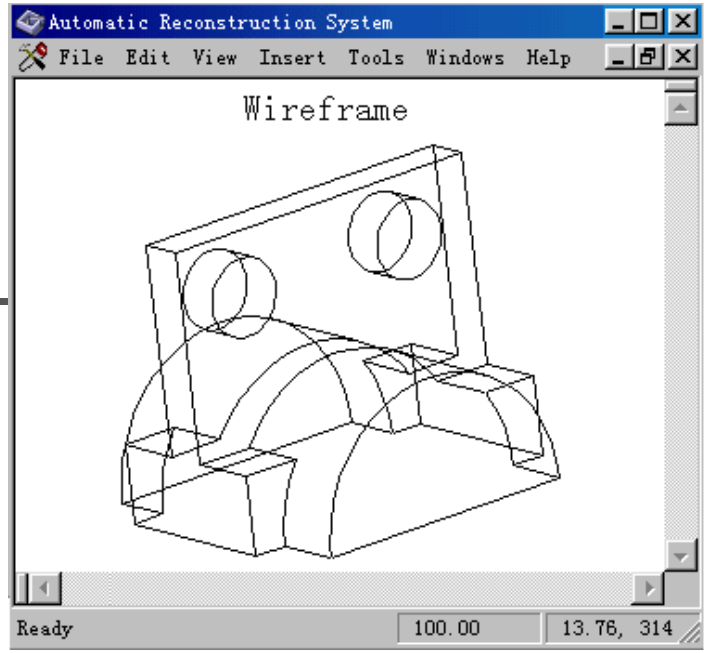
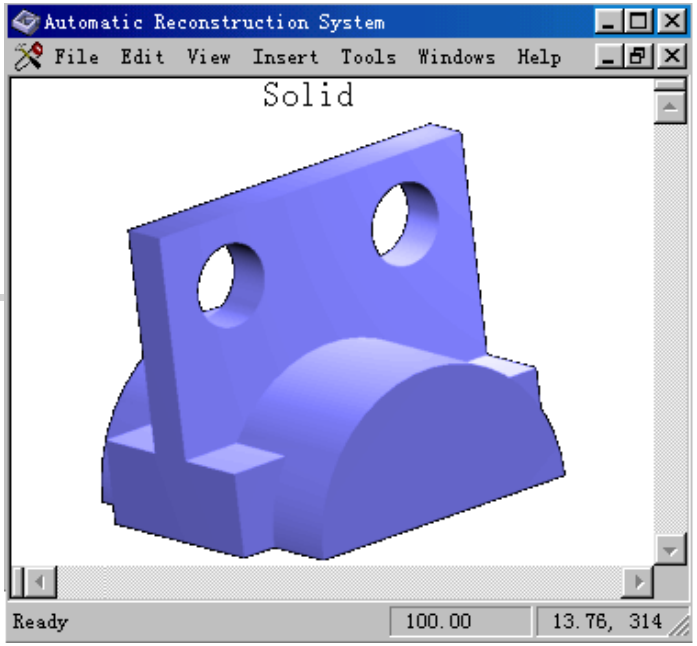
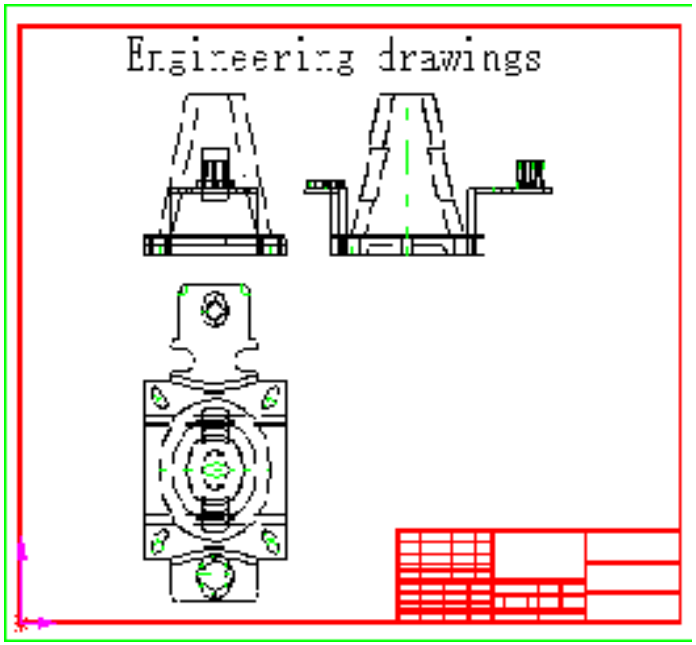
第四章
二维图形的生成技术

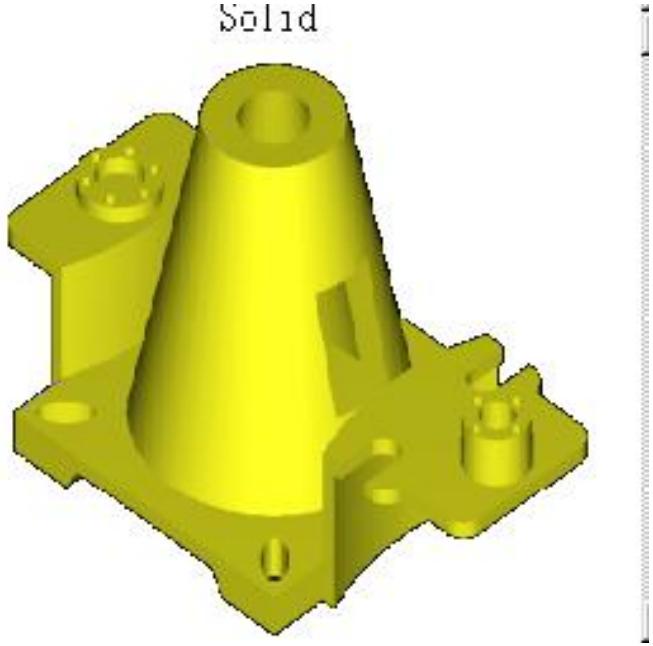
谢忠红
南京农业大学

复杂图形

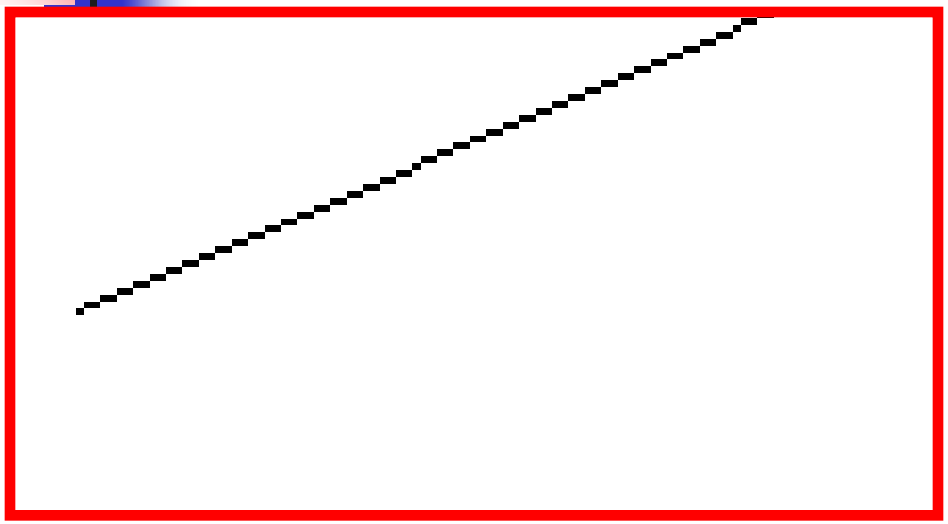

Automatic Reconstruction System
File Edit View Insert Tools Windows Help
Wireframe
Ready
100.00
13.76, 314

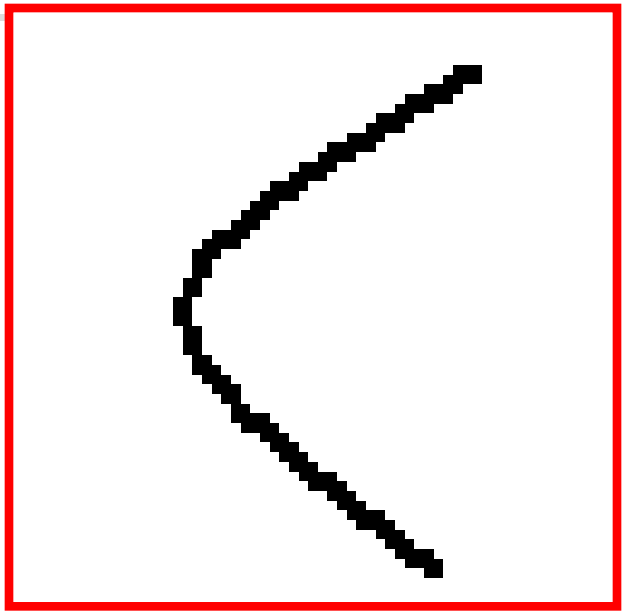

Automatic Reconstruction System
File Edit View Insert Tools Windows Help
Solid
Ready
100.00
13.76, 314

