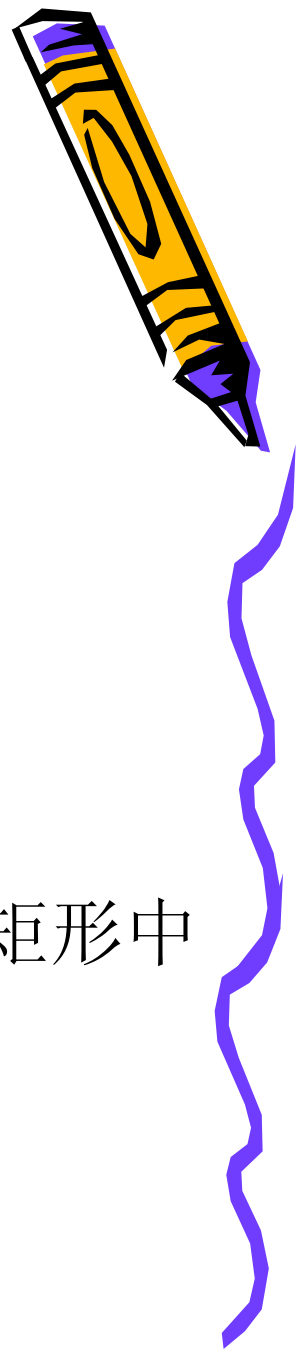


第二章 GIS算法的几何基础



- 2.1 维数扩展的9交集模型
- 2.2 矢量的概念
- 2.3 折线段的拐向判断
- 2.4 判断点是否在线段上
- 2.5 判断两线段是否相交
- 2.6 判断线段和直线是否相交
- 2.7 判断矩形是否包含点
- 2.8 判断线段、折线、多边形是否在矩形中
- 2.9 判断矩形是否在矩形中
- 2.10 判断圆是否在矩形中
- 2.11 判断点是否在多边形内
- 2.12 判断线段是否在多边形内



第二章 GIS算法的几何基础



2.13 判断折线是否在多边形内

2.14 判断多边形是否在多边形内

2.15 判断矩形是否在多边形内

2.16 判断圆是否在多边形内

2.17 判断点是否在圆内

2.18 判断线段、折线、矩形、多边形是否在圆内

2.19 判断圆是否在圆内

2.20 计算两条共线的线段的交点

2.21 计算线段或直线与线段的交点

2.22 求线段或直线与圆的交点



2.1 维数扩展的9交集模型(10-1)



模型介绍

设有现实世界中的两个简单实体 A 、 B , $B(A)$ 、 $B(B)$ 表示 A 、 B 的边界, $I(A)$ 、 $I(B)$ 表示 A 、 B 的内部, $E(A)$ 、 $E(B)$ 表示 A 、 B 外部。
Egenhofer(1993)构造出一个由边界、内部、外部的点集组成的9—交空间关系模型(9IM)如下:

$$\begin{bmatrix} I(A) \cap I(B) & I(A) \cap B(B) & I(A) \cap E(B) \\ B(A) \cap I(B) & B(A) \cap B(B) & B(A) \cap E(B) \\ E(A) \cap I(B) & E(A) \cap B(B) & E(A) \cap E(B) \end{bmatrix} \quad (1)$$

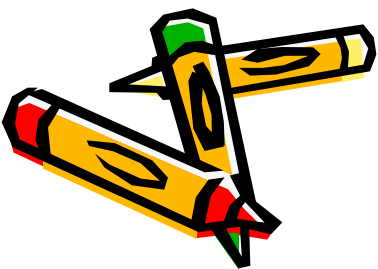


2.1 维数扩展的9交集模型(10-2)

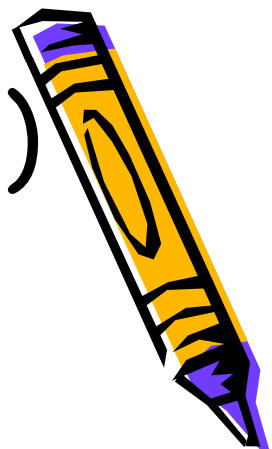


运用维数扩展法, 将9IM进行扩展, 利用点、线、面的边界、内部、余之间的交集的维数来作为空间关系描述的框架。对于几何实体的边界, 它是比其更低一维的几何实体的集合。为此, 点的边界为空集; 线的边界为线的两个端点, 当线为闭曲线时, 线的边界为空; 面的边界由构成面的所有线构成。若设 P 为一个点集, 定义点集的求维函数 DIM 如下:

$$DIM(P) = \begin{cases} -1 & P = \text{空} \\ 0 & P \text{ 不包含线、面, 但至少包含一个点} \\ 1 & P \text{ 不包含面, 但至少包含一条线} \\ 2 & P \text{ 至少包含一个面} \end{cases}$$



2.1 维数扩展的9交集模型(10-3)



利用维数扩展法, 式(1)可扩展为

$$\begin{bmatrix} \text{DIM}(I(A) \cap I(B)) & \text{DIM}(I(A) \cap B(B)) & \text{DIM}(I(A) \cap E(B)) \\ \text{DIM}(B(A) \cap I(B)) & \text{DIM}(B(A) \cap B(B)) & \text{DIM}(B(A) \cap E(B)) \\ \text{DIM}(E(A) \cap I(B)) & \text{DIM}(E(A) \cap B(B)) & \text{DIM}(E(A) \cap E(B)) \end{bmatrix} \quad (2)$$

根据DE-9IM, 对于点集拓扑空间 X , 当需要进行关系判别时, 可对矩阵的9元取值进行分析、比较。令 C 为各单元交的点集, 其取值 P 可能为 $\{T, F, *, 0, 1, 2\}$ 。各个取值的具体含义为:

- 1) $P=T$ $\text{DIM}(C) \in \{0, 1, 2\}$, 即交集 C 包含有点、线、面;
- 2) $P=F$ $\text{DIM}(C)=-1$, 即交集 C 为空;



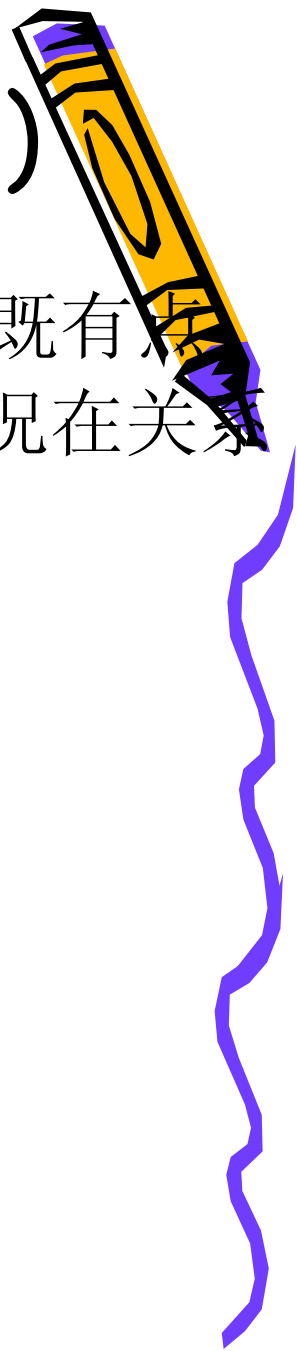
2.1 维数扩展的9交集模型(10-4)

3) $P=* \text{ DIM}(C) \in \{-1, 0, 1, 2\}$, 即两目标交集既有点、线、面, 又含有某些部分的交为空的情形, 该情况在关系判别时, 一般不予以考虑;

