计算机图形学

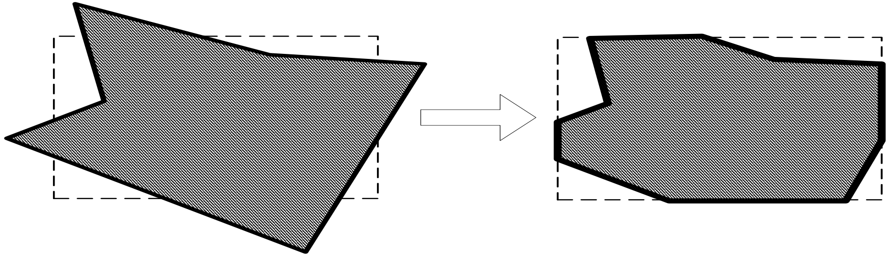
第二章：光栅图形学算法

光栅图形学算法的研究内容

- 直线段的扫描转换算法
- 多边形的扫描转换与区域填充算法
- 直线裁剪算法
- 反走样算法
- 消隐算法

一、裁剪

使用计算机处理图形信息时，计算机内部存储的图形往往比较大，而屏幕显示的只是图形的一部分。



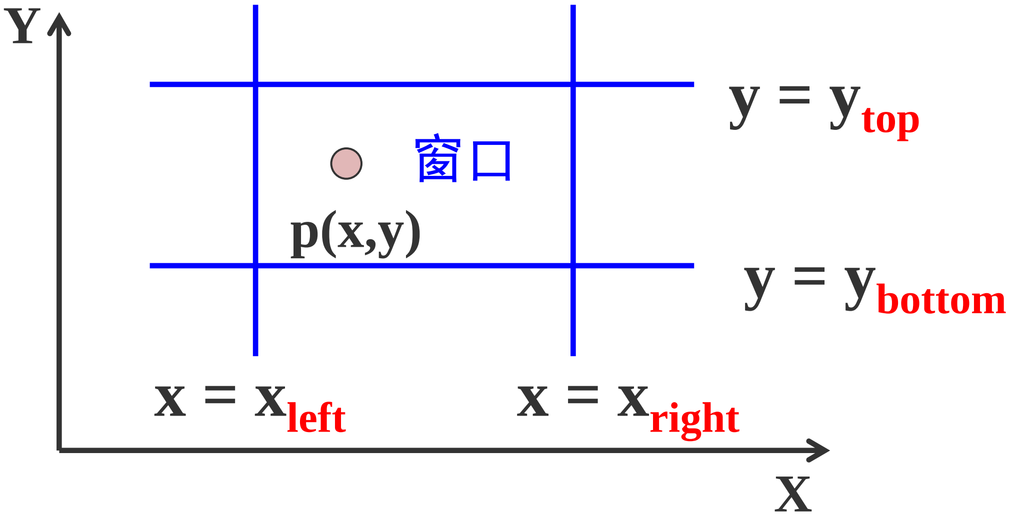
因此需要确定图形哪些部分落在显示区之内，哪些落在显示区之外。这个选择的过程就称为**裁剪**。

最简单的裁剪方法是把各种图形扫描转换为点之后，再判断点是否在窗口内。

1、点的裁剪

对于任意一点P（x，y），若满足下列两对不等式：

$$\begin{cases} x_{left} \le x \le x_{right} \\ y_{bottom} \le y \le y_{top} \end{cases}$$



则点P在矩形窗口内；否则，点P在矩形窗口之外

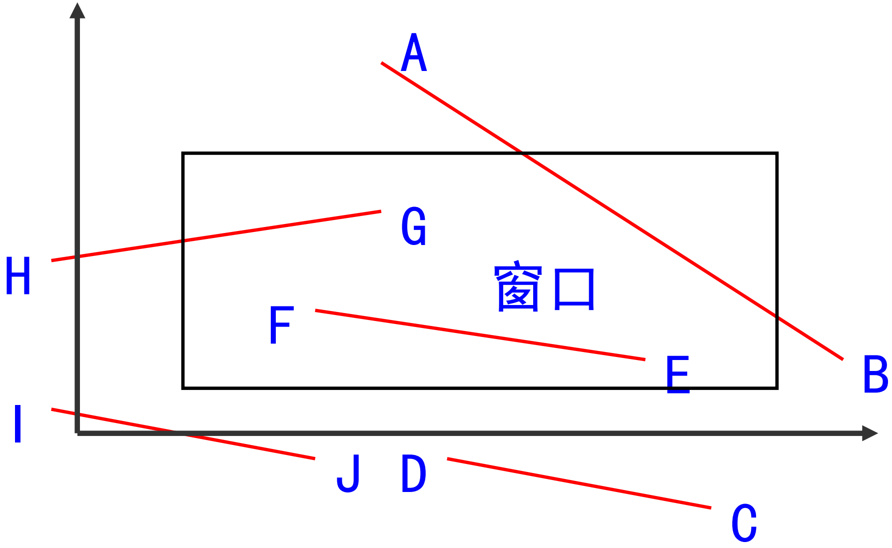
判断图形中每个点是否在窗口内，太费时，一般不可取

2、直线段的裁剪

直线段裁剪算法复杂图形裁剪的基础

直线段和剪裁窗口的可能关系：

- 完全落在窗口内
- 完全落在窗口外
- 与窗口边界相交

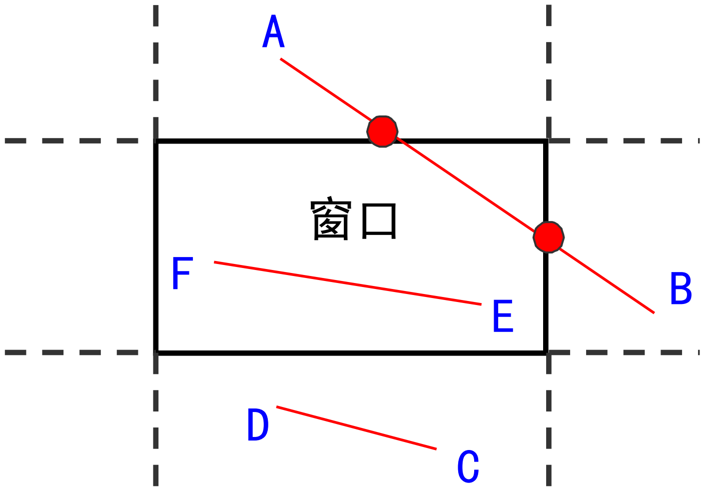


要裁剪一条直线段，首先要判断：

（1）它是否完全落在裁剪窗口内？

（2）它是否完全在窗口外？

（3）如果不满足以上两个条件，则计算它与一个或多个裁剪边界的交点

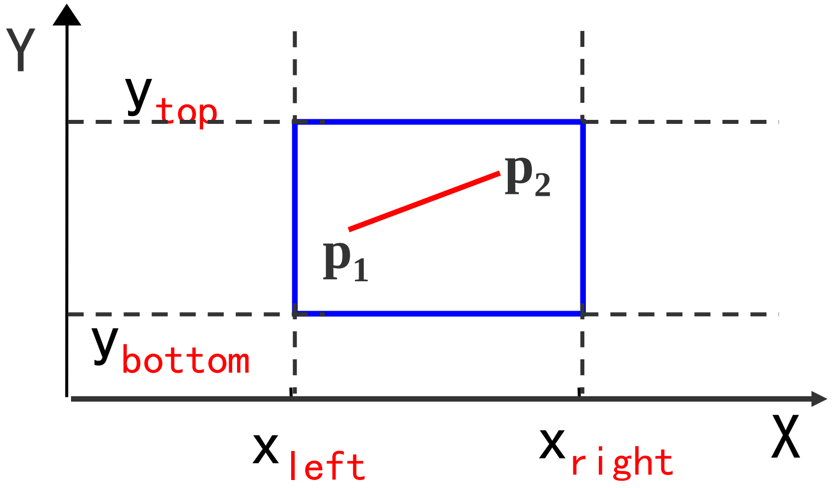


常用的裁剪算法有三种，即Cohen-Sutherland、中点分割法和Liang-Barsky裁剪算法

1、Cohen-Sutherland算法

本算法又称为编码裁剪算法，算法的基本思想是对每条直线段分三种情况处理：

（1）若点p_1和p_2完全在裁剪窗口内



“简取”之

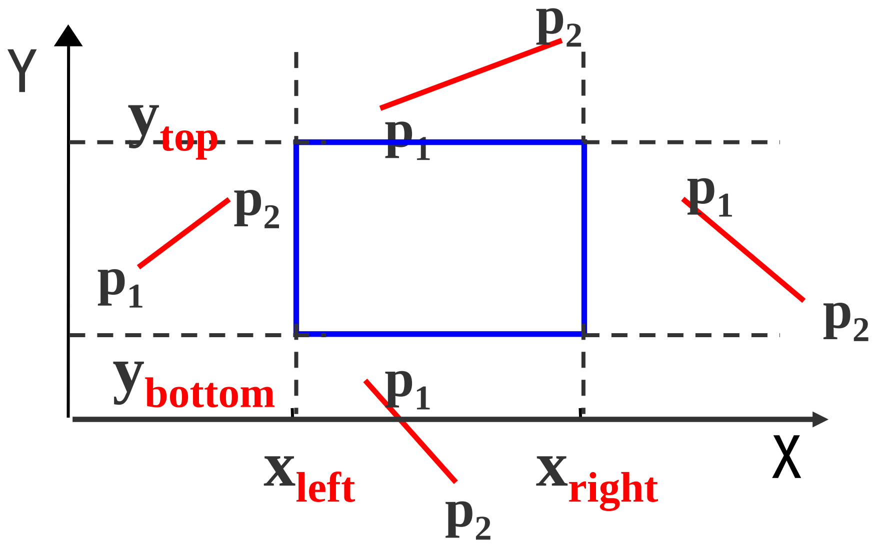
(2) 若点$p_1(x_1, y_1)$和$p_2(x_2, y_2)$均在窗口外，且满足下列四个条件之一：