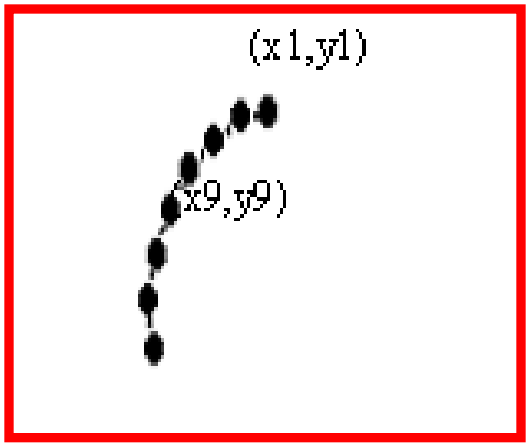

Engineering drawings


Solid

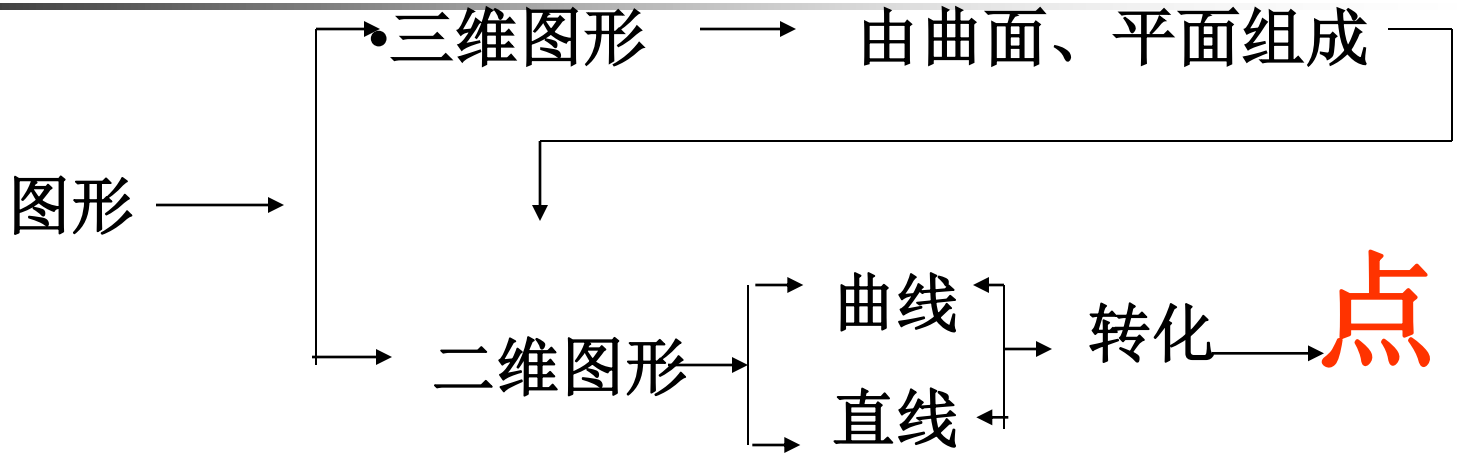
直线曲线都是点的集合

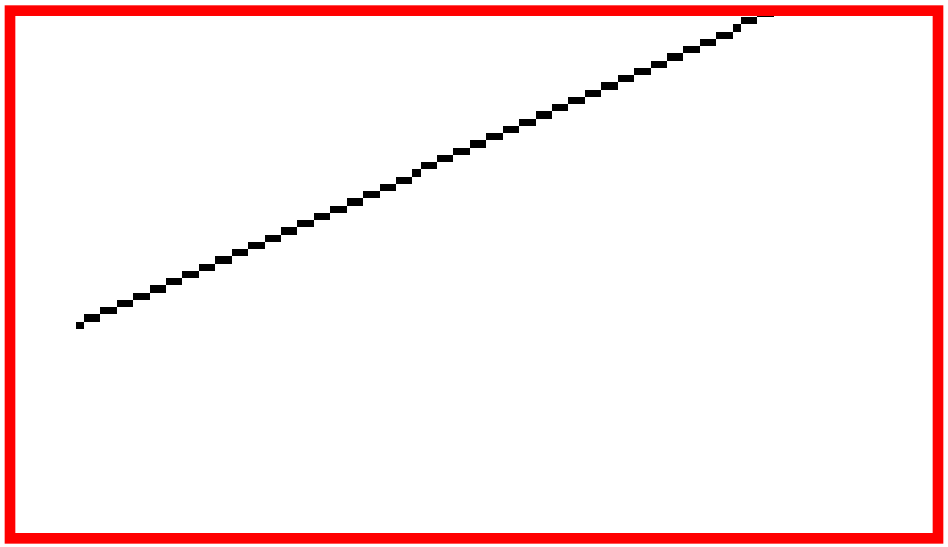






4.1基本图元素-点

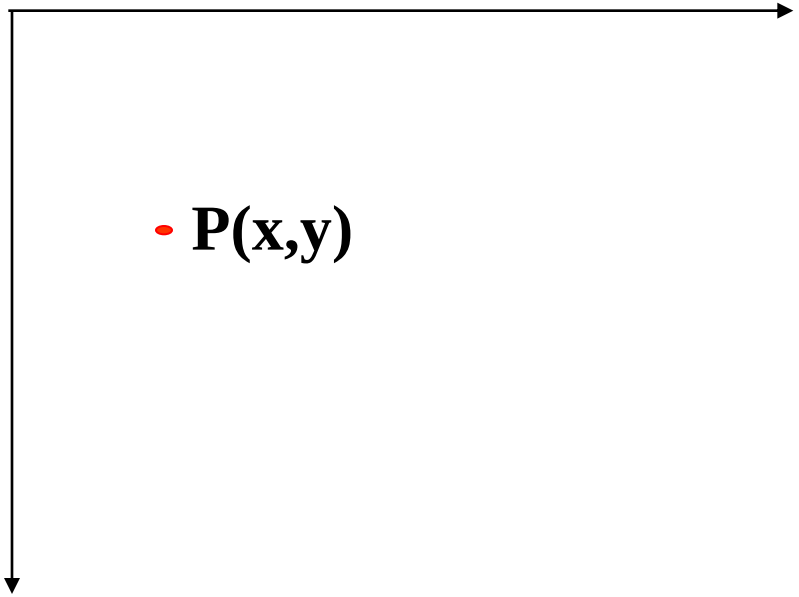




- 点是图形中最基本的图素，直线、曲线以及其它图元都是点的集合。

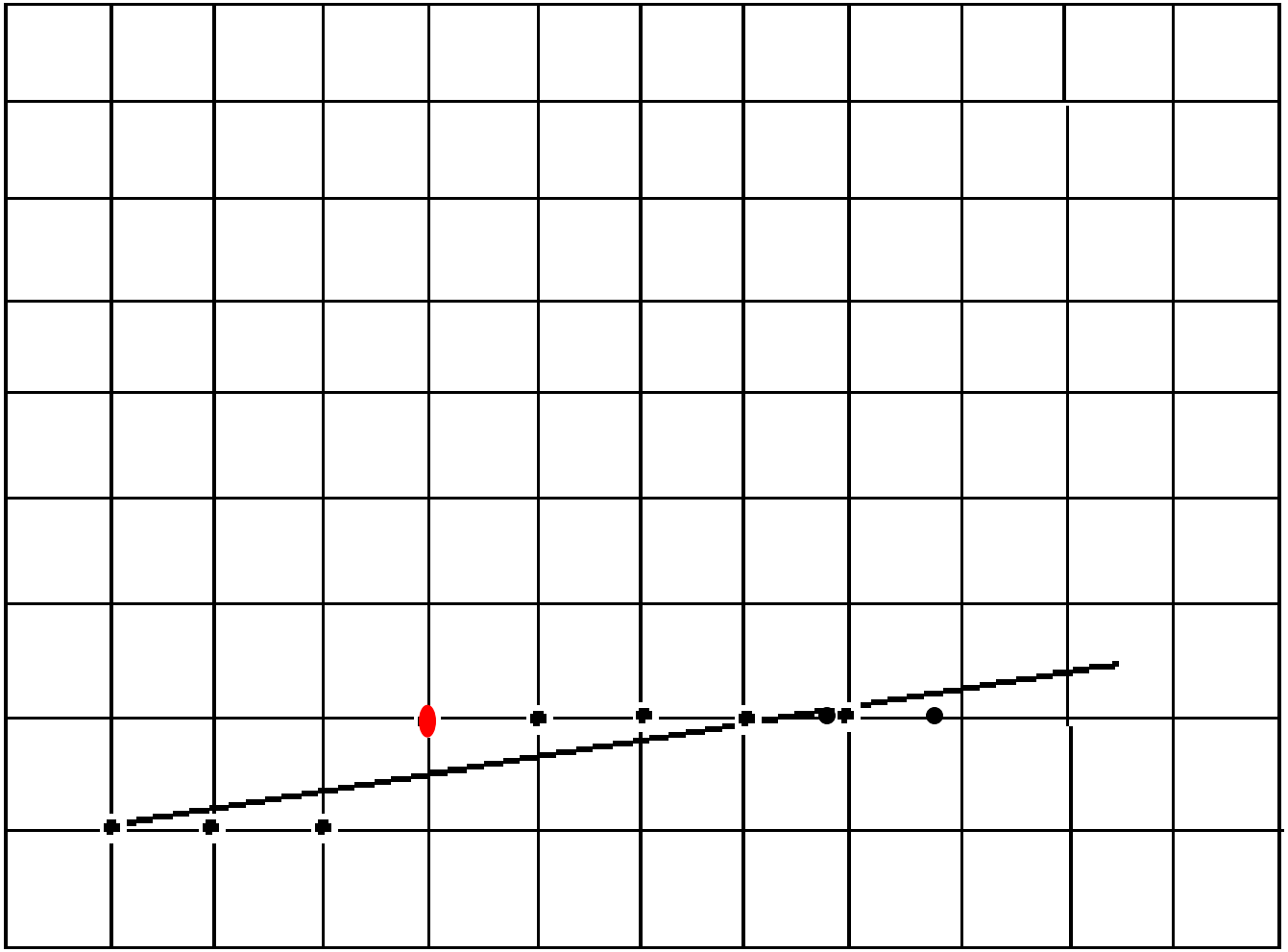
putpixel(x,y,color)

