Algorithm Library

CReatiQ

South China Normal University

June 1, 2025

Contents

数学

Set Xor-Min

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

```cpp
struct SetXorMin
{
    static constexpr int L=30;
    int tot=0;
    vector<array<int,2>> c;
    vector<int> s;
    set<i64> in;

    SetXorMin() {}
    SetXorMin(int n)
    {
        c.resize((n+1)*(L+1));
        s.resize((n+1)*(L+1));
    }

    void insert(i64 x)
    {
        if (in.count(x))
            return;
        in.insert(x);
        int p=0;
        for (int i=L;i>=0;i--)
        {
            bool o=x>>i&1;
            if (!c[p][o])
                c[p][o]=++tot;
            s[p=c[p][o]]++;
        }
    }

    void erase(i64 x)
    {
        if (!in.count(x))
            return;
        in.erase(x);
        int p=0;
        for (int i=L;i>=0;i--)
        {
            bool o=x>>i&1;
            s[p=c[p][o]]--;
        }
    }

    i64 QueryXorMin(i64 x)
    {
        int p=0;
        i64 r=0;
        for (int i=L;i>=0;i--)
        {
            bool o=x>>i&1;
            if (s[c[p][o]])
                p=c[p][o];
            else
            {
                p=c[p][o^1];
                r|=1ll<<i;
            }
        }
        return r;
    }
};
```

数据结构

半群 deque

维护一个半群的 deque，支持前后增删及求和。

```
template <class T>
struct SWAG
{
    vector<T> l,sl,r,sr;

    void push_front(const T &o)
    {
        sl.push_back(sl.empty()?o:o+sl.back());
        l.push_back(o);
    }

    void push_back(const T &o)
    {
        sr.push_back(sr.empty()?o:sr.back()+o);
        r.push_back(o);
    }

    void pop_front()
    {
        if (!l.empty())
        {
            l.pop_back();
            sl.pop_back();
            return;
        }
        int n=r.size(),m;
        if (m=n-1>>1)
        {
            l.resize(m);
            sl.resize(m);
            for (int i=1;i<=m;i++)
                l[m-i]=r[i];
            sl[0]=l[0];
            for (int i=1;i<m;i++)
                sl[i]=l[i]+sl[i-1];
        }
        for (int i=m+1;i<n;i++)
            r[i-(m+1)]=r[i];
        m=n-(m+1);
        r.resize(m);
        sr.resize(m);
        if (m)
        {
            sr[0]=r[0];
            for (int i=1;i<m;i++)
                sr[i]=sr[i-1]+r[i];
        }
    }

    void pop_back()
    {
        if (!r.empty())
        {
            r.pop_back();
            sr.pop_back();
        }
        else
        {
            int n=l.size(),m;
            if (m=n-1>>1)
            {
                r.resize(m);
                sr.resize(m);
                for (int i=1;i<=m;i++)
                    r[m-i]=l[i];
                sr[0]=r[0];
                for (int i=1;i<m;i++)
                    sr[i]=sr[i-1]+r[i];
            }
            for (int i=m+1;i<n;i++)
                l[i-(m+1)]=l[i];
            m=n-(m+1);
            l.resize(m);
            sl.resize(m);
            if (m)
            {
                sl[0]=l[0];
                for (int i=1;i<m;i++)
                    sl[i]=l[i]+sl[i-1];
            }
        }
    }

    T ask()
    {
        assert(l.size()||r.size());
        if (l.size()&&r.size())
            return sl.back()+sr.back();
        return l.size()?sl.back():sr.back();
    }
};

struct Info
{
    Z k,b;

    Info operator + (const Info &o) const
    {
        return {k*o.k,b*o.k+o.b};
    }
};

Z operator + (const Z &x,const Info &o)
{
    return o.k*x+o.b;
}
```

