计算几何

基础

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e3;

struct Point{             //定义点
    double x,y;
    Point(double x=0,double y=0):x(x),y(y){}  //构造函数，方便代码编写
};

struct polygon{          //定义多边形
    int n;     //顶点个数
    Point P[N];  //顶点集合 按顺时针或逆时针排列 N为顶点个数上限
};

struct Circle{          //定义圆
    double r;   //半径
    Point o;    //圆心
};

typedef Point Vector;
//程序上实现向量与点的结构体相同，一组(x,y)可以代表一点点或者一个向量，目前不做严格的区分

//重载实现基本运算
Vector operator + (Vector A,Vector B){return Vector(A.x+B.x,A.y+B.y);} //向量相加
Vector operator - (Vector A,Vector B){return Vector(A.x-B.x,A.y-B.y);} //向量相减
Vector operator * (Vector A,double B){return Vector(A.x*B,A.y*B);}     //向量乘以一个
数
Vector operator / (Vector A,double B){return Vector(A.x/B,A.y/B);}     //向量除以一个
数

const double eps=1e-10;
int dcmp(double x){       //判断数的精度
    if(fabs(x)<eps)return 0;
    return (x>0)?1:-1;
}

bool operator == (const Point&a,const Point &b){  //在精度允许的情况下，两点（向量）相等返
回true，反之false
    return dcmp(a.x-b.x)==0 && dcmp(a.y-b.y)==0;
}

/*向量的点积（又叫 标积 / 内积 / 数量积 /），a·b=|a||b|·cosθ
几何意义：向量a在向量b方向上的投影与向量b的模的乘
坐标公式：A.x*B.x+A.y*B.y;   */
double Dot(Vector A,Vector B){return A.x*B.x+A.y*B.y;} //向量点积
double Length(Vector A){return sqrt(Dot(A,A));}        //求向量长度（通过点积）
```

```
double Angle(Vector A,Vector B){return cos(Dot(A,B)/Length(A)/Length(B));} //求向量夹角（单位为弧度）

/*向量的叉积（又叫 矢积 / 外积 / 向量积 /），a×b=|a||b|·sinθ
几何意义：垂直a、b所在，向量a,b构成的平行四边形的面积
坐标公式： A.x*B.y-B.x*A.y   */
double Cross(Vector A,Vector B){return A.x*B.y-B.x*A.y;} //向量叉积

/*向量旋转  公式 x=x'*cos(rad)-y'*sin(rad)  y=x'*sin(rad)+y'*cos(rad) rad为要旋转的角度（单位为弧度）*/
Vector Rotate(Vector A,double rad){
    return Vector(A.x*cos(rad)-A.y*sin(rad) , A.x*sin(rad)+A.y*cos(rad));
}

/*计算向量的单位法线 ，左转90度，长度归一*/
Vector Normal(Vector A){
    double L=Length(A);
    return Vector(-A.y/L,A.x/L);
}
int main(){
    return 0;
}
```

二维计算几何

基本公式

正弦

余弦

点到直线距离

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
constexpr int inf=0x3f3f3f3f;
constexpr int N=2e5+10;
constexpr int M=1e3+100;
int a[N];

const double eps=1e-10;
//
struct Point{
//
    double x,y;
//
    Point(double x=0,double y=0):x(x),y(y){}
};
```

```
typedef Point Vector;
Vector operator + (Vector A,Vector B){return Vector(A.x+B.x,A.y+B.y);}
Vector operator - (Vector A,Vector B){return Vector(A.x-B.x,A.y-B.y);}
Vector operator * (Vector A,double B){return Vector(A.x*B,A.y*B);}
Vector operator / (Vector A,double B){return Vector(A.x/B,A.y/B);}

int dcmp(double x){if(fabs(x)<eps)return 0;return (x>0)?1:-1;}
bool operator == (const Vector A,const Vector B){
    return dcmp(A.x-B.x)==0 && dcmp(A.y-B.y)==0;
}
double Dot(Vector A,Vector B){return A.x*B.x+A.y*B.y;}  //计算向量点积
double Length(Vector A){return sqrt(Dot(A,A));}         //计算向量长度
//
double Cross(Vector A,Vector B){return A.x*B.y-B.x*A.y;}//计算向量叉积
//

double DistanceToSegment(Point P,Point A,Point B){
    if(A==B)return Length(P-A);
    Vector v1=B-A,v2=P-A,v3=P-B;
    if(dcmp(Dot(v1,v2))<0)return Length(v2);           //第二类第一小类
    else if(dcmp(Dot(v1,v3))>0)return Length(v3);      //第二类第二小类
    else return fabs(Cross(v1,v2))/Length(v1);
}
Point Typhoon[N];
Point shelters[N];
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);cout.tie(0);
    int n,m;
    cin>>m>>n;
    for(int i=1;i<=m;i++)
    {
        cin>>Typhoon[i].x>>Typhoon[i].y;
    }
    for(int i=1;i<=n;i++)
    {
        cin>>shelters[i].x>>shelters[i].y;
    }
    for(int i=1;i<=n;i++)
    {
        double minn=-1;
        for(int j=1;j<=m-1;j++)
        {
            double temp=DistanceToSegment(shelters[i],Typhoon[j],Typhoon[j+1]);
            if(j==1)minn=temp;
            else minn=min(minn,temp);
        }
        // printf("%.4lf\n",minn);
        cout<<fixed<<setprecision(4)<<minn<<"\n";
    }
```

```
    return 0;
}
```