- 函数作用：使屏幕上坐标(x,y)的点辉亮



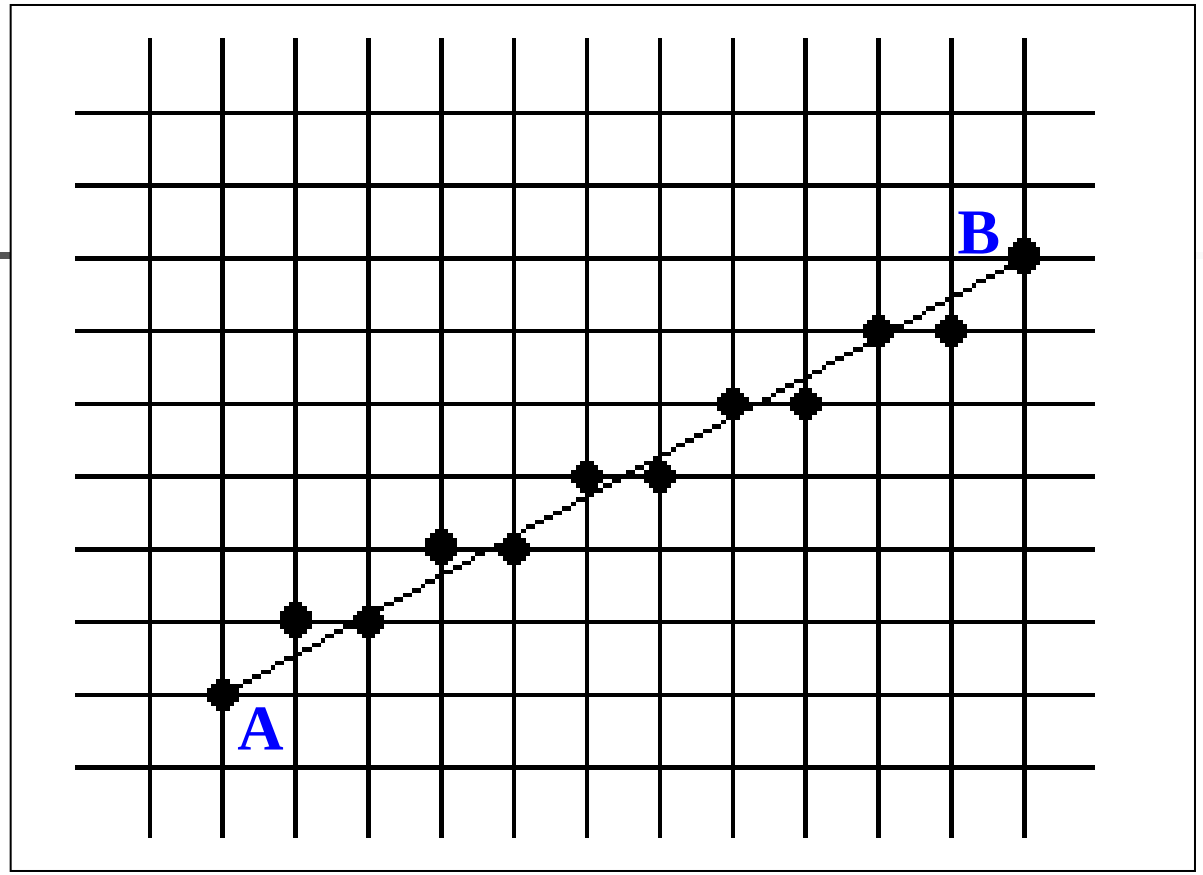
光栅显示器像素间为均匀网格

- **Putpixel (5 , 7 , RED)**



直线是点的集合，（几何学定义：直线被定义为两个点之间的最小距离）

- **图形学的直线：**
- **（1）图形显示器是由一个个排列有序的象素点构成的，象素之间为均匀的网格**
- **（2）点的坐标是指点所在象素点的位置，而象素点的位置是整型的，画一条直线实际上是依据一系列计算出来的位置得到靠近该位置的象素点。**



对于光栅显示器来说：

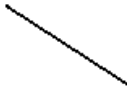
1、像素间为均匀网格

2、整型坐标系

(10.48, 20.51)→(10, 21)

