计算几何

计算几何全家桶

转载：https://www.cnblogs.com/Xing-Ling/p/12102489.html

```
#include<algorithm>
#include<cstdio>
#include<cmath>


/*一：【准备工作】*/
#define LD double
#define LL long long
#define Re register int
#define Vector Point
using namespace std;
const int N=262144+3;
const LD eps=1e-8,Pi=acos(-1.0);
inline int dcmp(LD a){return a<-eps?-1:(a>eps?1:0);}//处理精度
inline LD Abs(LD a){return a*dcmp(a);}//取绝对值
struct Point{
    LD x,y;Point(LD X=0,LD Y=0){x=X,y=Y;}
    inline void in(){scanf("%lf%lf",&x,&y);}
    inline void out(){printf("%.2lf %.2lf\n",x,y);}
};


/*二：【向量】*/
inline LD Dot(Vector a,Vector b){return a.x*b.x+a.y*b.y;}//【点积】
inline LD Cro(Vector a,Vector b){return a.x*b.y-a.y*b.x;}//【叉积】
inline LD Len(Vector a){return sqrt(Dot(a,a));}//【模长】
inline LD Angle(Vector a,Vector b){return acos(Dot(a,b)/Len(a)/Len(b));}//【两向量夹角】
```

```
inline Vector Normal(Vector a){return Vector(-a.y,a.x);}//【法向量】
inline Vector operator+(Vector a,Vector b){return Vector(a.x+b.x,a.y+b.y);}
inline Vector operator-(Vector a,Vector b){return Vector(a.x-b.x,a.y-b.y);}
inline Vector operator*(Vector a,LD b){return Vector(a.x*b,a.y*b);}
inline bool operator==(Point a,Point b){return !dcmp(a.x-b.x)&&!dcmp(a.y-
b.y);}//两点坐标重合则相等


/*三：【点、向量的位置变换】*/

/*1.【点、向量的旋转】*/
inline Point turn_P(Point a,LD theta){//【点A\向量A顺时针旋转theta(弧度)】
    LD x=a.x*cos(theta)+a.y*sin(theta);
    LD y=-a.x*sin(theta)+a.y*cos(theta);
    return Point(x,y);
}
inline Point turn_PP(Point a,Point b,LD theta){//【将点A绕点B顺时针旋转theta(弧度)】
    LD x=(a.x-b.x)*cos(theta)+(a.y-b.y)*sin(theta)+b.x;
    LD y=-(a.x-b.x)*sin(theta)+(a.y-b.y)*cos(theta)+b.y;
    return Point(x,y);
}


/*四：【图形与图形之间的关系】*/

/*1.【点与线段】*/
inline int pan_PL(Point p,Point a,Point b){//【判断点P是否在线段AB上】
    return !dcmp(Cro(p-a,b-a))&&dcmp(Dot(p-a,p-b))<=0;//做法一
//  return !dcmp(Cro(p-a,b-a))&&dcmp(min(a.x,b.x)-p.x)<=0&&dcmp(p.x-max(a.x,b.x))
<=0&&dcmp(min(a.y,b.y)-p.y)<=0&&dcmp(p.y-max(a.y,b.y))<=0;//做法二
    //PA,AB共线且P在AB之间(其实也可以用len(p-a)+len(p-b)==len(a-b)判断，但是精度损失较
大)
}
inline LD dis_PL(Point p,Point a,Point b){//【点P到线段AB距离】
    if(a==b)return Len(p-a);//AB重合
    Vector x=p-a,y=p-b,z=b-a;
    if(dcmp(Dot(x,z))<0)return Len(x);//P距离A更近
    if(dcmp(Dot(y,z))>0)return Len(y);//P距离B更近
    return Abs(Cro(x,z)/Len(z));//面积除以底边长
}

/*2.【点与直线】*/
inline int pan_PL_(Point p,Point a,Point b){//【判断点P是否在直线AB上】
    return !dcmp(Cro(p-a,b-a));//PA,AB共线
}
inline Point FootPoint(Point p,Point a,Point b){//【点P到直线AB的垂足】
    Vector x=p-a,y=p-b,z=b-a;
    LD len1=Dot(x,z)/Len(z),len2=-1.0*Dot(y,z)/Len(z);//分别计算AP,BP在AB,BA上的投影
    return a+z*(len1/(len1+len2));//点A加上向量AF
}
inline Point Symmetry_PL(Point p,Point a,Point b){//【点P关于直线AB的对称点】
    return p+(FootPoint(p,a,b)-p)*2;//将PF延长一倍即可
}

/*3.【线与线】*/
inline Point cross_LL(Point a,Point b,Point c,Point d){//【两直线AB,CD的交点】
    Vector x=b-a,y=d-c,z=a-c;
    return a+x*(Cro(y,z)/Cro(x,y));//点A加上向量AF
```

```
}
inline int pan_cross_L_L(Point a,Point b,Point c,Point d){//【判断直线AB与线段CD是否相交】
    return pan_PL(cross_LL(a,b,c,d),c,d);//直线AB与直线CD的交点在线段CD上
}