区间众数

```
template <class T>
struct Mode
{
    int n,ksz,m;
    vector<T> b;
    vector<vector<int>> pos,f;
    vector<int> a,blk,id,l;

    Mode(const vector<T> &c):n(c.size()),ksz(max<int>(1,sqrt(n))),
        m((n+ksz-1)/ksz),b(c),pos(n),f(m,vector<int>(m)),a(n),blk(n),id(n),l(m+1)
    {
        sort(b.begin(),b.end());
        b.erase(unique(b.begin(),b.end()),b.end());
        for (int i=0;i<n;i++)
        {
            a[i]=lower_bound(b.begin(),b.end(),c[i])-b.begin();
            id[i]=pos[a[i]].size();
            pos[a[i]].push_back(i);
        }
        for (int i=0;i<n;i++)
            blk[i]=i/ksz;
        for (int i=0;i<=m;i++)
            l[i]=min(i*ksz,n);

        vector<int> cnt(b.size());
        for (int i=0;i<m;i++)
        {
            cnt.assign(b.size(),0);
            pair<int,int> cur={0,0};
            for (int j=i;j<m;j++)
            {
                for (int k=l[j];k<l[j+1];k++)
                    cur=max(cur,{++cnt[a[k]],a[k]});
                f[i][j]=cur.second;
            }
        }
    }

    pair<T,int> ask(int L,int R)
    {
        int val=blk[L]==blk[R-1]?0:f[blk[L]+1][blk[R-1]-1],i;
        int cnt=lower_bound(pos[val].begin(),pos[val].end(),R)-
                lower_bound(pos[val].begin(),pos[val].end(),L);
        for (int i=min(R,l[blk[L]+1])-1;i>=L;i--)
        {
            auto &v=pos[a[i]];
            while (id[i]+cnt<v.size()&&v[id[i]+cnt]<R)
                cnt++,val=a[i];
            if (a[i]>val&&id[i]+cnt-1<v.size()&&v[id[i]+cnt-1]<R)
                val=a[i];
        }
        for (int i=max(L,l[blk[R-1]]);i<R;i++)
        {
            auto &v=pos[a[i]];
            while (id[i]>=cnt&&v[id[i]-cnt]>=L)
                cnt++,val=a[i];
            if (a[i]>val&&id[i]>=cnt-1&&v[id[i]-cnt+1]>=L)
                val=a[i];
        }
        return {b[val],cnt};
    }
};
```

李超树

```
constexpr i64 inf=9e18;

template <class Info>
struct SGT
{
    int cnt=0;
    vector<Info> a;
    vector<int> ls,rs;
    i64 z,y,L,R;

    SGT(int n,i64 l,i64 r)
    {
        int N=(n+7)*64;
        a.resize(N);
        ls.resize(N);
        rs.resize(N);
        L=l,R=r,cnt=1;
        a[1]={0,inf};
    }

private:
    void insert(int &p,i64 l,i64 r,Info v)
    {
        if (!p)
        {
            p=++cnt;
            a[p]={0,inf};
        }
        i64 m=(l+r)>>1;
        if (z<=l&&r<=y)
        {
            if (a[p].y(m)>v.y(m)) swap(a[p],v);
            if (a[p].y(l)>v.y(l)) insert(ls[p],l,m,v);
```

```cpp
            else if (a[p].y(r)>v.y(r)) insert(rs[p],m+1,r,v);
            return;
        }
        if (z<=m) insert(ls[p],l,m,v);
        if (y>m) insert(rs[p],m+1,r,v);
    }
public:
    void insert(i64 l,i64 r,const Info &v)
    {
        z=l,y=r;
        int p=1;
        insert(p,L,R,v);
    }

    i64 QueryMin(i64 p)
    {
        i64 res=a[1].y(p),l=L,r=R,x=1;
        while (l<r)
        {
            i64 m=(l+r)>>1;
            if (p<=m)
                x=ls[x],r=m;
            else
                x=rs[x],l=m+1;
            if (!x) return res;
            res=min(res,a[x].y(p));
        }
        return res;
    }
};

struct Info
{
    i64 k,b;

    i64 y(const i64 &x) const { return k*x+b; }
};
```

