

# Algorithm Library

CRatiQ

South China Normal University

June 1, 2025

# Contents

<b>数学</b>	<b>2</b>
Set Xor-Min . . . . .	2
<b>数据结构</b>	<b>3</b>
半群 deque . . . . .	3
区间众数 . . . . .	4
李超树 . . . . .	5
Splay . . . . .	6
<b>计算几何</b>	<b>10</b>
EPS . . . . .	10
Point . . . . .	10
Line . . . . .	11
距离 . . . . .	11
点绕中心旋转 . . . . .	11
关于线的对称点 . . . . .	11
位置关系判断 . . . . .	11
线段交点 . . . . .	12
过定点做圆的切线 . . . . .	12
两圆交点 . . . . .	12
多边形面积 . . . . .	13
自适应辛普森法 . . . . .	13
静态凸包 . . . . .	13
旋转卡壳求直径 . . . . .	14
半平面交 . . . . .	14
最小圆覆盖 . . . . .	15

# 数学

## Set Xor-Min

维护一个集合  $S$ , 可以求  $\min_{y \in S} (x \oplus y)$ 。

```
1  struct SetXorMin
2  {
3      static constexpr int L=30;
4      int tot=0;
5      vector<array<int,2>> c;
6      vector<int> s;
7      set<i64> in;
8
9      SetXorMin() {}
10     SetXorMin(int n)
11     {
12         c.resize((n+1)*(L+1));
13         s.resize((n+1)*(L+1));
14     }
15
16     void insert(i64 x)
17     {
18         if (in.count(x))
19             return;
20         in.insert(x);
21         int p=0;
22         for (int i=L;i>=0;i--)
23         {
24             bool o=x>>i&1;
25             if (!c[p][o])
26                 c[p][o]=++tot;
27             s[p=c[p][o]]++;
28         }
29     }
30
31     void erase(i64 x)
32     {
33         if (!in.count(x))
34             return;
35         in.erase(x);
36         int p=0;
37         for (int i=L;i>=0;i--)
38         {
39             bool o=x>>i&1;
40             s[p=c[p][o]]--;
41         }
42     }
43
44     i64 QueryXorMin(i64 x)
45     {
46         int p=0;
47         i64 r=0;
48         for (int i=L;i>=0;i--)
49         {
50             bool o=x>>i&1;
51             if (s[c[p][o]])
52                 p=c[p][o];
53             else
54             {
55                 p=c[p][o^1];
56                 r|=1ll<<i;
57             }
58         }
59         return r;
60     }
61 };
```