点到圆弧距离

```
#include<algorithm>
#include<iostream>
#include<cstring>
#include<cstdio>
#include<cmath>
using namespace std;
struct Point{
    double x,y;
    Point(double x=0,double y=0):x(x),y(y){}
};
typedef Point Vector;
Vector operator + (Vector A,Vector B){return Vector(A.x+B.x,A.y+B.y);}
Vector operator - (Vector A,Vector B){return Vector(A.x-B.x,A.y-B.y);}
Vector operator * (Vector A,double B){return Vector(A.x*B,A.y*B);}
Vector operator / (Vector A,double B){return Vector(A.x/B,A.y/B);}

double Dot(Vector A,Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Cross(Vector A,Vector B){return A.x*B.y-B.x*A.y;}

Vector Normal(Vector A){
    double l=Length(A);
    return Vector(-(A.y/l),A.x/l);
}
Point GetLineIntersection(Point P,Vector v,Point Q,Vector w){
    Vector u=P-Q;
    double t=Cross(w,u)/Cross(v,w);
    return P+v*t;
}
double Distance(Point a,Point b){
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
int main(){
    Point a,b,c,p,p1;
    int Case=0;

while(~scanf("%lf%lf%lf%lf%lf%lf%lf%lf",&a.x,&a.y,&b.x,&b.y,&c.x,&c.y,&p.x,&p.y)){
        Point mid1,mid2;
        mid1.x=(a.x+b.x)/2;mid1.y=(a.y+b.y)/2;
        mid2.x=(a.x+c.x)/2;mid2.y=(a.y+c.y)/2;
        Vector v1=(b-a);
        Vector v2=(c-a);
        v1=Normal(v1);
        v2=Normal(v2);
        p1=GetLineIntersection(mid1,v1,mid2,v2);//通过三点建立两直线，两直线的垂直平分线交点就是圆心
```

```
        double ans,ans1,ans2;
        double da,dc,dp1,r;
        dp1=Distance(p,p1);
        r=Distance(p1,a);
        da=Distance(p,a);
        dc=Distance(p,c);

        ans1=fabs(dp1-r);
        ans2=min(da,dc);

        double f1=atan2(a.y-p1.y,a.x-p1.x);
        double f2=atan2(b.y-p1.y,b.x-p1.x);
        double f3=atan2(c.y-p1.y,c.x-p1.x);
        double f4=atan2(p.y-p1.y,p.x-p1.x);
        if(f1<f3){
            if((f2<=f3&&f2>=f1) == (f4<=f3&&f4>=f1))ans=ans1;
            else ans=ans2;
        }else{
            if((f2>=f3&&f2<=f1) == (f4>=f3&&f4<=f1))ans=ans1;
            else ans=ans2;
        }
        printf("Case %d: %.3lf\n",++Case,ans);
    }
    return 0;
}
```

极坐标

极角序

两个人の星座

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
const int N=3e3+10;
int n;
int ans= 0;

struct node {
    int x, y, c;
    int a, b;
    bool operator < (const node& t)const {
        if(a * t.b - b * t.a == 0) return x < t.x;
        return (a * t.b - b * t.a > 0) ^ (t.a < 0) ^ (a < 0);
    }
};
node t[N];
```