$x_1 < x_{left}$ 且 $x_2 < x_{left}$

$x_1 > x_{right}$ 且 $x_2 > x_{right}$

$y_1 < y_{bottom}$ 且 $y_2 < y_{bottom}$

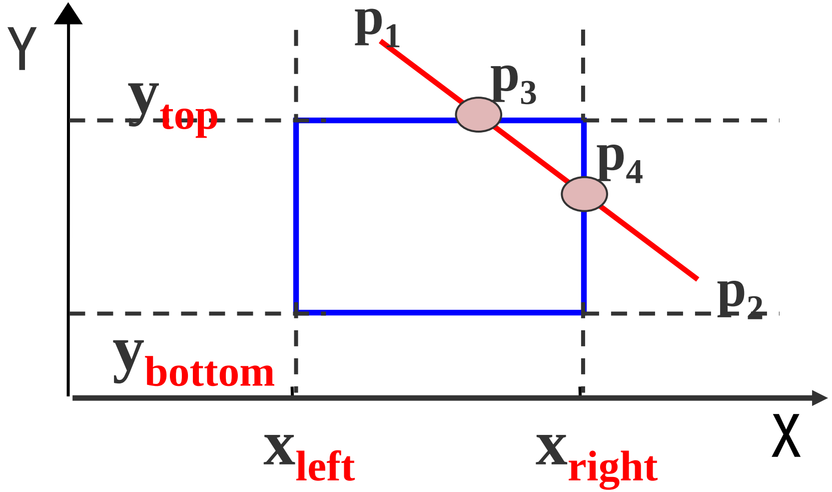
$y_1 > y_{top}$ 且 $y_2 > y_{top}$



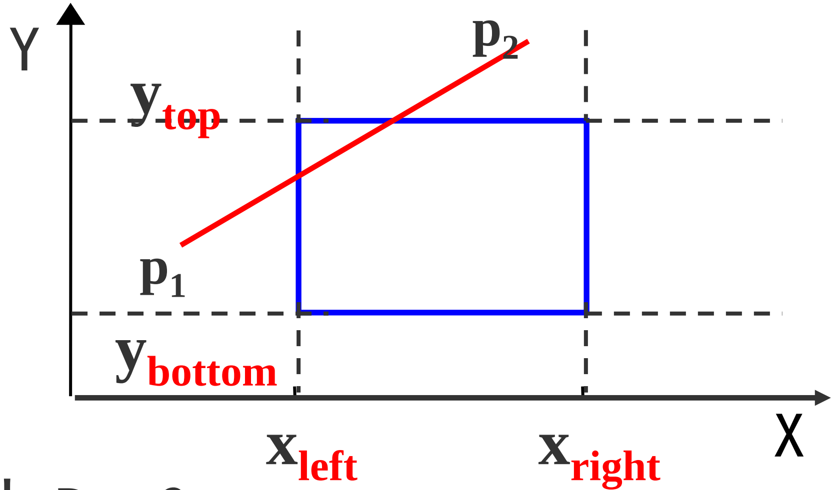
对这四种类型的直线，“简弃”之

（3）如果直线段既不满足“简取”的条件，也不满足“简弃”的条件？

需要对直线段按交点进行分段，分段后判断直线是“简取”还是“简弃”。

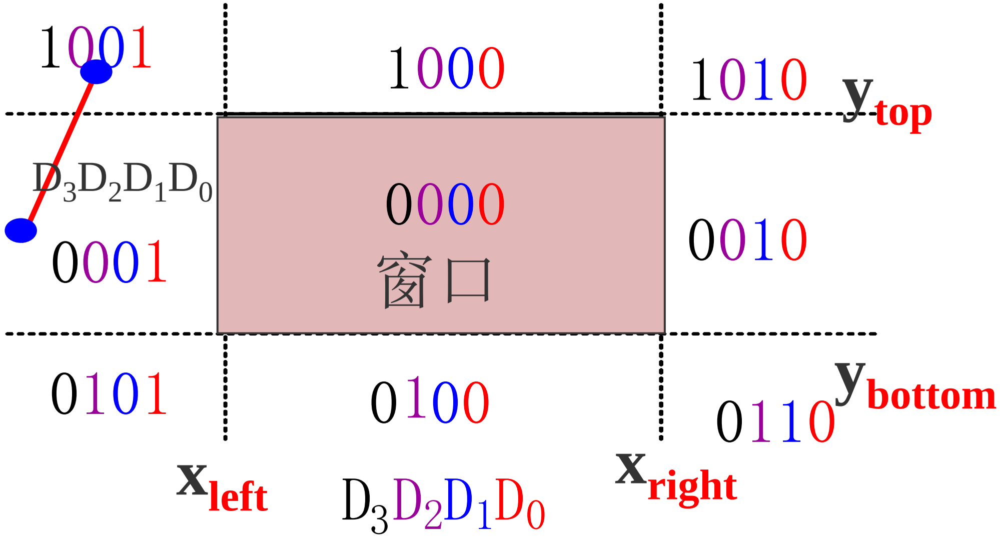


每条线段的端点都赋以四位二进制码$D_3D_2D_1D_0$，编码规则如下：



- 若$x<x_{left}$，　则 $D_0=1$，　否则 $D_0=0$
- 若$x>x_{right}$，则 $D_1=1$，　否则 $D_1=0$
- 若$y<y_{bottom}$，则 $D_2=1$，　否则 $D_2=0$
- 若$y>y_{top}$，　则 $D_3=1$，　否则 $D_3=0$

窗口及其延长线所构成了9个区域。根据该编码规则：



D_0对应窗口左边界

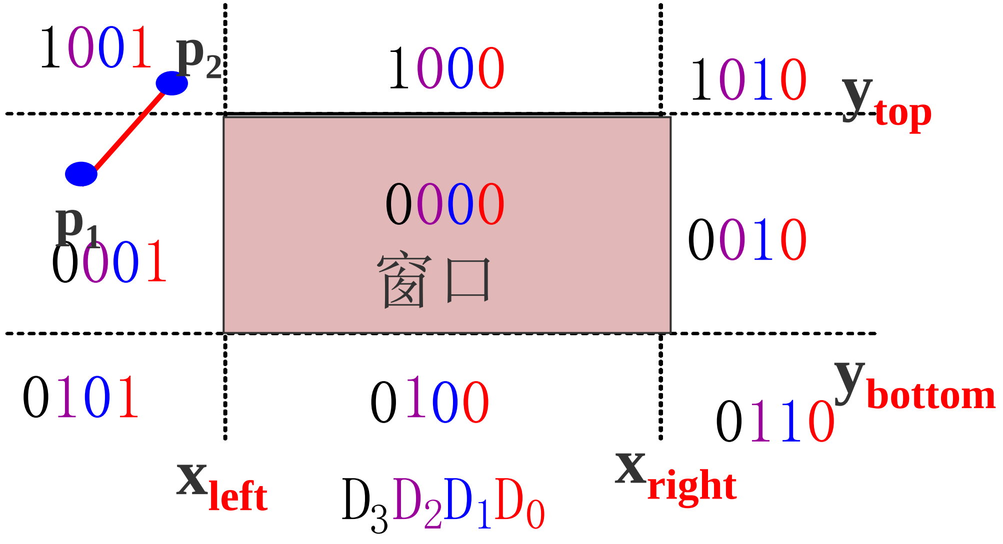
D_1对应窗口右边界

D_2对应窗口下边界

D_3对应窗口上边界

裁剪一条线段时，先求出端点p_1和p_2的编码$code_1$和$code_2$

然后进行二进制“或”运算和“与”运算

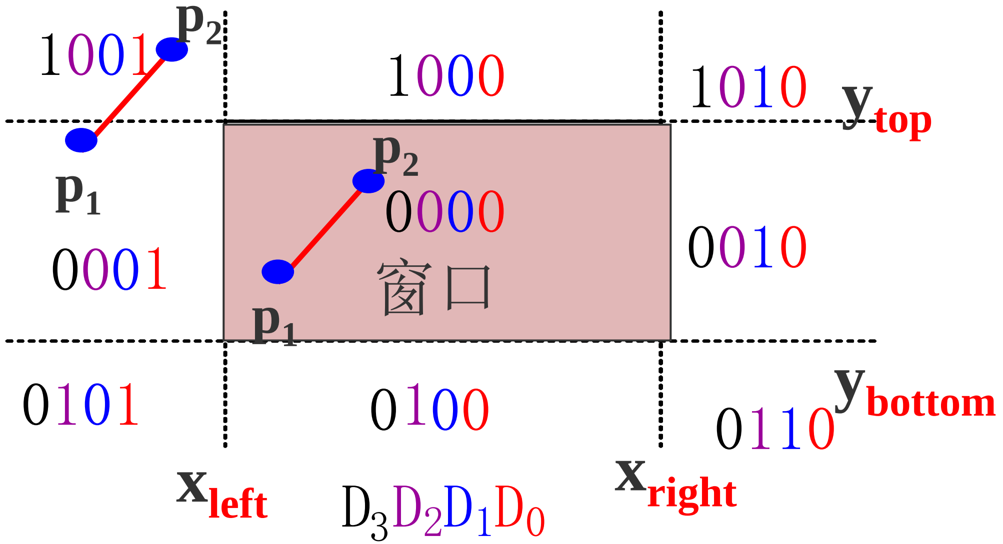


二进制运算

运算符	名称	例子	运算功能
~	位反	~b	求b的位反
&	与运算	b&c	b和c位与
|	或运算	b|c	b和c位或
^	异或运算	b^c	B和c位异或

（1）若$code_1|code_2=0$，对直线段应简取之

	0000
或	0000
	0000



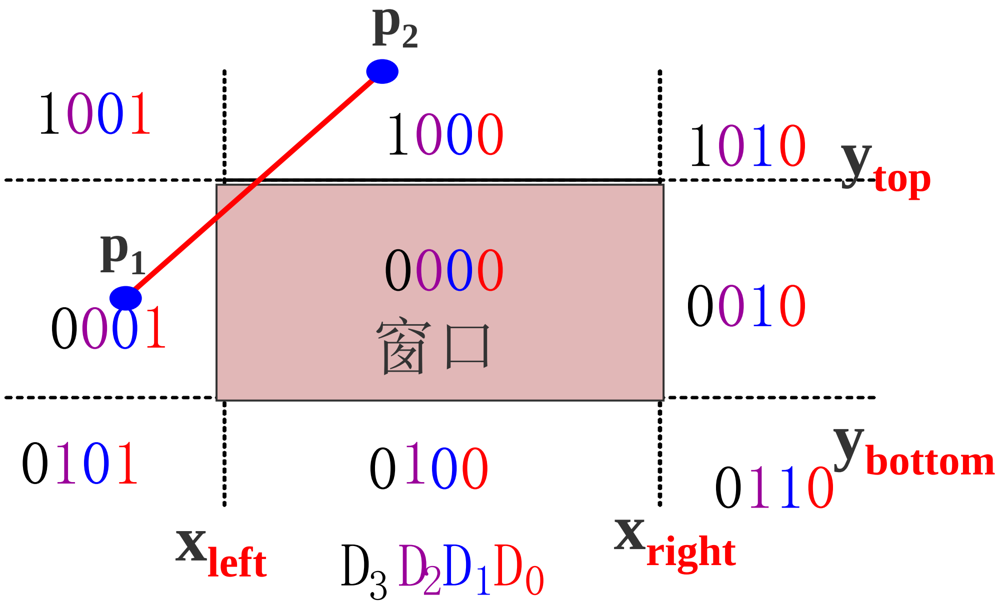
（2）若$code_1\&code_2\neq0$，对直线段可简弃之

	1001
与	0001
	0001

若上述两条件均不成立

或	0001 1000
	1001

与	0001 1000
	0000

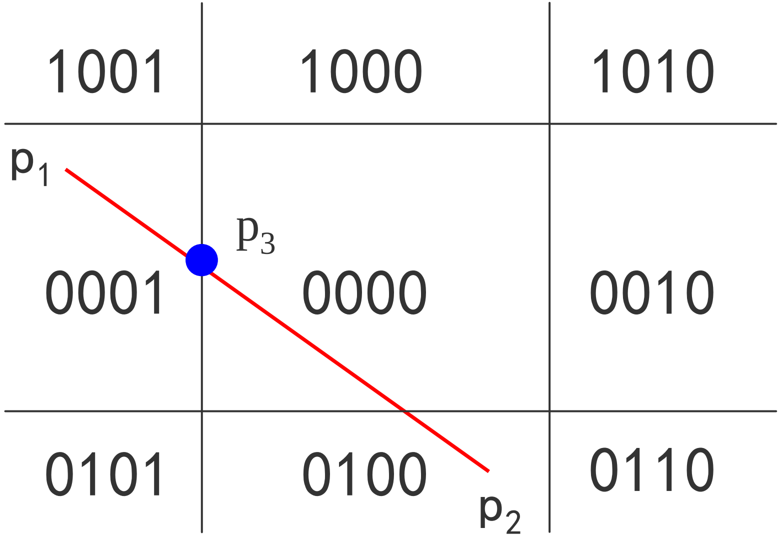


则需求出直线段与窗口边界的交点在交点处把线段一分为二

下面根据该算法步骤来裁剪如图所示的直线段P_1P_2：

首先对P_1P_2进行编码

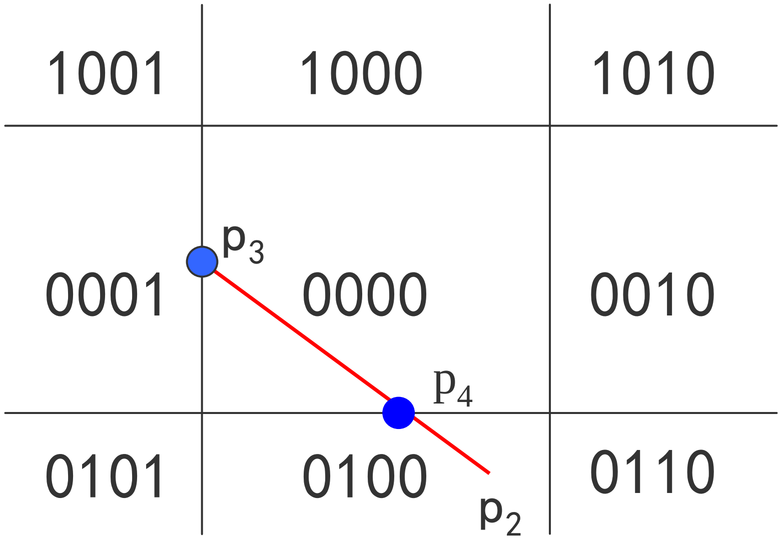
	或		与
	0001		0001
或	0100	与	0100
	0101		0000



按左、右、下、上的顺序求出直线段与窗口左边界的交点为P_3，P_1P_3必在窗口外，可简弃

对P_2P_3重复上述处理

$$\text{或}\ \begin{array}{r} 0000 \\ 0100 \\ \hline 0100 \end{array} \qquad \text{与}\ \begin{array}{r} 0000 \\ 0100 \\ \hline 0000 \end{array}$$



