return lim;
   }
   T solve(vector<P> A){
      sort(all(A),[&](P a,P b){return a.x<b.x;});
      return solve(0,sz(A)-1,A);
   }
}
struct C{
   P o;T r;C(){} C(P o,T r):o(o),r(r){}
   bool operator == (const C&b) const {return o==b.o&&sgn(r-b.r)==0;}
   // 通过圆心角（弧度）求圆上坐标
   P point(T rad) {return P(o.x + cos(rad) * r, o.y + sin(rad) * r);}
};
// 注意相等关系
// 相离4: 外切3: 相交2: 内切1: 内含0:
int relCC(C A,C B){
   T dis = abs(A.o - B.o);
   if(sgn(dis - (A.r + B.r)) == 1) return 4;
   if(sgn(dis - (A.r + B.r)) == 0) return 3;
   if(sgn(dis - fabs(A.r - B.r)) == 1) return 2;
   if(sgn(dis - fabs(A.r - B.r)) == 0) return 1;
   return 0;
}
vector<P> insCL(C c,L a){
   db x = (a.s-c.o)*(a.t-a.s) , y = norm(a.t-a.s);
   db d = x * x - y * (norm(a.s-c.o) - c.r*c.r);
   vector<P> res;
   if(sgn(d) < 0) return res;
   d = max(d,0.);
   P mid = a.s - (a.t - a.s) * (x / y);
   P del = (a.t - a.s) * (sqrt(d) / y);
   return {mid - del,mid + del}; // dir : a.s -> a.t
}
vector<P> insCC(C a,C b){
   vector<P> res;
   T x = norm(a.o - b.o);
   if(sgn(x)==0) return res;
   T  y = ((a.r * a.r - b.r * b.r) / x + 1) / 2 ,
      d = a.r * a.r / x - y * y;
   if(sgn(d) < 0) return res;
   d = max(d,0.);
   P mid = (b.o - a.o) * y + a.o ,
      del = ((b.o - a.o) * sqrt(d)).rot90();
   return {mid - del , mid + del};// counter-clockwise along a
}
vector<P> tanCP(C c,P p){
   db x = norm(p - c.o) , d = x - c.r * c.r;
   vector<P> res;
   if(sgn(d) < 0) return res;
   d = max(d,0.);
   P mid = c.o + (p - c.o) * (c.r * c.r / x) ,
      del = ((p-c.o)*(c.r*sqrt(d)/x)).rot90();
   return {mid - del ,mid + del}; // counter-clockwise
}
vector<pair<P,P> > tanCC(C c1,C c2){// need to unique
   vector<pair<P,P> > res;
   // extan
   if(!sgn(c1.r-c2.r)){
      P dir = c2.o-c1.o;
      dir = (dir*(c1.r/abs(dir))).rot90();
      res.pb({c1.o+dir,c2.o+dir});
      res.pb({c1.o-dir,c2.o-dir});
   } else {
      P p = (c2.o * c1.r - c1.o * c2.r) / (c1.r - c2.r);
      vector<P> ps = tanCP(c1 , p) , qs = tanCP(c2 , p);
      rep(i,0,min(sz(ps),sz(qs))) res.pb({ps[i],qs[i]});
   }
   // intan
   P p = (c1.o * c2.r + c2.o * c1.r) / (c1.r + c2.r);
   vector<P> ps = tanCP(c1 , p) , qs = tanCP(c2 , p);
   rep(i,0,min(sz(ps),sz(qs))) res.pb({ps[i],qs[i]});
   return res;
}
db areaCT(db r,P s,P t) { // need divide 2, maybe less than 0
   vector<P> p = insCL(C(P(0,0),r),L(s,t));
   if(!sz(p)) return r*r*rad(s,t);
   bool b1 = sgn(norm(s)-r*r) == 1 , b2 = sgn(norm(t)-r*r) == 1;
   if(b1 && b2) {
      if(sgn((s-p[0])*(t-p[0])) <= 0 && sgn((s-p[1])*(t-p[1])) <= 0)
         return r*r*(rad(s,p[0]) + rad(p[1],t)) + (p[0]/p[1]);
```

```
      else return r*r*rad(s,t);
   } else if(b1) return r*r*rad(s,p[0])+(p[0]/t);
   else if(b2) return r*r*rad(p[1],t)+(s/p[1]);
   return (s/t);
}