# 数据结构

## 半群 deque

维护一个半群的 deque，支持前后增删及求和。

```
1  template <class T>
2  struct SWAG
3  {
4      vector<T> l,sl,r,sr;
5
6      void push_front(const T &o)
7      {
8          sl.push_back(sl.empty()?o:o+sl.back());
9          l.push_back(o);
10     }
11
12     void push_back(const T &o)
13     {
14         sr.push_back(sr.empty()?o:sr.back()+o);
15         r.push_back(o);
16     }
17
18     void pop_front()
19     {
20         if (!l.empty())
21         {
22             l.pop_back();
23             sl.pop_back();
24             return;
25         }
26         int n=r.size(),m;
27         if (m=n-1>>1)
28         {
29             l.resize(m);
30             sl.resize(m);
31             for (int i=1;i<=m;i++)
32                 l[m-i]=r[i];
33             sl[0]=l[0];
34             for (int i=1;i<m;i++)
35                 sl[i]=l[i]+sl[i-1];
36         }
37         for (int i=m+1;i<n;i++)
38             r[i-(m+1)]=r[i];
39         m=n-(m+1);
40         r.resize(m);
41         sr.resize(m);
42         if (m)
43         {
44             sr[0]=r[0];
45             for (int i=1;i<m;i++)
46                 sr[i]=sr[i-1]+r[i];
47         }
48     }
49
50     void pop_back()
51     {
52         if (!r.empty())
53         {
54             r.pop_back();
55             sr.pop_back();
56         }
57         else
58         {
59             int n=l.size(),m;
60             if (m=n-1>>1)
61             {
62                 r.resize(m);
63                 sr.resize(m);
64                 for (int i=1;i<=m;i++)
65                     r[m-i]=l[i];
```

```

66         sr[0]=r[0];
67         for (int i=1;i<m;i++)
68             sr[i]=sr[i-1]+r[i];
69     }
70     for (int i=m+1;i<n;i++)
71         l[i-(m+1)]=l[i];
72     m=n-(m+1);
73     l.resize(m);
74     sl.resize(m);
75     if (m)
76     {
77         sl[0]=l[0];
78         for (int i=1;i<m;i++)
79             sl[i]=l[i]+sl[i-1];
80     }
81 }
82 }
83
84 T ask()
85 {
86     assert(l.size()||r.size());
87     if (l.size()&&r.size())
88         return sl.back()+sr.back();
89     return l.size()?sl.back():sr.back();
90 }
91 };
92
93 struct Info
94 {
95     Z k,b;
96
97     Info operator + (const Info &o) const
98     {
99         return {k*o.k,b*o.k+o.b};
100     }
101 };
102
103 Z operator + (const Z &x,const Info &o)
104 {
105     return o.k*x+o.b;
106 }

```

## 区间众数

```

1  template <class T>
2  struct Mode
3  {
4      int n,ksz,m;
5      vector<T> b;
6      vector<vector<int>> pos,f;
7      vector<int> a,blk,id,l;
8
9      Mode(const vector<T> &c):n(c.size()),ksz(max<int>(1,sqrt(n))),
10         m((n+ksz-1)/ksz),b(c),pos(n),f(m,vector<int>(m)),a(n),blk(n),id(n),l(m+1)
11  {
12      sort(b.begin(),b.end());
13      b.erase(unique(b.begin(),b.end()),b.end());
14      for (int i=0;i<n;i++)
15      {
16          a[i]=lower_bound(b.begin(),b.end(),c[i])-b.begin();
17          id[i]=pos[a[i]].size();
18          pos[a[i]].push_back(i);
19      }
20      for (int i=0;i<n;i++)
21          blk[i]=i/ksz;
22      for (int i=0;i<=m;i++)
23          l[i]=min(i*ksz,n);
24
25      vector<int> cnt(b.size());
26      for (int i=0;i<m;i++)
27      {

```

```

28         cnt.assign(b.size(),0);
29         pair<int,int> cur={0,0};
30         for (int j=i;j<m;j++)
31         {
32             for (int k=l[j];k<l[j+1];k++)
33                 cur=max(cur,{++cnt[a[k]],a[k]});
34             f[i][j]=cur.second;
35         }
36     }
37 }
38
39 pair<T,int> ask(int L,int R)
40 {
41     int val=blk[L]==blk[R-1]?0:f[blk[L]+1][blk[R-1]-1],i;
42     int cnt=lower_bound(pos[val].begin(),pos[val].end(),R)-
43         lower_bound(pos[val].begin(),pos[val].end(),L);
44     for (int i=min(R,l[blk[L]+1])-1;i>=L;i--)
45     {
46         auto &v=pos[a[i]];
47         while (id[i]+cnt<v.size()&&v[id[i]+cnt]<R)
48             cnt++,val=a[i];
49         if (a[i]>val&&id[i]+cnt-1<v.size()&&v[id[i]+cnt-1]<R)
50             val=a[i];
51     }
52     for (int i=max(L,l[blk[R-1]]);i<R;i++)
53     {
54         auto &v=pos[a[i]];
55         while (id[i]>=cnt&&v[id[i]-cnt]>=L)
56             cnt++,val=a[i];
57         if (a[i]>val&&id[i]>=cnt-1&&v[id[i]-cnt+1]>=L)
58             val=a[i];
59     }
60     return {b[val],cnt};
61 }
62 };