inline int pan_cross_LL(Point a,Point b,Point c,Point d){//【判断两线段AB,CD是否相交】
    LD c1=Cro(b-a,c-a),c2=Cro(b-a,d-a);
    LD d1=Cro(d-c,a-c),d2=Cro(d-c,b-c);
    return dcmp(c1)*dcmp(c2)<0&&dcmp(d1)*dcmp(d2)<0;//分别在两侧
}

/*4.【点与多边形】*/
inline int PIP(Point *P,Re n,Point a){//【射线法】判断点A是否在任意多边形Poly以内
    Re cnt=0;LD tmp;
    for(Re i=1;i<=n;++i){
        Re j=i<n?i+1:1;
        if(pan_PL(a,P[i],P[j]))return 2;//点在多边形上
        if(a.y>=min(P[i].y,P[j].y)&&a.y<max(P[i].y,P[j].y))//纵坐标在该线段两端点之间
            tmp=P[i].x+(a.y-P[i].y)/(P[j].y-P[i].y)*(P[j].x-
P[i].x),cnt+=dcmp(tmp-a.x)>0;//交点在A右方
    }
    return cnt&1;//穿过奇数次则在多边形以内
}
inline int judge(Point a,Point L,Point R){//判断AL是否在AR右边
    return dcmp(Cro(L-a,R-a))>0;//必须严格以内
}
inline int PIP_(Point *P,Re n,Point a){//【二分法】判断点A是否在凸多边形Poly以内
    //点按逆时针给出
    if(judge(P[1],a,P[2])||judge(P[1],P[n],a))return 0;//在P[1_2]或P[1_n]外
    if(pan_PL(a,P[1],P[2])||pan_PL(a,P[1],P[n]))return 2;//在P[1_2]或P[1_n]上
    Re l=2,r=n-1;
    while(l<r){//二分找到一个位置pos使得P[1]_A在P[1_pos],P[1_(pos+1)]之间
        Re mid=l+r+1>>1;
        if(judge(P[1],P[mid],a))l=mid;
        else r=mid-1;
    }
    if(judge(P[l],a,P[l+1]))return 0;//在P[pos_(pos+1)]外
    if(pan_PL(a,P[l],P[l+1]))return 2;//在P[pos_(pos+1)]上
    return 1;
}

/*5.【线与多边形】*/

/*6.【多边形与多边形】*/
inline int judge_PP(Point *A,Re n,Point *B,Re m){//【判断多边形A与多边形B是否相离】
    for(Re i1=1;i1<=n;++i1){
        Re j1=i1<n?i1+1:1;
        for(Re i2=1;i2<=m;++i2){
            Re j2=i2<m?i2+1:1;
            if(pan_cross_LL(A[i1],A[j1],B[i2],B[j2]))return 0;//两线段相交
            if(PIP(B,m,A[i1])||PIP(A,n,B[i2]))return 0;//点包含在内
        }
    }
    return 1;
}
```

```
/*五：【图形面积】*/

/*1.【任意多边形面积】*/
inline LD PolyArea(Point *P,Re n){//【任意多边形P的面积】
    LD S=0;
    for(Re i=1;i<=n;++i)S+=Cro(P[i],P[i<n?i+1:1]);
    return S/2.0;
}

/*2.【圆的面积并】*/

/*3.【三角形面积并】*/


/*六：【凸包】*/

/*1.【求凸包】*/
inline bool cmp1(Vector a,Vector b){return a.x==b.x?a.y<b.y:a.x<b.x;};//按坐标排序
inline int ConvexHull(Point *P,Re n,Point *cp){//【水平序Graham扫描法（Andrew算法）】求凸包
    sort(P+1,P+n+1,cmp1);
    Re t=0;
    for(Re i=1;i<=n;++i){//下凸包
        while(t>1&&dcmp(Cro(cp[t]-cp[t-1],P[i]-cp[t-1]))<=0)--t;
        cp[++t]=P[i];
    }
    Re St=t;
    for(Re i=n-1;i>=1;--i){//上凸包
        while(t>St&&dcmp(Cro(cp[t]-cp[t-1],P[i]-cp[t-1]))<=0)--t;
        cp[++t]=P[i];
    }
    return --t;//要减一
}
/*2.【旋转卡壳】*/

/*3.【半平面交】*/
struct Line{
    Point a,b;LD k;Line(Point A=Point(0,0),Point B=Point(0,0))
{a=A,b=B,k=atan2(b.y-a.y,b.x-a.x);}
    inline bool operator<(const Line &O)const{return dcmp(k-O.k)?dcmp(k-O.k)
<0:judge(O.a,O.b,a);}//如果角度相等则取左边的
}L[N],Q[N];
inline Point cross(Line L1,Line L2){return cross_LL(L1.a,L1.b,L2.a,L2.b);}//获取直线L1,L2的交点
inline int judge(Line L,Point a){return dcmp(Cro(a-L.a,L.b-L.a))>0;}//判断点a是否在直线L的右边
inline int halfcut(Line *L,Re n,Point *P){//【半平面交】
    sort(L+1,L+n+1);Re m=n;n=0;
    for(Re i=1;i<=m;++i)if(i==1||dcmp(L[i].k-L[i-1].k))L[++n]=L[i];
    Re h=1,t=0;
    for(Re i=1;i<=n;++i){
        while(h<t&&judge(L[i],cross(Q[t],Q[t-1])))--t;//当队尾两个直线交点不是在直线L[i]上或者左边时就出队
