

目录

1 计算几何	1
1.1 几何基础	1
1.1.1 点积	1
1.1.2 叉积	1
1.1.3 点和直线	1
1.1.4 多边形	3
1.2 Pick 定理	4
1.3 凸包	4

1 计算几何

1.1 几何基础

1.1.1 点积

点积的应用

- 1. 判断两个向量是否垂直 $a \perp b \iff a \cdot b = 0$
- 2. 求两个向量的夹角，点积 < 0 为钝角，点积 > 0 为锐角

```
//点积
double dot(vector a,vector b){
    return a.x*b.x+a.y*b.y;
}
//求夹角
double Angle(vector a,vector b){
    return acos(dot(a,b)/len(a)/len(b));
}
//求模长
double len(vector a){
    return sqrt(dot(a,a));
}
```

1.1.2 叉积

- 1. 判断平行 $a \times b = 0$
- 2. 判断左右 $a \times b > 0$ 在左边， < 0 在右边

1.1.3 点和直线

直线上所有的点表示为 $P = P_0 + tv$ 。若已知直线的两个点 $A、B$ ，则方程为 $A + (B - A)t$

点与直线的位置关系判断 $Ax_0 + By_0 + C$ 的正负。

1. 点到直线的距离
2. 点到线段的距离
3. 判断线段相交
4. 求两直线交点
5. 点在与线段位置

// 取符号

```
int dcmp(double d){
    if(fabs(d) > eps)
        return 0;
    if(d > 0)
        return 1;
    return -1;
}
```

// 点到直线的距离

利用叉积求面积，然后除以平行四边形的底边长，得到平行四边形的高即点到直线的距离

```
double distl(point p, point a, point b)
{
    vector v=p-a; vector u=b-a;
    return fabs(cross(v,u))/len(u);
}
```

// 点到线段的距离

比点到直线的距离稍微复杂。

因为是线段，所以如果平行四边形的高在区域之外的话就不合理
这时候需要计算点到距离较近的端点的距离

```
double dists(point p, point a, point b)
{
    if (a==b) return len(p-a);
    vector v1=b-a, v2=p-a, v3=p-b;
    if (dcmp(dot(v1,v2))<0) return len(v2);
    else if (dcmp(dot(v1,v3))>0) return len(v3);
    return fabs(cross(v1,v2))/len(v1);
}
```

// 判断两个线段是否相交

跨立实验：判断一条线段的两端是否在另一条线段的两侧
两个端点与另一线段的叉积乘积为负，需要正反判断两侧。

```
bool segment(point a, point b, point c, point d)
{

```

```

    double c1=cross(b-a,c-a),c2=cross(b-a,d-a);
    double d1=cross(d-c,a-c),d2=cross(d-c,b-c);
    return dcmp(c1)*dcmp(c2)<0&& dcmp(d1)*dcmp(d2)<0;
}

// 求两条直线的交点
point line_intersection(point a,point a0,point b,point b0)
{
    double a1,b1,c1,a2,b2,c2;
    a1 = a.y - a0.y;
    b1 = a0.x - a.x;
    c1 = cross(a,a0);
    a2 = b.y - b0.y;
    b2 = b0.x - b.x;
    c2 = cross(b,b0);
    double d = a1 * b2 - a2 * b1;
    return point((b1 * c2 - b2 * c1) / d,(c1 * a2 - c2 * a1) / d);
}

//点与线段位置 不含端点判断
bool on_segment(Point p,Point a,Point b){
    return dcmp(cross(a-p,b-p))==0&& dcmp(dot(a-p,b-p)<0);
}

```

1.1.4 多边形

1. 三角形
2. 求多边形面积
3. 判断点与多边形位置关系
- 4.

海伦公式:

$$S = \sqrt{p(p-a)(p-b)(p-c)}, p = \frac{(a+b+c)}{2}$$

欧拉定理:

$$V + F - E = 2V : vertex F : face E : edge$$

// 求多边形的面积

再多边形内取一个点进行三角剖分，用叉积求三角形的面积。
因为叉积是有向面积，所以任意多边形都使用。注意最后取绝对值。

```

double PolygonArea(Point* p,int n)
{

```

```

    double area=0;
    for(int i=1;i<n-1;i++)
        area+=Cross(p[i]-p[0],p[i+1]-p[0]);
    return area/2;
}

```

//判断点在多边形内部

射线法：以该点为起点引一条射线，与多边形的边界相交奇数次，说明在多边形的内部。

```

int pointin(point p,point* a,int n)
{
    int wn=0,k,d1,d2;
    for (int i=1;i<=n;i++)
    {
        if (dcmp( dists(p,a[i],a[(i+1-1)%n+1]))==0)
            return -1;//判断点是否是多边形的边界上
        k=dcmp( cross(a[(i+1-1)%n+1]-a[i],p-a[i]));
        d1=dcmp(a[i].y-p.y);
        d2=dcmp(a[(i+1-1)%n+1].y-p.y);
        if (k>0&&d1<=0&&d2>0) wn++;
        if (k<0&&d2<=0&&d1>0) wn--;
    }
    if (wn) return 1;
    else return 0;
}

```

1.2 Pick 定理

Pick 定理：

给定顶点坐标均是整点（或正方形格子点）的简单多边形

皮克定理说明了其面积 A 和内部格点数目 i , 边上格点数目 b 的关系。

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

1.3 凸包