```

## 李超树

```

1  constexpr i64 inf=9e18;
2
3  template <class Info>
4  struct SGT
5  {
6      int cnt=0;
7      vector<Info> a;
8      vector<int> ls,rs;
9      i64 z,y,L,R;
10
11      SGT(int n,i64 l,i64 r)
12      {
13          int N=(n+7)*64;
14          a.resize(N);
15          ls.resize(N);
16          rs.resize(N);
17          L=l,R=r,cnt=1;
18          a[1]={0,inf};
19      }
20
21  private:
22      void insert(int &p,i64 l,i64 r,Info v)
23      {
24          if (!p)
25          {
26              p=++cnt;
27              a[p]={0,inf};
28          }
29          i64 m=(l+r)>>1;
30          if (z<=l&&r<=y)
31          {
32              if (a[p].y(m)>v.y(m)) swap(a[p],v);
33              if (a[p].y(l)>v.y(l)) insert(ls[p],l,m,v);

```

```

34         else if (a[p].y(r)>v.y(r)) insert(rs[p],m+1,r,v);
35         return;
36     }
37     if (z<=m) insert(ls[p],l,m,v);
38     if (y>m) insert(rs[p],m+1,r,v);
39 }
40 public:
41 void insert(i64 l,i64 r,const Info &v)
42 {
43     z=l,y=r;
44     int p=1;
45     insert(p,L,R,v);
46 }
47
48 i64 QueryMin(i64 p)
49 {
50     i64 res=a[1].y(p),l=L,r=R,x=1;
51     while (l<r)
52     {
53         i64 m=(l+r)>>1;
54         if (p<=m)
55             x=ls[x],r=m;
56         else
57             x=rs[x],l=m+1;
58         if (!x) return res;
59         res=min(res,a[x].y(p));
60     }
61     return res;
62 }
63 };
64
65 struct Info
66 {
67     i64 k,b;
68
69     i64 y(const i64 &x) const { return k*x+b; }
70 };

```

## Splay

```

1  template <class Info,class Tag>
2  struct Splay
3  {
4      #define _rev
5      struct Node
6      {
7          Node *c[2],*f;
8          int siz;
9          Info s,v;
10         Tag t;
11
12         Node():c{},f(0),siz(1),s(),v(),t() {}
13         Node(Info x):c{},f(0),siz(1),s(x),v(x),t() {}
14
15         void operator += (const Tag &o)
16         {
17             s+=o,v+=o,t+=o;
18         }
19         #ifdef _rev
20         void swap() { swap(c[0],c[1]); }
21         #endif
22
23         void pushup()
24         {
25             if (c[0])
26                 s=c[0]->s+v,siz=c[0]->siz+1;
27             else s=v,siz=1;
28             if (c[1])
29                 s=s+c[1]->s,siz+=c[1]->siz;
30         }
31     }

```

```

32     void pushdown()
33     {
34         for (auto x:c)
35             if (x)
36                 *x+=t;
37         t=Tag();
38     }
39
40     void zigzag()
41     {
42         Node *y=f,*z=y->f;
43         bool isl=y->c[0]==this;
44         if (z) z->c[z->c[1]==y]=this;
45         f=z,y->f=this;
46         y->c[isl^1]=c[isl];
47         if (c[isl]) c[isl]->f=y;
48         c[isl]=y;
49         y->pushup();
50     }
51
52     //only used for makeroot
53     void splay(Node *tg)
54     {
55         for (Node *y=f;y!=tg;zigzag(),y=f)
56             if (Node *z=y->f;z!=tg)
57                 (z->c[1]==y^y->c[1]==this?this:y)->zigzag();
58         pushup();
59     }
60
61     void clear()
62     {
63         for (Node *x:c)
64             if (x)
65                 x->clear();
66         delete this;
67     }
68 };
69
70 Node *rt;
71 int shift;
72
73 Splay()
74 {
75     rt=new Node;
76     rt->c[1]=new Node;
77     rt->c[1]->f=rt;
78     rt->siz=2;
79 }
80
81 Splay(vector<Info> &a,int l,int r)
82 {
83     shift=l-1;
84     rt=new Node;
85     rt->c[1]=new Node;
86     rt->c[1]->f=rt;
87     if (l<r)
88     {
89         rt->c[1]->c[0]=build(a,l,r);
90         rt->c[1]->c[0]->f=rt->c[1];
91     }
92     rt->c[1]->pushup();
93     rt->pushup();
94 }
95
96 Node *build(vector<Info> &a,int l,int r)
97 {
98     if (l==r) return 0;
99     int m=(l+r)>>1;
100     Node *x=new Node(a[m]);
101     x->c[0]=build(a,l,m);
102     x->c[1]=build(a,m+1,r);