db areaCPoly(db r, polygon A) { // need divide 2, counter-clockwise
   db ans = 0;
   rep(i, 0, sz(A)) ans += areaCT(r, A[i], A[(i + 1) % sz(A)]);
   return ans;
}
P inC(P A,P B,P C){
   db a = abs(B - C) , b = abs(C - A) , c = abs(A - B);
   return (A * a + B * b + C * c) / (a + b + c);
}
P outC(P A,P B,P C){
   P b = B - A , c = C - A;
   db dB = norm(b) , dC = norm(c) , d = b / c * 2;
   return A - P(b.y * dC - c.y * dB , c.x * dB - b.x * dC) / d;
}
P othroC(P A,P B,P C){
   P b = B - A , c = C - A;
   db Y = b.y * c.y * (B - C).y,
      a = c / b,
      xx = (Y + c.x * b.y * B.x - b.x * c.y * C.x) / a,
      yy = -b.x * (xx - C.x) / b.y + c.y;
   return P(xx , yy);
}
C Mincir(P *p,int n){
   random_shuffle(p , p + n);
   P o = p[0];db r = 0;
   rep(i,1,n) {
      if(sgn(abs(o-p[i])-r) <= 0) continue;
      o = p[i] , r = 0;
      rep(j,0,i) {
         if(sgn(abs(o-p[j])-r) <= 0) continue;
         o = (p[i] + p[j]) / 2 , r = abs(o-p[j]);
         rep(k,0,j) {
            if(sgn(abs(o-p[k])-r) <= 0) continue;
            o = outC(p[i],p[j],p[k]) , r = abs(o-p[k]);
         }}}
   return C(o,r);
}
namespace CircleIntersection{
   struct E{
      P p;T ang;int delta;
      E(){} E(P p,T ang,int delta):p(p),ang(ang),delta(delta){}
      bool operator < (const E&b) const {return ang<b.ang;}
   };
   bool overlap(C a,C b) {return sgn(a.r-b.r-abs(a.o-b.o))>=0;}
   void solve(C *c,int n,T *ans) {
      memset(ans , 0 , sizeof(T) * (n + 1));
      rep(i,0,n) {
         int cnt=1;
         vector<E> evt;
         rep(j,0,i) if(c[i]==c[j]) cnt++;
         rep(j,0,n) if(j!=i&&!(c[i]==c[j])&&overlap(c[j],c[i])) cnt++;
         rep(j,0,n) if(j!=i){
            vector<P> pts=insCC(c[i],c[j]);
            if(sz(pts)) {
               T a[2];
               rep(j,0,2) a[j]=(pts[j]-c[i].o).arg();
               evt.pb(E(pts[0],a[0],1));
               evt.pb(E(pts[1],a[1],-1));
               cnt += a[0] > a[1];
            }
         }
         if(!sz(evt)) ans[cnt] += pi*c[i].r*c[i].r;
         else{
            sort(all(evt));
            evt.pb(evt.front());
            rep(j,0,sz(evt)-1) {
               cnt+=evt[j].delta;
               ans[cnt] += evt[j].p / evt[j+1].p / 2;
               db ang = evt[j + 1].ang - evt[j].ang;
               if(ang < 0) ang += pi * 2;
               ans[cnt] += ang * c[i].r * c[i].r / 2 - sin(ang) * c[i].r * c[i].r / 2;
            }}}}
namespace ConvecIntersection{
   const int N = 1005;
   struct Rec {
      P d[10];int dn;// d[dn] = d[0]
      P operator [] (const int&n) {return d[n];}
   }r[N];
   typedef pair<db,int> pdi;
   int n;pdi res[1000005];
   db getLoc(P a,P b,P p){
      if(sgn(b.x - a.x)) return (p.x - a.x) / (b.x - a.x);
      return (p.y - a.y) / (b.y - a.y);
   }
   db work() {
      db rt=0;
      rep(i,0,n) rep(j,0,r[i].dn){
         int sz=0;