- 后果：对坐标取整使得显示的线段具有锯齿现象
- 改进的方法：提高屏幕的分辨率

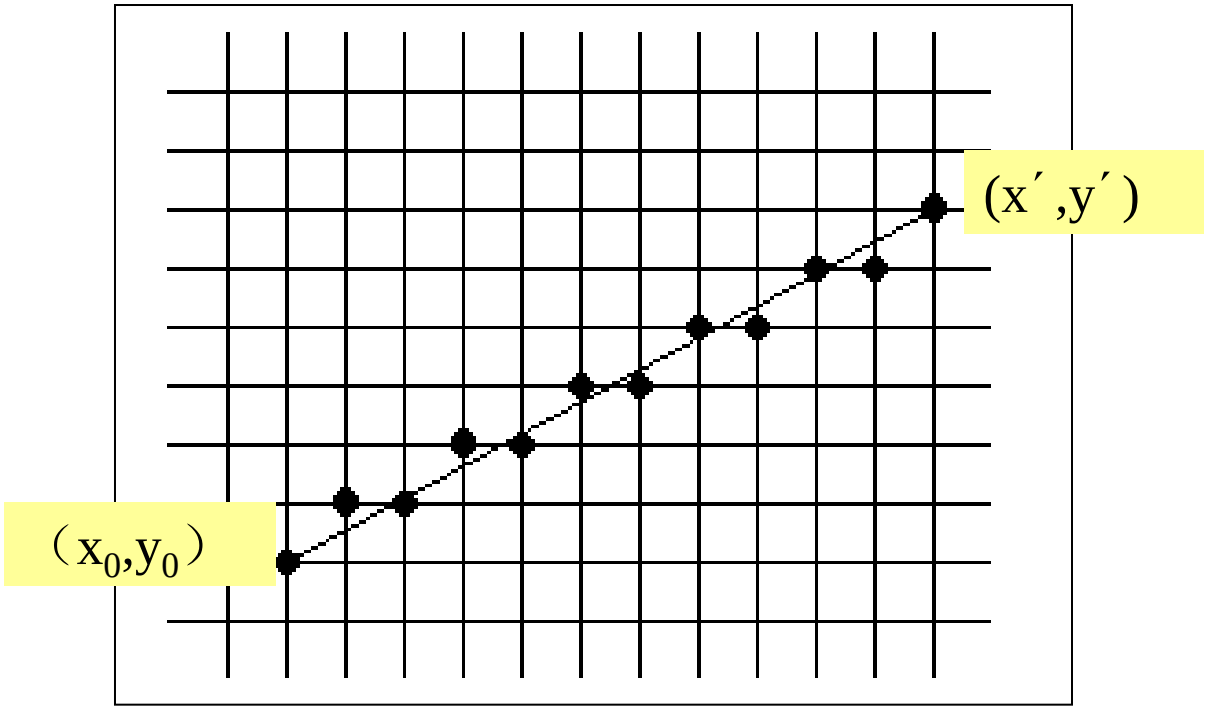
800*600

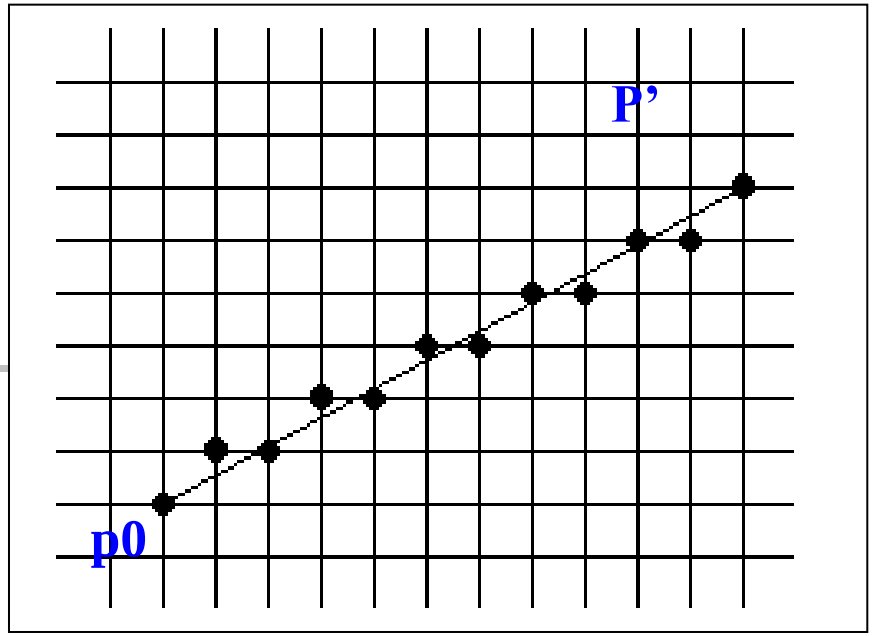


1024*1024

4.2直线段的生成

- 1.数值微分法(DDA)基本思想:
- 确定最佳逼近该直线的一组象素，并按扫描线的顺
- 序显示这组象素点.





- 已知条件：

待连线的两点：

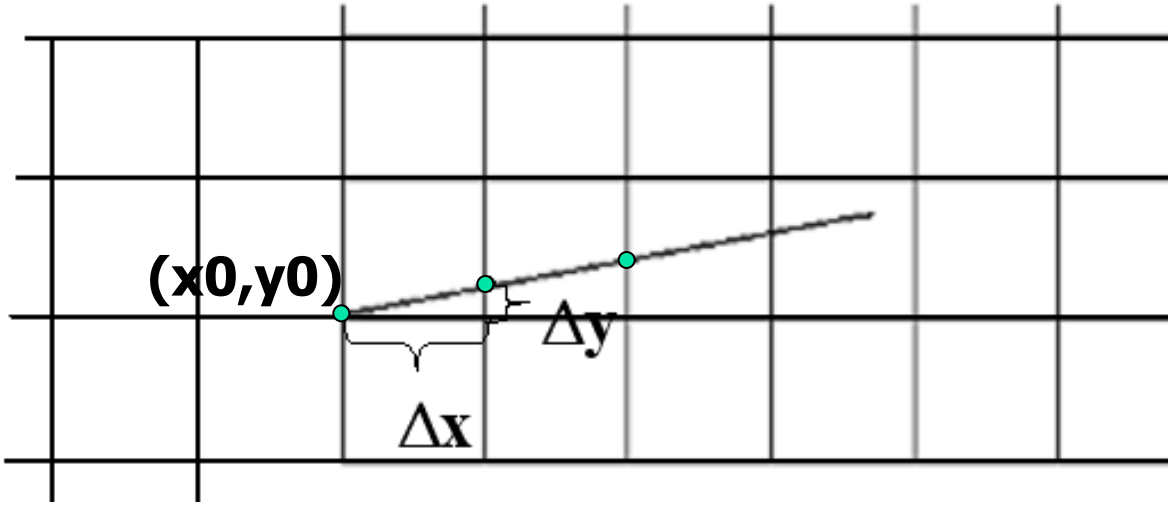
$$P0\ (x0, y0) \quad , \quad P'(x', y')$$

直线斜率为：

$$k = \frac{y' - y_0}{x' - x_0}$$

$$K = \Delta y / \Delta x$$

$$\Delta y = k \Delta x$$



$x_1 = x_0 + \Delta x$, $x_2 = x_1 + \Delta x$, …, $x_i = x_{i-1} + \Delta x$

$y_1 = y_0 + k \Delta x$, $y_2 = y_1 + k\Delta x$, …, $y_i = y_{i-1} + k \Delta x$

$$x_i = x_{i-1} + \Delta x$$

$$y_i = y_{i-1} + k\,\Delta x$$

设：$\Delta x = 1$

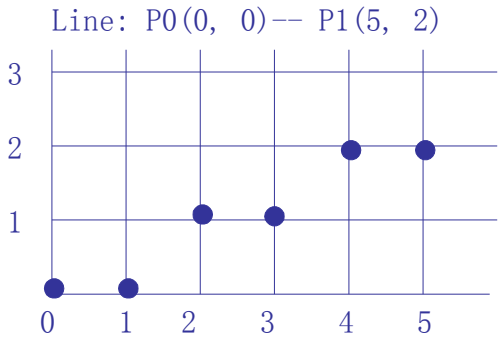
$$x_i = x_{i-1} + 1$$

$$y_i = y_{i-1} + k \quad (k<=1)$$

点的坐标：　(xi,　round(yi)　)

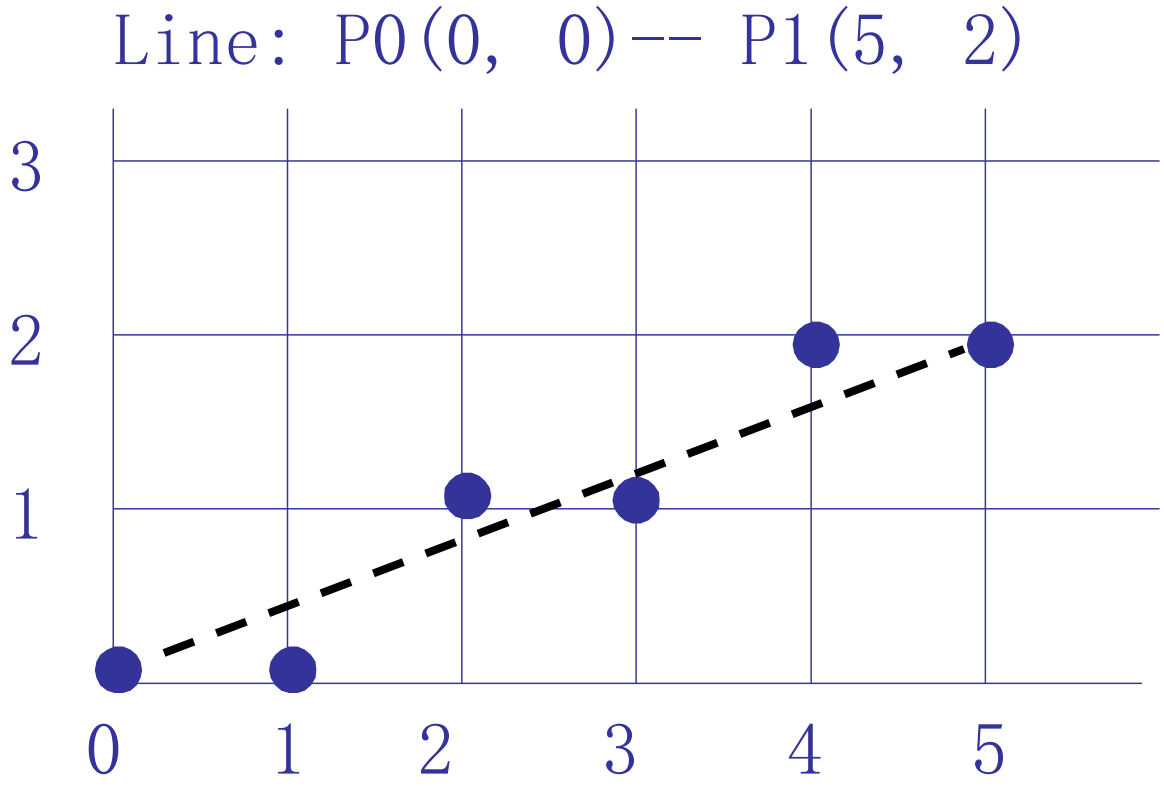
•即：当x每递增1，y递增k(即直线斜率)；

例：用DDA算法画直线段P0(0,0)--P1(5,2)

