计算几何：

1.1

Point定义

**const double** eps = 1e-8;

**const double** PI = acos(-1.0);

**int** sgn(**double** x) {

**if** (fabs(x) < eps)**return** 0;

**if** (x < 0)**return** -1; **else return** 1;

}

**struct** Point {

**double** x, y;

Point() {}

Point(**double** \_x, **double** \_y) {

x = \_x;

y = \_y;

}

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; }

绕原点旋转角度B（弧度值），后x,y的变化 **void** transXY(**double** B) {

**double** tx = x, ty = y;

x = tx \* cos(B) - ty \* sin(B);

y = tx \* sin(B) + ty \* cos(B);

}

};

1.2 Line定义

**struct** Line {

Point s, e;

Line() {}

Line(Point \_s, Point \_e) {

s = \_s;

e = \_e;

}

*/\**

两直线求交:

s=0,直线重合.

s=1,直线平行.

s=2,直线相交.

*\*/*

仅当s=2时,交点才有意义.

pair<**int**, Point>

**operator**&(**const** Line &b) **const** {

Point res = s;

**if** (sgn((s - e) ^ (b.s - b.e)) == 0) {

**if** (sgn((s - b.e) ^ (b.s - b.e)) == 0)

**return** make\_pair(0, res); *//*重合

**else return** make\_pair(1, res); *//*平行

}

**double** t = ((s - b.s) ^ (b.s - b.e)) / ((s - e) ^ (b.s - b.e));

res.x += (e.x - s.x) \* t;

res.y += (e.y - s.y) \* t;

**return** make\_pair(2, res);

}

};

1.3 判断线段交:

**bool** inter(Line l1, Line l2) {

**return** max(l1.s.x, l1.e.x) >= min(l2.s.x, l2.e.x) && max(l2.s.x, l2.e.x) >= min(l1.s.x, l1.e.x) &&

max(l1.s.y, l1.e.y) >= min(l2.s.y, l2.e.y) && max(l2.s.y, l2.e.y) >= min(l1.s.y, l1.e.y) &&

sgn((l2.s - l1.e) ^ (l1.s - l1.e)) \* sgn((l2.e - l1.e) ^ (l1.s - l1.e)) <= 0 &&

sgn((l1.s - l2.e) ^ (l2.s - l2.e)) \* sgn((l1.e - l2.e) ^ (l2.s - l2.e)) <= 0;

}

1.4:判断直线l1 和线段l2是否相交:

**bool** ZXJ(Line l1, Line l2)

{

return sgn((l2.s - l1.e )^(l1.s - l1.e ))\*sgn((l2.e - l1.e )^(l1.s - l1.e )) <= 0;

}

1.5:点到线段距离

Point DD ( Point P, Line L)

{

Point result;

**double** t = ((P - L.s )\*(L.e - L.s ))/((L.e - L.s )\*(L.e - L.s ));

**if** (t >= 0 && t <= 1)

{

result.x = L.s .x + (L.e .x - L.s .x )\*t;

result.y = L.s .y + (L.e .y - L.s .y )\*t;

}

**else**

{

**if** (dist(P,L.s ) < dist(P,L.e ))

result = L.s ;

**else** result = L.e ;

}

**return** result;

}

1.6:多边形面积：

{

**double** res = 0;

**for** ( **int** i = 0;i < n;i++) res += (p[i]^p[(i+1)%n])/2;

**return** fabs (res);

}

1.7：判断点是否在线段上：

{

**return** sgn((L.s - P)^(L.e - P)) == 0 &&

sgn((P.x - L.s .x ) \* (P.x - L.e .x )) <= 0

&&sgn((P.y - L.s .y ) \* (P.y - L.e .y )) <= 0;

}

1.8: 判断点在凸多边形内:

*/\**

点形成一个凸包，而且按逆时针排序（如果是顺时针把里面的 *<0* 改为 *>0* ）

*\*/*

*/\**

点的编号*:0~n-1.*

返回值：

*- 1:* 点在凸多边形外

*0:* 点在凸多边形边界上

*1:* 点在凸多边形内

*\*/*

**int** inConvexPoly(Point a, Point p[], **int** n) {

**for** (**int** i = 0; i < n; i++) {

**if** (sgn((p[i] - a) ^ (p[(i + 1) % n] - a)) < 0) **return** -1;

**else if** (OnSeg(a, Line(p[i], p[(i + 1) % n]))) **return** 0;

}

**return** 1;

}

1、外心：

Point waixin ( Point a, Point b, Point c)