        while(h<t&&judge(L[i],cross(Q[h],Q[h+1])))++h;//当队头两个直线交点不是在直线L[i]上或者左边时就出队
        Q[++t]=L[i];
    }
    while(h<t&&judge(Q[h],cross(Q[t],Q[t-1])))--t;
```

```
    while(h<t&&judge(Q[t],cross(Q[h],Q[h+1])))++h;
    n=0;
    for(Re i=h;i<=t;++i)P[++n]=cross(Q[i],Q[i<t?i+1:h]);
    return n;
}

/*4.【闵可夫斯基和】*/
Vector V1[N],V2[N];
inline int Mincowski(Point *P1,Re n,Point *P2,Re m,Vector *V){//【闵可夫斯基和】求两个凸包{P1},{P2}的向量集合{V}={P1+P2}构成的凸包
    for(Re i=1;i<=n;++i)V1[i]=P1[i<n?i+1:1]-P1[i];
    for(Re i=1;i<=m;++i)V2[i]=P2[i<m?i+1:1]-P2[i];
    Re t=0,i=1,j=1;V[++t]=P1[1]+P2[1];
    while(i<=n&&j<=m)++t,V[t]=V[t-1]+(dcmp(Cro(V1[i],V2[j]))>0?V1[i++]:V2[j++]);
    while(i<=n)++t,V[t]=V[t-1]+V1[i++];
    while(j<=m)++t,V[t]=V[t-1]+V2[j++];
    return t;
}

/*5.【动态凸包】*/

/*七：【圆】*/

/*1.【三点确定一圆】*/
#define S(a) ((a)*(a))
struct Circle{Point O;LD r;Circle(Point P,LD R=0){O=P,r=R;}};
inline Circle getCircle(Point A,Point B,Point C){//【三点确定一圆】暴力解方程
    LD x1=A.x,y1=A.y,x2=B.x,y2=B.y,x3=C.x,y3=C.y;
    LD D=((S(x2)+S(y2)-S(x3)-S(y3))*(y1-y2)-(S(x1)+S(y1)-S(x2)-S(y2))*(y2-y3))/((x1-x2)*(y2-y3)-(x2-x3)*(y1-y2));
    LD E=(S(x1)+S(y1)-S(x2)-S(y2)+D*(x1-x2))/(y2-y1);
    LD F=-(S(x1)+S(y1)+D*x1+E*y1);
    return Circle(Point(-D/2.0,-E/2.0),sqrt((S(D)+S(E)-4.0*F)/4.0));
}
inline Circle getcircle(Point A,Point B,Point C){//【三点确定一圆】向量垂心法
    Point P1=(A+B)*0.5,P2=(A+C)*0.5;
    Point O=cross_LL(P1,P1+Normal(B-A),P2,P2+Normal(C-A));
    return Circle(O,Len(A-O));
}

/*2.【最小覆盖圆】*/
inline int PIC(Circle C,Point a){return dcmp(Len(a-C.O)-C.r)<=0;}//判断点A是否在圆C内
inline void Random(Point *P,Re n){for(Re i=1;i<=n;++i)swap(P[i],P[rand()%n+1]);}//随机一个排列
inline Circle Min_Circle(Point *P,Re n){//【求点集P的最小覆盖圆】
//  random_shuffle(P+1,P+n+1);
    Random(P,n);Circle C=Circle(P[1],0);
    for(Re i=2;i<=n;++i)if(!PIC(C,P[i])){
        C=Circle(P[i],0);
        for(Re j=1;j<i;++j)if(!PIC(C,P[j])){
            C.O=(P[i]+P[j])*0.5,C.r=Len(P[j]-C.O);
            for(Re k=1;k<j;++k)if(!PIC(C,P[k]))C=getcircle(P[i],P[j],P[k]);
        }
    }
    return C;
}
```

```
/*3.【三角剖分】*/
inline LD calc(Point A,Point B,Point O,LD R){//【三角剖分】
    if(A==O||B==O)return 0;
    Re op=dcmp(Cro(A-O,B-O))>0?1:-1;LD ans=0;
    Vector x=A-O,y=B-O;
    Re flag1=dcmp(Len(x)-R)>0,flag2=dcmp(Len(y)-R)>0;
    if(!flag1&&!flag2)ans=Abs(Cro(A-O,B-O))/2.0;//两个点都在里面
    else if(flag1&&flag2){//两个点都在外面
        if(dcmp(dis_PL(O,A,B)-R)>=0)ans=R*R*Angle(x,y)/2.0;//完全包含了圆弧
        else{//分三段处理 △+圆弧+△
            if(dcmp(Cro(A-O,B-O))>0)swap(A,B);//把A换到左边
            Point F=FootPoint(O,A,B);LD lenx=Len(F-O),len=sqrt(R*R-lenx*lenx);
            Vector z=turn_P(F-O,Pi/2.0)*(len/lenx);Point B_=F+z,A_=F-z;
            ans=R*R*(Angle(A-O,A_-O)+Angle(B-O,B_-O))/2.0+Cro(B_-O,A_-O)/2.0;
        }
    }
    else{//一个点在里面，一个点在外面
        if(flag1)swap(A,B);//使A为里面的点，B为外面的点
        Point F=FootPoint(O,A,B);LD lenx=Len(F-O),len=sqrt(R*R-lenx*lenx);
        Vector z=turn_P(F-O,Pi/2.0)*(len/lenx);Point C=dcmp(Cro(A-O,B-O))>0?F-
z:F+z;
        ans=Abs(Cro(A-O,C-O))/2.0+R*R*Angle(C-O,B-O)/2.0;
    }
    return ans*op;
}

int main(){}
```