4) $P=0 \text{ DIM}(C)=0$;

5) $P=1 \text{ DIM}(C)=1$;

6) $P=2 \text{ DIM}(C)=2$ 。



2.1 维数扩展的9交集模型(10-5)

式(2)中各元素通过取值 $\{T, F, *, 0, 1, 2\}$, 可产生的情形为 $6^9=10077696$ 种, 关系非常复杂, 通过对大量的空间关系进行归纳和分类, 得出5种基本的空间关系: 相离关系(Disjoint)、相接关系(Touch)、相交关系(Cross)、真包含关系(Within)、叠置关系(Overlap), 并将这5种关系定义为空间关系的最小集, 其特征为:

- 1) 相互之间不能进行转化;
- 2) 能覆盖所有的空间关系模式;
- 3) 能应用于同维与不同维的几何目标;
- 4) 每一种关系对应于唯一的DE-9IM矩阵;
- 5) 任何其它的DE-9IM关系可以通过用这5种基本关系进行表达。

另外, 为了用户的使用方便, 还定义几个基本的空间关系即: 相等(Equal)、包含(Contain)、覆盖(Cover)、和被覆盖(CoveredBy)。



2.1 维数扩展的9交集模型(10-6)

在地理信息系统中,空间数据具有属性特征、空间特征和时间特征,基本数据类型包括属性数据、几何数据和空间关系数据。作为基本数据类型的空间关系数据主要指点/点、点/线、点/面、线/线、线/面、面/面之间的相互关系。

利用DE-9IM方法,识别规则为:

(1) 相离: $A. Disjoint(B)$ $A. Relate(B, "FF*FF****")$

(2) 相接: $A. Touch(B)$ $A. Relate(B, "FT*****")$

OR $A. Relate(B, "F**T*****")$

OR $A. Relate(B, "F***T*****")$ (如图)

(3) 相交: $A. Cross(B)$ $A. Relate(B, "P*T*****")$,

Case $A, B \in L, P=0, Else P=T$ (如图)



2.1 维数扩展的9交集模型(10-7)

(4) 叠置: $A. \text{Overlap}(B) \quad A. \text{Relate}(B, \text{"T*T***T**"}),$

Case $A, B \in L, P=0, \text{Else } P=T$ (如图)

(5) 真包含: $A. \text{Within}(B) \quad A. \text{Relate}(B, \text{"TF*F**F***"})$ (如图)

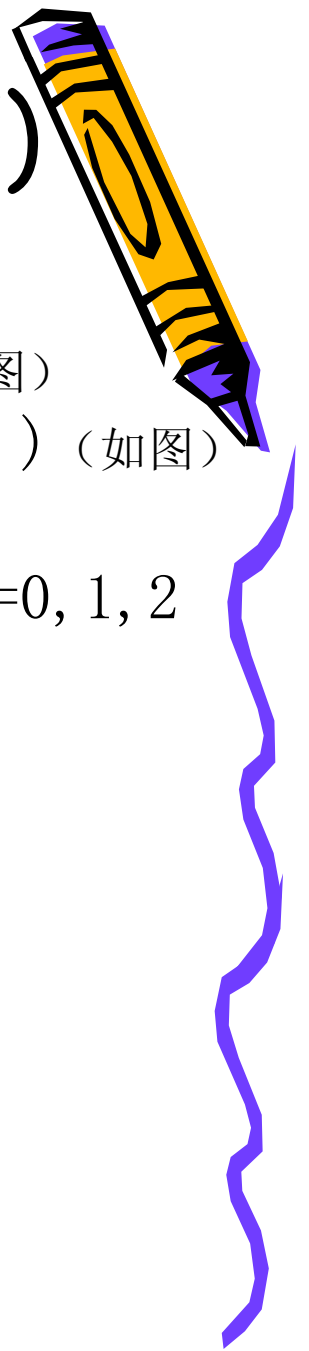
(6) 包含: $A. \text{Contain}(B) \quad B. \text{In}(A)$

(7) 相等: $A. \text{Equal}(B) \quad A. \text{Relate}(B, \text{"PF*FP*****"}), P=0, 1, 2$

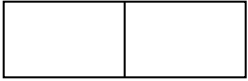
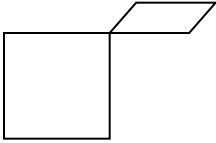
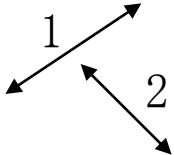
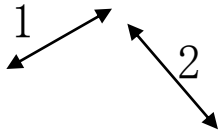
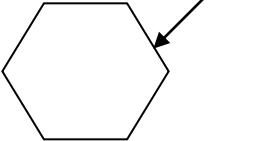
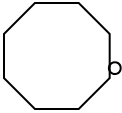
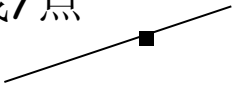
(8) 覆盖: $A. \text{Cover}(B) \quad (I(A) \cap I(B) = I(A) \text{ And}$

$(E(A) \cap E(B) \neq \Phi)$

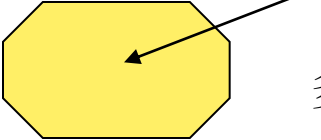
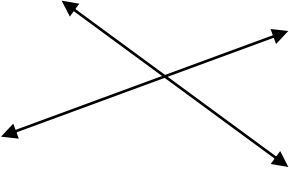
(9) 被覆盖: $A. \text{CoveredBy}(B) \quad B. \text{Cover}(A)$



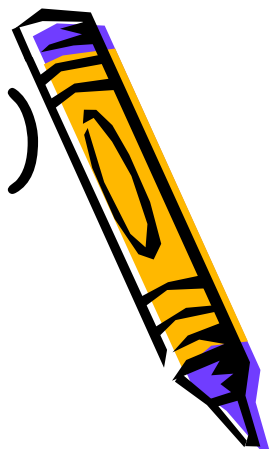
2.1 维数扩展的9交集模型(10-8)

<p>(a) </p> <p>多边形/多边形</p>	<p>(b) </p>	<p>多边形/线</p>
<p>(a) </p>	<p>(b) </p> <p>线/线</p>	<p></p>
<p>多边形/点</p>	<p></p>	<p>线/点</p> <p></p>