剩下的直线段（P_3P_4）再进行进一步判断，$code_1|code_2=0$，全在窗口中，简取之。

小 结

Cohen-Sutherland算法用编码的方法实现了对直线段的裁剪

编码的思想在图形学中甚至在计算机科学里也是非常重要的，一个很简单的思想可以带来很了不起的作用。

比较适合两种情况：一是大部分线段完全可见；二是大部分线段完全不可见。

二、中点分割算法

和上面讲到的Cohen-Sutherland算法一样，首先对直线段的端点进行编码。

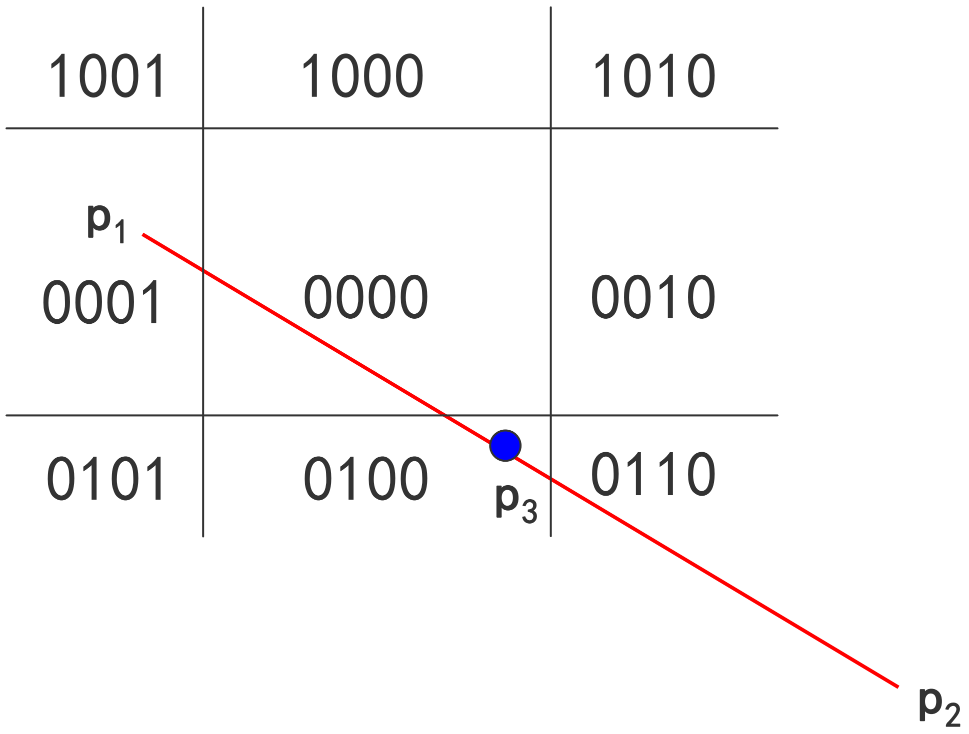
把线段和窗口的关系分成三种情况：

1、完全在窗口内

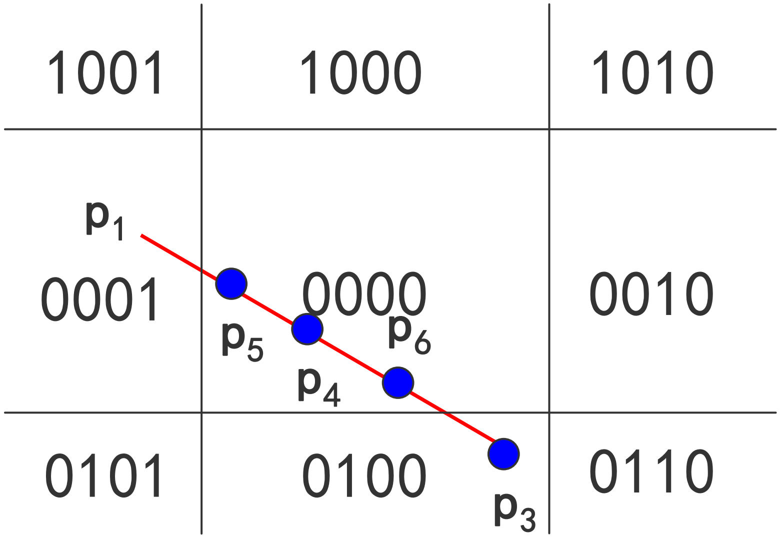
2、完全在窗口外

3、和窗口有交点

中点分割算法的**核心思想**是通过**二分逼近**来确定直线段与窗口的交点。

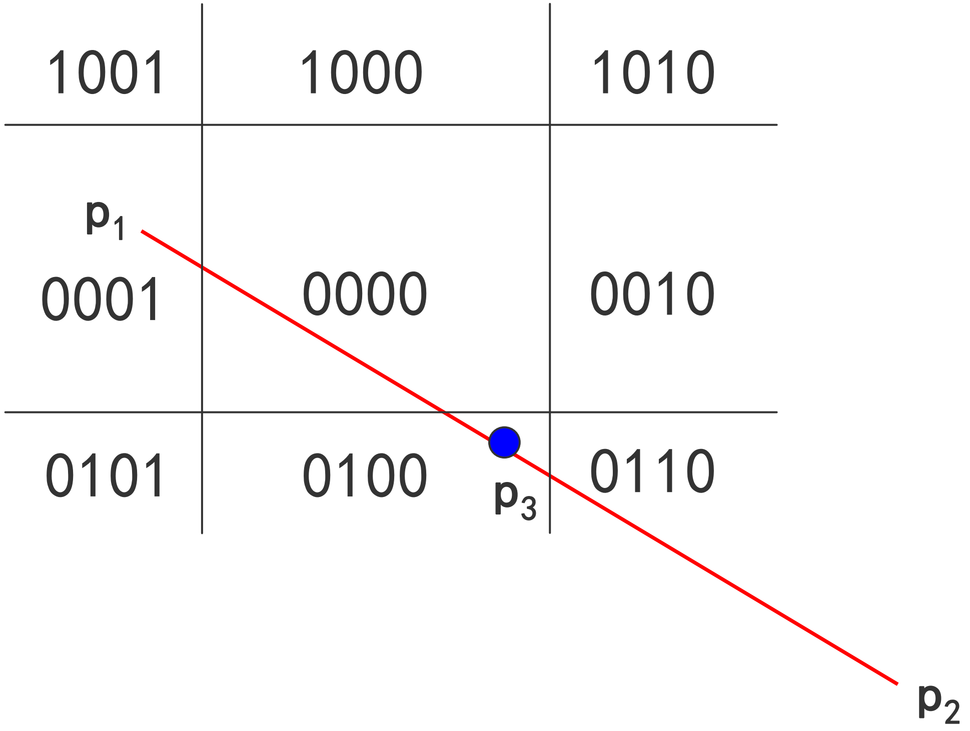


中点分割算法的**核心思想**是通过**二分逼近**来确定直线段与窗口的交点。

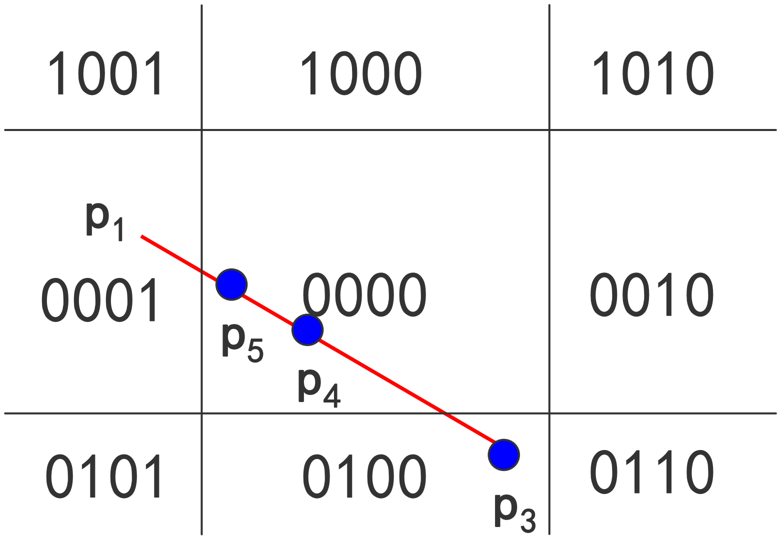


注意:

1、若中点不在窗口内,则把中点和离窗口边界最远点构成的线段丢掉,以线段上的另一点和该中点再构成线段求其中点



2、如中点在窗口内，则又以中点和最远点构成线段，并求其中点，直到中点与窗口边界的坐标值在规定的误差范围内相等



问题：

中点分割算法会不会无限循环二分下去？

三、Liang-Barsky算法

在Cohen-Sutherland算法提出后，梁友栋和Barsky又针对标准矩形窗口提出了更快的Liang-Barsky直线段裁剪算法。

上世纪80年代，梁友栋先生提出了著名的Liang-Barsky算法，至今仍是计算机图形学中最经典的算法之一，也是写进国内外主流《计算机图形学》教科书里的唯一一个以中国人命名的算法。

You-Dong Liang; Barsky, B.A. A new concept and method for line clipping, ACM Transactions on Graphics, Vol.3 1-22,1984

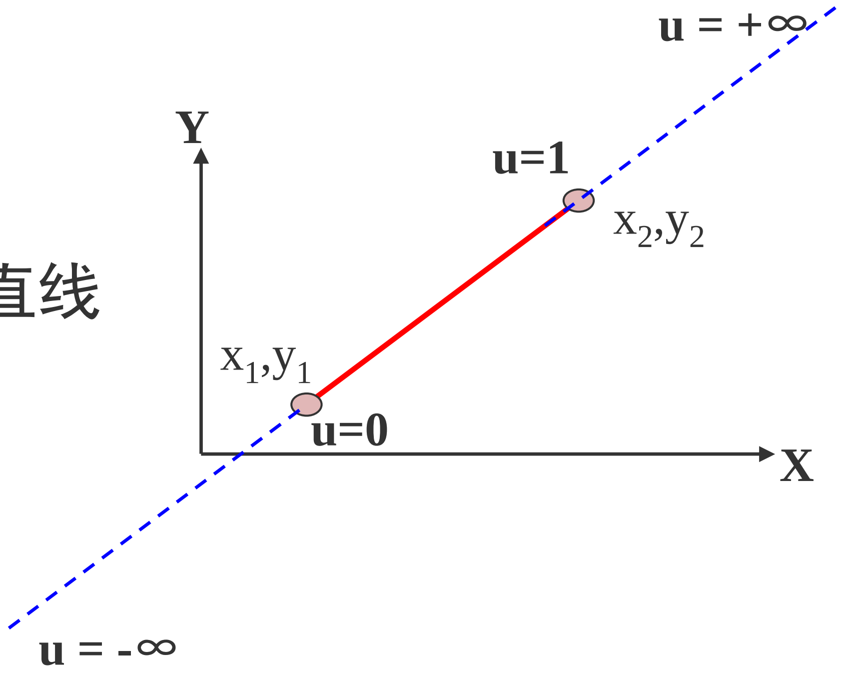
A New Concept and Method for Line Clipping

YOU-DONG LIANG and BRIAN A. BARSKY
University of California, Berkeley

A new concept and method for line clipping is developed that describes clipping in an exact and mathematical form. The basic ideas form the foundation for a family of algorithms for two-dimensional, three-dimensional, and four-dimensional (homogeneous coordinates) line clipping. The

梁算法的主要思想：

（1）用参数方程表示一条直线



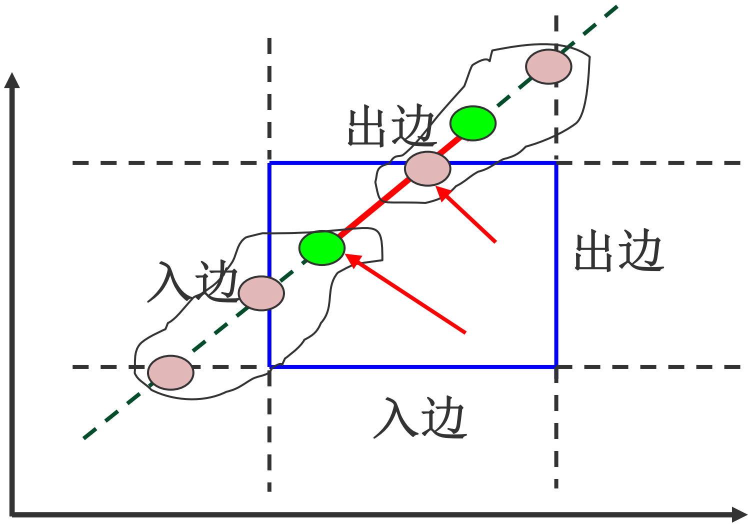
$$x = x_1 + u \cdot (x_2 - x_1) = \underline{x_1 + \Delta x \cdot u}$$
$$y = y_1 + u \cdot (y_2 - y_1) = \underline{y_1 + \Delta y \cdot u}$$