```



```

103     for (Node *y:x->c)
104         if (y) y->f=x;
105     x->pushup();
106     return x;
107 }
108
109 void makeroot(Node *u,Node *tg)
110 {
111     if (!tg) rt=u;
112     u->splay();
113 }
114
115 void findKth(int k,Node *tg)
116 {
117     Node *x=rt;
118     while (1)
119     {
120         x->pushdown();
121         int res=x->c[0]?x->c[0]->siz:0;
122         if (res+1==k)
123         {
124             x->splay(tg);
125             if (!tg) rt=x;
126             return;
127         }
128         if (res>=k) x=x->c[0];
129         else x=x->c[1],k-=res+1;
130     }
131 }
132
133 void split(int l,int r)
134 {
135     findKth(l,0);
136     findKth(r+2,rt);
137 }
138
139 #ifdef _rev
140 void reverse(int l,int r)
141 {
142     l-=shift;
143     r-=shift+1;
144     if (l>r) return;
145     split(l,r);
146     *(rt->c[1]->c[0])+=Tag(1);
147 }
148 #endif
149
150 //insert before pos
151 void insert(int pos,Info x)
152 {
153     pos-=shift;
154     split(pos,pos-1);
155     rt->c[1]->c[0]=new Node(x);
156     rt->c[1]->c[0]->f=rt->c[1];
157     rt->c[1]->pushup();
158     rt->pushup();
159 }
160
161 void insert(int pos,vector<Info> &a,int l,int r)
162 {
163     pos-=shift;
164     split(pos,pos-1);
165     rt->c[1]->c[0]=build(a,l,r);
166     rt->c[1]->c[0]->f=rt->c[1];
167     rt->c[1]->pushup();
168     rt->pushup();
169 }
170
171 void erase(int pos)
172 {
173     pos-=shift;

```

```

174     split(pos,pos);
175     delete rt->c[1]->c[0];
176     rt->c[1]->c[0]=0;
177     rt->c[1]->pushup();
178     rt->pushup();
179 }
180
181 void erase(int l,int r)
182 {
183     l-=shift,r-=shift+1;
184     if (l>r) return;
185     split(l,r);
186     rt->c[1]->c[0]->clear();
187     rt->c[1]->c[0]=0;
188     rt->c[1]->pushup();
189     rt->pushup();
190 }
191
192 void modify(int pos,Info x)
193 {
194     pos-=shift;
195     findKth(pos+1,0);
196     rt->v=x;
197     rt->pushup();
198 }
199
200 void rangeApply(int l,int r,Tag w)
201 {
202     l-=shift,r-=shift+1;
203     if (l>r) return;
204     split(l,r);
205     Node *x=rt->c[1]->c[0];
206     *x+=w;
207     rt->c[1]->pushup();
208     rt->pushup();
209 }
210
211 Info rangeQuery(int l,int r)
212 {
213     l-=shift,r-=shift+1;
214     split(l,r);
215     return rt->c[1]->c[0]->s;
216 }
217
218 ~Splay() { rt->clear(); }
219 #undef _rev
220 };
221
222 struct Tag
223 {
224     bool rev=0;
225
226     Tag() {}
227     Tag(bool c):rev(c) {}
228
229     void operator += (const Tag &o)
230     {
231         rev^=o.rev;
232     }
233 };
234
235 struct Info
236 {
237     i64 x=0;
238
239     void operator += (const Tag &o) const
240     {
241
242     }
243
244     Info operator + (const Info &o) const