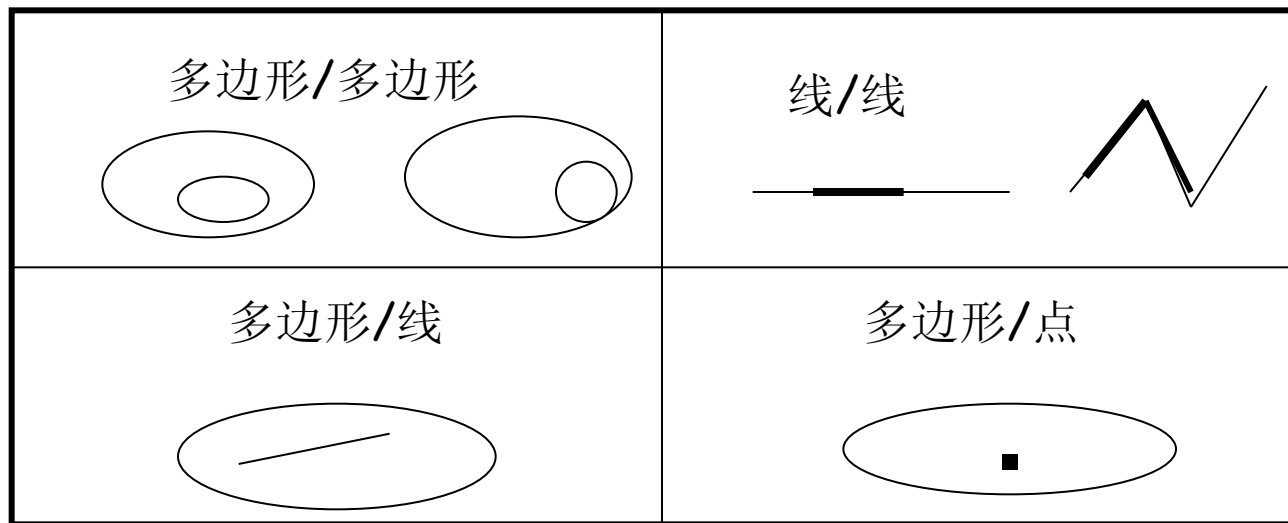
相接关系示例

<p></p> <p>多边形/线</p>	<p></p> <p>线/线</p>
---	---

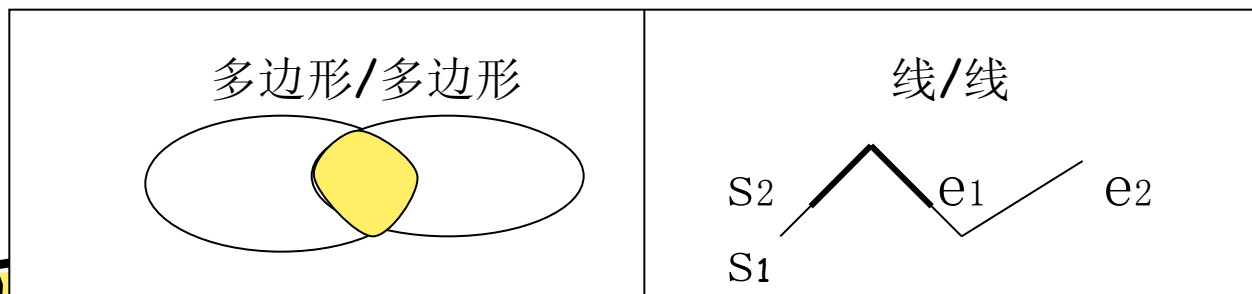
相交关系示例



2.1 维数扩展的9交集模型(10-9)



真包含关系示意图



叠置关系示例

2.1 维数扩展的9交集模型(10-10)

DE-9IM模版分析

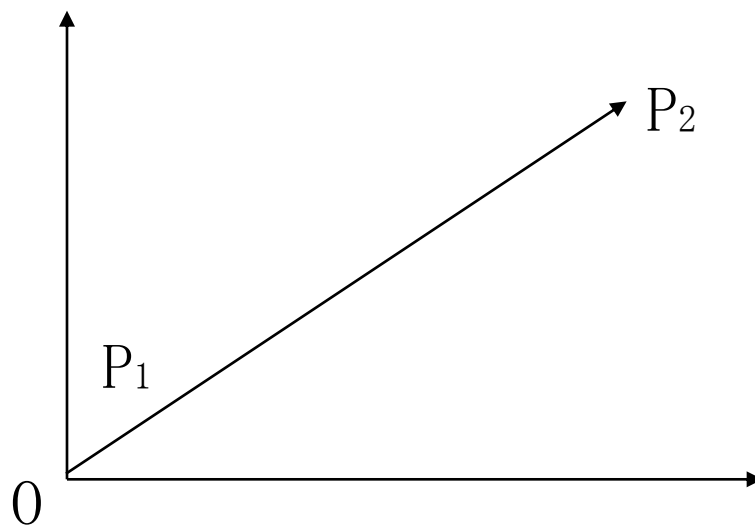
空间分析	几何对象	DE-9IM模版
相离 (Disjoint)	All	FF*FF****
相接 (Touches)	A/A	FT*****或 F**T*****或 F***T****
	L/L	
	L/A	
	P/A	
	P/L	
相交 (Crosses)	P/L	T*T*****
	P/A	
	L/L	0*****
	L/A	T*T*****
真包含 (Within)	All	T*F**F***
叠置 (Overlaps)	A/A	T*T***T**
	L/L	1*T***T**
	P/P	T*T***T**

2.2 矢量的概念(3-1)



一、矢量的概念

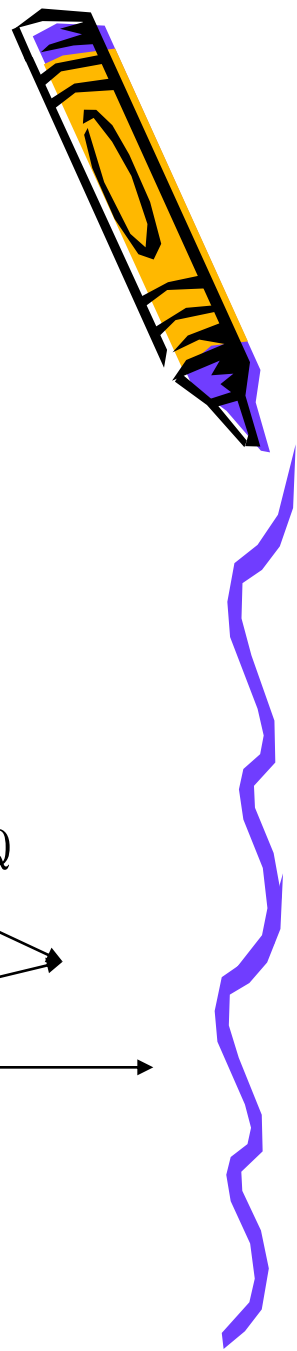
如果一条线段的端点是有次序之分的，我们把这种线段称为有向线段。如果有向线段 P_1P_2 的起点 P_1 在坐标原点，我们可以把它成为矢量 P_2 (如图)。



矢量的概念

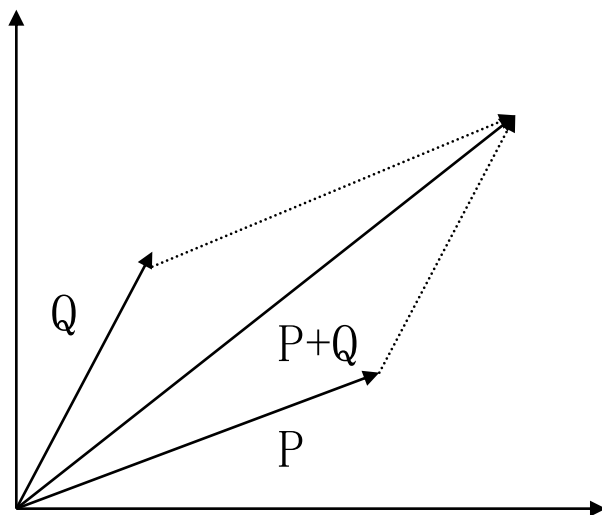


2.2 矢量的概念(3-2)