2020ICPC昆明 I-Mr. Main and Windmills

给定S,T点和n个点，每次询问第i个点与其他点连线中第k个落在ST线段上的点（从S到T排序）。

上面的板子精度爆了，再超一个。（求直线和线段的交点！！）

```
const int inf = 0x3f3f3f3f;
#define PI acos(-1)
const double eps = 1e-8;
const int maxn=1e6+100;
int n,m;
const double epsi=1e-10;
struct Point {
    double x,y;
    Point(double _x=0,double _y=0):x(_x),y(_y) {}
    Point operator -(const Point &op) const {
        return Point(x-op.x,y-op.y);
    }
    double operator ^(const Point &op) const {
        return x*op.y-y*op.x;
    }
};
inline int sign(const double &x) {
    if(x>epsi) return 1;
    if(x<-epsi) return -1;
    return 0;
}
```

```
inline double sqr( const double &x) {
    return x*x;
}
inline double dis(const Point &p1,const Point &p2) {
    return sqrt(sqr(p1.x-p2.x)+sqr(p1.y-p2.y));
}
inline double mul(const Point &p0,const Point &p1,const Point &p2) {
    return (p1-p0)^(p2-p0);
}
inline int cross(const Point &p1,const Point &p2,const Point &p3,const Point
&p4,Point &p) {
    double a1=mul(p1,p2,p3),a2=mul(p1,p2,p4);
    if(sign(a1)==0&&sign(a2)==0) return 2;//重叠
    if(sign(a1-a2)==0) return 0;//平行
    p.x=(a2*p3.x-a1*p4.x)/(a2-a1);
    p.y=(a2*p3.y-a1*p4.y)/(a2-a1);
    return 1;//相交
}
Point a[1100],s,e;
struct node {
    double x,y,d;
} b[1100];
bool cmp(node a,node b) {
    return a.d<b.d;
}
int main() {
    n=read,m=read;
    scanf("%lf%lf%lf%lf",&s.x,&s.y,&e.x,&e.y);
    rep(i,1,n) {
        scanf("%lf%lf",&a[i].x,&a[i].y);
    }
    double FUCK=dis(s,e);
    rep(i,1,m) {
        int idx=0,h=read,k=read;
        rep(j,1,n) {
            if(j==h) continue;
            Point t;
            int flag=cross(a[h],a[j],s,e,t);
            if(flag&&dis(t,s)<=FUCK&&dis(t,e)<=FUCK) {
                b[++idx]= {t.x,t.y,dis(t,s)};
            }
        }
        if(idx<k) puts("-1");
        else {
            sort(b+1,b+1+idx,cmp);
            printf("%.10lf %.10lf\n",b[k].x,b[k].y);
        }
    }
    return 0;
}
```

2022 牛客多校3 G Geometry

给定两个凸包的位置和运动向量，求相撞时间。

闵可夫斯基和+优秀的计算几何板子。

```
#include<bits/stdc++.h>
#define Vector Point
//#define inf 0x7f7f7f7f
//#define int long long
#define db long double
using namespace std;
typedef long long ll;
//const int N=1e5+7;
const db eps=1e-10,pi=acos(-1.0);
ll read(){ll x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}

//inline int dcmp(db a){return a<-eps?-1:(a>eps?1:0);}  //处理精度
//inline db Abs(db a){return a*dcmp(a);} //取绝对值

struct Point{
    db x,y;
    Point(){ SetZero(); }
    Point(db _x,db _y){ Set(_x,_y); }
    inline void Set(db _x,db _y){ x=_x,y=_y; }
    inline void SetZero(){ x=y=0; }
    inline Point operator +(const Point &v) const{ return Point(x+v.x,y+v.y); }
    inline Point operator +=(const Point &v){ return *this=*this+v; }
    inline Point operator -() const{ return Point(-x,-y); }
    inline Point operator -(const Point &v) const{ return Point(x-v.x,y-v.y); }
    inline Point operator -=(const Point &v){ return *this=*this-v; }
    inline Point operator *(db f) const{ return Point(f*x,f*y); }
    inline friend Point operator *(db f,const Point &v){ return v*f; }
    inline Point operator *=(db f){ return *this=*this*f; }
    inline Point operator /(db f) const{ return *this*(1.0/f); }
    inline Point operator /=(db f){ return *this=*this/f; }

    inline bool operator <(const Point &b){
        auto up=[](const Point &a){
            if(a.IsZero()) return -1;
            return int(a.y>eps||(a.y>-eps&&a.x>-eps));
        };
        int qa=up(*this),qb=up(b);
        if(qa!=qb) return qa>qb;
        return Cross(*this,b)>eps;
    }
    inline bool IsZero() const{ return abs(x)<=eps&&abs(y)<=eps; }
    inline bool operator ==(const Point &v) const{ return (*this-v).IsZero(); }
    inline friend db Cross(const Point &p,const Point &q){ return p.x*q.y-
p.y*q.x; }
    inline friend db Dot(const Point &p,const Point &q){ return p.x*q.x+p.y*q.y;
}
    inline Point Rot90() const{ return Point(-y,x); }
    inline Point Rot90CW() const{ return Point(y,-x); }
    inline db SqrLen() const{ return x*x+y*y; }
```

```
    inline db Length() const{ return sqrt(x*x+y*y); }
    inline Point Normalized() const{
        db len=Length();
        if(len<=eps) return Point(0,0);
        db invLen=1.0/len;
        return Point(x*invLen,y*invLen);
    }
    inline Point Normalize(){ return *this=this->Normalized(); }
    inline db Arg() const{ return atan2(y,x); }
    inline friend istream &operator >>(istream &is,Point &v){ return
is>>v.x>>v.y; }
    inline friend ostream &operator <<(ostream &os,const Point &v){
        os<<setiosflags(ios::fixed)<<setprecision(6);
        os<<"("<<setw(9)<<v.x<<','<<setw(9)<<v.y<<")";
        return os<<setprecision(6)<<resetiosflags(ios::fixed);
    }
};

struct Line{
    int u,v;
    Point ori,dir;
    Line(){ SetZero(); }
    Line(const Point &_ori,const Point &_dir){ Set(_ori,_dir); }
    inline void Set(const Point &_ori,const Point &_dir){
        ori=_ori,dir=_dir.Normalized();
    }
    inline void SetdbwoPoints(const Point &p1,const Point &p2){ Set(p1,p2-p1); }
    inline void SetZero(){ ori.SetZero(),dir.Set(1,0); }
    inline bool OnLeft(const Point &p){
        return Cross(dir,p-ori)>0;
    }
    inline db Arg() const{ return dir.Arg(); }
    inline friend Point Intersect(const Line &lA,const Line &lB){
        db k=Cross(lB.ori-lA.ori,lB.dir)/Cross(lA.dir,lB.dir);
        return lA.ori+lA.dir*k;
    }
};


vector<Point> minksum(vector<Point> &a,vector<Point> &b){
    int n=a.size(),m=b.size();
    vector<Point> ret(n+m+1);
    auto cmp=[](Point l,Point r){ return l.y<r.y||(l.y==r.y&&l.x<r.x); };
    rotate(a.begin(),min_element(a.begin(),a.end(),cmp),a.end());
    rotate(b.begin(),min_element(b.begin(),b.end(),cmp),b.end());
    ret[0]=a[0]+b[0];
    vector<Point> qa(n),qb(m);
    for(int i=0;i<n;++i) qa[i]=a[(i+1)%n]-a[i];
    for(int i=0;i<m;++i) qb[i]=b[(i+1)%m]-b[i];
    merge(qa.begin(),qa.end(),qb.begin(),qb.end(),ret.begin()+1);
    for(int i=1;i<=n+m;++i) ret[i]+=ret[i-1];
    return ret;
}


bool InPoly(const vector<Point> &ps,Point p){  //是否在凸包里边
    int n=ps.size();
    if(n<3) return 0;
```

```
    for(int i=0;i<n;++i){
        if(Cross(ps[i]-p,ps[(i+1)%n]-p)<-eps) return 0;
    }
    return 1;
}

signed main(){
    int n=read();
    vector<Point> A(n);
    for(auto &p:A) p.x=read(),p.y=read();
    int m=read();
    vector<Point> B(m);
    for(auto &p:B) p.x=-read(),p.y=-read();
    Point v;
    v.x=-read(),v.y=-read();
    v.x+=read(),v.y+=read();
    auto H=minksum(A,B);
    if(InPoly(H,Point())) return puts("0"),0;
    if(v.IsZero()) return puts("-1"),0;
    int N=H.size();
    db ans=1e19;
    for(int i=0;i<N;++i){
        Point p=H[i],q=H[(i+1)%N];
        Line l(Point(),v);
        if(Cross(q,v)*Cross(p,v)<=eps){
            auto s=Intersect(l,Line(p,q-p));
            if(Dot(s,v)<-eps) continue;
            ans=min(ans,s.Length()/v.Length());
        }
    }
    if(ans>1e18) puts("-1");
    else printf("%.10Lf",ans);
    return 0;
}
```

