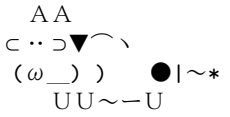
Nocow Library

へクト

July 14, 2017



## Contents



# 1 幾何

## 1.1 注意事項

- sgn 関数の定義はちゃんと写す
- istream を使う時は p の代入を忘れない
- 交点を求める前に交差判定が必要 iss(a,b) ill(a,b) == parallel(a,b)
- angle の定義には気をつける
- complex の比較関数は namespace std の中で書く

### 1.2 ベクトル

```
// Description: ベクトル
// Verifyed: various problem
using namespace placeholders;
using R = long double;
const R EPS = 1e-9L; // [-1000,1000]->EPS=1e-8 [-10000,10000]->EPS=1e-7
inline int sgn(const R& r) {return (r > EPS) - (r < -EPS);}
inline R sq(R x) {return sqrt(max(x, 0.0L));}

const R INF = 1E40L;
const R PI = acos(-1.0L);
using P = complex<R>;
using L = struct {P s, t;};
using VP = vector<P>;
using C = struct {P c; R r;};

#define at(a,i) (a[(i + a.size()) % a.size()])

auto& operator >> (istream& is, P& p) { R x, y; is >> x >> y, p = P(x, y); return is;}
auto& operator << (ostream& os, P& p) { os << real(p) << " " << imag(p); return os;}

namespace std {
bool operator <  (const P& a, const P& b) { return sgn(real(a - b)) ? real(a - b) < 0 : sgn(imag(a - b)) < 0;}
bool operator == (const P& a, const P& b) { return sgn(real(a - b)) == 0 && sgn(imag(a - b)) == 0;}
}

inline R dot(P o, P a, P b) {return real(conj(a - o) * (b - o));}
inline R det(P o, P a, P b) {return imag(conj(a - o) * (b - o));}
inline P vec(L l) {return l.t - l.s;}
auto sdot = bind(sgn, bind(dot, _1, _2, _3));
auto sdet = bind(sgn, bind(det, _1, _2, _3));

//projection verify AOJ CGL_1_A
P proj(L l, P p) { R u = real((p - l.s) / vec(l)); return (1 - u) * l.s + u * l.t;}
```

### 1.3 点集合

#### 1.3.1 凸包

```
// convex_hull Verify AOJ CGL_4_A
VP convex_hull(VP pol){
    int n=pol.size(),k=0,t=1;

```

```
    auto cmp_x=[](P a,P b)->bool{
        int sr = sgn(real(a-b)), si=sgn(imag(a-b));
        return sr ? sr < 0:si < 0;
    };

    sort(begin(pol),end(pol),cmp_x);
    VP res(2*n);

    auto push = [&](P p)->void{
        while(k>t and sdet(res[k-1],res[k-2],p)<=-1) k--;
        res[k++]=p;
    };

    for_each(begin(pol),end(pol),push);
    t = k;
    for_each(rbegin(pol)+1,rend(pol),push);
    res.resize(k-1);
    return res;
}
```

### 1.3.2 最近点对

```
// closest point pair Verify AOJ CGL_5_A
R cpp(VP a, int flag = 1) {
    const int n = a.size(), m = n / 2;
    if (n <= 1) return INF;

    auto cmp_x = [](P a, P b)->bool{
        int sr = sgn(real(a - b)), si = sgn(imag(a - b));
        return sr ? sr < 0 : si < 0;
    };

    if (flag) sort(begin(a), end(a), cmp_x);

    VP b(begin(a), begin(a) + m), c(begin(a) + m, end(a));
    R x = real(a[m]), d = min(cpp(b, 0), cpp(c, 0));


    auto cmp_y = [](P a, P b)->bool{
        int sr = sgn(real(a - b)), si = sgn(imag(a - b));
        return si ? si < 0 : sr < 0;
    };

    sort(begin(a), end(a), cmp_y);
```

```
    deque<P> e;

    for (auto &p : a) {
        if (abs(real(p) - x) >= d) continue;

        for (auto &q : e) {
            if (imag(p - q) >= d) break;
            d = min(d, abs(p - q));
        }
        e.push_front(p);
    }
    return d;
}
```