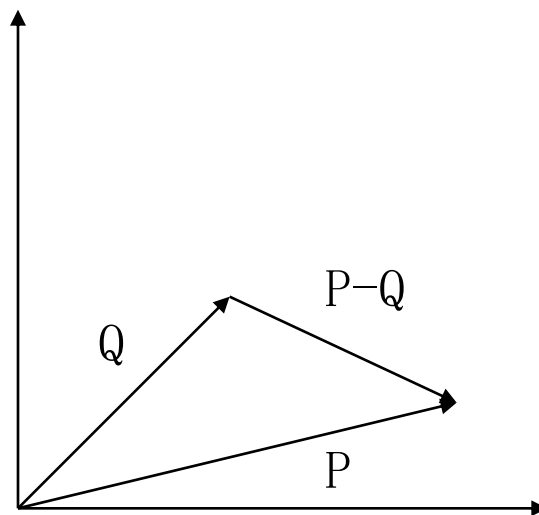


二、矢量加减法

设二维矢量 $P = (x_1, y_1)$ ， $Q = (x_2, y_2)$ ，则：



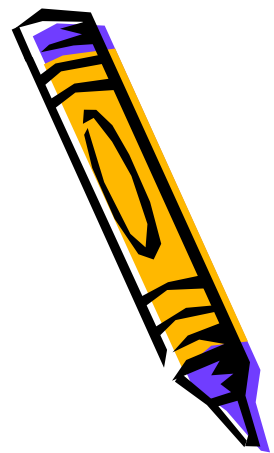
矢量加法



矢量减法



2.2 矢量的概念(3-3)



三、矢量叉积

设二维矢量 $P = (x_1, y_1)$ ， $Q = (x_2, y_2)$ ，则矢量叉积定义为：

由 $(0, 0)$ 、 P 、 Q 和 PQ 所组成的平行四边形的带符号的面积，即：

$P \times Q = x_1 \cdot y_2 - x_2 \cdot y_1$ 其结果是一个标量。显然有性质：

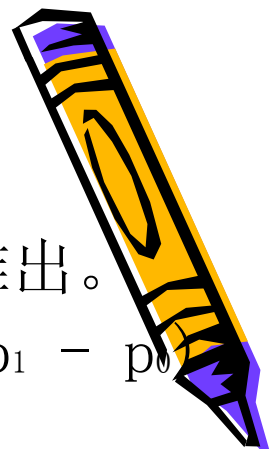
$$P \times Q = -(Q \times P) \text{ 和 } P \times (-Q) = -(P \times Q)$$

叉积的一个非常重要的性质是可以通过它的符号判断两矢量相互之间的顺逆时针关系：

- (1) 若 $P \times Q > 0$ ，则 P 在 Q 的顺时针方向；
- (2) 若 $P \times Q < 0$ ，则 P 在 Q 的逆时针方向；
- (3) 若 $P \times Q = 0$ ，则 P 与 Q 共线，但可能同向也可能反向。



2.3 折线段的拐向判断



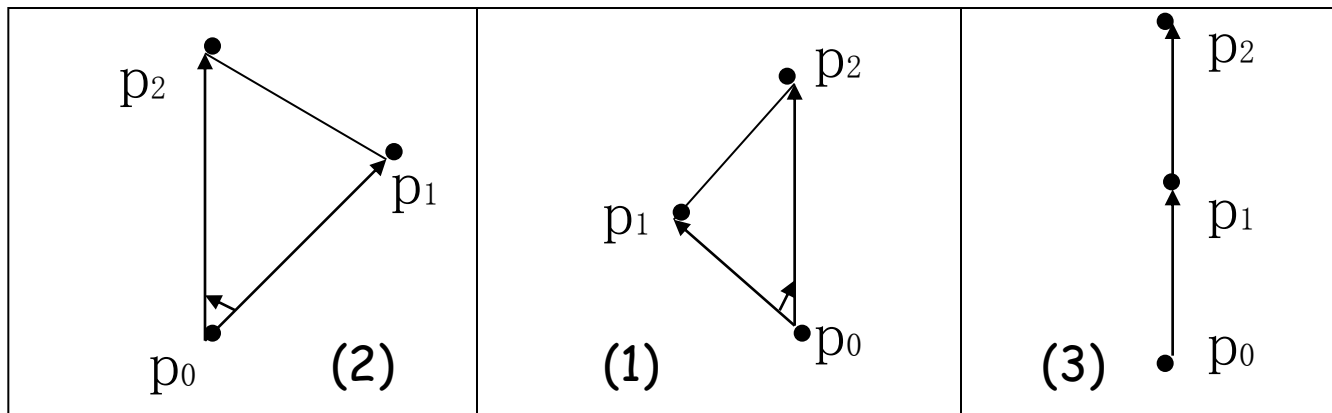
折线段的拐向判断方法可以直接由矢量叉积的性质推出。对于有公共端点的线段 p_0p_1 和 p_1p_2 ，通过计算 $(p_2 - p_0) \times (p_1 - p_0)$ 的符号便可以确定折线段的拐向：

(1) 若 $(p_2 - p_0) \times (p_1 - p_0) > 0$ ，则 p_0p_1 在 p_1 点拐向右侧后得到 p_1p_2 。

(2) 若 $(p_2 - p_0) \times (p_1 - p_0) < 0$ ，则 p_0p_1 在 p_1 点拐向左侧后得到 p_1p_2 。

(3) 若 $(p_2 - p_0) \times (p_1 - p_0) = 0$ ，则 p_0 、 p_1 、 p_2 三点共线。

具体情况可参考下图：



2.4判断点是否在线段上



设点为 Q ，线段为 P_1P_2 ，判断点 Q 在该线段上的依据是：
 $(Q - P_1) \times (P_2 - P_1) = 0$ 且 Q 在以 P_1, P_2 为对角顶点的矩形内。前者保证 Q 点在直线 P_1P_2 上，后者是保证 Q 点不在线段 P_1P_2 的延长线或反向延长线上，对于这一步骤的判断可以用以下过程实现：

ON-SEGMENT (p_i, p_j, p_k)

```
if  $\min(x_i, x_j) \leq x_k \leq \max(x_i, x_j)$  and  $\min(y_i, y_j) \leq y_k \leq \max(y_i, y_j)$   
  then return true;  
  else return false;
```

特别要注意的是，由于需要考虑水平线段和垂直线段两种特殊情况， $\min(x_i, x_j) \leq x_k \leq \max(x_i, x_j)$ 和 $\min(y_i, y_j) \leq y_k \leq \max(y_i, y_j)$ 两个条件必须同时满足才能返回真值。



2.5 判断两线段是否相交 (3-1)



我们分两步确定两条线段是否相交：

(1) 快速排斥试验

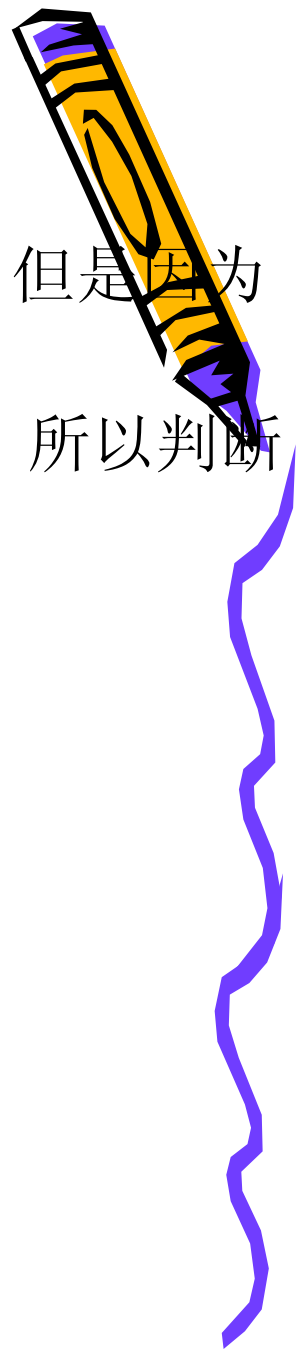
设以线段 P_1P_2 为对角线的矩形为 R ， 设以线段 Q_1Q_2 为对角线的矩形为 T ， 如果 R 和 T 不相交， 显然两线段不会相交。