Splay

```cpp
template <class Info,class Tag>
struct Splay
{
#define _rev
    struct Node
    {
        Node *c[2],*f;
        int siz;
        Info s,v;
        Tag t;

        Node():c{},f(0),siz(1),s(),v(),t() {}
        Node(Info x):c{},f(0),siz(1),s(x),v(x),t() {}

        void operator += (const Tag &o)
        {
            s+=o,v+=o,t+=o;
#ifdef _rev
            if (o.rev) swap(c[0],c[1]);
#endif
        }

        void pushup()
        {
            if (c[0])
                s=c[0]->s+v,siz=c[0]->siz+1;
            else s=v,siz=1;
            if (c[1])
                s=s+c[1]->s,siz+=c[1]->siz;
        }

        void pushdown()
        {
            for (auto x:c)
                if (x)
                    *x+=t;
            t=Tag();
        }

        void zigzag()
        {
            Node *y=f,*z=y->f;
            bool isl=y->c[0]==this;
            if (z) z->c[z->c[1]==y]=this;
            f=z,y->f=this;
            y->c[isl^1]=c[isl];
            if (c[isl]) c[isl]->f=y;
            c[isl]=y;
            y->pushup();
        }

        //only used for makeroot
        void splay(Node *tg)
        {
            for (Node *y=f;y!=tg;zigzag(),y=f)
                if (Node *z=y->f;z!=tg)
                    (z->c[1]==y^y->c[1]==this?this:y)->zigzag();
            pushup();
        }

        void clear()
        {
            for (Node *x:c)
                if (x)
                    x->clear();
            delete this;
        }
    };

    Node *rt;
    int shift;

    Splay()
    {
        rt=new Node;
        rt->c[1]=new Node;
        rt->c[1]->f=rt;
        rt->siz=2;
    }

    Splay(vector<Info> &a,int l,int r)
    {
        shift=l-1;
        rt=new Node;
        rt->c[1]=new Node;
        rt->c[1]->f=rt;
        if (l<r)
        {
            rt->c[1]->c[0]=build(a,l,r);
            rt->c[1]->c[0]->f=rt->c[1];
        }
        rt->c[1]->pushup();
        rt->pushup();
    }

    Node *build(vector<Info> &a,int l,int r)
    {
        if (l==r) return 0;
        int m=(l+r)>>1;
        Node *x=new Node(a[m]);
        x->c[0]=build(a,l,m);
        x->c[1]=build(a,m+1,r);
        for (Node *y:x->c)
            if (y) y->f=x;
        x->pushup();
        return x;
    }

    void makeroot(Node *u,Node *tg)
    {
        if (!tg) rt=u;
        u->splay();
    }

    void findKth(int k,Node *tg)
    {
        Node *x=rt;
        while (1)
        {
            x->pushdown();
            int res=x->c[0]?x->c[0]->siz:0;
            if (res+1==k)
            {
                x->splay(tg);
                if (!tg) rt=x;
                return;
            }
            if (res>=k) x=x->c[0];
            else x=x->c[1],k-=res+1;
        }
    }

    void split(int l,int r)
    {
        findKth(l,0);
        findKth(r+2,rt);
    }

#ifdef _rev
    void reverse(int l,int r)
    {
        l-=shift;
        r-=shift+1;
        if (l>r) return;
        split(l,r);
        *(rt->c[1]->c[0])+=Tag(1);
    }
#endif

    //insert before pos
    void insert(int pos,Info x)
    {
        pos-=shift;
        split(pos,pos-1);
        rt->c[1]->c[0]=new Node(x);
        rt->c[1]->c[0]->f=rt->c[1];
        rt->c[1]->pushup();
        rt->pushup();
    }

    void insert(int pos,vector<Info> &a,int l,int r)
    {
        pos-=shift;
        split(pos,pos-1);
        rt->c[1]->c[0]=build(a,l,r);
        rt->c[1]->c[0]->f=rt->c[1];
        rt->c[1]->pushup();
        rt->pushup();
    }

    void erase(int pos)
    {
        pos-=shift;
        split(pos,pos);
        delete rt->c[1]->c[0];
        rt->c[1]->c[0]=0;
        rt->c[1]->pushup();
        rt->pushup();
    }

    void erase(int l,int r)
    {
        l-=shift,r-=shift+1;
        if (l>r) return;
        split(l,r);
        rt->c[1]->c[0]->clear();
        rt->c[1]->c[0]=0;
        rt->c[1]->pushup();
        rt->pushup();
    }

    void modify(int pos,Info x)
    {
        pos-=shift;
        findKth(pos+1,0);
        rt->v=x;
        rt->pushup();
    }

    void rangeApply(int l,int r,Tag w)
    {
        l-=shift,r-=shift+1;
        if (l>r) return;
        split(l,r);
        Node *x=rt->c[1]->c[0];
        *x+=w;
        rt->c[1]->pushup();
        rt->pushup();
    }

    Info rangeQuery(int l,int r)
    {
        l-=shift,r-=shift+1;
        split(l,r);
        return rt->c[1]->c[0]->s;
    }

    ~Splay() { rt->clear(); }
#undef _rev
};

struct Tag
{
    bool rev=0;

    Tag() {}
    Tag(bool c):rev(c) {}

    void operator += (const Tag &o)
    {
        rev^=o.rev;
    }
};

struct Info
{
    i64 x=0;

    void operator += (const Tag &o) const
    {

    }

    Info operator + (const Info &o) const
```

```cpp
    {
        return {x+o.x};
    }
};
```