### 1.3.3 最遠点対

```
// farthest point pair Verify AOJ CGL_4_B
R fpp(VP pol) {
    int n = pol.size(), i = 0, j = 0;
    if (n <= 2) return abs(pol[0] - pol[1]);
    R res = 0.0;

    auto cmp_x = [](P a, P b)->bool{
        int sr = sgn(real(a - b)), si = sgn(imag(a - b));
        return sr ? sr < 0 : si < 0;
    };

    rep(k, n) {
        if (!cmp_x(pol[i], pol[k]))i = k;
        if (cmp_x(pol[j], pol[k]))j = k;
    }

    int si = i, sj = j;
    while (i != sj || j != si) {
        res = max(res, abs(pol[i] - pol[j]));
        P li = vec(L{at(pol, i), at(pol, i + 1)});
        P lj = vec(L{at(pol, j), at(pol, j + 1)});
        if(sdet(0, li, lj) < 0)
            i = (i + 1) % n;
        else
            j = (j + 1) % n;
    }
    return res;
}
```

### 1.4 直線と線分

```
// vertical parallel
// verified: AOJ CGL_2_A
bool vertical(L a, L b) {return sdot(0, vec(a), vec(b)) == 0;}
bool parallel(L a, L b) {return sdet(0, vec(a), vec(b)) == 0;}
bool eql(L a, L b) { return parallel(a, b) and sdet(a.s, a.t, b.s) == 0;}

// crossing determination
// verified: AOJ CGL_2_B
bool iss(L a, L b) {
    int sa = sdet(a.s, a.t, b.s) * sdet(a.s, a.t, b.t);
    int sb = sdet(b.s, b.t, a.s) * sdet(b.s, b.t, a.t);
    return max(sa, sb) < 0;
}

// crossing point
// verified: AOJ CGL_2_C
P cross(L a, L b) {
    R u = det(a.s, b.s, b.t) / det(0, vec(a), vec(b));
    return (1 - u) * a.s + u * a.t;
}

// distance
// verified: AOJ CGL_2_D
R dsp(L l, P p) {
    P h = proj(l, p);
    if (sdot(l.s, l.t, p) <= 0) h = l.s;
    if (sdot(l.t, l.s, p) <= 0) h = l.t;
    return abs(p - h);
}

R dss(L a, L b) {
    if(iss(a,b)) return 0;
    return min({dsp(a, b.s), dsp(a, b.t), dsp(b, a.s), dsp(b, a.t)});
}
```

### 1.5 多角形

```
// Polygon

```

```cpp
// area
// verified: AOJ 1100 CGL_3_A
R area(const VP& pol) {
    R sum = 0.0;
    rep(i, pol.size()) sum += det(0, at(pol, i), at(pol, i + 1));
    return abs(sum / 2.0L);
}

// convex_polygon determination
// verified: CGL_3_B
bool is_convex(const VP& pol) {
    rep(i, pol.size()){
        if(sdet(at(pol, i), at(pol, i + 1), at(pol, i + 2)) < 0){
            return false;
        }
    }
    return true;
}

// polygon realation determination  in  2 on 1 out 0   (possible non-convex)
// verified: AOJ CGL_3-C
int in_polygon(const VP& pol, const P& p) {
    int res = 0;
    auto simag = [](const P & p) {return sgn(imag(p));};
    rep(i, pol.size()) {
        P a = at(pol, i), b = at(pol, i + 1);
        if (sdet(p, a, b) == 0 and sdot(p, a, b) <= 0) return 1;
        bool f = simag(p - a) >= 0, s = simag(p - b) < 0;
        if (simag(b - a)*sdet(a, b, p) == 1 and f == s) res += (2 * f - 1);
    }
    return res ? 2 : 0;
}

// polygon realation determination   (possible non-convex)
// verified: not AOJ 2514
bool in_polygon(const VP& pol, const L& l) {
    VP check = {l.s, l.t};
    rep(i, pol.size()) {
        L edge = {at(pol, i), at(pol, i + 1)};
        if (iss(l, edge)) check.emplace_back(cross(l, edge));
    }

    auto cmp_x = [](P a, P b)->bool{
        int sr = sgn(real(a - b)), si = sgn(imag(a - b));
        return sr ? sr < 0 : si < 0;
    };
```

```
    sort(begin(check), end(check), cmp_x);
    rep(i, check.size() - 1) {
        P m = (at(check, i) + at(check, i + 1)) / 2.0L;
        if (in_polygon(pol, m) == false) return false;
    }
    return true;
}

// convex_cut
// verified: AOJ CGL_4_C
VP convex_cut(const VP& pol, const L& l) {
    VP res;
    rep(i, pol.size()) {
        P a = at(pol, i), b = at(pol, i + 1);
        int da = sdet(l.s, l.t, a), db = sdet(l.s, l.t, b);
        if (da >= 0) res.emplace_back(a);
        if (da * db < 0) res.emplace_back(cross({a, b}, l));
    }
    return res;
}
```