```

```

245     {
246         return {x+o.x};
247     }
248 };

```

## 计算几何

### EPS

```

1  const double eps=1e-8;
2  int sgn(double x)
3  {
4      if (fabs(x)<eps) return 0;
5      if (x>0) return 1;
6      return -1;
7  }

```

### Point

```

1  template <class T>
2  struct Point
3  {
4      T x,y;
5      Point(T x_=0,T y_=0):x(x_),y(y_) {}
6
7      Point &operator += (Point p) &
8      {
9          x+=p.x;
10         y+=p.y;
11         return *this;
12     }
13
14     Point &operator -= (Point p) &
15     {
16         x-=p.x;
17         y-=p.y;
18         return *this;
19     }
20
21     Point &operator *= (T v) &
22     {
23         x*=v;
24         y*=v;
25         return *this;
26     }
27
28     Point operator - () const { return Point(-x,-y); }
29
30     friend Point operator + (Point a,Point b) { return a+=b; }
31     friend Point operator - (Point a,Point b) { return a-=b; }
32     friend Point operator * (Point a,T b) { return a*=b; }
33     friend Point operator * (T a,Point b) { return b*=a; }
34
35     friend bool operator == (Point a,Point b) { return a.x==b.x&& a.y==b.y; }
36
37     friend istream &operator >> (istream &is,Point &p) { return is>>p.x>>p.y; }
38
39     friend ostream &operator << (ostream &os,Point p) { return os<<'('<<p.x<<','<<p.y<<')'; }
40 };
41
42 template <class T>
43 int sgn(const Point<T> &a) { return a.y>0||(a.y==0&&a.x>0)?1:-1; }
44
45 template <class T>
46 T dot(Point<T> a,Point<T> b) { return a.x*b.x+a.y*b.y; }
47
48 template <class T>
49 T cross(Point<T> a,Point<T> b) { return a.x*b.y-a.y*b.x; }
50

```

```

51 template <class T>
52 T square(Point<T> p) { return dot(p,p); }
53
54 template <class T>
55 double length(Point<T> p) { return sqrt(double(square(p))); }
56
57 long double length(Point<long double> p) { return sqrt(square(p)); }

```

## Line

```

1 template <class T>
2 struct Line
3 {
4     Point<T> a,b;
5     Line(Point<T> a_=Point<T>(),Point<T> b_=Point<T>()):a(a_),b(b_) {}
6 };

```

## 距离

```

1 template <class T>
2 double dis_PP(Point<T> a,Point<T> b) { return length(a-b); }
3
4 template <class T>
5 double dis_PL(Point<T> a,Line<T> l) { return fabs(cross(a-l.a,a-l.b))/dis_PP(l.a,l.b); }
6
7 template <class T>
8 double dis_PS(Point<T> a,Line<T> l)
9 {
10     if (dot(a-l.a,l.b-l.a)<0) return dis_PP(a,l.a);
11     if (dot(a-l.b,l.a-l.b)<0) return dis_PP(a,l.b);
12     return dis_PL(a,l);
13 }

```

## 点绕中心旋转

```

1 template <class T>
2 Point<T> rotate(Point<T> a,double alpha)
3 { return Point<T>(a.x*cos(alpha)-a.y*sin(alpha),a.x*sin(alpha)+a.y*cos(alpha)); }