         res[sz++] = pdi(0,0);res[sz++] = pdi(1,0);
         rep(t,0,n) {
            if(t == i) continue;
            rep(g,0,r[t].dn) {
               int du = sgn((r[i][j+1] - r[i][j]) / (r[t][g] - r[i][j]));
               int dv = sgn((r[i][j+1] - r[i][j]) / (r[t][g+1] - r[i][j]));
               if(!du && !dv) {
                  if(sgn((r[i][j+1] - r[i][j]) * (r[t][g+1] - r[t][g])) < 0 || i < t){
                     res[sz++] = pdi(getLoc(r[i][j] , r[i][j+1] , r[t][g]) , 1);
                     res[sz++] = pdi(getLoc(r[i][j] , r[i][j+1] , r[t][g+1]) , -1);
                  }} else {
                     db s1 = (r[i][j] - r[t][g]) / (r[t][g+1] - r[t][g]);
                     db s2 = (r[t][g+1] - r[t][g]) / (r[i][j+1] - r[t][g]);
                     if(du >= 0 && dv < 0) res[sz++] = pdi(s1 / (s1 + s2) , 1);
                     else if(du < 0 && dv >= 0) res[sz++] = pdi(s1 / (s1 + s2) , -1);
                  }}}
         sort(res , res + sz);
```

```
        int cnt = 0; --sz;
        rep(t,0,sz) {
           cnt += res[t].se;
           if(cnt == 0 && sgn(res[t].fi - res[t+1].fi)) {
              db a = res[t].fi;
              if(a < 0) a = 0; if(a > 1) break;
              db b = res[t+1].fi;
              if(b < 0) continue; if(b > 1) b = 1;
              rt += ((r[i][j+1] - r[i][j]) * a + r[i][j]) / ((r[i][j+1]-r[i][j]) * b +
              r[i][j]);
           }}}
      return rt / 2;}}
```

1.3 Geo3D

```
// didn't vertify
typedef double T;
const T eps = 1e-8;
int sgn(T x){return (x>eps)-(x<-eps);}
struct P{
  T x,y,z;P(){} P(T x,T y,T z):x(x),y(y),z(z){}
  P operator - (const P&b) const {return P(x-b.x,y-b.y,z-b.z);}
  P operator + (const P&b) const {return P(x+b.x,y+b.y,z+b.z);}
  P operator * (const T&k) const {return P(x*k,y*k,z*k);}
  P operator / (const T&k) const {return P(x/k,y/k,z/k);}
  T operator * (const P&b) const {return x*b.x+y*b.y+z*b.z;}
  P operator / (const P&b) const {return P(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);}
  bool operator < (const P&b) const {return tie(x,y,z)<tie(b.x,b.y,b.z);}
  bool operator == (const P&b) const {return sgn(x-b.x)==0&&sgn(y-b.y)==0&&sgn(z-b.z)
  ==0;}
  T len() const {return sqrtl(x*x+y*y+z*z);}
};
T norm(P a){return a*a;}
P outC(P A,P B,P C){
  T d = 2 * norm((A-B)/(B-C));
  if(sgn(d) == 0) {
    if((B-A).len()<(C-A).len()) swap(B,C);
    if((B-A).len()<(C-B).len()) swap(A,C);
    return (A+B)/2;
  }
  T a = (A-B)*(A-C)*norm(B-C);
  T b = (B-C)*(B-A)*norm(C-A);
  T c = (C-A)*(C-B)*norm(A-B);
  return (A*a+B*b+C*c)/d;
}
T mix(P a,P b,P c){return a / b * c;}
T area(P a,P b,P c){return ((b - a) / (c - a)).len();}
namespace Convex{
  typedef tuple<int,int,int> F;
  const int N = 1010;
  int mark[N][N] , n , cnt;
  vector<F> face;// (p[a]-p[b])/(p[c]-p[b]) inward ?