计算几何

EPS

```cpp
const double eps=1e-8;
int sgn(double x)
{
    if (fabs(x)<eps) return 0;
    if (x>0) return 1;
    return -1;
}
```

Point

```cpp
template <class T>
struct Point
{
    T x,y;
    Point(T x_=0,T y_=0):x(x_),y(y_) {}

    Point &operator += (Point p) &
    {
        x+=p.x;
        y+=p.y;
        return *this;
    }

    Point &operator -= (Point p) &
    {
        x-=p.x;
        y-=p.y;
        return *this;
    }

    Point &operator *= (T v) &
    {
        x*=v;
        y*=v;
        return *this;
    }

    Point operator - () const { return Point(-x,-y); }

    friend Point operator + (Point a,Point b) { return a+=b; }
    friend Point operator - (Point a,Point b) { return a-=b; }
    friend Point operator * (Point a,T b) { return a*=b; }
    friend Point operator * (T a,Point b) { return b*=a; }

    friend bool operator == (Point a,Point b) { return a.x==b.x&&a.y==b.y; }

    friend istream &operator >> (istream &is,Point &p) { return is>>p.x>>p.y; }

    friend ostream &operator << (ostream &os,Point p) { return os<<'('<<p.x<<','<<p.y<<')'; }
};

template <class T>
int sgn(const Point<T> &a) { return a.y>0||(a.y==0&&a.x>0)?1:-1; }

template <class T>
T dot(Point<T> a,Point<T> b) { return a.x*b.x+a.y*b.y; }

template <class T>
T cross(Point<T> a,Point<T> b) { return a.x*b.y-a.y*b.x; }

```

```cpp
template <class T>
T square(Point<T> p) { return dot(p,p); }

template <class T>
double length(Point<T> p) { return sqrt(double(square(p))); }

long double length(Point<long double> p) { return sqrt(square(p)); }
```

Line

```cpp
template <class T>
struct Line
{
    Point<T> a,b;
    Line(Point<T> a_=Point<T>(),Point<T> b_=Point<T>()):a(a_),b(b_) {}
};
```

距离

```cpp
template <class T>
double dis_PP(Point<T> a,Point<T> b) { return length(a-b); }

template <class T>
double dis_PL(Point<T> a,Line<T> l) { return fabs(cross(a-l.a,a-l.b))/dis_PP(l.a,l.b); }

template <class T>
double dis_PS(Point<T> a,Line<T> l)
{
    if (dot(a-l.a,l.b-l.a)<0) return dis_PP(a,l.a);
    if (dot(a-l.b,l.a-l.b)<0) return dis_PP(a,l.b);
    return dis_PL(a,l);
}
```

点绕中心旋转

```cpp
template <class T>
Point<T> rotate(Point<T> a,double alpha)
{ 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

```cpp
template <class T>
Point<T> lineRoot(Point<T> a,Line<T> l)
{
    Point<T> v=l.b-l.a;
    return l.a+v*(dot(a-l.a,v)/dot(v,v));
}

template <class T>
Point<T> symmetry_PL(Point<T> a,Line<T> l) { return a+(lineRoot(a,l)-a)*2; }
```

位置关系判断

```cpp
template <class T>
bool pointOnSegment(Point<T> a,Line<T> l)
{ return (sgn(cross(a-l.a,a-l.b))==0)&&(sgn(dot(a-l.a,a-l.b))<=0); }

template <class T>
bool lineCrossLine(Line<T> a,Line<T> b)
{
    double f1=cross(b.a-a.a,a.b-a.a),f2=cross(b.b-a.a,a.b-a.a);
    double g1=cross(a.a-b.a,b.b-b.a),g2=cross(a.b-b.a,b.b-b.a);
    return ((f1<0)^(f2<0))&&((g1<0)^(g2<0));
}

template <class T>
bool pointOnLineLeft(Point<T> a,Line<T> l) { return cross(l.b-l.a,a-l.a)>0; }

```