(2) 跨立试验

如果两线段相交， 则两线段必然相互跨立对方。 若 P_1P_2 跨立 Q_1Q_2 ， 则矢量 $(P_1 - Q_1)$ 和 $(P_2 - Q_1)$ 位于矢量 $(Q_2 - Q_1)$ 的两侧， 即 $(P_1 - Q_1) \times (Q_2 - Q_1) * (P_2 - Q_1) \times (Q_2 - Q_1) < 0$ 。 上式可改写成 $(P_1 - Q_1) \times (Q_2 - Q_1) * (Q_2 - Q_1) \times (P_2 - Q_1) > 0$ 。



2.5 判断两线段是否相交(3-2)



- (1) 当 $(P_1 - Q_1) \times (Q_2 - Q_1) = 0$ 时, 说明 $(P_1 - Q_1)$ 和 $(Q_2 - Q_1)$ 共线, 但是因为已经通过快速排斥试验, 所以 P_1 一定在线段 Q_1Q_2 上;
- (2) 当 $(Q_2 - Q_1) \times (P_2 - Q_1) = 0$ 时, 说明 P_2 一定在线段 Q_1Q_2 上。所以判断 P_1P_2 跨立 Q_1Q_2 的依据是:

$$(P_1 - Q_1) \times (Q_2 - Q_1) * (Q_2 - Q_1) \times (P_2 - Q_1) \geq 0$$

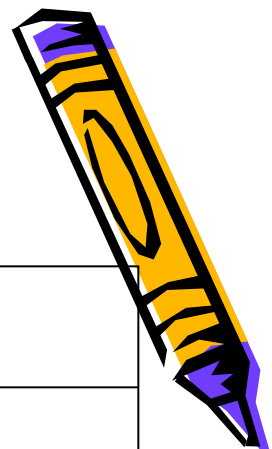
- (3) 同理判断 Q_1Q_2 跨立 P_1P_2 的依据是:

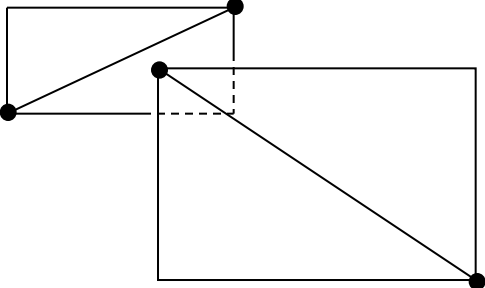
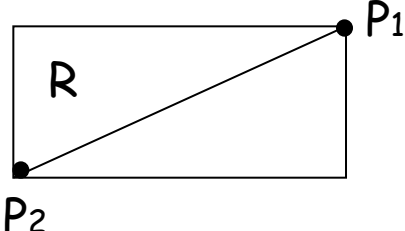
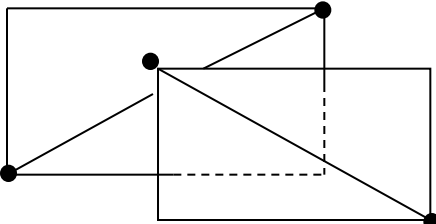
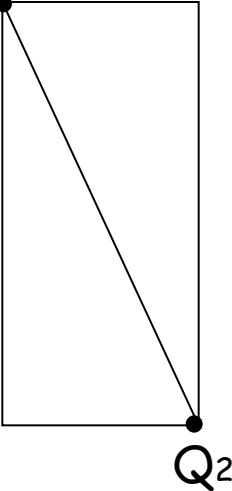
$$(Q_1 - P_1) \times (P_2 - P_1) * (P_2 - P_1) \times (Q_2 - P_1) \geq 0。$$

具体情况如下图所示:



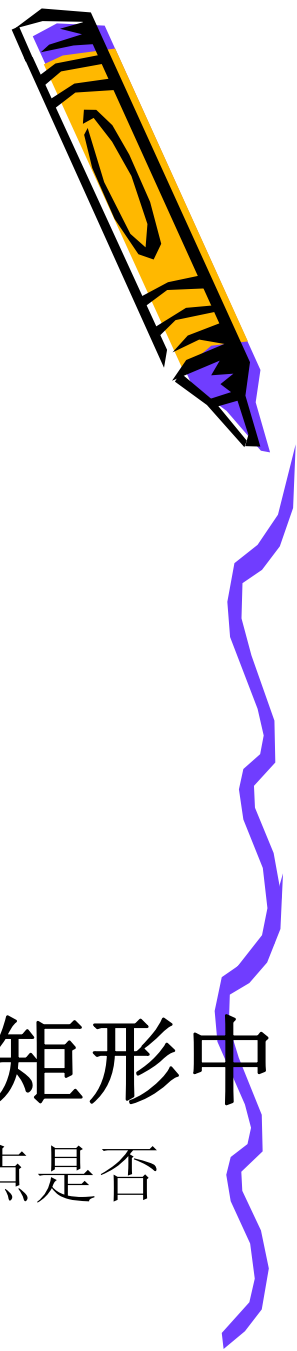
2.5 判断两线段是否相交(3-3)