  void build(vector<P> p){
    sort(all(p));p.erase(unique(all(p)),p.end());
    n = sz(p);
    random_shuffle(all(p));
    face.clear();
    auto volume = [&](int a,int b,int c,int d)
    {return mix(p[b]-p[a],p[c]-p[a],p[d]-p[a]);};
    auto insert = [&](int a,int b,int c)
    {face.pb(make_tuple(a,b,c));};
    auto find = [&](){
      rep(i,2,n){
        P dir = (p[0] - p[i]) / (p[1] - p[i]);
        if(dir == P(0 , 0 , 0)) continue;
        swap(p[i] , p[2]);
        rep(j,i+1,n) if(sgn(volume(0,1,2,j))) {
          swap(p[j],p[3]);
          insert(0,1,2);
          insert(0,2,1);
          return 1;
        }
      }
      return 0;
    };
    auto add = [&](int d){
      vector<F> tmp;
      int a, b, c;
      cnt++;
      for(auto f : face){
        tie(a , b , c) = f;
        if (sgn(volume(d, a, b, c)) < 0)
          mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b]
            = mark[c][a] = mark[a][c] = cnt;
        else tmp.pb(f);
      }
      face = tmp;
      for(auto f : tmp){
        tie(a , b , c) = f;
        if (mark[a][b] == cnt) insert(b,a,d);
        if (mark[b][c] == cnt) insert(c,b,d);
        if (mark[c][a] == cnt) insert(a,c,d);
      }
    };
    if(find()){
      rep(i,0,n) memset(mark[i],0,sizeof(int)*n);
      cnt = 0;
      rep(i,3,n) add(i);
    }}}
struct L{ P a,b; L(){} L(P a,P b):a(a),b(b){}};
struct PL{
  P a,b,c;
  PL(){} PL(P a,P b,P c):a(a),b(b),c(c){}
  P pvec() {return (b-a)/(c-a);}
  T area() {return pvec().len();}
};
T dis(P a,P b){return (b-a).len();}
bool PonL(P a,L l) {return sgn(((l.b-l.a)/(a-l.a)).len())==0;}
bool PonS(P a,L l){
  return PonL(a,l) &&
```

```
    sgn((l.a.x-a.x)*(l.b.x-a.x)) <= 0 &&
    sgn((l.a.y-a.y)*(l.b.y-a.y)) <= 0 &&
    sgn((l.a.z-a.z)*(l.b.z-a.z)) <= 0;
}
bool PonPL(P a,PL s) {return sgn(s.pvec()*(a-s.a))==0;}
bool PonTri(P a,PL s){
  return sgn(s.area()-PL(a,s.a,s.b).area()-PL(a,s.b,s.c).area()-PL(a,s.c,s.a).area())
  ==0;
}
bool PonPL(vector<P> p){ // distinct points
  int n=sz(p);
  if(n<4) return true;
  int c=-1;rep(i,2,n) if(!PonL(p[0],L(p[1],p[i]))) {c=i;break;}
  if(c==-1) return true;
  PL s(p[0],p[1],p[c]);
  rep(i,2,n) if(!PonPL(p[i],s)) return false;
  return true;
}
bool same_side(P a,P b,L l){ // coplanar, sgn(pvec()) to prove precision
  return sgn(PL(l.a,l.b,a).pvec()*PL(l.a,l.b,b).pvec()) > 0;
}
bool same_side(P a,P b,PL s){
  return sgn((s.pvec()*(a-s.a))*(s.pvec()*(b-s.a))) > 0;
}
bool opposite_side(P a,P b,L l){ // coplanar, sgn(pvec()) to prove precision
  return sgn(PL(l.a,l.b,a).pvec()*PL(l.a,l.b,b).pvec()) < 0;
}
bool opposite_side(P a,P b,PL s){
  return sgn((s.pvec()*(a-s.a))*(s.pvec()*(b-s.a))) < 0;
}
bool isSSr(L u,L v){
  return PonPL(u.a,PL(u.b,v.a,v.b)) &&
    opposite_side(u.a,u.b,v) && opposite_side(v.a,v.b,u);
}
bool isSS(L u,L v){
  if(!PonPL(u.a,PL(u.b,v.a,v.b))) return false;
  if(!PonL(u.a,v) || !PonL(u.b,v)) return isSSr(u,v);
  return PonS(u.a,v) || PonS(u.b,v) || PonS(v.a,u) || PonS(v.b,u);
}
bool isSTri(L l,PL s){ // can't coplanar
  return !same_side(l.a,l.b,s) && !same_side(s.a,s.b,PL(l.a,l.b,s.c)) &&
    !same_side(s.b,s.c,PL(l.a,l.b,s.a)) && !same_side(s.c,s.a,PL(l.a,l.b,s.b));
}
bool isSTrir(L l,PL s){
  return opposite_side(l.a,l.b,s) && opposite_side(s.a,s.b,PL(l.a,l.b,s.c)) &&