```

## 关于线的对称点

```

1 template <class T>
2 Point<T> lineRoot(Point<T> a,Line<T> l)
3 {
4     Point<T> v=l.b-l.a;
5     return l.a+v*(dot(a-l.a,v)/dot(v,v));
6 }
7
8 template <class T>
9 Point<T> symmetry_PL(Point<T> a,Line<T> l) { return a+(lineRoot(a,l)-a)*2; }

```

## 位置关系判断

```

1 template <class T>
2 bool pointOnSegment(Point<T> a,Line<T> l)
3 { return (sgn(cross(a-l.a,a-l.b))==0)&&(sgn(dot(a-l.a,a-l.b))<=0); }
4
5 template <class T>
6 bool lineCrossLine(Line<T> a,Line<T> b)
7 {
8     double f1=cross(b.a-a.a,a.b-a.a),f2=cross(b.b-a.a,a.b-a.a);
9     double g1=cross(a.a-b.a,b.b-b.a),g2=cross(a.b-b.a,b.b-b.a);
10    return ((f1<0)^(f2<0))&&((g1<0)^(g2<0));
11 }
12
13 template <class T>
14 bool pointOnLineLeft(Point<T> a,Line<T> l) { return cross(l.b-l.a,a-l.a)>0; }
15

```

```

16 //适用任意多边形, O(n)
17 template <class T>
18 bool pointInPolygon(Point<T> a, const vector<Point<T>> &p)
19 {
20     int n=p.size();
21     for (int i=0; i<n; i++)
22         if (pointOnSegment(a, Line<T>(p[i], p[(i+1)%n])))
23             return 1;
24     bool t=0;
25     for (int i=0; i<n; i++)
26     {
27         Point<T> u=p[i], v=p[(i+1)%n];
28         if (u.x<a.x&&v.x>a.x&&pointOnLineLeft(a, Line<T>(v, u))) t^=1;
29         if (u.x>a.x&&v.x<a.x&&pointOnLineLeft(a, Line<T>(u, v))) t^=1;
30     }
31     return t;
32 }
33
34 //适用凸多边形, O(log n)
35 template <class T>
36 bool pointInPolygon_(Point<T> a, const vector<Point<T>> &p)
37 {
38     int n=p.size();
39     if (cross(a-p[0], p[1]-p[0])<0 || cross(a-p[0], p[n-1]-p[0])>0) return 0;
40     if (pointOnSegment(a, Line<T>(p[0], p[1])) || pointOnSegment(a, Line<T>(p[n-1], p[0]))) return 1;
41     int l=1, r=n-1;
42     while (l<r)
43     {
44         int mid=(l+r)>>1;
45         if (cross(a-p[l], p[mid]-p[l])<0) l=mid;
46         else r=mid;
47     }
48     if (cross(a-p[l], p[r]-p[l])>0) return 0;
49     if (pointOnSegment(a, Line<T>(p[l], p[r]))) return 1;
50     return 1;
51 }

```

## 线段交点

```

1 //小心 平行
2 template <class T>
3 Point<T> lineIntersection(Line<T> a, Line<T> b)
4 {
5     Point<T> u=a.a-b.a, v=a.b-a.a, w=b.b-b.a;
6     double t=cross(u, w)/cross(w, v);
7     return a.a+t*v;
8 }

```

## 过定点做圆的切线

```

1 template <class T>
2 vector<Line<T>> tan_PC(Point<T> a, Point<T> c, T r)
3 {
4     Point<T> v=c-a;
5     vector<Line<T>> res;
6     int dis=dis_PP(a, c);
7     if (sgn(dis-r)==0) res.push_back(rotate(v, acos(-1)/2));
8     else if (dis>r)
9     {
10         double alpha=asin(r/dis);
11         res.push_back(rotate(v, alpha));
12         res.push_back(rotate(v, -alpha));
13     }
14     return res;
15 }