```
//适用任意多边形,O(n)
template <class T>
bool pointInPolygon(Point<T> a,const vector<Point<T>> &p)
{
    int n=p.size();
    for (int i=0;i<n;i++)
        if (pointOnSegment(a,Line<T>(p[i],p[(i+1)%n])))
            return 1;
    bool t=0;
    for (int i=0;i<n;i++)
    {
        Point<T> u=p[i],v=p[(i+1)%n];
        if (u.x<a.x&&v.x>=a.x&&pointOnLineLeft(a,Line<T>(v,u))) t^=1;
        if (u.x>=a.x&&v.x<a.x&&pointOnLineLeft(a,Line<T>(u,v))) t^=1;
    }
    return t;
}

//适用凸多边形,O(log n)
template <class T>
bool pointInPolygon_(Point<T> a,const vector<Point<T>> &p)
{
    int n=p.size();
    if (cross(a-p[0],p[1]-p[0])<0||cross(a-p[0],p[n-1]-p[0])>0) return 0;
    if (pointOnSegment(a,Line<T>(p[0],p[1]))||pointOnSegment(a,Line<T>(p[n-1],p[0]))) return 1;
    int l=1,r=n-1;
    while (l+1<r)
    {
        int mid=(l+r)>>1;
        if (cross(a-p[1],p[mid]-p[1])<0) l=mid;
        else r=mid;
    }
    if (cross(a-p[l],p[r]-p[l])>0) return 0;
    if (pointOnSegment(a,Line<T>(p[l],p[r]))) return 1;
    return 1;
}
```

线段交点

```
//小 心 平 行
template <class T>
Point<T> lineIntersection(Line<T> a,Line<T> b)
{
    Point<T> u=a.a-b.a,v=a.b-a.a,w=b.b-b.a;
    double t=cross(u,w)/cross(w,v);
    return a.a+t*v;
}
```

过定点做圆的切线

```
template <class T>
vector<Line<T>> tan_PC(Point<T> a,Point<T> c,T r)
{
    Point<T> v=c-a;
    vector<Line<T>> res;
    int dis=dis_PP(a,c);
    if (sgn(dis-r)==0) res.push_back(rotate(v,acos(-1)/2));
    else if (dis>r)
    {
        double alpha=asin(r/dis);
        res.push_back(rotate(v,alpha));
        res.push_back(rotate(v,-alpha));
    }
    return res;
}
```

两圆交点

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

```cpp
{
    auto get=[&](Point<T> c,T r,double alpha)->Point<T>
    { return Point<T>(c.x+cos(alpha)*r,c.y+sin(alpha)*r); };

    auto angle=[&](Point<T> a)->double { return atan2(a.x,a.y); };

    vector<Point<T>> res;
    double d=dis_PP(c1,c2);
    if (sgn(d)==0) return res;
    if (sgn(r1+r2-d)<0) return res;
    if (sgn(fabs(r1-r2)-d)>0) return res;
    double alpha=angle(c2-c1);
    double beta=acos((r1*r1-r2*r2+d*d)/(r1*d*2));
    Point<T> p1=get(c1,r1,alpha-beta),p2=get(c1,r1,alpha+beta);
    res.push_back(p1);
    if (p1!=p2) res.push_back(p2);
    return res;
}
```

多边形面积

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

自适应辛普森法

```cpp
//注意边界函数值不能小于 eps
double f(double x) { return pow(x,0.5); }
double calc(double l,double r)
{
    double mid=(l+r)/2.0;
    return (r-l)*(f(l)+f(r)+f(mid)*4.0)/6.0;
}
double simpson(double l,double r,double lst)
{
    double mid=(l+r)/2.0;
    double fl=calc(l,mid),fr=calc(mid,r);
    if (sgn(fl+fr-lst)==0) return fl+fr;
    else return simpson(l,mid,fl)+simpson(mid,r,fr);
}
```

静态凸包

```cpp
template <class T>
vector<Point<T>> getHull(vector<Point<T>> p)
{
    vector<Point<T>> h,l;
    sort(p.begin(),p.end(),[&](auto a,auto b)
    {
        if (a.x!=b.x) return a.x<b.x;
        else return a.y<b.y;
    });
    p.erase(unique(p.begin(),p.end()),p.end());
    if (p.size()<=1) return p;
    for (auto a:p)
    {
        while (h.size()>1&&sgn(cross(a-h.back(),a-h[h.size()-2]))<=0) h.pop_back();
        while (l.size()>1&&sgn(cross(a-l.back(),a-l[l.size()-2]))>=0) l.pop_back();
        l.push_back(a);
        h.push_back(a);
    }
    l.pop_back();
    reverse(h.begin(),h.end());
    h.pop_back();
    l.insert(l.end(),h.begin(),h.end());
    return l;
}
```

旋转卡壳求直径