    opposite_side(s.b,s.c,PL(l.a,l.b,s.a)) && opposite_side(s.c,s.a,PL(l.a,l.b,s.b));
}
// | parallel , ^ perpendicular , & intersection
bool operator | (L a,L b) {return sgn(((a.b-a.a)/(b.b-b.a)).len())==0;}
bool operator ^ (L a,L b) {return sgn((a.b-a.a)*(b.b-b.a))==0;}
bool operator | (PL a,PL b) {return sgn((a.pvec()/b.pvec()).len())==0;}
bool operator ^ (PL a,PL b) {return sgn(a.pvec()*b.pvec())==0;}
bool operator | (L l,PL s) {return sgn(((l.b-l.a)/s.pvec()).len())==0;}
bool operator ^ (L l,PL s) {return sgn((l.b-l.a)*s.pvec())==0;}
P operator & (L a,L b) { // can't parallel
  P s = a.a - b.a , v = a.b - a.a , w = b.b - b.a;
  db k = (s / w) * (w / v) / ((w / v) * (w / v));
  return a.a + v * k;
}
P operator & (L l,PL s){ // can't parallel
  return l.a+(l.b-l.a)*((s.pvec()*(s.a-l.a))/(s.pvec()*(l.b-l.a)));
}
L operator & (PL s,PL t){ // can't parallel
  P a=L(s.a,(L(s.a,s.b)|t)?s.c:s.b)&t;
  return L(a,a+s.pvec()/t.pvec());
}
P PtoS(P a,L l){
  P b=l.a,c=l.b;
  rep(i,0,50) {
    P d=(b+c)*0.5,e=(d+c)*0.5;
    if(dis(a,d)<=dis(a,e)) b=e;
    else c=d;
  }
  return b;
}
// - distance , + projection
T operator - (P a,L l) {return ((a-l.a)/(l.b-l.a)).len() / dis(l.a,l.b);}
P operator + (P a,L l) {P s=l.a,d=l.b-l.a;return s+d*((a-s)*d/(d*d));}
T operator - (P a,PL s) {return fabs((a-s.a)*s.pvec()/s.pvec().len());}
P operator + (P a,PL s) {P d=s.pvec();return a+d*((s.a-a)*d/(d*d));}
T operator - (L u,L v) {P t=(u.b-u.a)/(v.b-v.a);return fabs((v.a-u.a)*t/t.len());}
P operator + (L a,L b) {return a & b;}
db angle(P a,P b) {return acos(max(-1.,min(1.,a*b/a.len()/b.len())));}
db angle(PL a,PL b) {return angle(a.pvec(),b.pvec());}
db angle(L l,PL s) {return asin(max(-1.,min(1.,(l.b-l.a)*s.pvec()/(l.b-l.a).len()/s.pvec
().len())));}
struct SP{
  P o;T r;
  T dis(P a,P b){ return angle(a-o,b-o)*r;}
  P to(T lng,T lat){ return P(cos(lng)*cos(lat)*r,sin(lng)*cos(lat)*r,sin(lat)*r);}
};
vector<P> operator & (L l,SP sp){
  if(sgn((sp.o-l)-sp.r)>0) return vector<P>();
  P s=l.a,d=l.b-l.a;
  T A=d*d,B=(s-sp.o)*d*2,C=(s-sp.o)*(s-sp.o)-sp.r*sp.r;
  T delta=sqrt(max(0.,B*B-4*A*C)),k1=(-B-delta)/(2*A),k2=(-B+delta)/(2*A);
  return {s+d*k1,s+d*k2};
}
```

1.4 Hull

```
typedef complex<ll> P;
typedef map<ll,P> hull;
#define X real()
#define Y imag()
ll cross(const P&a,const P&b){return (conj(a)*b).Y;}
struct Hull{
  hull h1,h2;
  bool in(hull&h,ll x,ll y){
    if(!sz(h)) return false;
```

```
    if(x < h.begin()->se.X || x > h.rbegin()->se.X) return false;
    auto l = h.lower_bound(x);
    if(x == l->se.X) return y <= l->se.Y;
    auto r = l--;
    return cross(r->se - l->se , P(x,y) - l->se) <= 0;
  }
  void ins(hull&h,ll x,ll y){
    if(in(h , x , y)) return;
    P p(x,y);h[x] = p;
    auto LL = h.find(x) , RR = LL , L = LL , R = L;
    if(L != h.begin()) for(--LL;(L = LL) != h.begin();){
      --(LL = L);
      if(cross(p - LL->se , L->se - LL->se) <= 0) h.erase(L);
      else break;
    }
    if(*R != *h.rbegin()) for(++RR;*(R = RR) != *h.rbegin();){
      ++(RR = R);
      if(cross(p - RR->se , R->se - RR->se) >= 0) h.erase(R);
      else break;
    }
  }
  void ins(ll x,ll y){ ins(h1,x,y);ins(h2,x,-y); }
  bool in(ll x,ll y){ return in(h1,x,y) && in(h2,x,-y); }
};
```