```
node g[N];
int t1[3];int t2[3];
int bl[N];
void work(int p) {
    t1[0]=t1[1]=t1[2]=0;
    t2[0]=t2[1]=t2[2]=0;
    memcpy(g,t,sizeof(t));
    swap(g[p],g[n]);
    for(int i=1;i<n;i++)
    {
        g[i].a=g[i].x-g[n].x,
        g[i].b=g[i].y-g[n].y;
    }
    sort(g+1,g+n);
    // cout<<g[n].x<<" "<<g[n].y<<endl;
    for(int i=1;i<n;i++)
    {
        cout<<g[i].a<<" "<<g[i].b<<endl;
    }
    cout<<"****\n";
    for(int i=1;i<n;i++)
    {
        if(g[i].a >= 0)
        {
            bl[i]=0;
            t1[g[i].c]++;
        }
        else
        {
            bl[i]=1;
            t2[g[i].c]++;
        }
    }

    for(int i=1;i<n;i++)
    {
        if(bl[i])
            t2[g[i].c] --;
        else
            t1[g[i].c] --;

        int tmp = 1;

        for(int j = 0; j <= 2; j ++)
        {
            if(j != g[n].c) tmp=tmp*t1[j];
        }
        for(int j = 0; j <= 2; j ++)
        {
            if(j != g[i].c) tmp=tmp*t2[j];
        }
```

```
        ans += tmp;
        tmp = 1;

        for(int j = 0; j <= 2; j ++)
        {
            if(j != g[n].c) tmp=tmp*t2[j];
        }
        for(int j = 0; j <= 2; j ++)
        {
            if(j != g[i].c) tmp=tmp*t1[j];
        }

        ans += tmp;
        bl[i] ^= 1;
        if(bl[i]) t2[g[i].c] ++;
        else t1[g[i].c] ++;
    }
}
int main() {
    cin>>n;
    for(int i = 1; i <= n; i ++)
    {
        cin>>t[i].x;
        cin>>t[i].y;
        cin>>t[i].c;
    }

    for(int i = 1; i <= n; i ++) work(i);
    printf("%lld\n", ans / 4);
    return 0;
}
```

距离

曼哈顿距离

切比雪夫

点线面

Pick定理

$$A = i + \frac{1}{2} * b - 1$$

$$\text{面积} = \text{内部格点数目} + \frac{1}{2} * \text{边上格点数目} - 1$$

欧拉公式

$$V - E + F = 2$$

$$\text{点} - \text{边} + \text{面} = 2$$

凸包

求凸包上的点/求凸包的边长

[数论小白都能看懂的平面凸包详解 - ShineEternal的笔记小屋 - 洛谷博客 (luogu.com.cn)]

[Convex Hull - 洛谷 | 计算机科学教育新生态 (luogu.com.cn)]

```
#include <bits/stdc++.h>
using namespace std;
const int N=1e5+5;
const double eps=1e-7;
int n;
struct point {
    double x, y;
    point () {}
    point (double a, double b) : x (a), y (b) {}
    bool operator<(const point &a)const{
        if(a.x==x)
        {
            return a.y>y;
        }
        return a.x>x;
    }
    point operator - (const point &b) {
        return point (x - b.x, y - b.y);
    }
};
point p[N], sp[N];
int cmp (double x) {
    if (fabs (x) < eps) return 0;
    return x > 0 ? 1 : -1;
}
double dis (point a, point b) {
```