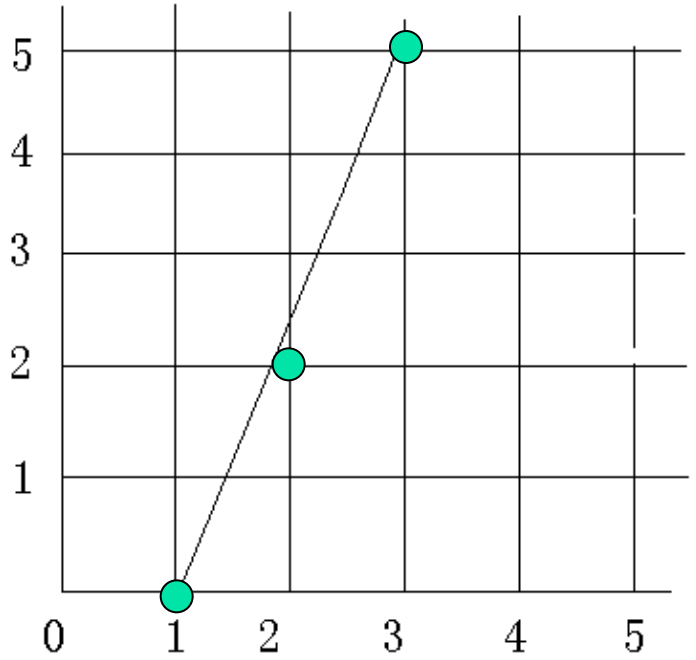


k=(2-0)/(5-0)=0.4

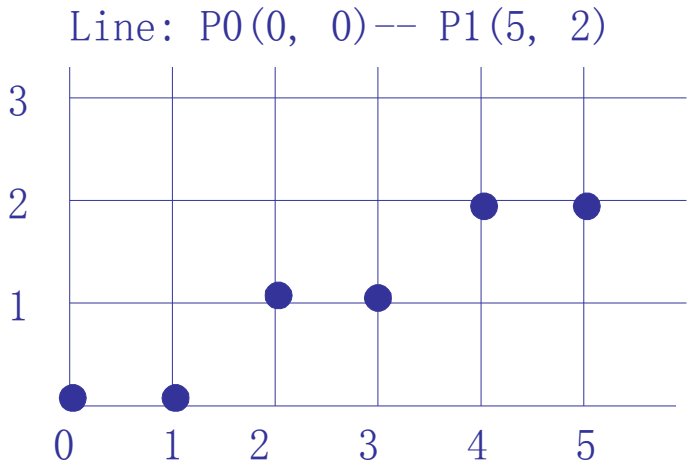
i	$X_i=x_{i-1}+1$	$y_i=y_{i-1}+k$	round (y_i)	坐标
0	0	0	round(0)=0	(0,0)
1	1	0+0.4=0.4	round(0.4)=0	(1,0)
2	2	0.4+0.4=0.8	round(0.8)=1	(2,1)
3	3	0.8+0.4=1.2	round(1.2)=1	(3,1)
4	4	1.2+0.4=1.6	round(1.6)=2	(4,2)
5	5	1.6+0.4=2.0	round(2.0)=2	(5,2)

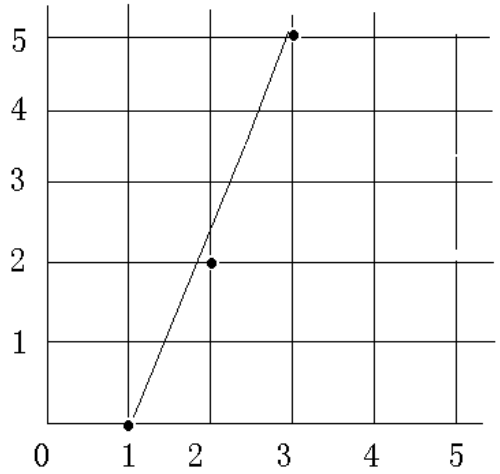

Line: P0(0, 0)-- P1(5, 2)
3
2
1
0
1
2
3
4
5

该方法能画出
所有直线吗？



- 分析：
- 上述分析的算法仅适用于|k| ≤1的情形。
- 原因：
- 当|k|<=1时，x每增加1(Δx=1),y最多增加1(|Δy|=|k|x ≤ 1)。如果，
- 当 |k| >1时， x每增加1，y可能会增加大于1的数(|Δy|=|k| Δx>1)，这时象素点会稀疏，所画直线会断断续续。

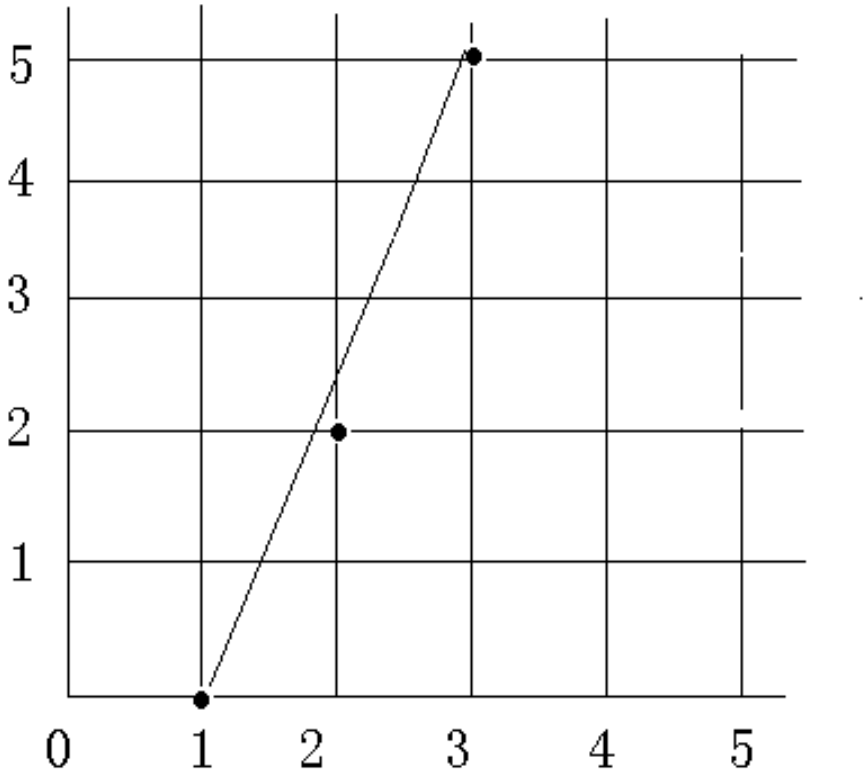




K>1时使用如下公式的效果图

$$x_i = x_{i-1} + 1$$

$$y_i = y_{i-1} + k \quad (k>1)$$



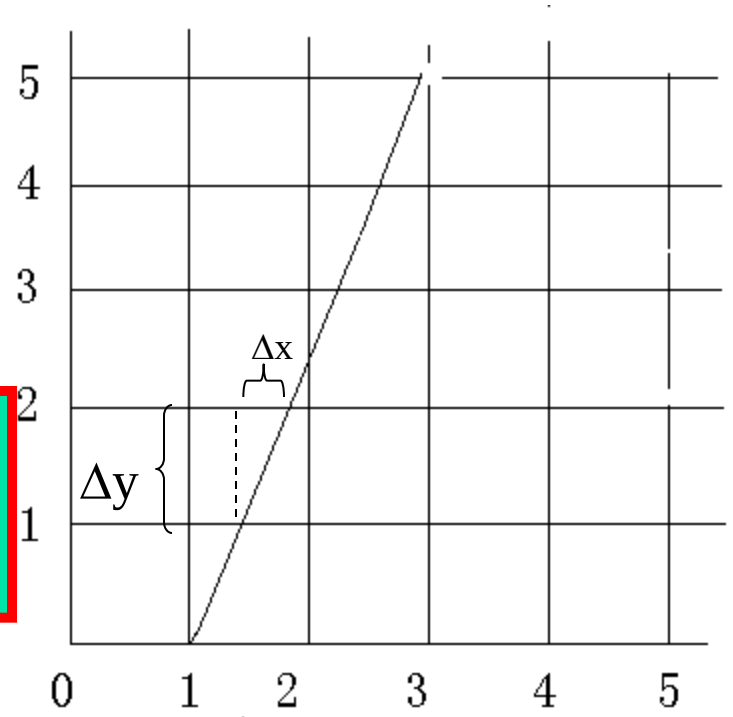

K>1时怎么办?

