计算几何

Ebola

Institute of Mathematics,
Zhejiang University.

Jan, 2024

❶ 二维几何基础

❶ 二维几何基础

点与向量

二维平面上的任何一个点，可以用坐标 (x, y) 表示。

点与向量

二维平面上的任何一个点，可以用坐标 (x, y) 表示。

向量是一个“具有方向和长度的箭头”，它不规定起点和终点。
二维平面上的任何一个向量，也可以用坐标 (x, y) 表示。

点与向量

二维平面上的任何一个点，可以用坐标 (x, y) 表示。

向量是一个“具有方向和长度的箭头”，它不规定起点和终点。
二维平面上的任何一个向量，也可以用坐标 (x, y) 表示。

计算机存储点与向量没有区别，所以我们都可以用下面的结构体来存储。

```
1  struct Point{
2      double x,y;
3      Point(double x=0, double y=0): x(x), y(y) {}
4  };
5  #define Vector Point
6  // 在计算机里，Vector 就是 Point，但为了从逻辑上区分，我们赋予它们不同的名字
```

浮点数比大小

浮点数是有限精度的，在运算过程中，难免会产生误差，相信大家深有被卡精度的体会。但是在计算几何中，我们经常需要判断浮点数的大小。

浮点数比大小

浮点数是有限精度的，在运算过程中，难免会产生误差，相信大家深有被卡精度的体会。但是在计算几何中，我们经常需要判断浮点数的大小。这里我们引入如下的比较函数：

```
1  #define eps 1e-12
2  int dcmp(double x)
3  {
4      if(fabs(x)<=eps) return 0;
5      else if(x<0) return -1;
6      else return 1;
7  }
```

向量的基本运算

我们重载一些运算符来实现向量基本运算。

```
1  Vector operator + (Vector a, Vector b){return Vector(a.x+b.x, a.y+b.y);}
2  Vector operator - (Vector a, Vector b){return Vector(a.x-b.x, a.y-b.y);}
3  Vector operator - (Vector b){return Vector(-b.x, -b.y);}
4  Vector operator * (Vector a, double x){return Vector(a.x*x, a.y*x);}
5  Vector operator * (double x, Vector a){return Vector(a.x*x, a.y*x);}
6  double Angle(Vector a){return atan2(a.y, a.x);}
7
8  bool operator < (Point a, Point b){
9      return a.x < b.x || (a.x == b.x && a.y < b.y);
10 }
11 bool operator == (Point a, Point b){
12     return dcmp(a.x-b.x) == 0 && dcmp(a.y-b.y) == 0;
13 }
14
15 double Dot(Vector a, Vector b){return a.x*b.x + a.y*b.y;}
16 double Length(Vector a){return sqrt(a.x*a.x + a.y*a.y);}
```

向量的叉乘

二维向量叉乘写作 $\mathbf{a} \times \mathbf{b}$，定义如下：

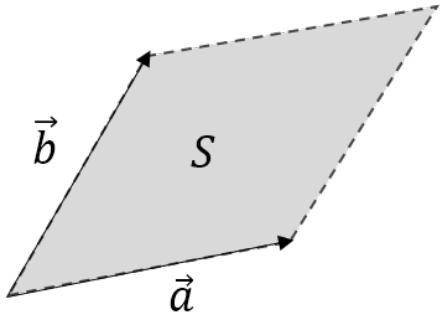
```
1    double Cross(Vector a,Vector b){return a.x*b.y - a.y*b.x;}
```

向量的叉乘

二维向量叉乘写作 $\mathbf{a} \times \mathbf{b}$，定义如下：

```
1    double Cross(Vector a,Vector b){return a.x*b.y - a.y*b.x;}
```

在几何中，叉乘是向量 a 与 b 构成的平行四边形的有向面积。
如果 b 在 a 的**逆时针**方向，结果就是**正**的；否则就是负的。



$\vec{a} \times \vec{b} = S$　　$\vec{b} \times \vec{a} = -S$

叉乘的应用：将凸多边形的顶点按逆时针排序

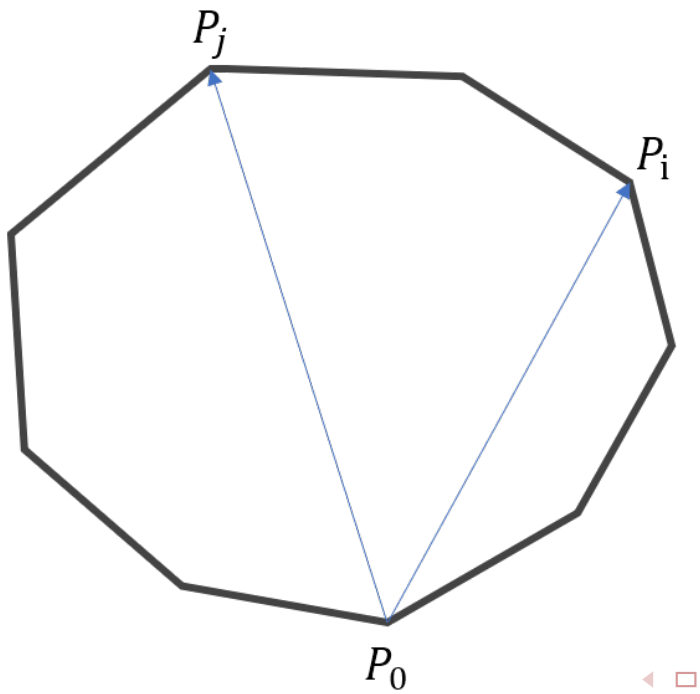
给定凸 n 边形的所有顶点，请将它们按逆时针排序，起点随意。
（提示：使用 `sort` 函数，考虑如何定义 `cmp`）

叉乘的应用：将凸多边形的顶点按逆时针排序

给定凸 n 边形的所有顶点，请将它们按逆时针排序，起点随意。
（提示：使用 `sort` 函数，考虑如何定义 `cmp`）

先随意固定一个起点 P_0，P_i 排在 P_j 前面，当且仅当

$$\overrightarrow{P_0P_i} \times \overrightarrow{P_0P_j} > 0.$$

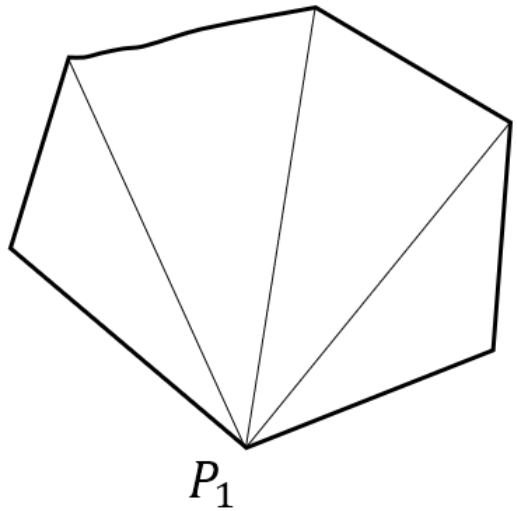


叉乘的应用：求凸多边形的面积

给定凸 n 边形的所有顶点，它们已经按逆时针排好了序，求图形的面积。

叉乘的应用：求凸多边形的面积

给定凸 n 边形的所有顶点，它们已经按逆时针排好了序，求图形的面积。



依次叉乘并累加即可。

```
1    double area = 0;
2    for(int i = 2; i <= n-1; i++)
3        area += 0.5 * cross(p[i]-p[1], p[i+1]-p[1]);
```

Thank You