题杂

圆的周长并

```
#include<bits/stdc++.h>
using namespace std;
const int N = 2005;
const double pi = acos(-1.0);
int read() {
    char ch = getchar(); int x = 0, f = 1;
    while(ch < '0' || ch > '9') {if(ch == '-') f = -1; ch = getchar();}
    while(ch >= '0' && ch <= '9') {x = x * 10 + ch - '0'; ch = getchar();}
    return x * f;
}
int cnt, n;
struct dat {double l, r;}opt[N];
bool operator < (dat a, dat b) {return a.l < b.l;}
struct C {
    double x, y, r;
    C(double a = 0, double b = 0, double c = 0) : x(a), y(b), r(c) {}
```

```
    C operator - (C a) {return C(x - a.x, y - a.y, 0);}
    double operator ^ (C a) {return x * a.x + y * a.y;}
}p[N];
double sqr(double x) {return x * x;}
double sqr(C a) {return a ^ a;}
double angle(C p) {return atan2(p.y, p.x);}
double angle(double a, double b, double c) {return acos((a * a + b * b - c * c)
/ (2 * a * b));}
void add(double a, double b) {opt[++cnt].l = a; opt[cnt].r = b;}
void Solve(C u, C v, double dis) {
    double t1 = angle(u - v), t2 = angle(u.r, dis, v.r);
    double l = t1 - t2, r = t1 + t2;
    if(l < 0) l += 2 * pi;
    if(r < 0) r += 2 * pi;
    if(l > r) {add(0, r); add(l, 2 * pi);}
    else add(l, r);
}
double calc() {
    sort(opt + 1, opt + cnt + 1); double l = -10, r = -10, ret = 0;
    for(int i = 1;i <= cnt; ++i)
    if(opt[i].l > r) {ret += r - l; l = opt[i].l; r = opt[i].r;}
    else r = max(r, opt[i].r);
    ret += r - l; return 2 * pi - ret;
}

signed main() {
    scanf("%d", &n);
    for(int i = n; i; --i) scanf("%lf%lf%lf", &p[i].x, &p[i].y, &p[i].r);
    double ans = 0;
    for(int i = 1, j;i <= n; ++i) {
        cnt = 0;
        for(j = 1;j < i; ++j) {
            double dis = sqrt(sqr(p[j] - p[i]));
            if(p[j].r - p[i].r > dis) break;
            if(p[j].r + p[i].r > dis && fabs(p[j].r - p[i].r) < dis) Solve(p[i],
p[j], dis);
        }
        if(j == i) ans += p[i].r * calc();
    }
    printf("%.12lf\n", ans);
    return 0;
}
```