已知直线　y=kx+b　　|k|>1

$$K = \Delta y / \Delta x$$

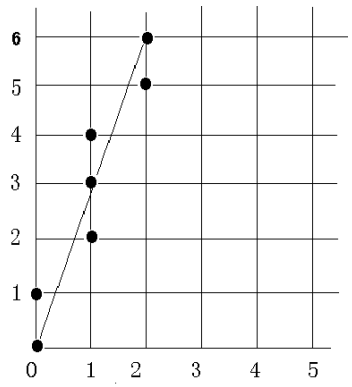
令 $\begin{cases} \Delta y = 1 \\ \Delta x = \Delta y * 1/k \end{cases}$

即: $\begin{cases} y_i = y_{i-1} + 1 \\ x_i = x_{i-1} + 1/k \end{cases}$



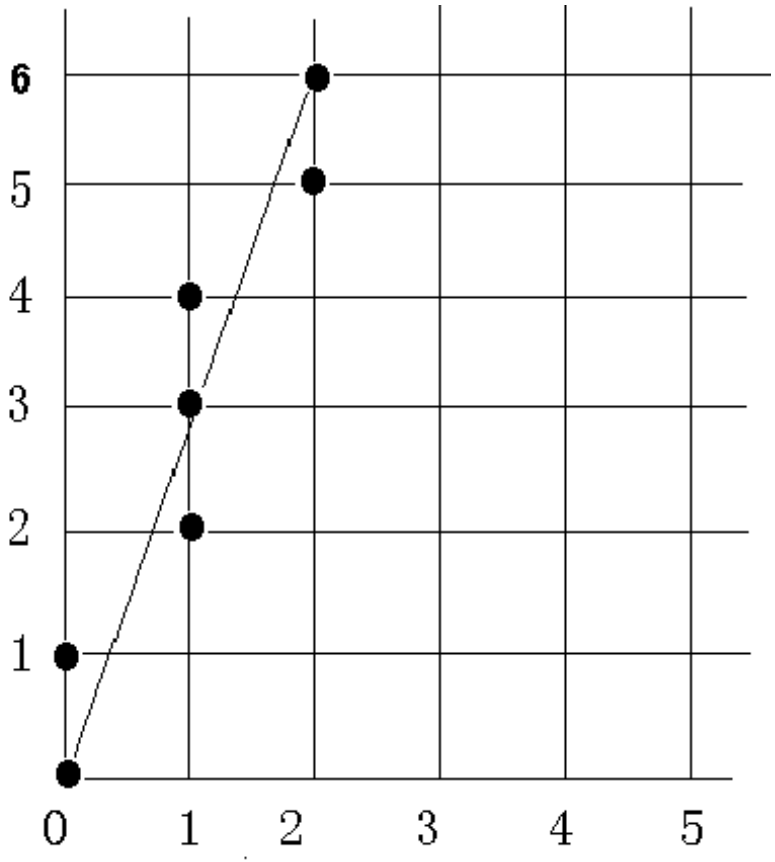
(round (xi), yi)

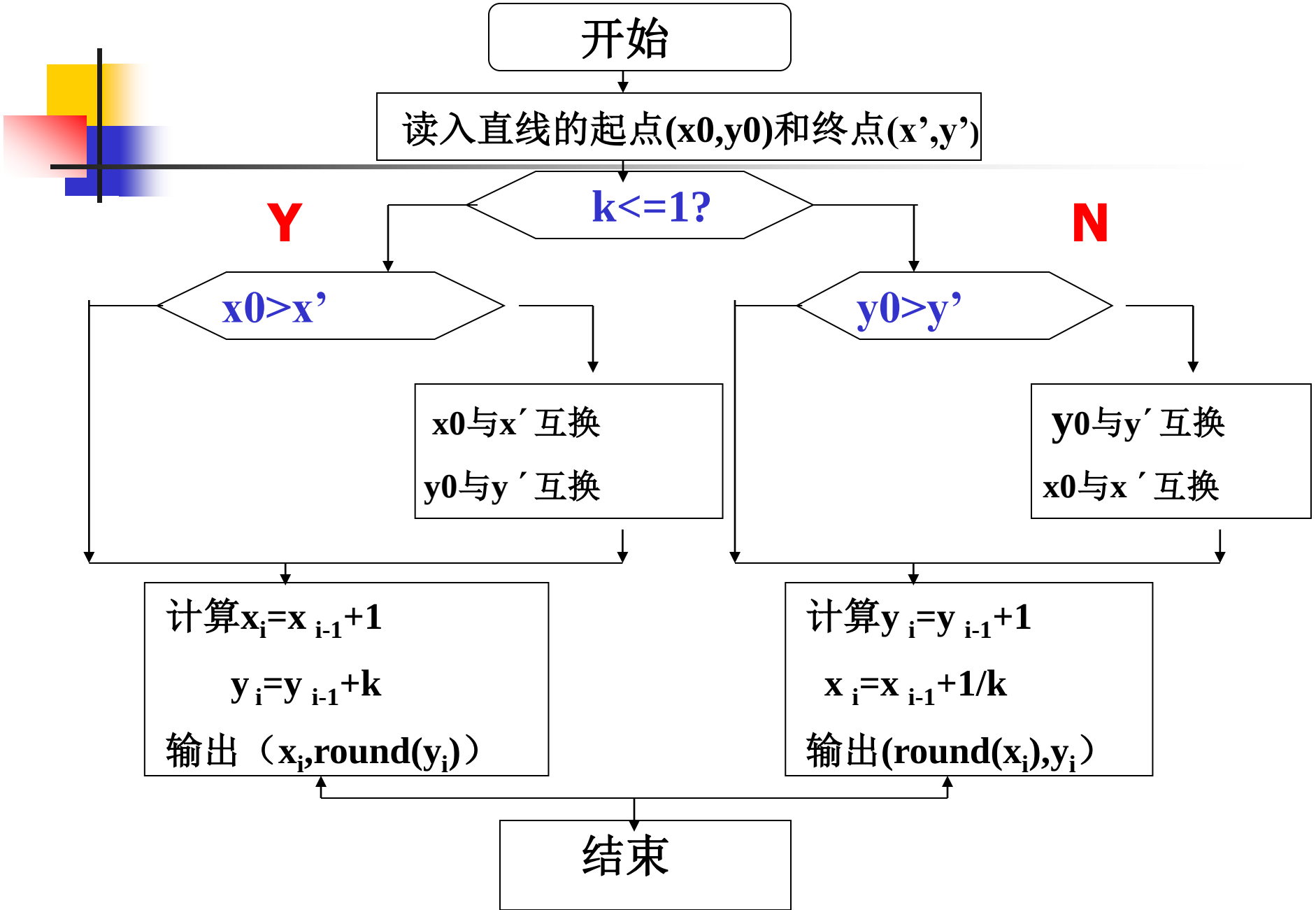
- 例：画出p0(0,0)和p′(2,6)之间的一条直线。
- K=(6-0)/(2-0)=3>1
- 采用通过yi的变化确定xi的变化
- 即： Δy =1 Δx=1/k
- 1/k=1/3=0.33

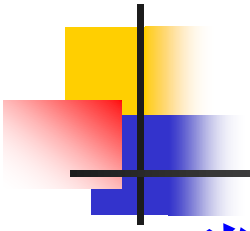


i	yi=yi-1+1	xi=xi-1+1/k	round (xi)	坐标
0	0	0	round(0)=0	(0,0)
1	1	0+0.33=0.33	round(0.33)=0	(0,1)
2	2	0.33+0.33=0.66	round(0.66)=1	(1,2)
3	3	0.66+0.33=0.99	round(0.99)=1	(1,3)
4	4	0.99+0.33=1.32	round(1.32)=1	(1,4)
5	5	1.32+0.33=1.65	round(1.65)=2	(2,5)
6	6	1.65+0.33=1.98	round(1.98)=2	(2,6)

Line(0 , 0 , 2 , 6)