$$0 \le u \le 1$$

梁算法的主要思想：

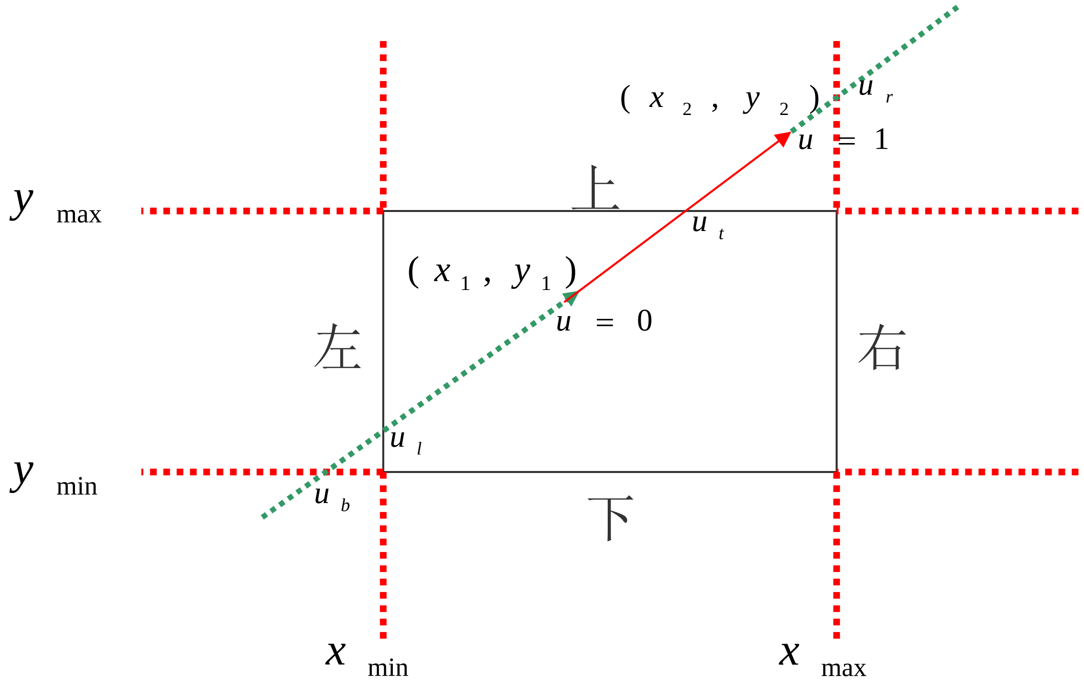
（2）把被裁剪的红色直线段看成是一条有方向的线段，把窗口的四条边分成两类：

入边和出边



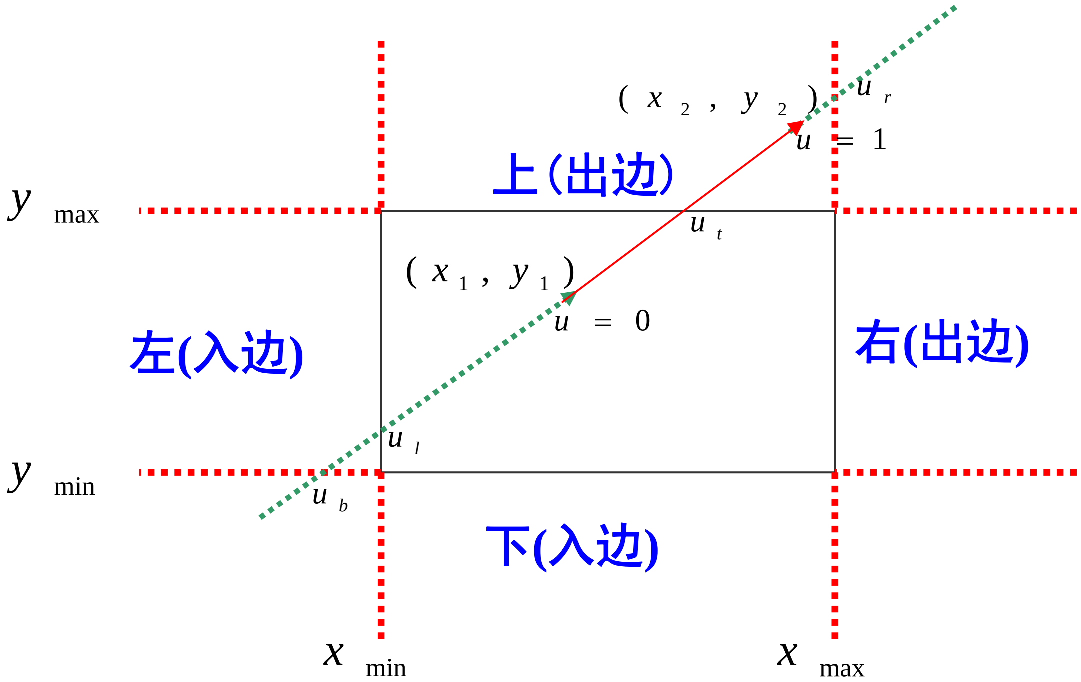
裁剪结果的线段起点是直线和两条入边的交点以及始端点三个点里最前面的一个点，即参数u最大的那个点；

裁剪线段的终点是和两条出边的交点以及端点最后面的一个点，取参数u最小的那个点。



值得注意的是，当u从-∞到+∞遍历直线时，首先对裁剪窗口的两条边界直线（下边和左边）从外面向里面移动，再对裁剪窗口两条边界直线（上边和右边）从里面向外面移动。

如果用u_1，u_2分别表示线段（$u_1 \leqslant u_2$）可见部分的开始和结束



$$u_1 = \max(0, u_l, u_b)$$

$$u_2 = \min(1, u_t, u_r)$$

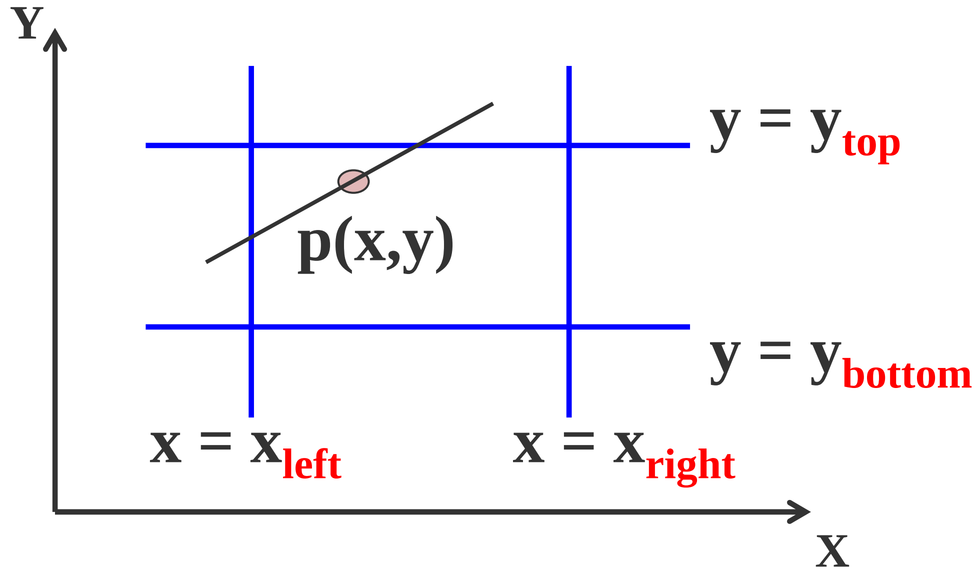
这就是梁先生的重大发现！

Liang-Barsky算法的基本出发点是直线的参数方程

$$x = x_1 + u \cdot (x_2 - x_1)$$
$$y = y_1 + u \cdot (y_2 - y_1)$$
$$0 \le u \le 1$$

$$x_{left} \le x_1 + u \cdot \Delta x \le x_{right}$$

$$y_{bottom} \le y_1 + u \cdot \Delta y \le y_{top}$$



$$x_{left} \le x_1 + u \cdot \Delta x \le x_{right}$$

$$y_{bottom} \le y_1 + u \cdot \Delta y \le y_{top}$$

$$u \cdot (-\Delta x) \le x_1 - x_{left}$$

$$u \cdot \Delta x \le x_{right} - x_1$$

$$u \cdot (-\Delta y) \le y_1 - y_{bottom}$$

$$u \cdot \Delta y \le y_{top} - y_1$$

$$u \cdot (-\Delta x) \leq x_1 - x_{left}$$

$$u \cdot \Delta x \leq x_{right} - x_1$$

$$u \cdot (-\Delta y) \leq y_1 - y_{bottom}$$

$$u \cdot \Delta y \leq y_{top} - y_1$$

令：

$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{left}$$

$$p_2 = \Delta x \qquad q_2 = x_{right} - x_1$$

$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{bottom}$$

$$p_4 = \Delta y \qquad q_4 = y_{top} - y_1$$

令：

$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{left}$$
$$p_2 = \Delta x \qquad q_2 = x_{right} - x_1$$
$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{bottom}$$
$$p_4 = \Delta y \qquad q_4 = y_{top} - y_1$$

于是有：$u \cdot p_k \leq q_k$ 其中，k = 1，2，3，4

$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{left} \qquad p_2 = \Delta x \qquad q_2 = x_{right} - x_1$$
$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{bottom} \qquad p_4 = \Delta y \qquad q_4 = y_{top} - y_1$$

入边：左边和下边

出边：右边和上边

Liang-Barsky算法的基本出发点是直线的参数方程

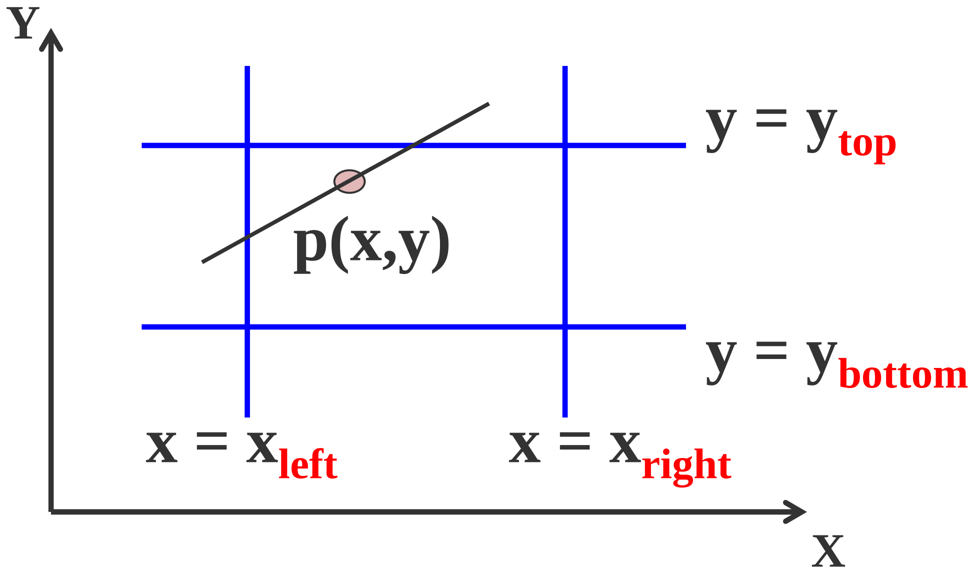
$$x = x_1 + u \cdot (x_2 - x_1)$$

$$y = y_1 + u \cdot (y_2 - y_1)$$

$$0 \le u \le 1$$

$$x_{left} \le x_1 + u \cdot \Delta x \le x_{right}$$

$$y_{bottom} \le y_1 + u \cdot \Delta y \le y_{top}$$



令：

$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{left}$$
$$p_2 = \Delta x \qquad q_2 = x_{right} - x_1$$
$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{bottom}$$
$$p_4 = \Delta y \qquad q_4 = y_{top} - y_1$$

于是有：$u \cdot p_k \leq q_k$ 其中，k = 1，2，3，4

$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{left} \qquad p_2 = \Delta x \qquad q_2 = x_{right} - x_1$$
$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{bottom} \qquad p_4 = \Delta y \qquad q_4 = y_{top} - y_1$$

入边：左边和下边

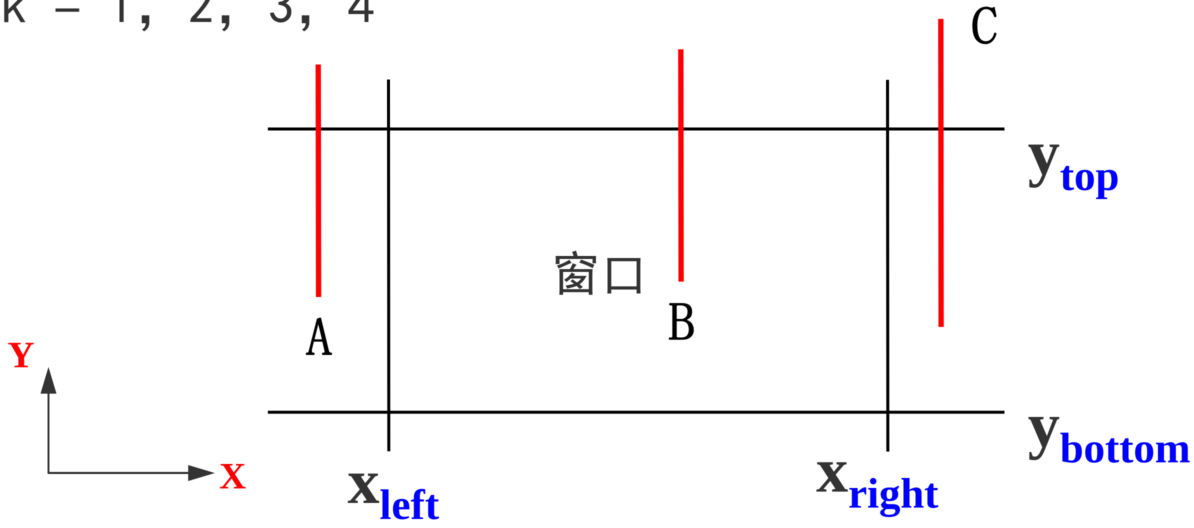
出边：右边和上边

$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{left}$$
$$p_2 = \Delta x \qquad q_2 = x_{right} - x_1$$
$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{bottom}$$
$$p_4 = \Delta y \qquad q_4 = y_{top} - y_1$$