```
template <class T>
double getDiameter(vector<Point<T>> p)
{
    double res=0;
    if (p.size()==2) return dis_PP(p[0],p[1]);
    int n=p.size();
    p.push_back(p.front());
    int j=2;
    for (int i=0;i<n;i++)
    {
        while (sgn(cross(p[i+1]-p[i],p[j]-p[i])-cross(p[i+1]-p[i],p[j+1]-p[i]))<0)
            j=(j+1)%n;
        res=max({res,dis_PP(p[i],p[j]),dis_PP(p[i+1],p[j])});
    }
    return res;
}
```

半平面交

```
template <class T>
vector<Point<T>> hp(vector<Line<T>> lines)
{
    sort(lines.begin(),lines.end(),[&](auto l1,auto l2)
    {
        auto d1=l1.b-l1.a;
        auto d2=l2.b-l2.a;

        if (sgn(d1)!=sgn(d2)) return sgn(d1)==1;
        return cross(d1,d2)>0;
    });

    deque<Line<T>> ls;
    deque<Point<T>> ps;
    for (auto l:lines)
    {
        if (ls.empty())
        {
            ls.push_back(l);
            continue;
        }
        while (!ps.empty()&&!pointOnLineLeft(ps.back(),l))
        {
            ps.pop_back();
            ls.pop_back();
        }
        while (!ps.empty()&&!pointOnLineLeft(ps[0],l))
        {
            ps.pop_front();
            ls.pop_front();
        }
        if (cross(l.b-l.a,ls.back().b-ls.back().a)==0)
        {
            if (dot(l.b-l.a,ls.back().b-ls.back().a)>0)
            {
                if (!pointOnLineLeft(ls.back().a,l))
                {
                    assert(ls.size()==1);
                    ls[0]=l;
                }
                continue;
            }
            return {};
        }
        ps.push_back(lineIntersection(ls.back(),l));
        ls.push_back(l);
    }
    while (!ps.empty()&&!pointOnLineLeft(ps.back(),ls[0]))
    {
        ps.pop_back();
        ls.pop_back();
    }
    if (ls.size()<=2) return {};
    ps.push_back(lineIntersection(ls[0],ls.back()));
    return vector(ps.begin(),ps.end());
}
```

最小圆覆盖

期望时间复杂度为 $\mathcal{O}(n)$。

```
using Real=long double;

//only for 3*3
Real det(vector<vector<Real>> a)
{
    Real res=0;
    for (int i=0;i<3;i++)
    {
        Real tmp=1;
        for (int j=0;j<3;j++)
            tmp*=a[j][(i+j)%3];
        res+=tmp;
    }
    for (int i=0;i<3;i++)
    {
        Real tmp=1;
        for (int j=0;j<3;j++)
            tmp*=a[j][(i+j*2)%3];
        res-=tmp;
    }
    return res;
}

mt19937_64 rnd(chrono::steady_clock::now().time_since_epoch().count());

tuple<Point<Real>,Real> Coverage(vector<Point<Real>> p)
{
    int n=p.size();
    shuffle(p.begin(),p.end(),rnd);
    Point<Real> C=p[0];
    Real r=0;
    for (int i=0;i<n;i++)
        if (dis_PP(C,p[i])>r)
        {
            C=p[i],r=0;
            for (int j=0;j<i;j++)
                if (dis_PP(C,p[j])>r)
                {
                    C=(p[i]+p[j])*0.5;
                    r=dis_PP(p[i],p[j])*0.5;
                    for (int k=0;k<j;k++)
                        if (dis_PP(C,p[k])>r)
                        {
                            array<Real,3> x,y;
                            x[0]=p[i].x,y[0]=p[i].y;
                            x[1]=p[j].x,y[1]=p[j].y;
                            x[2]=p[k].x,y[2]=p[k].y;
                            vector<vector<Real>> a(3,vector<Real>(3)),b(a),c(a);
                            for (int t=0;t<3;t++)
                            {
                                a[t][0]=b[t][0]=x[t]*x[t]+y[t]*y[t];
                                c[t][0]=b[t][1]=x[t];
                                a[t][1]=c[t][1]=y[t];
                                a[t][2]=b[t][2]=c[t][2]=1;
                            }
                            Real px=det(a)/det(c)/2.0,py=-det(b)/det(c)/2.0;
                            C={px,py};
                            r=dis_PP(C,p[i]);
                        }
                }
        }
    return {C,r};
}
```