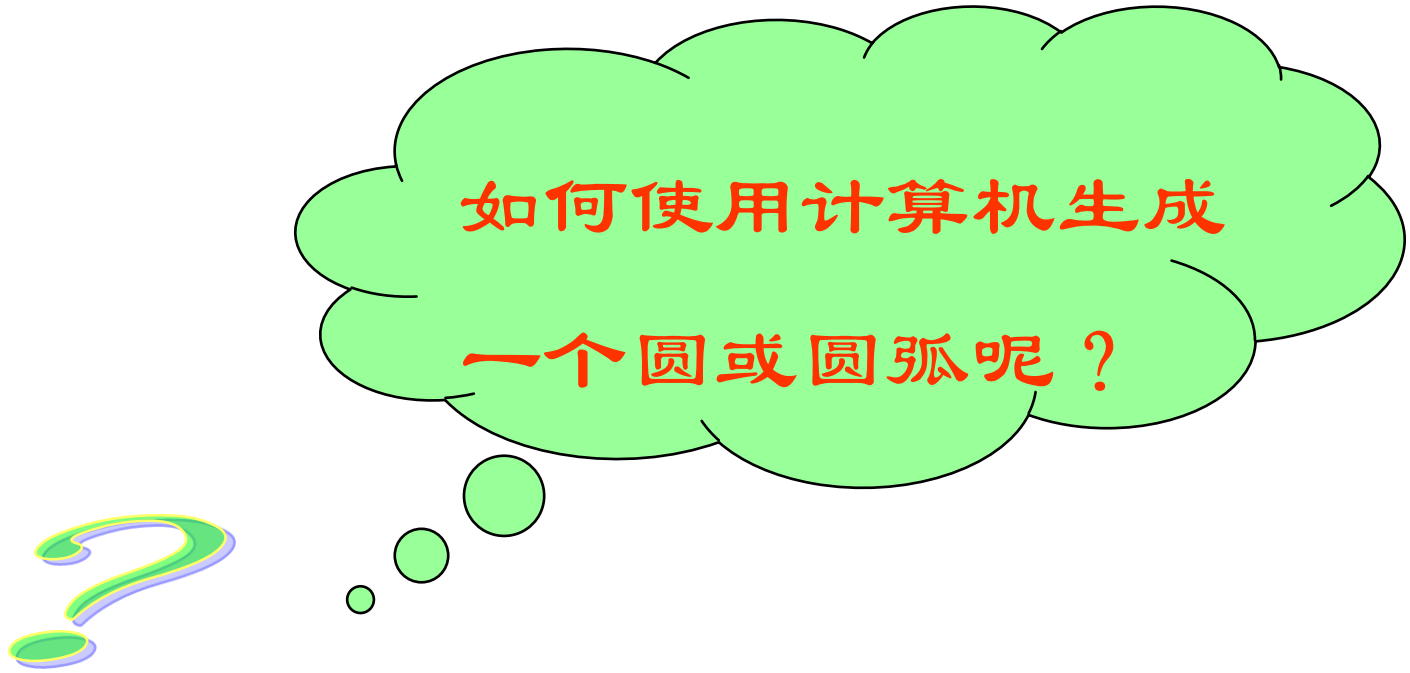



开始
读入直线的起点(x0,y0)和终点(x',y')
k<=1?
Y
N
x0>x'
y0>y'
x0与x′互换
y0与y′互换
y0与y′互换
x0与x′互换
计算x_i=x_i-1+1
y_i=y_i-1+k
输出（x_i,round(y_i)）
计算y_i=y_i-1+1
x_i=x_i-1+1/k
输出(round(x_i),y_i）
结束



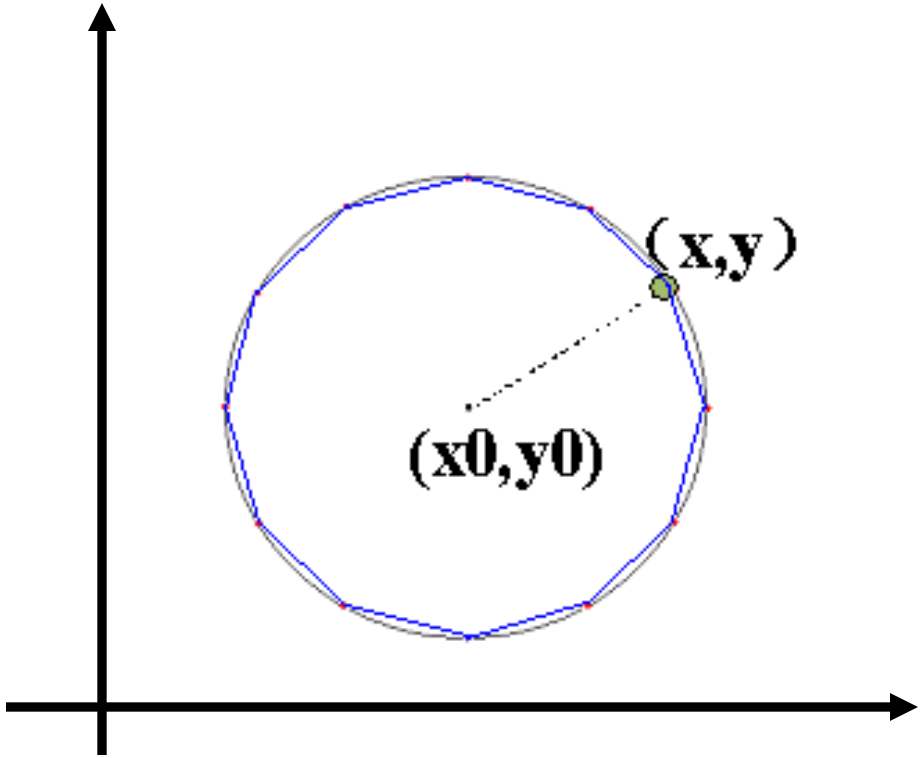
- 数值微分(DDA)法缺点：
- 在此算法中，y、k必须是float，且每一步都必须对y进行舍入取整，不利于硬件实现。

DDA圆的生成算法



角度DDA法生成圆

原理： 把圆或者圆弧分成N等分。用N段相邻的直线来逼近圆弧或圆。



圆心在 （x0,y0）

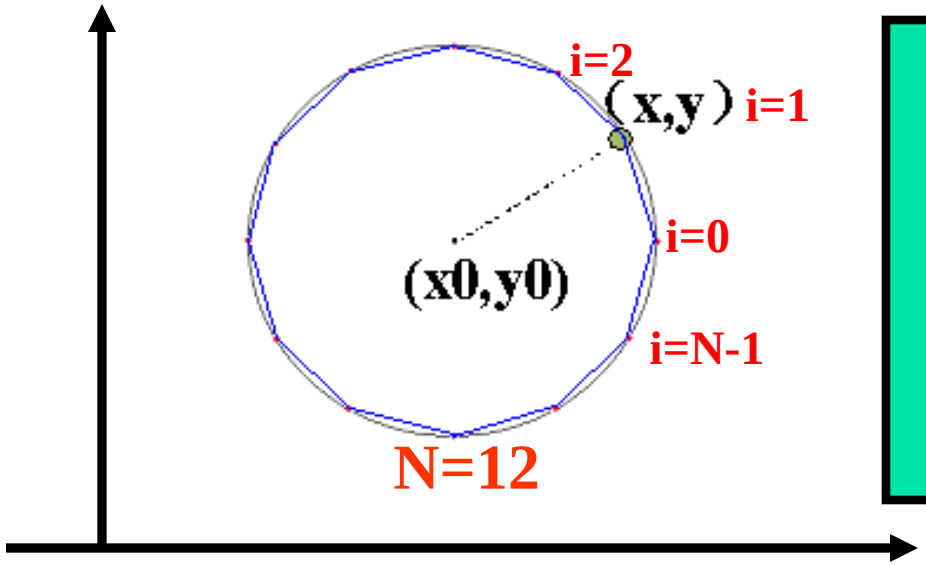
半径为 R

圆的参数方程

$$\begin{cases} x = x_0 + R\cos\theta \\ y = y_0 + R\sin\theta \end{cases}$$

若将整个圆分成N等分，$\Delta\theta=2\pi/N$，则有参数式递推公式：

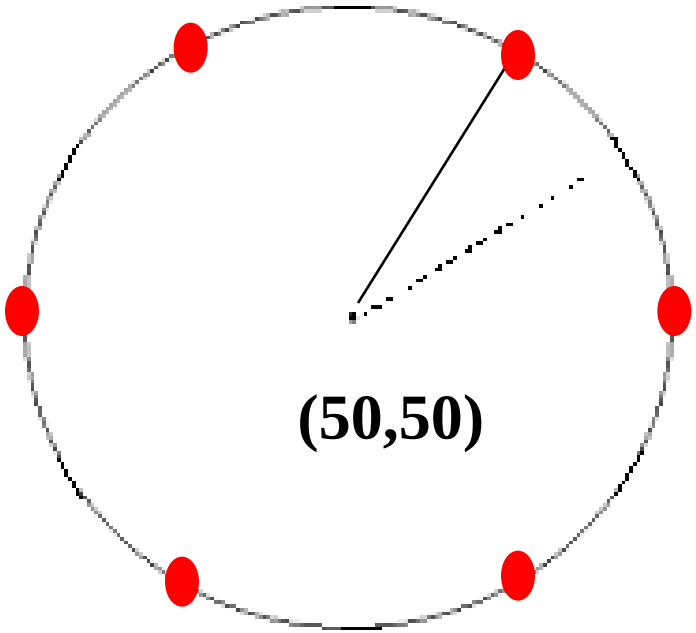
- 其中$\theta_i= i*\Delta\theta \quad i=0,1,2,3,\ldots N-1$



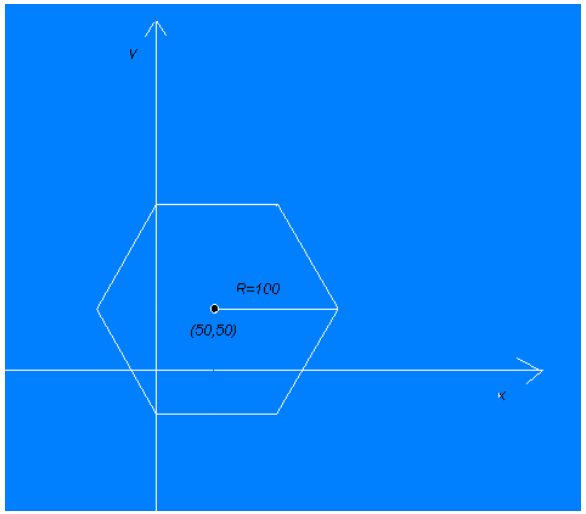
$$\begin{cases} X_i=x_0+R\cos\theta_i \\ Y_i=y_0+R\sin\theta_i \end{cases}$$