```
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
double cp (point a, point b) {
    return a.x * b.y - a.y * b.x;
}
int Andrew () {
    sort(p+1,p+n+1);
    int len=0;
    for (int i=1;i<=n;i++){
        while(len>1&&cmp(cp(sp[len]-sp[len-1],p[i]-sp[len-1]))<0)
        {
            len--;
        }
        sp[++len]=p[i];
    }
    int k=len;
    for(int i=n-1;i>=1;i--)
    {
        while(len>k&&cmp(cp(sp[len]-sp[len-1],p[i]-sp[len-1]))<0)
        {
            len--;
        }
        sp[++len]=p[i];
    }
    return len;
}
int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        cin>>n;
        char c;
        for(int i=1;i<=n;i++)
        {
            cin>>p[i].x>>p[i].y>>c;
        }
        int t=Andrew();
        cout<<t-1<<endl;
        for(int i=1;i<t;i++)
        {
            printf("%.0lf %.0lf\n",sp[i].x,sp[i].y);
        }
    }
    return 0;
}
```

三维凸包 （坑！！）

表面积

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cmath>
using namespace std;
const int N=2010;
const double eps=1e-9;
int n,cnt,vis[N][N];
double ans;
double Rand() {return rand()/(double)RAND_MAX;}
double reps() {return (Rand()-0.5)*eps;}
struct Node
{
    double x,y,z;
    void shake() {x+=reps();y+=reps();z+=reps();}
    double len() {return sqrt(x*x+y*y+z*z);}
    Node operator - (Node A) {return (Node){x-A.x,y-A.y,z-A.z};}
    Node operator * (Node A) {return (Node){y*A.z-z*A.y,z*A.x-x*A.z,x*A.y-y*A.x};}
    double operator & (Node A) {return x*A.x+y*A.y+z*A.z;}
}A[N];
struct Face
{
    int v[3];
    Node Normal() {return (A[v[1]]-A[v[0]])*(A[v[2]]-A[v[0]]);}
    double area() {return Normal().len()/2.0;}
}f[N],C[N];
int see(Face a,Node b) {return ((b-A[a.v[0]])&a.Normal())>0;}
void Convex_3D()
{
    f[++cnt]=(Face){1,2,3};
    f[++cnt]=(Face){3,2,1};
    for(int i=4,cc=0;i<=n;i++)
    {
        for(int j=1,v;j<=cnt;j++)
        {
            if(!(v=see(f[j],A[i]))) C[++cc]=f[j];
            for(int k=0;k<3;k++) vis[f[j].v[k]][f[j].v[(k+1)%3]]=v;
        }
        for(int j=1;j<=cnt;j++)
            for(int k=0;k<3;k++)
            {
                int x=f[j].v[k],y=f[j].v[(k+1)%3];
                if(vis[x][y]&&!vis[y][x]) C[++cc]=(Face){x,y,i};
            }
        for(int j=1;j<=cc;j++) f[j]=C[j];
        cnt=cc;cc=0;
    }
}
int main()
```

```
{
    cin>>n;
    for(int i=1;i<=n;i++) cin>>A[i].x>>A[i].y>>A[i].z,A[i].shake();
    Convex_3D();
    for(int i=1;i<=cnt;i++) ans+=f[i].area();
    printf("%.3f\n",ans);
}
```

体积

```
#include <bits/stdc++.h>
using namespace std;

//Start
typedef long long ll;
typedef double db;
#define mp(a,b) make_pair(a,b)
#define x first
#define y second
#define be(a) a.begin()
#define en(a) a.end()
#define sz(a) int((a).size())
#define pb(a) push_back(a)
const int inf=0x3f3f3f3f;
const ll INF=0x3f3f3f3f3f3f3f3f;

//Data
const int N=2000;
const db eps=1e-9;
int n,m;
db ans;

//Convex
mt19937 orz(time(0));
db reps(){return (1.*(orz()%98)/97-.5)*eps;}
struct point{
    db x,y,z;
    void shake(){x+=reps(),y+=reps(),z+=reps();}
    db len(){return sqrt(x*x+y*y+z*z);}
    point operator-(point p){return (point){x-p.x,y-p.y,z-p.z};}
    point operator*(point p){return (point){y*p.z-p.y*z,z*p.x-p.z*x,x*p.y-p.x*y};}
    db operator^(point p){return x*p.x+y*p.y+z*p.z;}
}a[N];
struct plane{
    int v[3];
    point flag(){return (a[v[1]]-a[v[0]])*(a[v[2]]-a[v[0]]);}
    db area(){return flag().len()/2;}
    db dist(point p){return fabs(((p-a[v[0]])^flag())/flag().len());}
    int see(point p){return ((p-a[v[0]])^flag())>0;}
}f[N],g[N];
int vis[N][N];
```

```
void Convex(){
    #define ft f[j].v[t]
    #define bk f[j].v[(t+1)%3]
    f[m++]=(plane){0,1,2},f[m++]=(plane){2,1,0};
    for(int i=3;i<n;i++){
        int cnt=0,b;
        for(int j=0;j<m;j++){
            if(!(b=f[j].see(a[i]))) g[cnt++]=f[j];
            for(int t=0;t<3;t++) vis[ft][bk]=b;
        }
        for(int j=0;j<m;j++)
            for(int t=0;t<3;t++)
                if(vis[ft][bk]&&!vis[bk][ft]) g[cnt++]=(plane){ft,bk,i};
        m=cnt;
        for(int j=0;j<m;j++) f[j]=g[j];
    }
}

//Main
int main(){
    ios::sync_with_stdio(0);
    cin.tie(0),cout.tie(0);
    cin>>n;
    for(int i=0;i<n;i++) cin>>a[i].x>>a[i].y>>a[i].z,a[i].shake();
    Convex();
    for(int i=0;i<m;i++) ans+=f[i].area()*f[i].dist(a[0])/3;
    cout.precision(2);
    cout<<fixed<<ans<<'\n';
    return 0;
}
```