```

## 两圆交点

```

1 template <class T>
2 vector<Point<T>> circleIntersection(Point<T> c1, T r1, Point<T> c2, T r2)

```

```

3 {
4     auto get=[&](Point<T> c,T r,double alpha)->Point<T>
5     { return Point<T>(c.x+cos(alpha)*r,c.y+sin(alpha)*r); };
6
7     auto angle=[&](Point<T> a)->double { return atan2(a.x,a.y); };
8
9     vector<Point<T>> res;
10    double d=dis_PP(c1,c2);
11    if (sgn(d)==0) return res;
12    if (sgn(r1+r2-d)<0) return res;
13    if (sgn(fabs(r1-r2)-d)>0) return res;
14    double alpha=angle(c2-c1);
15    double beta=acos((r1*r1-r2*r2+d*d)/(r1*d*2));
16    Point<T> p1=get(c1,r1,alpha-beta),p2=get(c1,r1,alpha+beta);
17    res.push_back(p1);
18    if (p1!=p2) res.push_back(p2);
19    return res;
20 }

```

## 多边形面积

```

1 template <class T>
2 double polygonArea(const vector<Point<T>> &p)
3 {
4     int n=p.size();
5     double res=0;
6     for (int i=1;i<n-1;i++) res+=cross(p[i]-p[0],p[i+1]-p[0]);
7     return fabs(res/2);
8 }

```

## 自适应辛普森法

```

1 //注意边界函数值不能小于 eps
2 double f(double x) { return pow(x,0.5); }
3 double calc(double l,double r)
4 {
5     double mid=(l+r)/2.0;
6     return (r-l)*(f(l)+f(r)+f(mid)*4.0)/6.0;
7 }
8 double simpson(double l,double r,double lst)
9 {
10    double mid=(l+r)/2.0;
11    double fl=calc(l,mid),fr=calc(mid,r);
12    if (sgn(fl+fr-lst)==0) return fl+fr;
13    else return simpson(l,mid,fl)+simpson(mid,r,fr);
14 }

```

## 静态凸包

```

1 template <class T>
2 vector<Point<T>> getHull(vector<Point<T>> p)
3 {
4     vector<Point<T>> h,l;
5     sort(p.begin(),p.end(),[&](auto a,auto b)
6     {
7         if (a.x!=b.x) return a.x<b.x;
8         else return a.y<b.y;
9     });
10    p.erase(unique(p.begin(),p.end()),p.end());
11    if (p.size()<=1) return p;
12    for (auto a:p)
13    {
14        while (h.size()>1&&sgn(cross(a-h.back(),a-h[h.size()-2]))<=0) h.pop_back();
15        while (l.size()>1&&sgn(cross(a-l.back(),a-l[l.size()-2]))>=0) l.pop_back();
16        l.push_back(a);
17        h.push_back(a);
18    }
19    l.pop_back();
20    reverse(h.begin(),h.end());

```

```

21     h.pop_back();
22     l.insert(l.end(),h.begin(),h.end());
23     return l;
24 }

```

## 旋转卡壳求直径

```

1  template <class T>
2  double getDiameter(vector<Point<T>> p)
3  {
4      double res=0;
5      if (p.size()==2) return dis_PP(p[0],p[1]);
6      int n=p.size();
7      p.push_back(p.front());
8      int j=2;
9      for (int i=0;i<n;i++)
10     {
11         while (sgn(cross(p[i+1]-p[i],p[j]-p[i])-cross(p[i+1]-p[i],p[j+1]-p[i]))<0)
12             j=(j+1)%n;
13         res=max({res,dis_PP(p[i],p[j]),dis_PP(p[i+1],p[j])});
14     }
15     return res;
16 }