### 1.6 円

```
// Circle // verified: AOJ 1183
enum RCC {OUT = 2, ON_OUT = 1, ISC = 0, ON_IN = -1, IN = -2};
int rcc(C a, C b) {
    R d = abs(a.c - b.c);
    return sgn(d - a.r - b.r) + sgn(d - abs(a.r - b.r));
}

// circle crossing determination
bool icp(C c, P p, int end = 0) {return sgn(abs(p - c.c) - c.r) <= -end;}
bool ics(C c, L s, int end = 0) {
    if (sgn(dsp(s, c.c) - c.r) > end) return false;
    if (icp(c, s.s, end) and icp(c, s.t, end)) return false;
    return true;
}
// common area between circles
R area(C a, C b) {
    int r = rcc(a, b);
    if (r >= ON_OUT) return 0.0L;
    if (r <= ON_IN) return min(norm(a.r), norm(b.r)) * PI;
    R d = abs(b.c - a.c), rc = (norm(d) + norm(a.r) - norm(b.r)) / (2.0 * d);
```

```
    R t = acos(rc / a.r), p = acos((d - rc) / b.r);
    return norm(a.r) * t + norm(b.r) * p - d * a.r * sin(t);
}

// cross point between circle and line
// verified: AOJ CGL_7_D
P cir(C c, R t) {return c.c + polar(c.r, t);}
VP cross(C c, L l) {
    P h = proj(l, c.c);
    P e = polar(sq(norm(c.r) - norm(h - c.c)), arg(vec(l)));
    return VP{h - e, h + e};
}

// cross point between circles
// verified: AOJ CGL_7_E
VP cross(C a, C b) {
    P d = b.c - a.c;
    P w = (norm(d) + norm(a.r) - norm(b.r)) / (2.0L * norm(d)) * d;
    return cross(a, {a.c + w, a.c + w + 1il * d});
}

// circle tangent
// verified: AOJ CGL_7_F
L tan(C c, P p) {return L{p, p + 1il * (p - c.c)};}

P helper(C c, P d, R r, P j) {
    P tmp = sq(norm(d) - norm(r)) * j;
    P dir = (r + tmp) / norm(d) * d;
    return c.c + c.r * dir;
}

VP contact(C c, P p) {
    VP ret;
    P d = p - c.c;
    for (P j : { -1il, 1il}) ret.emplace_back(helper(c, d, c.r, j));
    sort(begin(ret), end(ret));
    ret.erase(unique(begin(ret), end(ret)), end(ret));
    return ret;
}

// circle tangent
// Verified: AOJ CGL_7_G
VP contact(C a, C b) {
    VP ret;
    P d = b.c - a.c;
    for (int s : { -1, 1}) {
```

```
        if (rcc(a, b) >= s) {
            for (P j : { -1i, 1i}) {
                R r = a.r + s * b.r;
                ret.emplace_back(helper(a, d, r, j));
            }
        }
    }
    sort(begin(ret), end(ret));
    ret.erase(unique(begin(ret), end(ret)), end(ret));
    return ret;
}

// common area of circle and polygon
// verified: AOJ CGL_7_H
R area_helper(C c, P a, P b) {
    if (icp(c, a) and icp(c, b)) return det(0, a, b) / 2.0l;
    return norm(c.r) * arg(conj(a) * b) / 2.0l;
}

R area(C c, P a, P b) {
    L l = {a, b};

    if (sgn(min({c.r, abs(a), abs(b), abs(b - a)})) == 0) return 0.0;
    if (ics(c, l) == false) return area_helper(c, a, b);

    R res = 0.0; VP ary;
    ary.push_back(a);
    for (auto &p : cross(c, l)) if (sdot(p, a, b) < 0) ary.push_back(p);
    ary.push_back(b);

    rep(i, ary.size() - 1) res += area_helper(c, at(ary, i), at(ary, i + 1));
    return res;
}

R area(C c, VP pol) {
    R res = 0;
    rep(i, pol.size()) {
        P a = at(pol, i) - c.c , b = at(pol, i + 1) - c.c;
        res += area(C{0.0L, c.r}, a, b);
    }
    return res;
}
```