面积

凸多边形中找最大三角形面积（jiangly）

```
#include<bits/stdc++.h>

#define X first
#define Y second
using namespace std;

typedef long long ll;
typedef pair<ll, ll> pll;
inline pll operator-(const pll &a, const pll &b) { return {a.X - b.X, a.Y - b.Y}; }
inline ll cross(const pll &a, const pll &b) { return a.X * b.Y - a.Y * b.X; }

int main() {
    int n;
    cin >> n;
    vector<pll> ps(n);
    for (auto &[x, y]: ps) cin >> x >> y;
```

```
    auto nxt = [&](int x) { return (x + 1) % n; };
    auto pre = [&](int x) { return (x + n - 1) % n; };
    auto area = [&](int u, int v, int w) { return cross(ps[v] - ps[u], ps[w] -
ps[u]); };

    int r = 0, s = 1, t = 2;
    // 固定i=1，找到一个初始的极大
    ll maxArea = -1;
    for (int j = 1, k = 2; k < n; ++k) {
        while (j + 1 != k && area(0, j + 1, k) > area(0, j, k)) ++j;
        ll S = area(0, j, k);
        if (S > maxArea) maxArea = S, s = j, t = k;
    }

    // 局部调整
    while (true) {
        auto S = area(r, s, t);
        if (area(r, s, nxt(t)) > S) t = nxt(t);
        else if (area(r, s, pre(t)) > S) t = pre(t);
        else if (area(r, nxt(s), t) > S) s = nxt(s);
        else if (area(r, pre(s), t) > S) s = pre(s);
        else if (area(nxt(r), s, t) > S) r = nxt(r);
        else if (area(pre(r), s, t) > S) r = pre(r);
        else break;
    }
    cout << r + 1 << ' ' << s + 1 << ' ' << t + 1 << endl;
    return 0;
}
```

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int inf=0x3f3f3f3f;
const int N=2e5+10;
const int M=1e3+10;
const int mod=1e9+7;
#define pii pair<int,int>
    vector<pii>p;
int area(int k,int q,int r)
{
    pii a=p[k];
    pii b=p[q];
    pii c=p[r];
    int X1=a.first-b.first;
    int X2=a.first-c.first;
    int Y1=a.second-b.second;
    int Y2=a.second-c.second;
    return abs(X1*Y2-X2*Y1);
}
```

```
signed main()
{
    ios::sync_with_stdio(false);
    int n;
    cin>>n;

    for(int i=1;i<=n;i++)
    {
        int x,y;
        cin>>x>>y;
        p.push_back({x,y});
    }
    int r=0;int s=1;int t=2;
    int S=0;

    while(true)
    {
        bool ok=true;
        if(area((r-1+n)%n,s,t)>area(r,s,t))
        {
            r=(r-1+n)%n;
            ok=false;
        }
        if(area((r+1)%n,s,t)>area(r,s,t))
        {
            r=(r+1)%n;
            ok=false;
        }
        if(area(r,(s-1+n)%n,t)>area(r,s,t))
        {
            s=(s-1+n)%n;
            ok=false;
        }
        if(area(r,(s+1)%n,t)>area(r,s,t))
        {
            s=(s+1)%n;
            ok=false;
        }
        if(area(r,s,(t-1+n)%n)>area(r,s,t))
        {
            t=(t-1+n)%n;
            ok=false;
        }
        if(area(r,s,(t+1)%n)>area(r,s,t))
        {
            t=(t+1)%n;
            ok=false;
        }

        if(ok)break;
    }
    if(r>s)swap(r,s);
```

```
    if(r>t)swap(r,t);
    if(s>t)swap(s,t);
    cout<<r+1<<" "<<s+1<<" "<<t+1;
    return 0;
}
```