圆的面积并

```
//minamoto
#include<bits/stdc++.h>
#define R register
#define inline __inline__ __attribute__((always_inline))
#define fp(i,a,b) for(R int i=(a),I=(b)+1;i<I;++i)
#define fd(i,a,b) for(R int i=(a),I=(b)-1;i>I;--i)
#define go(u) for(int i=head[u],v=e[i].v;i;i=e[i].nx,v=e[i].v)
template<class T>inline bool cmax(T&a,const T&b){return a<b?a=b,1:0;}
template<class T>inline bool cmin(T&a,const T&b){return a>b?a=b,1:0;}
using namespace std;
```

```
const int N=1005;const double Pi=acos(-1.0);
struct Point{
    int x,y;
    inline Point(){}
    inline Point(R int xx,R int yy):x(xx),y(yy){}
    inline Point operator +(const Point &b)const{return Point(x+b.x,y+b.y);}
    inline Point operator -(const Point &b)const{return Point(x-b.x,y-b.y);}
    inline bool operator <(const Point &b)const{return x<b.x||(x==b.x&&y<b.y);}
    inline bool operator ==(const Point &b)const{return x==b.x&&y==b.y;}
    inline double norm(){return sqrt(x*x+y*y);}
};
struct Cir{
    Point p;int r;
    inline bool operator <(const Cir &b)const{return p<b.p||p==b.p&&r<b.r;}
    inline bool operator ==(const Cir &b)const{return p==b.p&&r==b.r;}
    inline double oint(R double t1,R double t2){
        return r*(r*(t2-t1)+p.x*(sin(t2)-sin(t1))-p.y*(cos(t2)-cos(t1)));
    }
}c[N];
pair<double,int>st[N<<1];int n;double res;
double calc(int id){
    int top=0,cnt=0;
    fp(i,1,n)if(i!=id){
        double dis=(c[i].p-c[id].p).norm();
        if(c[id].r+dis<=c[i].r)return 0;
        if(c[i].r+dis<=c[id].r||c[i].r+c[id].r<=dis)continue;
        double del=acos((c[id].r*c[id].r+dis*dis-
c[i].r*c[i].r)/(2*c[id].r*dis));
        double ang=atan2(c[i].p.y-c[id].p.y,c[i].p.x-c[id].p.x);
        double l=ang-del,r=ang+del;
        if(l<-Pi)l+=2*Pi;if(r>=Pi)r-=2*Pi;
        if(l>r)++cnt;
        st[++top]=make_pair(l,1),st[++top]=make_pair(r,-1);
    }
    st[0]=make_pair(-Pi,0),st[++top]=make_pair(Pi,0);
    sort(st+1,st+1+top);
    double res=0;
    for(R int i=1;i<=top;cnt+=st[i++].second)
        if(!cnt)res+=c[id].oint(st[i-1].first,st[i].first);
    return res;
}
int main(){
//  freopen("testdata.in","r",stdin);
    scanf("%d",&n);
    fp(i,1,n)scanf("%d%d%d",&c[i].p.x,&c[i].p.y,&c[i].r);
    sort(c+1,c+1+n),n=unique(c+1,c+1+n)-c-1;
    fp(i,1,n)res+=calc(i);
    printf("%.3lf\n",res*0.5);
    return 0;
}
```

凸包期望面积

给定每个点的坐标和出现概率，求形成凸包的期望面积。

```
//By BLADEVIL
#include <cstdio>
```

```
#define maxn 110

using namespace std;

int n;
int x[maxn],y[maxn];
double p[maxn];

bool judge(int i,int j,int k) {
    return ((x[j]-x[i])*(y[k]-y[i])-(y[j]-y[i])*(x[k]-x[i]))<0;
}

signed main() {
    scanf("%d",&n);
    double ans=0;
    for (int i=1;i<=n;i++) scanf("%d%d%lf",&x[i],&y[i],&p[i]);
    for (int i=1;i<=n;i++)
        for (int j=1;j<=n;j++)
            if (i!=j) {
                double q=p[i]*p[j];
                for (int k=1;k<=n;k++)
                    if ((k!=i)&&(k!=j)&&(judge(i,j,k)))
                        q*=1-p[k];
                ans+=q*(x[i]*y[j]-y[i]*x[j]);
            }
    ans/=2;
    printf("%.6f",ans+1e-8);
    fclose(stdin); fclose(stdout);
    return 0;
}
```