$u \cdot p_k \leq q_k$ 其中，k = 1，2，3，4

（1）分析P_k=0的情况

若P_1= P_2 = 0



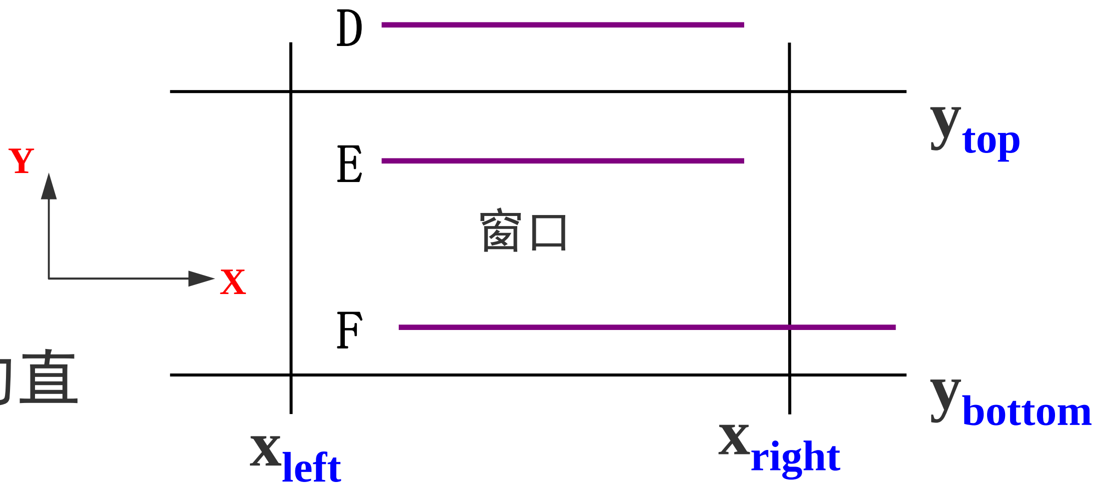
$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{left}$$
$$p_2 = \Delta x \qquad q_2 = x_{right} - x_1$$
$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{bottom}$$
$$p_4 = \Delta y \qquad q_4 = y_{top} - y_1$$

$u \cdot p_k \le q_k$ 其中，k = 1，2，3，4

（1）分析P_k=0的情况

若P_3=P_4=0

任何平行于窗口某边界的直线，其p_k=0



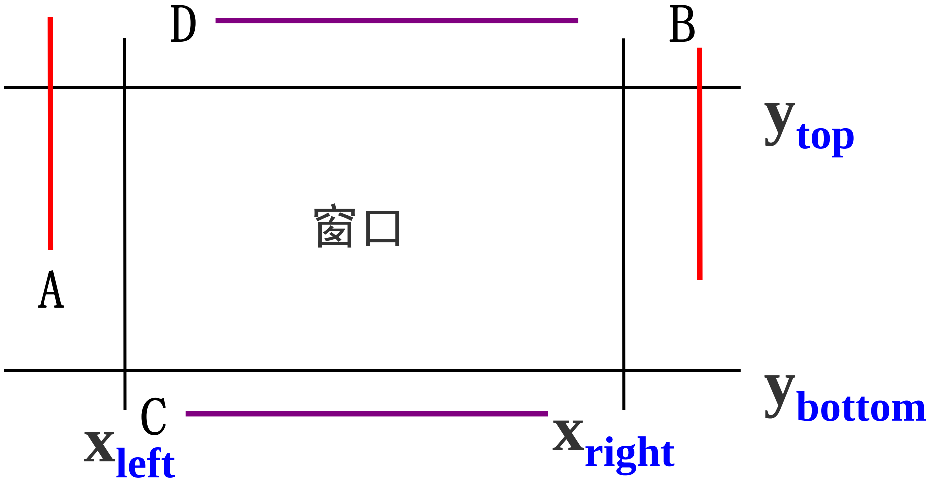
$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{left}$$
$$p_2 = \Delta x \qquad q_2 = x_{right} - x_1$$
$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{bottom}$$
$$p_4 = \Delta y \qquad q_4 = y_{top} - y_1$$

$u \cdot p_k \leq q_k$ 其中，k = 1，2，3，4

（1）分析P_k=0的情况

如果还满足q_k<0

则线段完全在边界外，应舍弃该线段



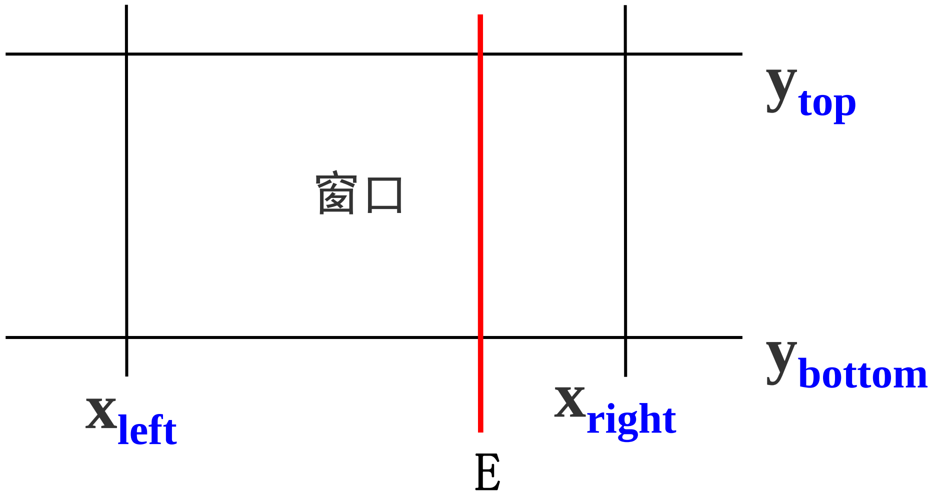
$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{left}$$
$$p_2 = \Delta x \qquad q_2 = x_{right} - x_1$$
$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{bottom}$$
$$p_4 = \Delta y \qquad q_4 = y_{top} - y_1$$

$u \cdot p_k \le q_k$ 其中，k = 1，2，3，4

（1）分析P_k=0的情况

如果$q_k \geq 0$

则进一步判断



$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{left}$$

$$p_2 = \Delta x \qquad q_2 = x_{right} - x_1$$

$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{bottom}$$

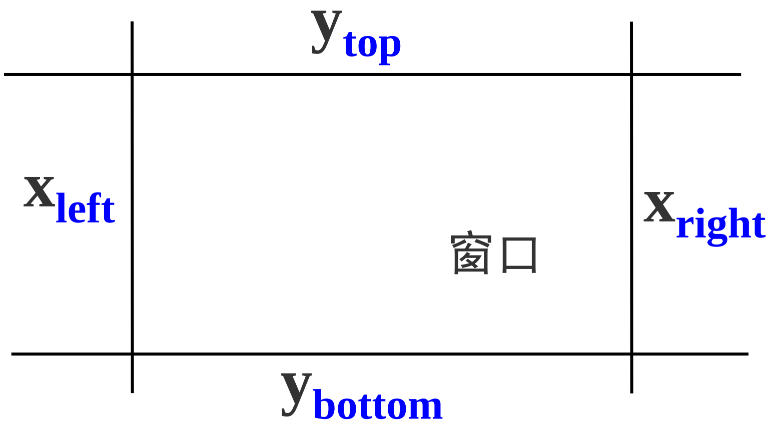
$$p_4 = \Delta y \qquad q_4 = y_{top} - y_1$$

$u \cdot p_k \le q_k$ 其中，k = 1，2，3，4

（2）当$p_k \neq 0$时：

当$p_k<0$时

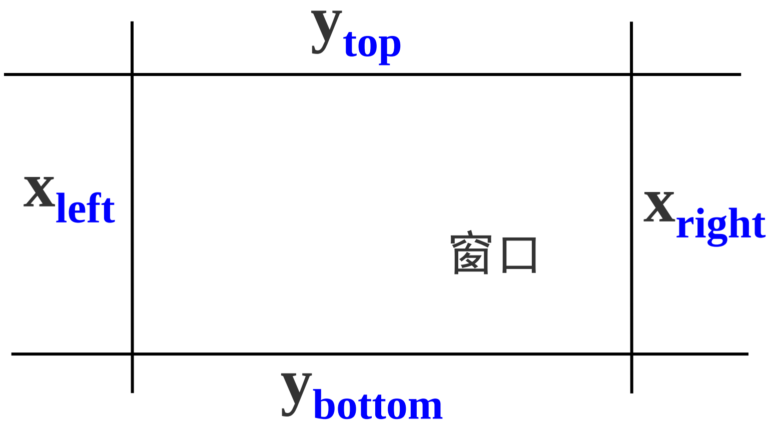
线段从裁剪边界延长线的外部延伸到内部，是入边交点



$$
\begin{aligned}
p_1 &= -\Delta x & q_1 &= x_1 - x_{left} \\
p_2 &= \Delta x & q_2 &= x_{right} - x_1 \\
p_3 &= -\Delta y & q_3 &= y_1 - y_{bottom} \\
p_4 &= \Delta y & q_4 &= y_{top} - y_1
\end{aligned}
$$

$u \cdot p_k \le q_k$ 其中，$k = 1，2，3，4$

（2）当pk≠0时：



当p_k ＞ 0时

线段从裁剪边界延长线的内部延伸到外部，是出边交点

线段和窗口边界一共有四个交点，根据p_k的符号，就知道哪两个是入交点，哪两个是出交点

当$p_k < 0$时：对应入边交点

当$p_k > 0$时：对应出边交点

一共四个u值，再加上u=0、u=1两个端点值，总共六个值

把$p_k<0$的两个u值和0比较去找最大的，把$p_k>0$的两个u值和1比较去找最小的，这样就得到两个端点的参数值

$$u_k = \frac{q_k}{p_k} \qquad (p_k \neq 0, k = 1,2,3,4)$$

u_k是窗口边界及其延长线的交点的对应参数值

分别计算u_{max}和u_{min}：

$$u_{max} = \max(0, u_k|_{pk<0}, u_k|_{pk<0})$$

$$u_{min} = \min(u_k|_{pk>0}, u_k|_{pk>0}, 1)$$

注意：$p_k < 0$，代表入边；$p_k > 0$代表出边

若$u_{max}>u_{min}$，则直线段在窗口外，删除该直线

若$u_{max} \leqslant u_{min}$，将$u_{max}$和$u_{min}$代回直线参数方程，即求出直线与窗口的两实交点坐标。

$$x = x_1 + u \cdot (x_2 - x_1)$$
$$y = y_1 + u \cdot (y_2 - y_1)$$

注意：因为对于实交点$0 \leqslant u \leqslant 1$，因此$u_{max}$不能小于0，$u_{min}$不能大于1

下面写出Liang-Barsky裁剪算法步骤：

（1）输入直线段的两端点坐标（x_1, y_1）、（x_2, y_2），以及窗口的四条边界坐标：wxl、wxr、wyb和wyt

（2）若△X=0，则$p_1=p_2=0$，此时进一步判断是否满足$q_1<0$或$q_2<0$，若满足，则该直线段不在窗口内，算法转（7）-结束。否则，满足$q_1 \geqslant 0$且$q_2 \geqslant 0$，则进一步计算u_{max}和u_{min}：

$$u_{\max} = \max(0, \; u_k \big|_{pk<0})$$

$$u_{\min} = \min(u_k \big|_{pk>0}, 1)$$

其中，$u_k = \dfrac{q_k}{p_k} \quad (p_k \neq 0, k = 3,4)$ 。算法转（5）

（3）若△y=0，则$p_3=p_4=0$, 此时进一步判断是否满足$q_3<0$或$q_4<0$，若满足，则该直线段不在窗口内，算法转（7）。否则，满足$q_3 \geqslant 0$且$q_4 \geqslant 0$，则进一步计算u_{max}和u_{min}：

$$u_{\max} = \max(0, \; u_k|_{pk<0})$$

$$u_{\min} = \min(u_k|_{pk>0}, 1)$$

其中，$u_k = \dfrac{q_k}{p_k}$ $(p_k \neq 0, k = 3,4)$ 。算法转（5）

（4）若上述两条均不满足，则有$p_k \neq 0$（k=1, 2, 3, 4），此时计算u_{max}和u_{min}：

$$u_{max} = \max（0,\ u_k|_{pk<0},\ u_k|_{pk<0}）$$

$$u_{min} = \min（u_k|_{pk>0},\ u_k|_{pk>0},\ 1）$$

其中，$u_k = \frac{q_k}{p_k} \ (p_k \neq 0, k = 1,2,3,4)$

（5）求得u_{max}和u_{min}后，进行判断：若$u_{max}>u_{min}$，则直线段在窗口外，算法转（7）。若$u_{max}\leqslant u_{min}$，利用直线的参数方程：