## 1.7 線分アレンジメント

```cpp
// segments arrangement AOJ 1050
G segment_arrangement(const vector<L> &seg, vector<P> &point){
    int n=seg.size();
    rep(i,n){
        auto &l=seg[i];
        point.emplace_back({l.s,l.t});
        rep(j,i) if(iss(seg[i],seg[j],1)) point.emplace_back(cross(l,seg[j]));
    }

    uniq(point);
    int m=point.size();
    G graph(m);

    for(auto &l:seg){
        vector<int> idx;
        rep(j,m) if(sdot(point[j],l.s,l.t)<0)  idx.emplace_back(j);

        sort(_all(idx),[&](int i,int j){return norm(point[i]-l.s)<norm(point[j]-l.s)});
        rep(j,1,idx.size){
            int a=idx[j-1],b=idx[j];
            add_edge(graph,a,b,abs(point[a]-point[b]));
        }
    }
    return graph;
}
```

## 1.8 円アレンジメント

```cpp
const int vmax=5010;
struct node{int to;R cost;};
vector<node> graph[vmax];

// Points not verify
R toRagian(R degree){ return degree*PI/180.0;}
R ang (P p){return arg(p);}
R ang (P bs,P a,P b) {R res=arg((b-bs)/(a-bs));return res<0?res+2*PI:res;}
P rot (P bs,P a,R tht){P tar=a-bs;return bs+polar(abs(tar),arg(tar)+tht);}

const int vmax=5010;
struct node{int to;R cost;};
vector<node> graph[vmax];

inline void add_edge(int f,int t,R c){reg(graph[f],{t,c}),reg(graph[t],{f,c});}

```

```cpp
// AOJ 1352

void circle_arrangement(const VC &circle,VP &point){
    VP candiate;
    auto can=[&](P p){
        for(auto &c:circle)if(icp(c,p,1)) return;
        reg(candiate,p);
    };

    auto check1=[&](P p){
        for(auto &c:circle)if(icp(c,p,1)) return false;
        return true;
    };

    auto check2=[&](L s){
        for(auto &c:circle)if(ics(c,s,1)) return false;
        return true;
    };

    for(auto &c1:circle){
        rep(j,4) can(cir(c1,j*PI/2.0));
        for(auto &p:point) for(auto &l:tan(c1,p)) can(proj(l,c1.c));
        for(auto &c2:circle){
            if(rcc(c1,c2)==ISC) for(auto &p:pcc(c1,c2)) can(p);
            for(auto &l:tan(c1,c2)) can(proj(l,c1.c)),can(proj(l,c2.c));
        }
    }

    uniq(candiate),move(_all(candiate),back_inserter(point));
    for(auto &c:circle){
        vector<pair<R,int>> idx;
        rep(i,point.size()){
            if(sgn(norm(c.c-point[i])-norm(c.r))==0)
                reg(idx,{arg(point[i]-c.c),i});
        }
        sort(_all(idx)),reg(idx,{idx[0].first+2*PI,idx[0].second});
        rep(i,1,idx.size()){
            R a1=idx[i-1].first,a2=idx[i].first;
            P mid=cir(c,(a1+a2)/2.0);
            if(check1(mid)) add_edge(idx[i-1].second,idx[i].second,c.r*(a2-a1));
        }
    }
    rep(i,point.size())rep(j,i){
        L l={point[i],point[j]};
        if(check2(l)) add_edge(i,j,abs(l.t-l.s));
    }
```

```
}
```

### 1.9 ボロノイ図

```
VP normalize_polygon(VP pol) {
    rep(i, pol.size()) {
        if (ccw(pol[(i + n - 1) % n], pol[i], pol[(i + 1) % n]) == ON)
            pol.erase(begin(pol) + i--);
    }
    return pol;
}

L bisector(P a, P b) {
    const P mid = (a + b) / P(2, 0);
    return L{mid, mid + (b - a)*P(0, 1)};
}

VP voronoi_cell(VP pol, VP v, int s) {
    rep(i, v.size()) {
        if (i == s) continue;
        pol = convex_cut(pol, bisector(v[s], v[i]));
    }
    return pol;
}
```