【模板】半平面交

```
#include<bits/stdc++.h>
#define N 100006
#define eps 1e-13
using namespace std;

inline double abs(const double &a){return a>0?a:-a;}
struct Vector{
    double x,y;
    inline void input(){scanf("%lf%lf",&x,&y);}
    inline double len(){return std::sqrt(x*x+y*y);}
    inline void operator += (const Vector &a){x+=a.x;y+=a.y;}
    inline void operator -= (const Vector &a){x-=a.x;y-=a.y;}
    inline void operator *= (const double &a){x*=a;y*=a;}
    inline void operator /= (const double &a){x/=a;y/=a;}
};
inline Vector operator + (const Vector &a,const Vector &b){return (Vector)
{a.x+b.x,a.y+b.y};}
inline Vector operator - (const Vector &a,const Vector &b){return (Vector){a.x-
b.x,a.y-b.y};}
inline Vector operator * (const Vector &a,const double &b){return (Vector)
{a.x*b,a.y*b};}
```

```
inline Vector operator / (const Vector &a,const double &b){return (Vector)
{a.x/b,a.y/b};}
inline double dot(const Vector &a,const Vector &b){return a.x*b.x+a.y*b.y;}
inline double cross(const Vector &a,const Vector &b){return a.x*b.y-a.y*b.x;}
struct Line{
    Vector p,way;
    double ang;
    inline void makeLine(const Vector &a,const Vector &b)
{p=a;way=b;ang=atan2(b.y,b.x);}
};
inline int onRight(const Line &a,const Vector &b){return cross(a.way,b-a.p)<=-
eps;}
inline int cmp(const Line &a,const Line &b){return a.ang<b.ang;}
inline Vector intersect(const Line &a,const Line &b){
    double x=cross(b.way,a.p-b.p)/cross(a.way,b.way);
    return a.p+a.way*x;
}
inline double polygonArea(int n,Vector *a){
    double S=0;
    for(int i=1;i<n;i++) S+=cross(a[i],a[i+1]);
    S+=cross(a[n],a[1]);
    return S/2;
}
int lf,rg;
Line que[N];

inline int halfPlane(int n,Line *a,Vector *p){
    std::sort(a+1,a+1+n,cmp);
    lf=rg=0;que[0]=a[1];
    for(int i=2;i<=n;i++){
        while(lf<rg&&onRight(a[i],p[rg])) rg--;
        while(lf<rg&&onRight(a[i],p[lf+1])) lf++;
        que[++rg]=a[i];
        if(fabs(cross(que[rg].way,que[rg-1].way))<=eps){//平行
            if(onRight(que[rg],que[rg-1].p)&&dot(que[rg].way,que[rg-1].way)<=-
eps) return 0;
            rg--;
            if(!onRight(que[rg],a[i].p)) que[rg]=a[i];
        }
        if(lf<rg) p[rg]=intersect(que[rg],que[rg-1]);
    }
    while(lf<rg&&onRight(que[lf],p[rg])) rg--;
    if(rg-lf<=1) return 0;
    p[lf]=intersect(que[lf],que[rg]);
    return 1;
}
Vector p[N],in[N];
Line a[N];
int main(){
    int t,m,o=0;
    scanf("%d",&t);
    while(t--){
        scanf("%d",&m);
        for(int i=1;i<=m;i++) in[i].input();
        for(int i=1;i<m;i++) a[++o].makeLine(in[i],in[i+1]-in[i]);
        a[++o].makeLine(in[m],in[1]-in[m]);
    }
    if(!halfPlane(o,a,p)) puts("0.000");
```

```
    else printf("%.3lf\n",polygonArea(rg-lf+1,p+lf-1));
    return 0;
}
```

[HNOI2008]水平可见直线]

在二维直角坐标系上，给 n 条直线的斜率和截距，问你从在这个坐标系 y 轴方向无穷远处，能看到多少条直线，从小到大输出其标号。

```
#include<bits/stdc++.h>
using namespace std;
const int N=50005;

struct Nod{
    int a,b,id;
    bool operator <(const Nod &p)const{
        if(a^p.a) return a>p.a;
        return b>p.b;
    }
}s[N];

int n,stk[N],top,ans[N];

double calc(int i,int j){ //计算交点
    return ((1.0*(s[i].b-s[j].b))/(s[j].a-s[i].a));
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n;
    for(int i=1;i<=n;++i){
        cin>>s[i].a>>s[i].b;
        s[i].id=i;
    }
    stable_sort(s+1,s+1+n);
    for(int i=1;i<=n;++i){
        if(s[i].a==s[i-1].a&&i>1) continue;
        while(top>1&&calc(stk[top],i)>=calc(stk[top],stk[top-1])) --top;
        stk[++top]=i;
        ans[top]=s[i].id;
    }
    stable_sort(ans+1,ans+1+top);
    for(int i=1;i<=top;++i) cout<<ans[i]<<" \n"[i==top];
    return 0;
}
```

三维

两球相交部分体积（球缺）

```
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<iostream>
#define CLR(a,b) memset(a,b,sizeof(a));
const int inf=0x3f3f3f3f;
using namespace std;
const double PI = acos(-1.0);
typedef unsigned long long ll;
const int maxn= 110;
typedef struct point {
    double x,y,z;
    point() {
    }
    point(double a, double b,double c) {
        x = a;
        y = b;
        z = c;
    }
    point operator -(const point &b)const {     //返回减去后的新点
        return point(x - b.x, y - b.y,z-b.z);
    }
    point operator +(const point &b)const {     //返回加上后的新点
        return point(x + b.x, y + b.y,z+b.z);
    }
    //数乘计算
    point operator *(const double &k)const {    //返回相乘后的新点
        return point(x * k, y * k,z*k);
    }
    point operator /(const double &k)const {    //返回相除后的新点
        return point(x / k, y / k,z/k);
    }
    double operator *(const point &b)const {    //点乘
        return x*b.x + y*b.y+z*b.z;
    }
}point;

double dist(point p1, point p2) {       //返回平面上两点距离
    return sqrt((p1 - p2)*(p1 - p2));
}
typedef struct sphere {//球
    double r;
    point centre;
}sphere;
sphere s,a[maxn];
void SphereInterVS(sphere a, sphere b,double &v,double &s) {
    double d = dist(a.centre, b.centre);//球心距
    double t = (d*d + a.r*a.r - b.r*b.r) / (2.0 * d);//
    double h = sqrt((a.r*a.r) - (t*t)) * 2;//h1=h2，球冠的高
    double angle_a = 2 * acos((a.r*a.r + d*d - b.r*b.r) / (2.0 * a.r*d));  //余弦公式计算r1对应圆心角，弧度
    double angle_b = 2 * acos((b.r*b.r + d*d - a.r*a.r) / (2.0 * b.r*d));  //余弦公式计算r2对应圆心角，弧度
```

```
    double l1 = ((a.r*a.r - b.r*b.r) / d + d) / 2;
    double l2 = d - l1;
    double x1 = a.r - l1, x2 = b.r - l2;//分别为两个球缺的高度
    double v1 = PI*x1*x1*(a.r - x1 / 3);//相交部分r1圆所对应的球缺部分体积
    double v2 = PI*x2*x2*(b.r - x2 / 3);//相交部分r2圆所对应的球缺部分体积
     v = v1 + v2;//相交部分体积
    double s1 = PI*a.r*x1;  //r1对应球冠表面积
    double s2 = PI*a.r*x2;  //r2对应球冠表面积
     s = 4 * PI*(a.r*a.r + b.r*b.r) - s1 - s2;//剩余部分表面积
}
int t, n;
double x, y, z, r;
int cas = 1;
int main()
{
    cin >> t;
    while(t--)
    {
        cin >> n;
        for(int i = 1; i <= n; i++)
        {
            scanf("%lf%lf%lf%lf",&x,&y,&z,&a[i].r);
            a[i].centre = {x,y,z};
        }
        scanf("%lf%lf%lf%lf",&x,&y,&z,&r);
        s.r = r;
        s.centre = {x,y,z};
        double ans = 0, v = 0;
        for(int i = 1; i <= n; i++)
        {
            double ss, dis = dist(s.centre, a[i].centre);

            if(dis >= s.r + a[i].r)continue;  //在外部
            if(dis + min(s.r, a[i].r) <= max(s.r, a[i].r))  //在内部
            {
                ans += 4.0 / 3.0 * PI * min(s.r,a[i].r) * min(s.r,a[i].r) *
min(s.r,a[i].r);
                continue;
            }
            SphereInterVS(s, a[i], v, ss); //相交部分
            ans += v;
        }
        printf("Case #%d: %.14f\n",cas++,ans);
    }
}
```