	通过快速排斥试验	未通过快速排斥试验
未通过快速排斥试验		
通过快速排斥试验		



2.6 判断线段和直线是否相交



有了上面的基础，这个算法就很容易了。如果线段 P_1P_2 和直线 Q_1Q_2 相交，则 P_1P_2 跨立 Q_1Q_2 ，即：

$$(P_1 - Q_1) \times (Q_2 - Q_1) * (Q_2 - Q_1) \times (P_2 - Q_1) \geq 0。$$

2.7 判断矩形是否包含点

只要判断该点的横坐标和纵坐标是否夹在矩形的左右边和上下边之间。

2.8 判断线段、折线、多边形是否在矩形中

因为矩形是个凸集，所以只要判断所有端点是否都在矩形中就可以了。



2.9 判断矩形是否在矩形中

只要比较左右边界和上下边界就可以了。

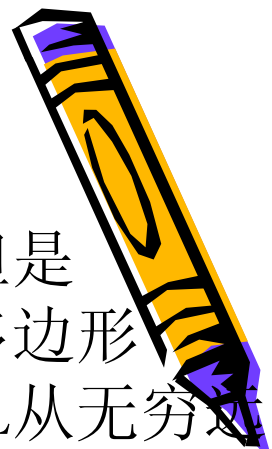


2.10 判断圆是否在矩形中

很容易证明，圆在矩形中的充要条件是：圆心在矩形中且圆的半径小于等于圆心到矩形四边的距离的最小值。



2.11 判断点是否在多边形内

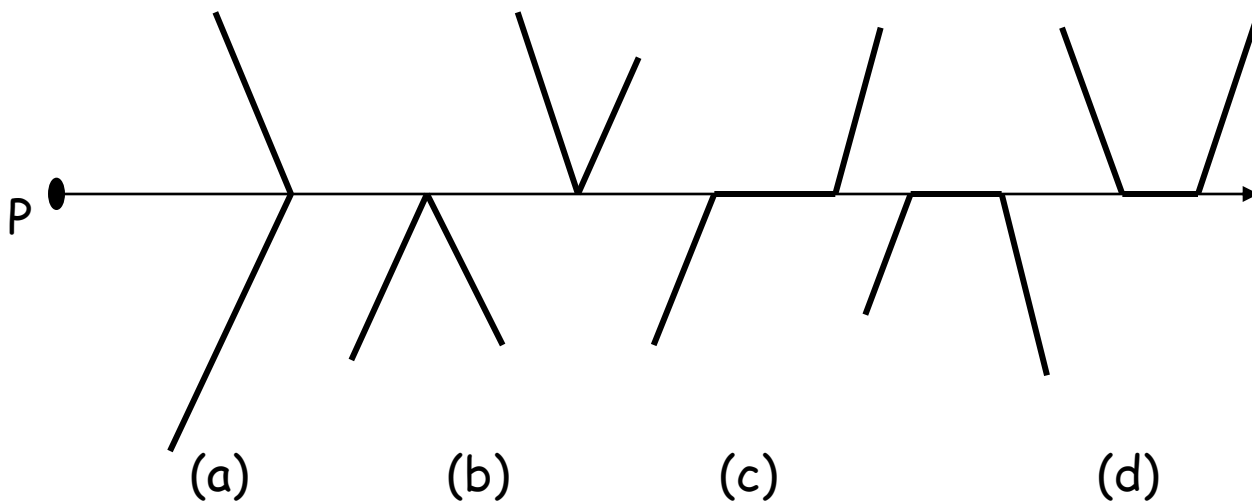


判断点P是否在多边形中是计算几何中一个非常基本但是十分重要的算法。以点P为端点，向左方作射线L，由于多边形是有界的，所以射线L的左端一定在多边形外，考虑沿着L从无穷远处开始自左向右移动，遇到和多边形的第一个交点的时候，进入到了多边形的内部，遇到第二个交点的时候，离开了多边形，……所以很容易看出当L和多边形的交点数目C是奇数的时候，P在多边形内，是偶数的话P在多边形外。

但是有些特殊情况要加以考虑。如图下图(a) (b) (c) (d)所示。在图(a)中，L和多边形的顶点相交，这时候交点只能计算一个；在图(b)中，L和多边形顶点的交点不应被计算；在图(c)和(d)中，L和多边形的一条边重合，这条边应该被忽略不计。如果L和多边形的一条边重合，这条边应该被忽略不计。




2.11 判断点是否在多边形内



为了统一起见，我们在计算射线 L 和多边形的交点的时候，
1。对于多边形的水平边不作考虑；2。对于多边形的顶点和 L 相交的情况，如果该顶点是其所属的边上纵坐标较大的顶点，则计数，否则忽略；3。对于 P 在多边形边上的情形，直接可判断 P 属于多边形。由此得出算法的伪代码如下：






```
count ← 0;
以P为端点，作从右向左的射线L;
for 多边形的每条边s
do if P在边s上
    then return true;
    if s不是水平的
    then if s的一个端点在L上
        if 该端点是s两端点中纵坐标较大的端点
        then count ← count+1
        else if s和L相交
        then count ← count+1;
if count mod 2 = 1
    then return true;
else return false;
```

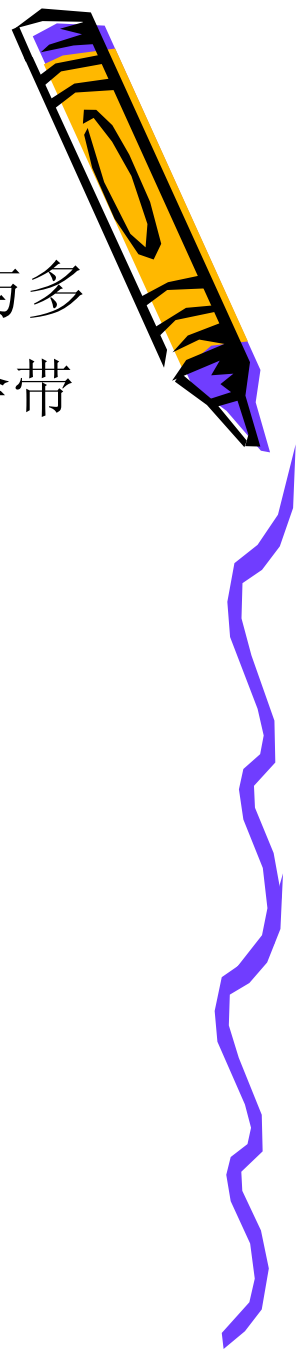
其中做射线L的方法是：设P'的纵坐标和P相同，横坐标为正无穷大（很大的一个正数），则P和P'就确定了射线L。

判断点是否在多边形中的这个算法的时间复杂度为 $O(n)$ 。



2.11 判断点是否在多边形内

另外还有一种算法是用带符号的三角形面积之和与多边形面积进行比较，这种算法由于使用浮点数运算所以会带来一定误差，不推荐大家使用。

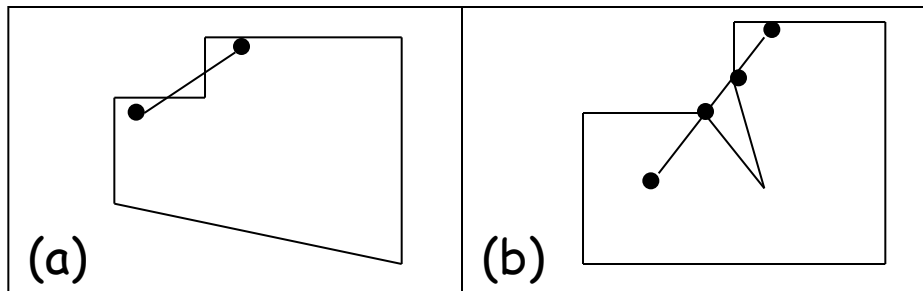


2.12 判断线段是否在多边形内(5-1)



线段在多边形内的一个必要条件是线段的两个端点都在多边形内，但由于多边形可能为凹，所以这不能成为判断的充分条件。如果线段和多边形的某条边内交（两线段内交是指两线段相交且交点不在两线段的端点），因为多边形的边的左右两侧分属多边形内外不同部分，所以线段一定会有一部分在多边形外(见图a)。于是我们得到线段在多边形内的第二个必要条件：线段和多边形的所有边都不内交。

线段和多边形交于线段的两个端点并不会影响线段是否在多边形内；但是如果多边形的某个顶点和线段相交，还必须判断两相邻交点之间的线段是否包含于多边形内部（反例见图b）。



2.12 判断线段是否在多边形内(5-2)



因此我们可以先求出所有和线段相交的多边形的顶点，然后按照X-Y坐标排序(X坐标小的排在前面，对于X坐标相同的点，Y坐标小的排在前面，这种排序准则也是为了保证水平和垂直情况的判断正确)，这样相邻的两个点就是在线段上相邻的两交点，如果任意相邻两点的中点也在多边形内，则该线段一定在多边形内。



2.12 判断线段是否在多边形内(5-3)



证明如下：

命题1：

如果线段和多边形的两相邻交点 P_1 ， P_2 的中点 P' 也在多边形内，则 P_1 ， P_2 之间的所有点都在多边形内。

证明：

假设 P_1 ， P_2 之间含有不在多边形内的点，不妨设该点为 Q ，在 P_1 ， P' 之间，因为多边形是闭合曲线，所以其内外部之间有界，而 P_1 属于多边形内部， Q 属于多边形外部， P' 属于多边形内部， P_1 - Q - P' 完全连续，所以 P_1Q 和 QP' 一定跨越多边形的边界，因此在 P_1 ， P' 之间至少还有两个该线段和多边形的交点，这和 P_1P_2 是相邻两交点矛盾，故命题成立。证毕。