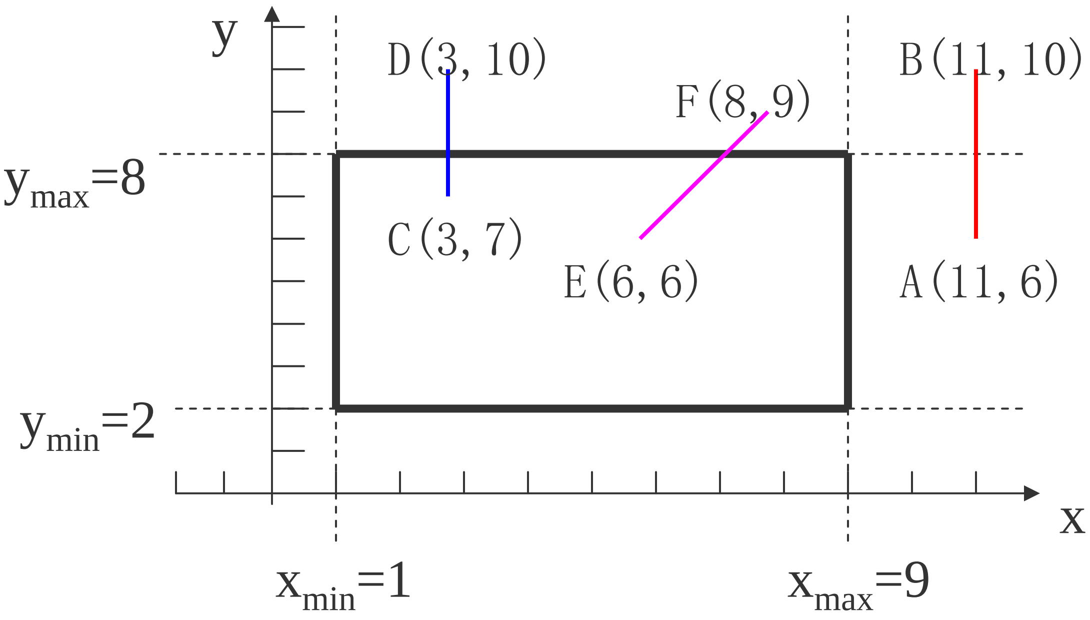
$$x = x_1 + u \cdot (x_2 - x_1)$$

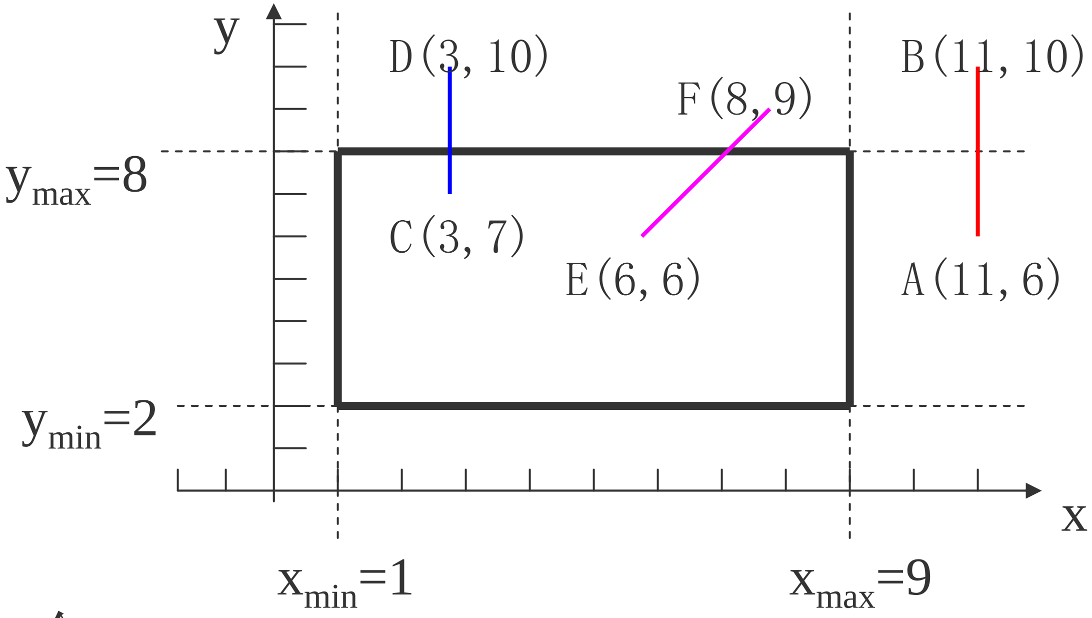
$$y = y_1 + u \cdot (y_2 - y_1)$$

（6）利用直线的扫描转换算法绘制在窗口内的直线段。

（7）算法结束。

用Liang-Barsky算法裁减直线段举例





对于直线AB，有：

$p_1 = 0$　　$q_1 = 10$

$p_2 = 0$　　$q_2 = -2$

$p_3 = -4$　　$q_3 = 4$

$p_4 = 4$　　$q_4 = 2$

令：

$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{left}$$
$$p_2 = \Delta x \qquad q_2 = x_{right} - x_1$$
$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{bottom}$$
$$p_4 = \Delta y \qquad q_4 = y_{top} - y_1$$