两球面积交

前导知识：

球冠：球面被平面所截得的一部分叫做球冠.

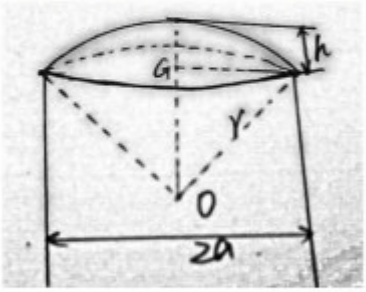
截得的圆得的圆叫做球.

垂直于截面的直径被截得的一段叫做球冠的高.

如图球冠：

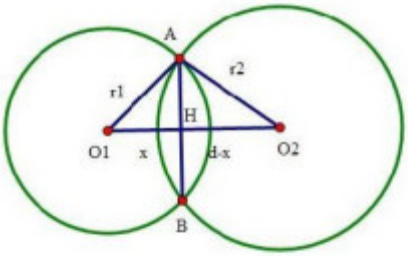
体积$V=\frac{\pi}{6}h(3a^2+h^2)=\frac{\pi}{3}h^2(3r-h)$

r为球半径，a为拱底圆半径，h为拱高



那么两个半径分别为$r1,r2$的球相交，

求相交部分的体积即求两个球冠的体积之和：



可以求得球$O1$的球冠的参数为$r=r_1$，

$$h=r_1-\frac{r_1^2-r_2^2+d^2}{2d}$$

球$O2$的球冠的参数为$r=r_2$，

$$h=r_2-\frac{r_2^2-r_1^2+d^2}{2d}$$

带入上述的球冠体积公式得：

$$V=V_1+V_2$$

$$=\frac{\pi}{3}(r_1-\frac{r_1^2-r_2^2+d^2}{2d})^2(3r_1-(r_1-\frac{r_1^2-r_2^2+d^2}{2d}))$$

$$+\frac{\pi}{3}(r_2-\frac{r_2^2-r_1^2+d^2}{2d})^2(3r_2-(r_2-\frac{r_2^2-r_1^2+d^2}{2d}))$$

化简得……

三维空间中两线段的最短点对

一、 (本题 15 分) 设 L_1 和L_2是空间中两异面直线.设在标准直角坐标系下直线L_1过坐标为a的点，以单位向量v为直线方向；直线L_2过坐标为b的点，以单位向量w为直线方向.

1) 证明：存在唯一点$P \in L_1$ 和$Q \in L_2$ 使得两点连线PQ 同时垂直于 L_1 和L_2.

2)求P点和Q点坐标(用a, b, v, w表示).

解：1）过直线L_2上一点和线性无关向量 v 和w 做平面σ，则直线L_2落在平面σ上，且直线L_1平行于平面σ 。过L_1做平面τ 垂直于平面σ ，记两平面交线为L_1^*。设两直线L_1^*和L_2的交点为Q, 过Q做平面σ 的法线，交直线L_1为P, 则 PQ 同时垂直于L_1和L_2。 .....(4分)

设$X = P + sv \in L_1$ 和$Y = Q + tw \in L_2$ 也使得XY同时垂直于L_1和L_2，则有

$\overrightarrow{XY} = \overrightarrow{PQ} - sv + tw$垂直于 v 和w ，故有 $-s + (v \cdot w)t = 0$ 和 $-s(v \cdot w) + t = 0$ 。由于$(v \cdot w)^2 < 1$ ，我们得到$s = t = 0$, 即$X = P$，$Y = Q$，这样的P和Q存在且唯一。 .....(8分)

2）设$P = a + sv \in L_1$ 和$Q = b + tw \in L_2$ 。因为$\overrightarrow{PQ} = \lambda v \times w$ ，我们得到

$$(b - a) - sv + tw = \lambda v \times w,$$

.....(11分)

于是有

$$(b - a) \cdot v - s + t(v \cdot w) = 0, (b - a) \cdot w - s(v \cdot w) + t = 0$$

故有

$$s = \frac{(b - a) \cdot (v - (v \cdot w)w)}{1 - (v \cdot w)^2}, t = \frac{(a - b) \cdot (w - (v \cdot w)v)}{1 - (v \cdot w)^2}$$

得到

$$P = a + \frac{(b - a) \cdot (v - (v \cdot w)w)}{1 - (v \cdot w)^2} v, Q = b + \frac{(a - b) \cdot (w - (v \cdot w)v)}{1 - (v \cdot w)^2} w.$$

.....(15分)