2.12 判断线段是否在多边形内(5-4)



由命题1直接可得出推论：

推论2：

设多边形和线段PQ的交点依次为 P_1, P_2, \dots, P_n ，其中 P_i 和 P_{i+1} 是相邻两交点，线段PQ在多边形内的充要条件是： P, Q 在多边形内且对于 $i = 1, 2, \dots, n-1$ ， P_i, P_{i+1} 的中点也在多边形内。

在实际编程中，没有必要计算所有的交点，首先应判断线段和多边形的边是否内交，倘若线段和多边形的某条边内交则线段一定在多边形外；如果线段和多边形的每一条边都不内交，则线段和多边形的交点一定是线段的端点或者多边形的顶点，只要判断点是否在线段上就可以了。

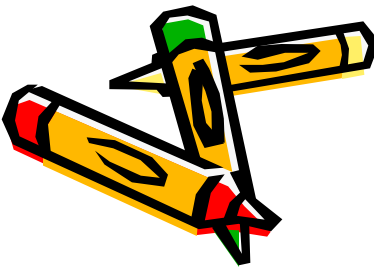


(5-5)

至此我们得出算法如下：

```
if 线端PQ的端点不都在多边形内
  then return false;
else
  点集pointSet初始化为空;
  for 多边形的每条边s
    do if 线段的某个端点在s上
        then 将该端点加入pointSet;
        else if s的某个端点在线段PQ上
            then 将该端点加入pointSet;
            else if s和线段PQ相交 /*这时候已经可以肯定是内交了*/
                then return false;
        else
          将pointSet中的点按照X-Y坐标排序;
          for pointSet中每两个相邻点pointSet[i],
                                pointSet[i+1]
            do if pointSet[i] , pointSet[ i+1] 的
                中点不在多边形中
                then return false;
                else return true
```

这个过程中的排序因为交点数目肯定远小于多边形的顶点数目 n ，所以最多是常数级的复杂度，几乎可以忽略不计。因此算法的时间复杂度也是 $O(n)$ 。



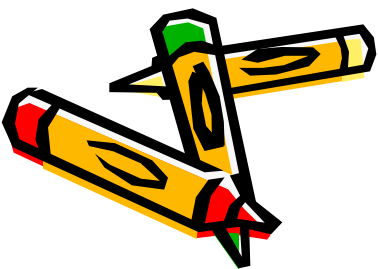
2.13 判断折线是否在多边形内



只要判断折线的每条线段是否都在多边形内即可。设折线有 m 条线段，多边形有 n 个顶点，则该算法的时间复杂度为 $O(m \times n)$ 。

2.14 判断多边形是否在多边形内

只要判断多边形的每条边是否都在多边形内即可。判断一个有 m 个顶点的多边形是否在一个有 n 个顶点的多边形内复杂度为 $O(m \times n)$ 。



2.15 判断矩形是否在多边形内

将矩形转化为多边形，然后再判断是否在多边形内。

2.16 判断圆是否在多边形内

只要计算圆心到多边形的每条边的最短距离，如果该距离大于等于圆半径则该圆在多边形内。计算圆心到多边形每条边最短距离的算法在后文阐述。

2.17 判断点是否在圆内

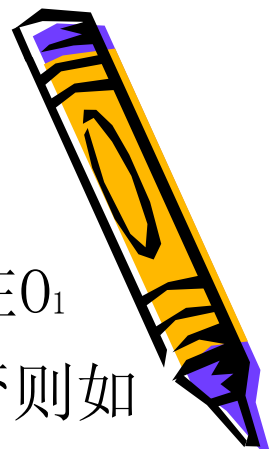
计算圆心到该点的距离，如果小于等于半径则该点在圆内。

2.18 判断线段、折线、矩形、多边形 是否在圆内

因为圆是凸集，所以只要判断是否每个顶点都在圆内即可。



2.19 判断圆是否在圆内



设两圆为 O_1, O_2 ，半径分别为 r_1, r_2 ，要判断 O_2 是否在 O_1 内。先比较 r_1, r_2 的大小，如果 $r_1 < r_2$ 则 O_2 不可能在 O_1 内；否则如果两圆心的距离大于 $r_1 - r_2$ ，则 O_2 不在 O_1 内；否则 O_2 在 O_1 内。



2.20 计算两条共线的线段的交点(2-1)



对于两条共线的线段，它们之间的位置关系有下图所示的几种情况。图(a)中两条线段没有交点；图(b)和(d)中两条线段有无穷焦点；图(c)中两条线段有一个交点。

设 $line1$ 是两条线段中较长的一条， $line2$ 是较短的一条。

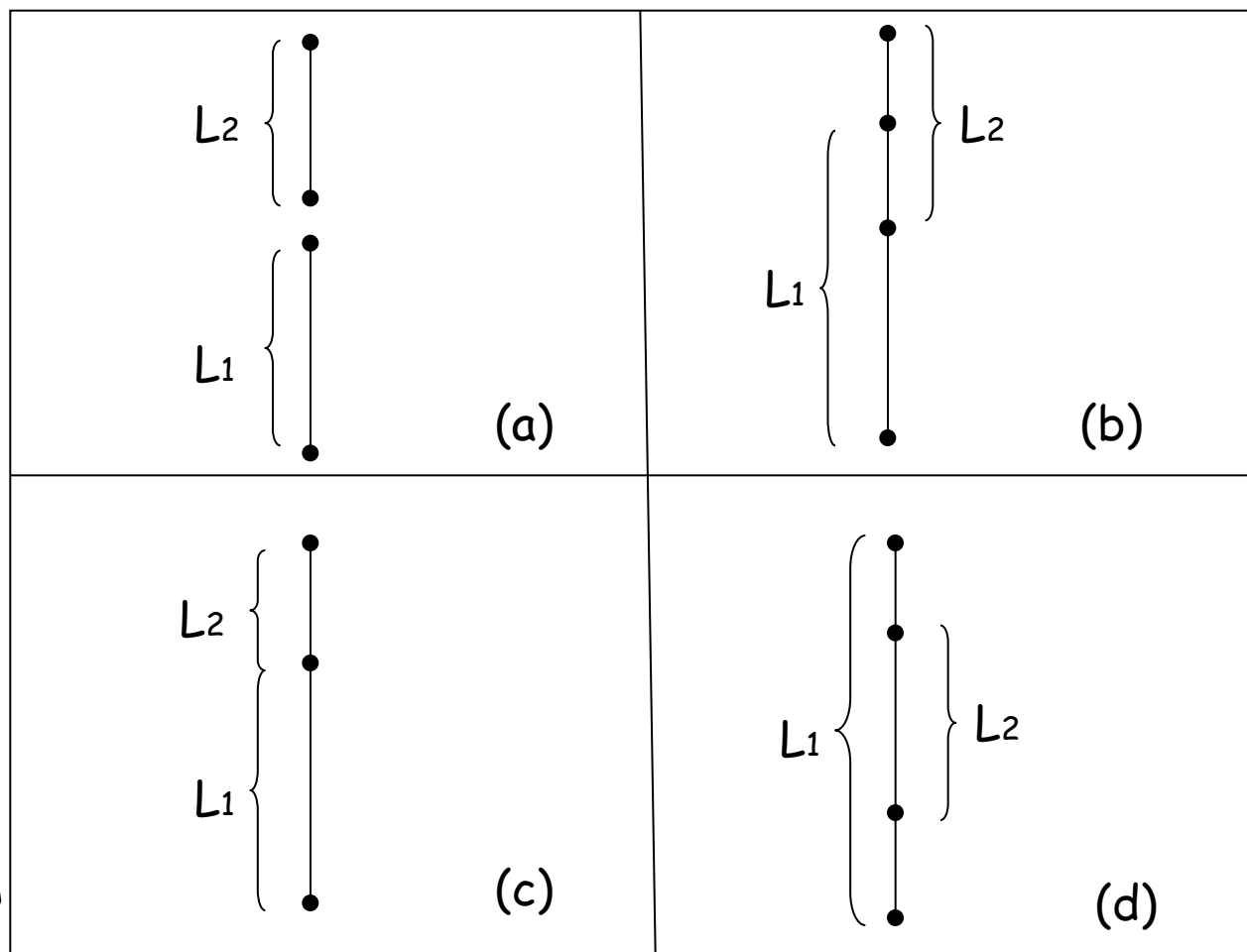
(1) 如果 $line1$ 包含了 $line2$ 的两个端点，则是图(d)的情况，两线段有无穷交点；