AB完全在右边界之右

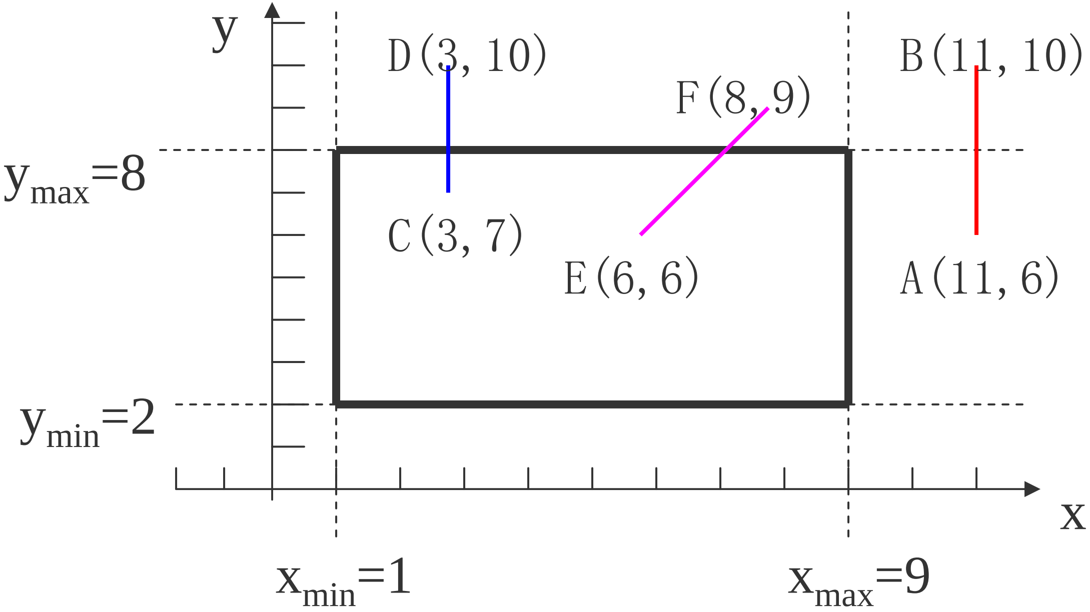
对于直线CD，有：

$p_1 = 0 \qquad q_1 = 2$

$p_2 = 0 \qquad q_2 = 6$

$p_3 = -3 \qquad q_3 = 5$

$p_4 = 3 \qquad q_4 = 1$



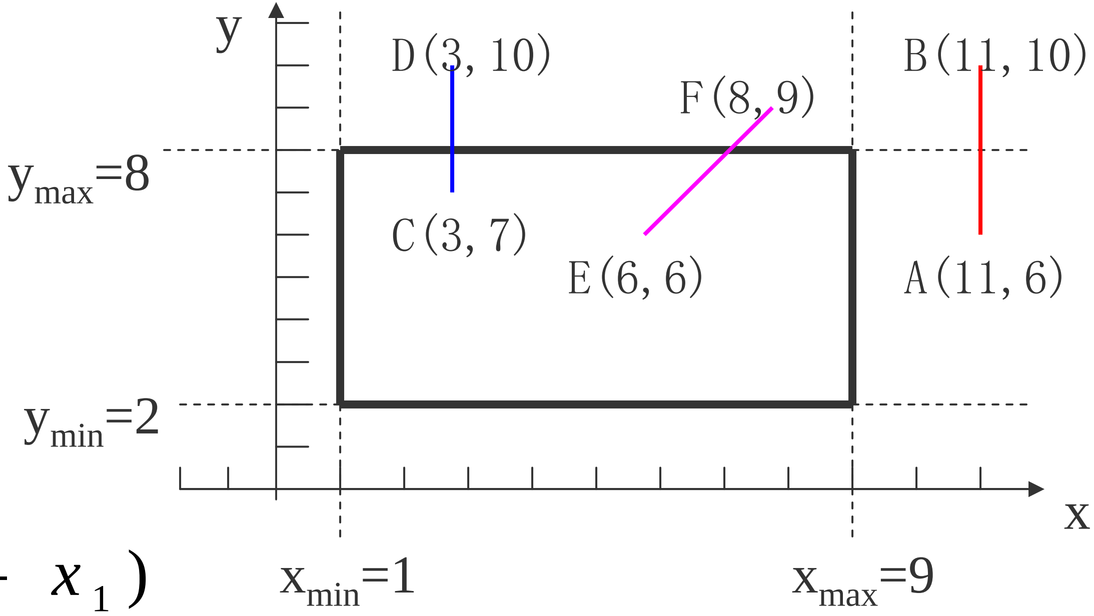
$u_3 = -5/3 \quad u_4 = 1/3$

$$u_{max} = \max(0, -5/3) = 0$$

$$u_{min} = \min(1, 1/3) = 1/3$$

$$u_{max} = 0$$

$$u_{min} = 1/3$$



$$x = x_1 + u \cdot (x_2 - x_1)$$

$$y = y_1 + u \cdot (y_2 - y_1)$$

裁减后直线的两个端点是（3，7）和（3，8）。

对于直线EF，有：

$p_1 = -2$　　$q_1 = 5$

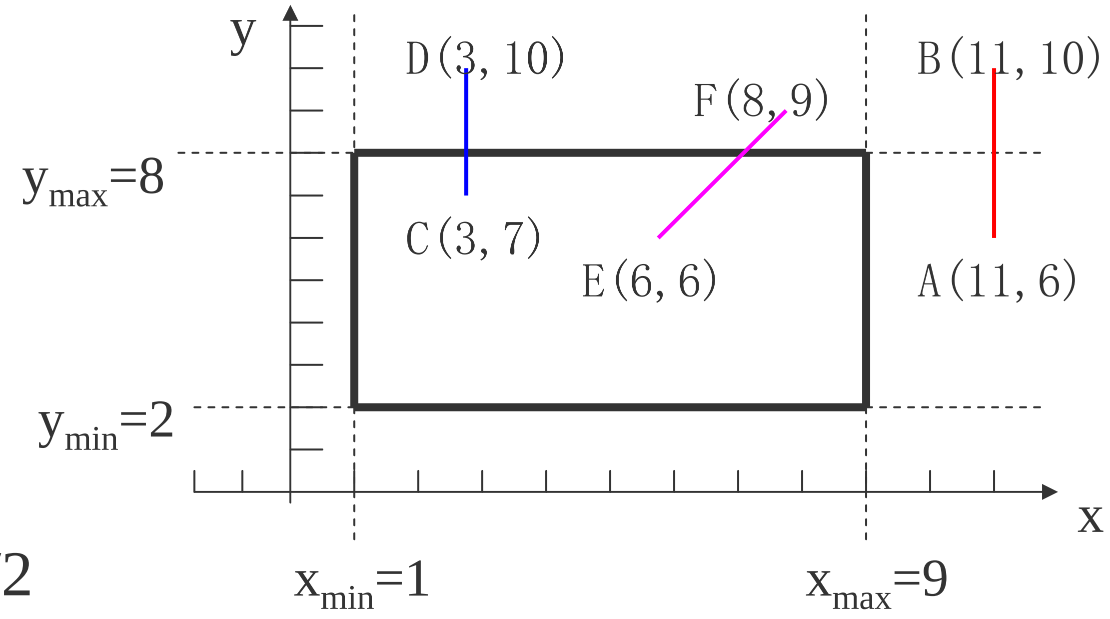
$p_2 = 2$　　$q_2 = 3$

$p_3 = -3$　　$q_3 = 4$

$p_4 = 3$　　$q_4 = 2$

$u_1 = -5/2$　　$u_2 = 3/2$

$u_3 = -4/3$　　$u_4 = 2/3$



因此：

$u_{max} = \max(0, -5/2, -4/3) = 0$

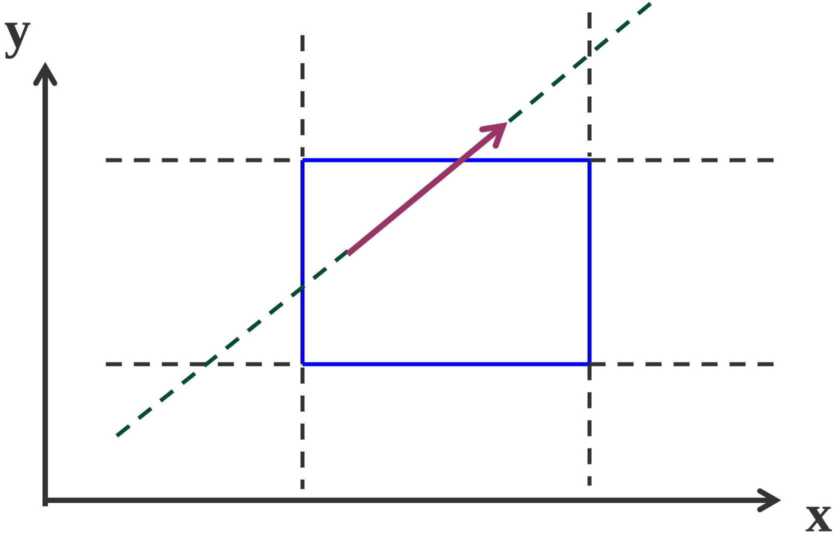
$u_{min} = \min(1, 3/2, 2/3) = 2/3$

$u_{max} < u_{min}$

裁剪后的直线的两个端点是（6，6）和（7.33，8）

Liang-Barsky算法小结

1、直线段看成是有方向的



2、直线参数化

$$x = x_1 + u \cdot (x_2 - x_1)$$

$$y = y_1 + u \cdot (y_2 - y_1)$$

3、判断线段上一点是否在窗口内，需满足下面两个不等式

$$x_{left} \le x_1 + u \cdot \Delta x \le x_{right}$$

$$y_{bottom} \le y_1 + u \cdot \Delta y \le y_{top}$$

$$u \cdot p_k \le q_k$$

4、线段和窗口边界一共有四个交点

$$u_k = \frac{q_k}{p_k} \qquad (p_k \neq 0, k = 1,2,3,4)$$

$$u_{max} = \max(0, u_k|_{pk<0}, \ u_k|_{pk<0})$$

$$u_{min} = \min(u_k|_{pk>0}, \ u_k|_{pk>0}, 1)$$

$$x = x_1 + u \cdot (x_2 - x_1)$$

$$y = y_1 + u \cdot (y_2 - y_1)$$

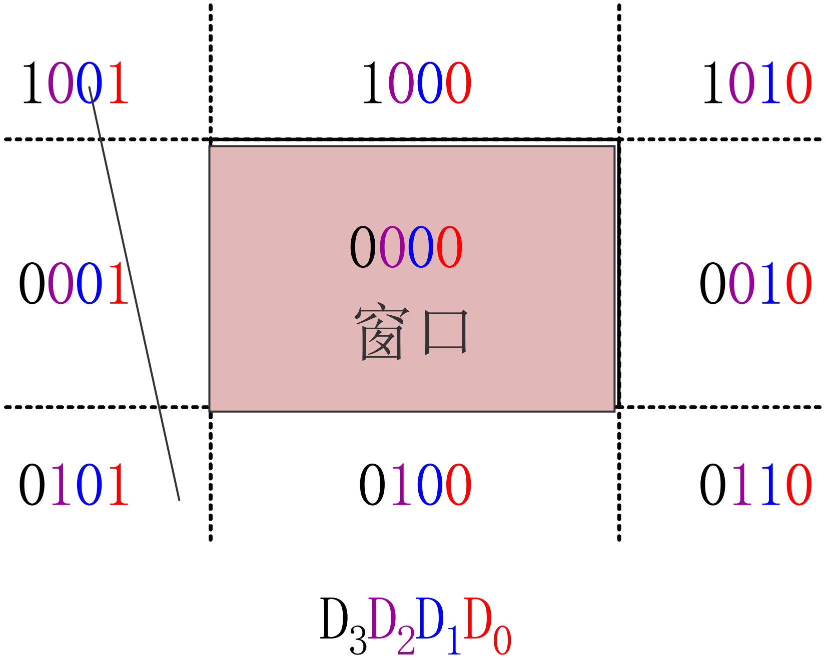
梁算法引进了一种新的思想，把窗口的4条边根据线段的方向走向分成入边和出边，裁剪的算法就变得比较简单。

它首先把线段的裁剪变成了射线的裁剪，也就是把线段赋以方向。发现最终裁剪结果的起始端点肯定是入边的两个交点和线段的起始点里面的一点；而裁剪结果的终点肯定是出边的两个交点和线段的终点里面的一点；前三个点取最大，后三个点取最小。这样就把一个经典的裁剪问题变成了解不等式。

这也是中国人的算法第一次出现在了所有图形学教科书都必须提的一个算法。这就叫原始创新！

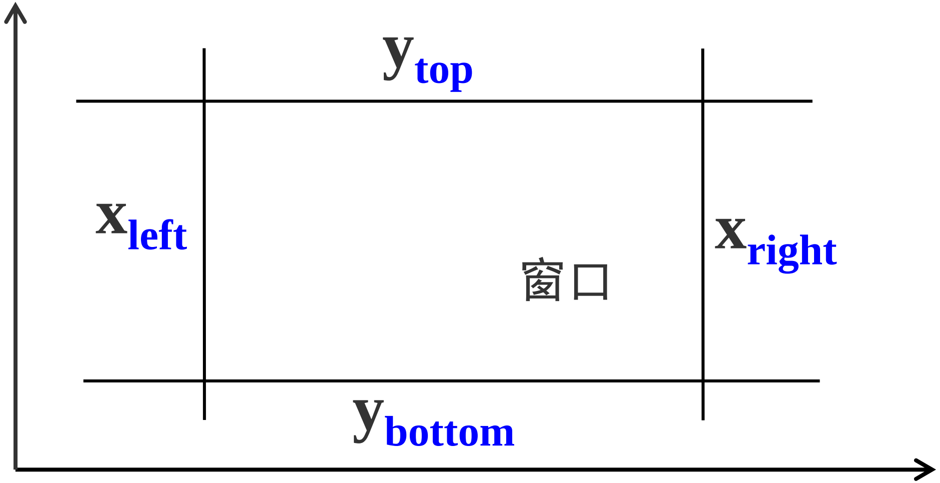
Cohen-Sutherland PK Liang-Barsky裁剪算法比较

1、Cohen-Sutherland算法的核心思想是编码



窗口及区域编码

2、如果被裁剪的图形大部分线段要么在窗口内或者要么完全在窗口外，很少有贯穿窗口的。Cohen-Sutherland算法效果非常好



3、在一般情况下，Liang-Barsky裁剪算法的效率则优于Cohen-Sutherland算法

4、Cohen-Sutherland和Liang-Barsky只能应用于矩形窗口

裁剪算法是非常底层的算法，任何一个图形显示的算法和软件都离不开这些底层算法

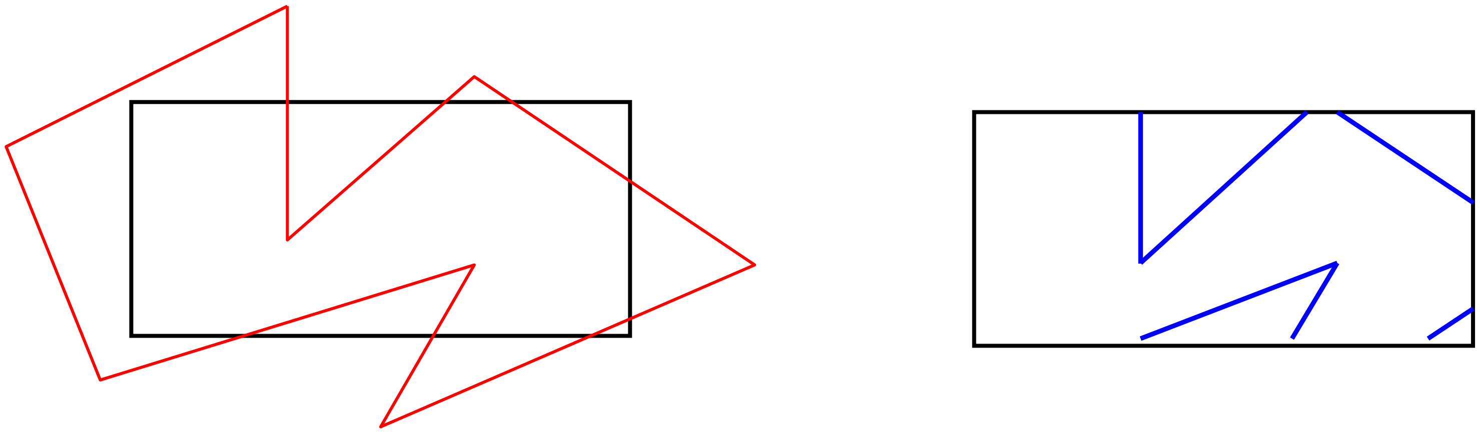
这些底层算法目前都已经固化到计算机硬件里了。现在做一个图形软件，不需要再研究裁剪算法

一是了解图形学底层算法的思想

另一方面，改进底层算法对提高图形显示和处理效率具有至关重要的作用

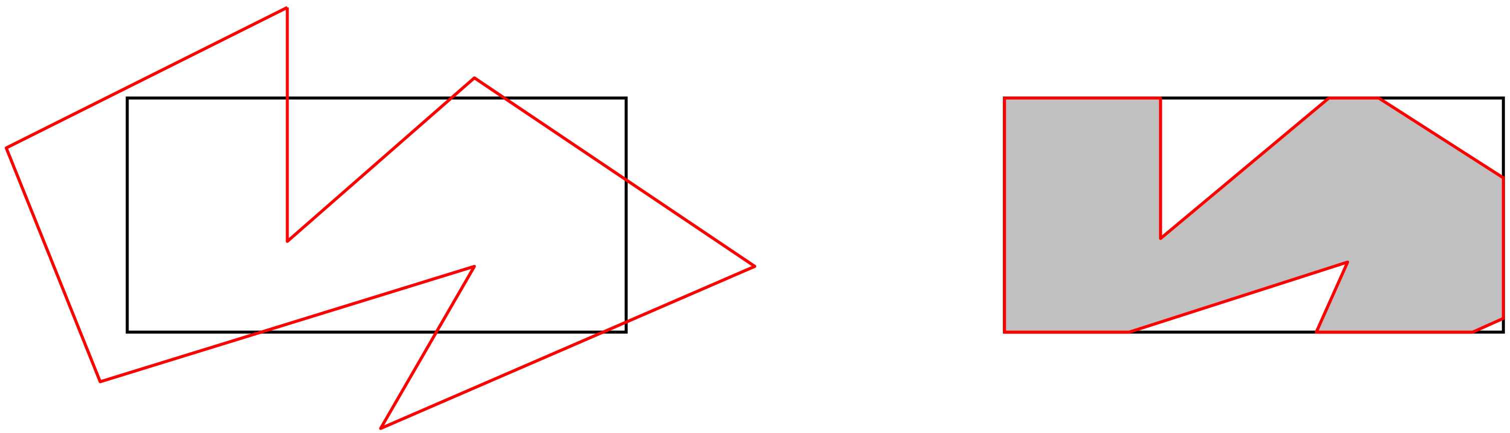
多边形、字符裁剪

一、多边形的裁剪



即得到一系列不连续的直线段！

而实际上，应该得到的是下图所示的有边界的区域：

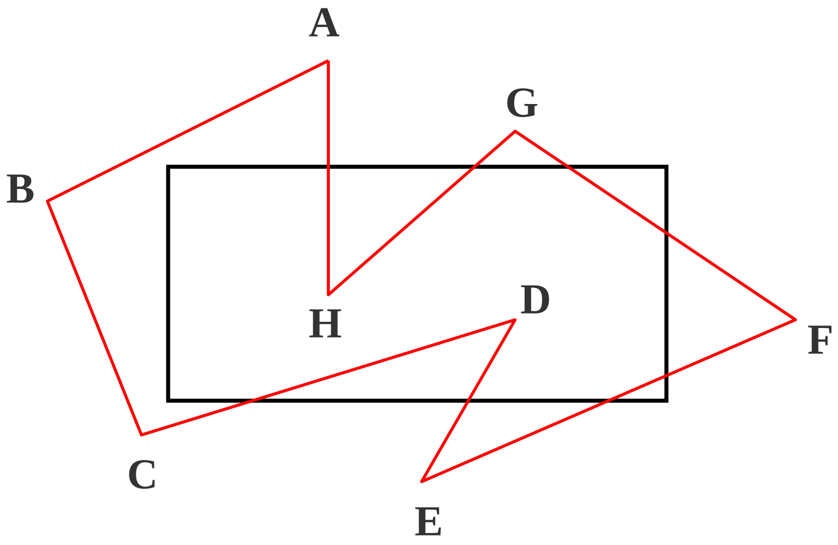


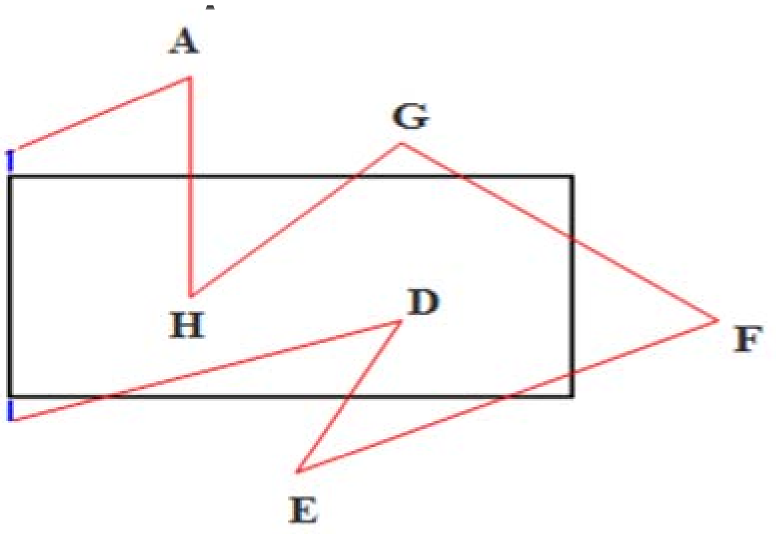
多边形裁剪算法的输出应该是裁剪后的多边形边界的顶点序列！

需要构造能产生一个或多个封闭区域的多边形裁剪算法

二、Sutherland-Hodgeman多边形裁剪

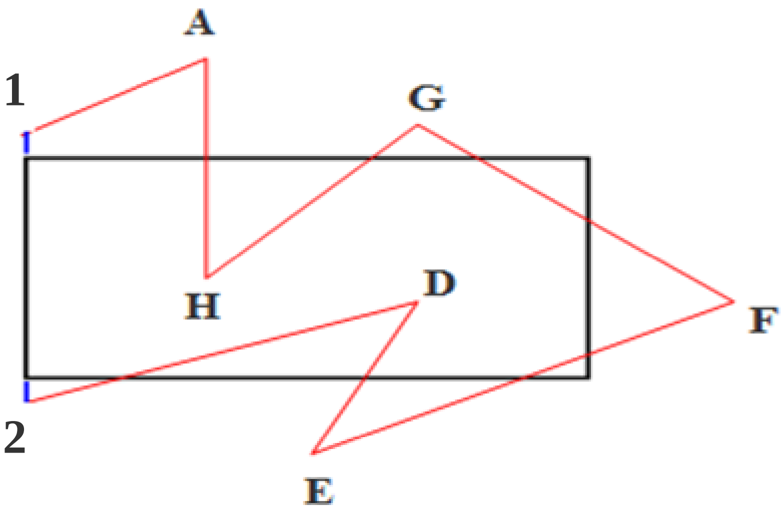
该算法的基本思想是将多边形边界作为一个整体，每次用窗口的一条边对要裁剪的多边形和中间结果多边形进行裁剪，体现一种分而治之的思想