{

**double** a1 = b. x - a. x , b1 = b. y - a. y , c1 = (a1\*a1 + b1\*b1)/2;

**double** a2 = c. x - a. x , b2 = c. y - a. y , c2 = (a2\*a2 + b2\*b2)/2;

**double** d = a1\*b2 - a2\*b1;

**return** Point (a. x + (c1\*b2 - c2\*b1)/d, a. y + (a1\*c2 - a2\*c1)/d);

}

最近点对：

. . .

typedef pair<db,db> P;

P A[N];

bool cmp(P a,P b) {

return a.second < b.second;

}

db cp(P \*a, int n){

if(n<=1) return 1.0\*mod\*mod;

int m=n/2;

db x=a[m].first;

db d=min(cp(a,m),cp(a+m,n-m));

inplace\_merge(a,a+m,a+n,cmp);//归并排序

vector<P> b;

for(int i=0;i<n;i++){

if(fabs(a[i].first-x) >= d) continue;

for(int j=0;j<b.size();j++){

db dx=a[i].first -b[b.size()-1-j].first;

db dy=a[i].second-b[b.size()-1-j].second;

if(dy>=d) break;

d=min(d,sqrt(dx\*dx+dy\*dy));

}

b.push\_back(a[i]);

}

return d;

}

int main()

{

int n;

while(scanf("%d",&n)&&n!=0)

{

for(int i=0;i<n;i++) cd(A[i].first),cd(A[i].second);

sort(A,A+n);

db ans=cp(A,n);

printf("%.2f\n",ans);

}

return 0;

}

求凸包：

* 找凸包，建立一圈围墙，要求围墙上每个点，距离凸包上最近点大于等于l，问围墙的周长
* 就是求个凸包，再加上以L为半径的圆周长，看题中图很清楚

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = 1005;

#define M\_PI 3.1415926535

struct P{

int x,y;

};

int st[N],top;

P a[N];

int rk[N];

int n,T,l;

ll cross(const P &a,const P &b,const P &c){

return 1ll\*(b.x-a.x)\*(c.y-a.y)-1ll\*(c.x-a.x)\*(b.y-a.y);

}

ll cross(int x,int y,int z){

return cross(a[x],a[y],a[z]);

}

double dis(const P &a,const P &b){

return sqrt(1.0\*(a.x-b.x)\*(a.x-b.x)+1.0\*(a.y-b.y)\*(a.y-b.y));

}

bool cmp(int x,int y){

ll m = cross(a[rk[0]],a[x],a[y]);

if (m>0)return 1;

else if (m==0&&dis(a[rk[0]],a[x])<=dis(a[rk[0]],a[y]))return 1;

else return 0;

}

void solve(){

scanf("%d%d",&n,&l);

for (int i=0;i<n;i++){

scanf("%d%d",&a[i].x,&a[i].y);

rk[i]=i;

}

for (int i=1;i<n;i++){

if (a[rk[i]].y<a[rk[0]].y||a[rk[i]].y==a[rk[0]].y&&a[rk[i]].x<a[rk[0]].x)swap(rk[i],rk[0]);

}

sort(rk+1,rk+n,cmp);

top=2;

st[0]=rk[0];

st[1]=rk[1];

for (int i=2;i<n;i++){

while (cross(st[top-2],st[top-1],rk[i])<0)top--;

st[top++] =rk[i];

}

double ans =0;

for (int i=1;i<top;i++){

ans+=dis(a[st[i]],a[st[i-1]]);

}

ans+=dis(a[st[0]],a[st[top-1]]);

ans+=2\*M\_PI\*l;

printf("%.0lf\n",ans);

}

int main(){

scanf("%d",&T);

while (T--){

solve();

if (T!=0)printf("\n");

}

return 0;

}

圆的反演：

typedef double db;

const db PI = 3.14159265358979323846;

const db eps = 1E-10;

const db R2 = 1.0;

int sgn(db x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }

struct P {

db x, y;

P(db x = 0, db y = 0): x(x), y(y) {}

P operator \* (db k) { return P(x \* k, y \* k); }

P operator / (db k) { return P(x / k, y / k); }

string prt() const {

char s[100];

sprintf(s, "(%.2f, %.2f)", x, y);

return string(s);

}

};

typedef P V;

P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }

struct C {

P p;

db r;

C(db x = 0, db y = 0, db r = 0): p(x, y), r(r) {}

};

db dist(V v) { return sqrt(v.x \* v.x + v.y \* v.y); }

C inv(C c, const P& o) {

db d = dist(c.p - o);

assert(sgn(d) != 0);

db a = 1 / (d - c.r);

db b = 1 / (d + c.r);

c.r = (a - b) / 2 \* R2;

c.p = o + (c.p - o) \* ((a + b) \* R2 / 2 / d);

return c;

}