(2) 如果 $line1$ 只包含 $line2$ 的一个端点，那么如果 $line1$ 的某个端点等于被 $line1$ 包含的 $line2$ 的那个端点，则是图(c)的情况，这时两线段只有一个交点，否则就是图(b)的情况，两线段也是有无穷的点；

(3) 如果 $line1$ 不包含 $line2$ 的任何端点，则是图(a)的情况，这时两线段没有交点。



2.20 计算两条共线的线段的交点(2-2)



共线线段的位置关系

2.21 计算线段或直线与线段的交点(4-1)



设一条线段为 $L_0=P_1P_2$ ，另一条线段或直线为 $L_1=Q_1Q_2$ ，要计算的就是 L_0 和 L_1 的交点。步骤如下：

1. 首先判断 L_0 和 L_1 是否相交（方法已在前文讨论过），如果不相交则没有交点，否则说明 L_0 和 L_1 一定有交点，下面就将 L_0 和 L_1 都看作直线来考虑。

2. 如果 P_1 和 P_2 横坐标相同，即 L_0 平行于Y轴

a) 若 L_1 也平行于Y轴，

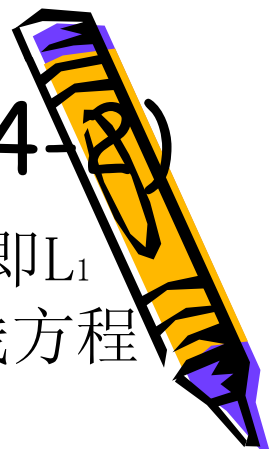
i. 若 P_1 的纵坐标和 Q_1 的纵坐标相同，说明 L_0 和 L_1 共线，假如 L_1 是直线的话他们有无穷的点，假如 L 是线段的话可用“计算两条共线线段的交点”的算法求他们的交点（该方法在前文已讨论过）；

ii. 否则说明 L_0 和 L_1 平行，他们没有交点；

b) 若 L_1 不平行于Y轴，则交点横坐标为 P_1 的横坐标，代入到 L_1 的直线方程中可以计算出交点纵坐标；



2.21 计算线段或直线与线段的交点(4-2)



3. 如果 P_1 和 P_2 横坐标不同，但是 Q_1 和 Q_2 横坐标相同，即 L_1 平行于Y轴，则交点横坐标为 Q_1 的横坐标，代入到 L_0 的直线方程中可以计算出交点纵坐标；

4. 如果 P_1 和 P_2 纵坐标相同，即 L_0 平行于X轴

a) 若 L_1 也平行于X轴，

i. 若 P_1 的横坐标和 Q_1 的横坐标相同，说明 L_0 和 L_1 共线，假如 L_1 是直线的话他们有无穷的点，假如 L_1 是线段的话可用“计算两条共线线段的交点”的算法求他们的交点（该方法在前文已讨论过）；

ii. 否则说明 L_0 和 L_1 平行，他们没有交点；

b) 若 L_1 不平行于X轴，则交点纵坐标为 P_1 的纵坐标，代入到 L_1 的直线方程中可以计算出交点横坐标；

5. 如果 P_1 和 P_2 纵坐标不同，但是 Q_1 和 Q_2 纵坐标相同，即 L_1 平行于X轴，则交点纵坐标为 Q_1 的纵坐标，代入到 L_0 的直线方程中可以计算出交点横坐标；



2.21 计算线段或直线与线段的交点(4-3)



6. 剩下的情况就是 L_1 和 L_0 的斜率均存在且不为0

的情况

a) 计算出 L_0 的斜率 K_0 , L_1 的斜率 K_1 ;

b) 如果 $K_1=K_2$

i. 如果 Q_1 在 L_0 上, 则说明 L_0 和 L_1 共线, 假如 L_1 是直线的话有无穷交点, 假如 L_1 是线段的话可用“计算两条共线线段的交点”的算法求他们的交点(该方法在前文已讨论过);

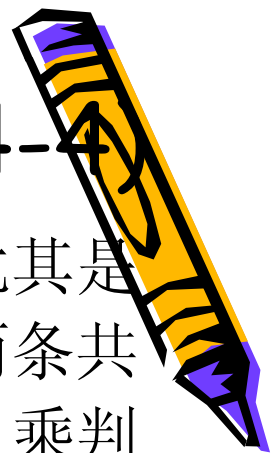
ii. 如果 Q_1 不在 L_0 上, 则说明 L_0 和 L_1 平行, 他们没有交点。

c) 联立两直线的方程组可以解出交点来



2.21 计算线段或直线与线段的交点(4-4)

这个算法并不复杂，但是要分情况讨论清楚，尤其是当两条线段共线的情况需要单独考虑，所以在前文将求两条共线线段的算法单独写出来。另外，一开始就先利用矢量叉乘判断线段与线段（或直线）是否相交，如果结果是相交，那么在后面就可以将线段全部看作直线来考虑。需要注意的是，我们可以将直线或线段方程改写为 $ax+by+c=0$ 的形式，这样一来上述过程的部分步骤可以合并，缩短了代码长度，但是由于先要求出参数，这种算法将花费更多的时间。



2.22 求线段或直线与圆的交点

设圆心为 O ，圆半径为 r ，直线(或线段) L 上的两点为 P_1, P_2 。

1. 如果 L 是线段且 P_1, P_2 都包含在圆 O 内，则没有交点；否则进行下一步。
2. 如果 L 平行于 Y 轴，
 - a) 计算圆心到 L 的距离 dis ;
 - b) 如果 $dis > r$ 则 L 和圆没有交点;
 - c) 利用勾股定理，可以求出两交点坐标，但要注意考虑 L 和圆的相切情况。
3. 如果 L 平行于 X 轴，做法与 L 平行于 Y 轴的情况类似;
4. 如果 L 既不平行 X 轴也不平行 Y 轴，可以求出 L 的斜率 K ，然后列出 L 的点斜式方程，和圆方程联立即可求解出 L 和圆的两个交点;
5. 如果 L 是线段，对于2, 3, 4中求出的交点还要分别判断是否属于该线段的范围内。