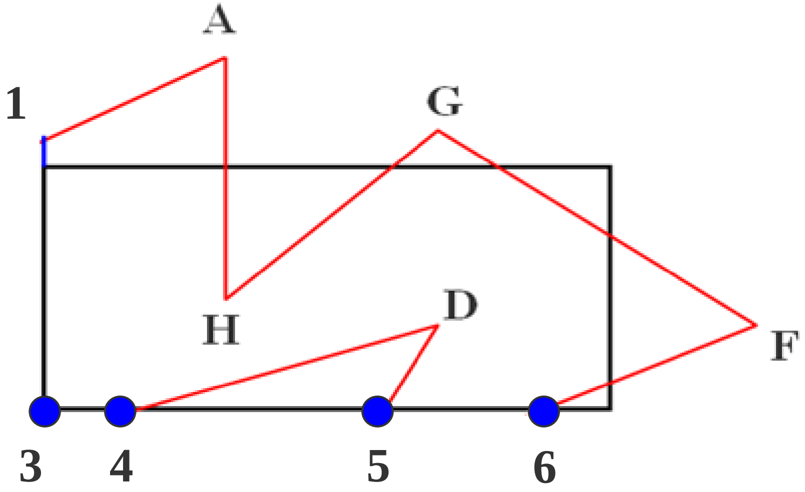




算法的输入：**ABCDEFGH**　　　　输出：**A12DEFGHA**

把平面分为两个区域：包含有窗口区域的一个域称为可见侧；不包含窗口区域的域为不可见侧





输入：A12DEFGH　　　　**输出：A134D56FGH**

裁剪得到的结果多边形的顶点有两部分组成：

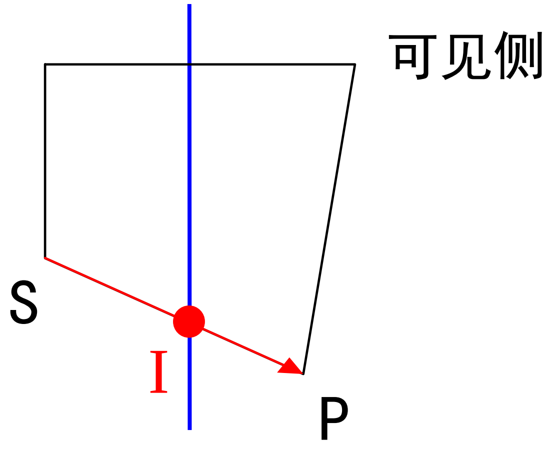
（1）落在可见一侧的原多边形顶点

（2）多边形的边与裁剪窗口边界的交点

根据多边形每一边与窗口边所形成的位置关系，沿着多边形依次处理顶点会遇到四种情况：

（1）第一点S在不可见侧面，而第二点P在可见侧

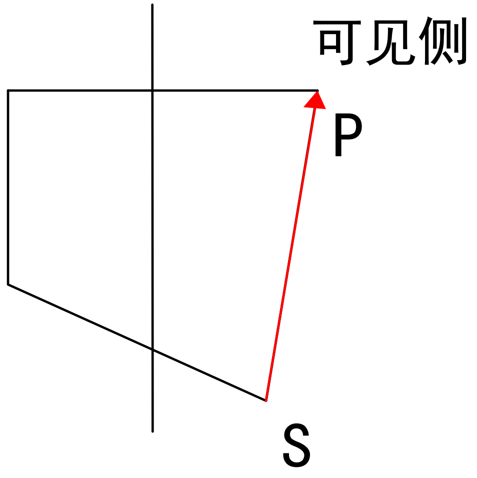
交点I与点P均被加入到输出顶点表中。



（1）输出I、P

（2）是S和P都在可见侧

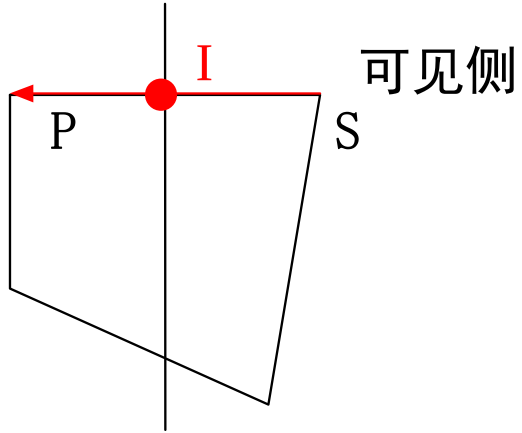
则P被加入到输出顶点表中



（2）输出P

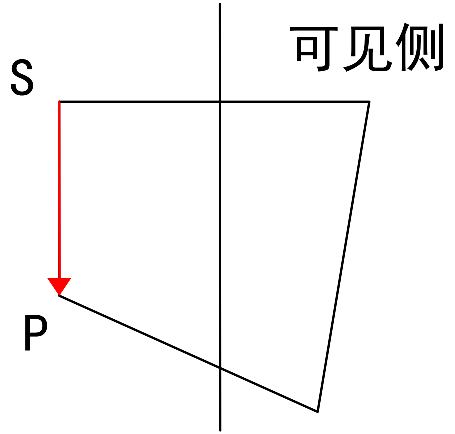
（3）S在可见侧，而P在不可见侧

则交点I被加入到输出顶点表中



(3) 输出I

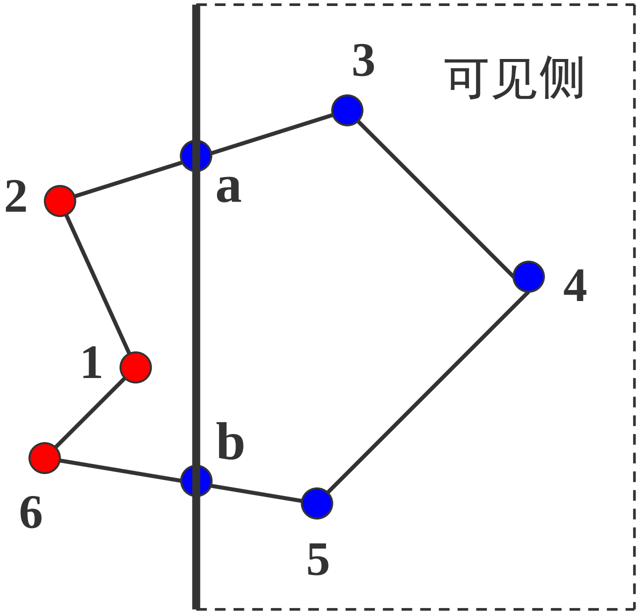
（4）如果S和P都在不可见侧



(4) 不输出

输出顶点表中不增加任何顶点

在窗口的一条裁剪边界处理完所有顶点后，其输出顶点表将用窗口的下一条边界继续裁剪



输入：1 2 3 4 5 6

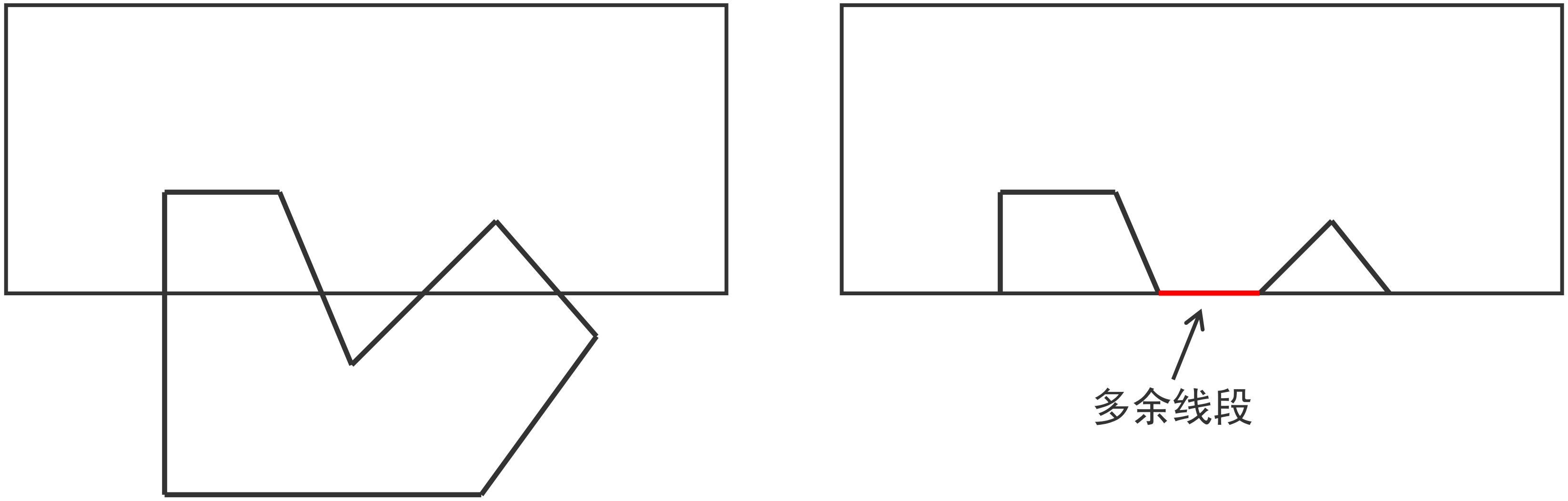
输出：

a 3 4 5 b

```
while对于每一个窗口边或面 do
  begin
  if P1 在窗口边的可见一侧 then 输出P1
  for i=1 to n do
      begin
      if P1 在窗口边的可见一侧 then
         if P1+1 在窗口边的可见一侧 then 输出 P1+1
         else 计算交点并输出交点
      else if Pi+1 在窗口可见一侧, then 计算交点
           并输出交点，同时输出Pi+1
      end
  end
end
```

Sutherland-Hodgeman算法不足之处

利用Sutherland-Hodgeman裁剪算法对凸多边形进行裁剪可以获得正确的裁剪结果，但是。。。



三、文字裁剪

屏幕上显示的不仅仅是多边形和直线等，还显示字符。字符如何来处理？

字符并不是由直线段组成的。文字裁剪包括以下几种：

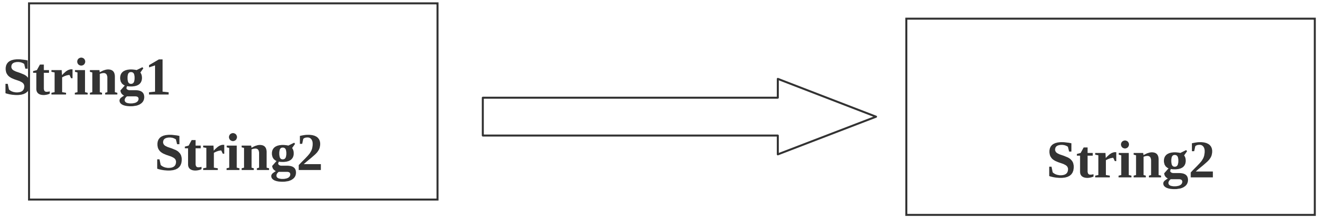
串精度裁剪

字符精度裁剪

笔划/像素精度裁剪

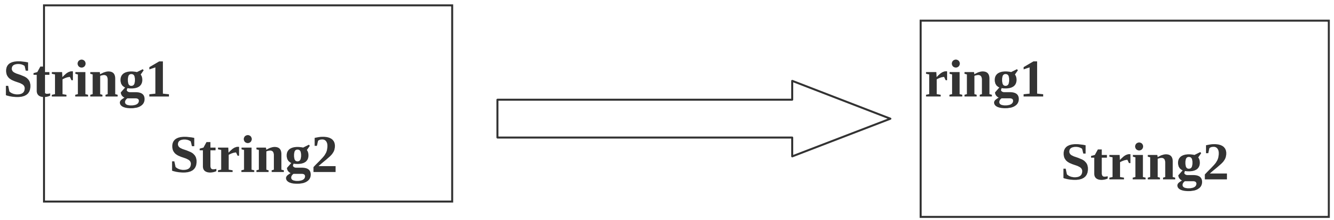
1、串精度裁剪

当字符串中的所有字符都在裁剪窗口内时，就全部保留它，否则舍弃整个字符串



2、字符精度裁剪

在进行裁剪时，任何与窗口有重叠或落在窗口边界以外的字符都被裁剪掉



3、笔画/象素精度裁剪

将笔划分解成直线段对窗口作裁剪。需判断字符串中各字符的哪些像素、笔划的哪一部分在窗口内，保留窗口内部分，裁剪掉窗口外的部分