```

## 半平面交

```

1  template <class T>
2  vector<Point<T>> hp(vector<Line<T>> lines)
3  {
4      sort(lines.begin(),lines.end(),[&](auto l1,auto l2)
5      {
6          auto d1=l1.b-l1.a;
7          auto d2=l2.b-l2.a;
8
9          if (sgn(d1)!=sgn(d2)) return sgn(d1)==1;
10         return cross(d1,d2)>0;
11     });
12
13     deque<Line<T>> ls;
14     deque<Point<T>> ps;
15     for (auto l:lines)
16     {
17         if (ls.empty())
18         {
19             ls.push_back(l);
20             continue;
21         }
22         while (!ps.empty()&&!pointOnLineLeft(ps.back(),l))
23         {
24             ps.pop_back();
25             ls.pop_back();
26         }
27         while (!ps.empty()&&!pointOnLineLeft(ps[0],l))
28         {
29             ps.pop_front();
30             ls.pop_front();
31         }
32         if (cross(l.b-l.a,ls.back().b-ls.back().a)==0)
33         {
34             if (dot(l.b-l.a,ls.back().b-ls.back().a)>0)
35             {
36                 if (!pointOnLineLeft(ls.back().a,l))
37                 {
38                     assert(ls.size()==1);
39                     ls[0]=l;
40                 }
41                 continue;
42             }
43             return {};
44         }
45     }

```

```

45         ps.push_back(lineIntersection(ls.back(),l));
46         ls.push_back(l);
47     }
48     while (!ps.empty())&&!pointOnLineLeft(ps.back(),ls[0]))
49     {
50         ps.pop_back();
51         ls.pop_back();
52     }
53     if (ls.size()<=2) return {};
54     ps.push_back(lineIntersection(ls[0],ls.back()));
55     return vector(ps.begin(),ps.end());
56 }

```

## 最小圆覆盖

期望时间复杂度为  $\mathcal{O}(n)$ 。

```

1  using Real=long double;
2
3  //only for 3*3
4  Real det(vector<vector<Real>> a)
5  {
6      Real res=0;
7      for (int i=0;i<3;i++)
8      {
9          Real tmp=1;
10         for (int j=0;j<3;j++)
11             tmp*=a[j][(i+j)%3];
12         res+=tmp;
13     }
14     for (int i=0;i<3;i++)
15     {
16         Real tmp=1;
17         for (int j=0;j<3;j++)
18             tmp*=a[j][(i+j*2)%3];
19         res-=tmp;
20     }
21     return res;
22 }
23
24 mt19937_64 rnd(chrono::steady_clock::now().time_since_epoch().count());
25
26 tuple<Point<Real>,Real> Coverage(vector<Point<Real>> p)
27 {
28     int n=p.size();
29     shuffle(p.begin(),p.end(),rnd);
30     Point<Real> C=p[0];
31     Real r=0;
32     for (int i=0;i<n;i++)
33         if (dis_PP(C,p[i])>r)
34         {
35             C=p[i],r=0;
36             for (int j=0;j<i;j++)
37                 if (dis_PP(C,p[j])>r)
38                 {
39                     C=(p[i]+p[j])*0.5;
40                     r=dis_PP(p[i],p[j])*0.5;
41                     for (int k=0;k<j;k++)
42                         if (dis_PP(C,p[k])>r)
43                         {
44                             array<Real,3> x,y;
45                             x[0]=p[i].x,y[0]=p[i].y;
46                             x[1]=p[j].x,y[1]=p[j].y;
47                             x[2]=p[k].x,y[2]=p[k].y;
48                             vector<vector<Real>> a(3,vector<Real>(3)),b(a),c(a);
49                             for (int t=0;t<3;t++)
50                             {
51                                 a[t][0]=b[t][0]=x[t]*x[t]+y[t]*y[t];
52                                 c[t][0]=b[t][1]=x[t];
53                                 a[t][1]=c[t][1]=y[t];
54                                 a[t][2]=b[t][2]=c[t][2]=1;

```



```

55     }
56     Real px=det(a)/det(c)/2.0,py=-det(b)/det(c)/2.0;
57     C={px,py};
58     r=dis_PP(C,p[i]);
59 }
60 }
61 }
62 return {C,r};
63 }

```