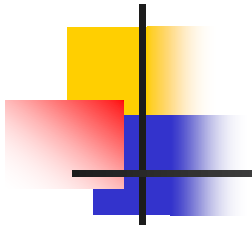
- 最后将相邻的（x_i,y_i）（x_{i+1},y_{i+1}）用直线段一一相连就可近似地画出圆了。

- 例：圆心$(x_0,y_0)=(50,50)$
- R=100　N=6
- $\Delta\theta = 2\pi/6 = \pi/3$



- $x_0=50+100\cos(0*\pi/3)$
  $y_0=50+100\sin(0*\pi/3)$
- $x_1=50+100\cos(1*\pi/3)$
  $y_1=50+100\sin(1*\pi/3)$
- $x_2=50+100\cos(2*\pi/3)$
  $y_2=50+100\sin(2*\pi/3)$
- $x_3=50+100\cos(3*\pi/3)$
  $y_3=50+100\sin(3*\pi/3)$
- $x_4=50+100\cos(4*\pi/3)$
  $y_4=50+100\sin(4*\pi/3)$
- $x_5=50+100\cos(5*\pi/3)$
  $y_5=50+100\sin(5*\pi/3)$
- $x_6=50+100\cos(6*\pi/3)$
  $y_6=50+100\sin(6*\pi/3)$
- [P DDA_circle]





- 请写出DDA画圆算法！

```
void DDA_circle(int n ,int x0,int y0,int radius)
  // 把圆分成n等分，(x0,y0)为圆心坐标，radius为圆的半径
  {int i;
  int x_start, y_start ;   //线段的起点坐标
  Int  x_end,y_end;        //线段的中点坐标
   for(i=0;i<n;i++)//循环地算初起点和终点坐标并连接这两点
     {
        x_start=x0+radius*cos(i*2*3.14/n) ;
        y_start=y0+radius*sin(i*2*3.14/n)  ;
        x_end=x0+radius*cos((i+1)*2*3.14/n) ;
        y_end=y0+radius*sin((i+1)*2*3.14/n) ;
        my_line(x_start,y_start,x_end,y_end) ;
     }

  }
```

还能改进吗？ ？

```
void DDA_circle(int n ,int x0,int y0,int radius)
  // 把圆分成n等分，(x0,y0)为圆心坐标，radius为圆的半径
 {int i;
   Int  x_end,y_end;        //线段的中点坐标
   my_Moveto (x0+radius, y0)
   for(i=1;i<=n;i++)
 //循环地算终点坐标并连接这两点
    {
        x_end=x0+radius*cos(i*2*3.14/n) ;
        y_end=y0+radius*sin(i*2*3.14/n) ;
        my_lineto(x_end,y_end) ;
    }
   }
```

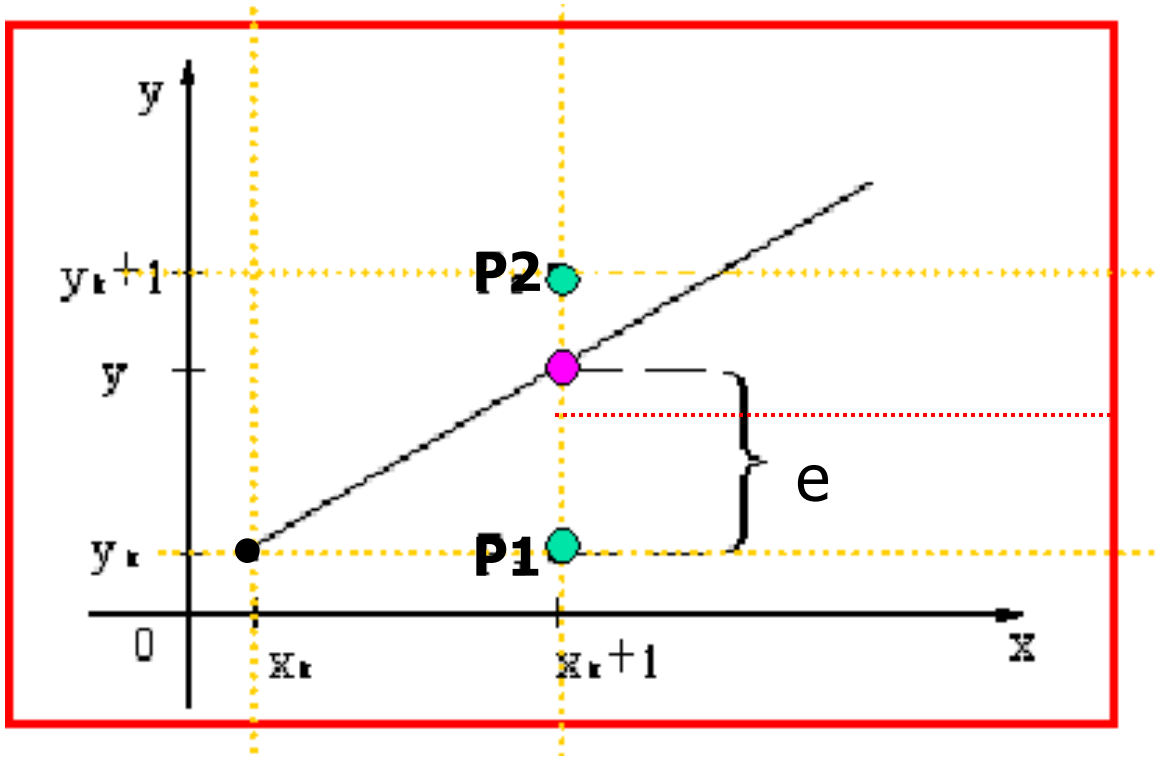
直线Bresenham算法

- 特点：
- -使用最广泛
- -由误差项符号决定下一个象素取正右方像素还是右上方像素
- 优点
  - 整数运算，速度快
  - 精度高

基本思想

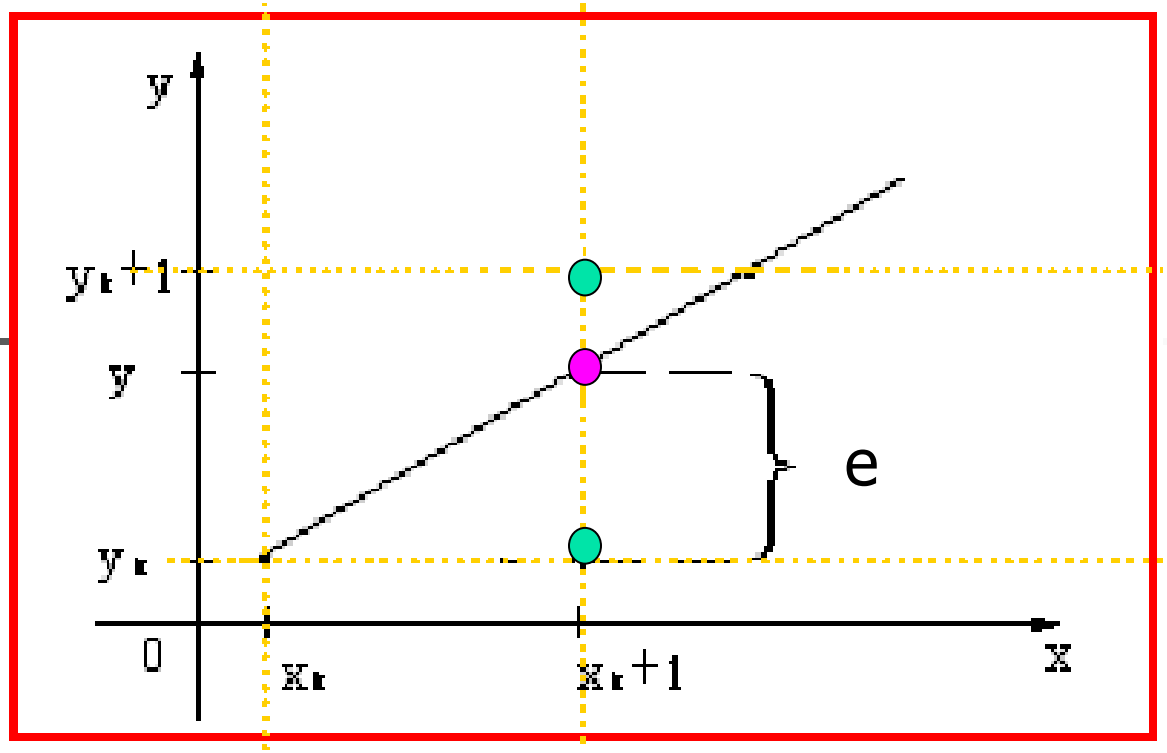
根据理想直线到位于直线下方的像素的距离e, 确定与理想直线最近的象素

以K<=1为例



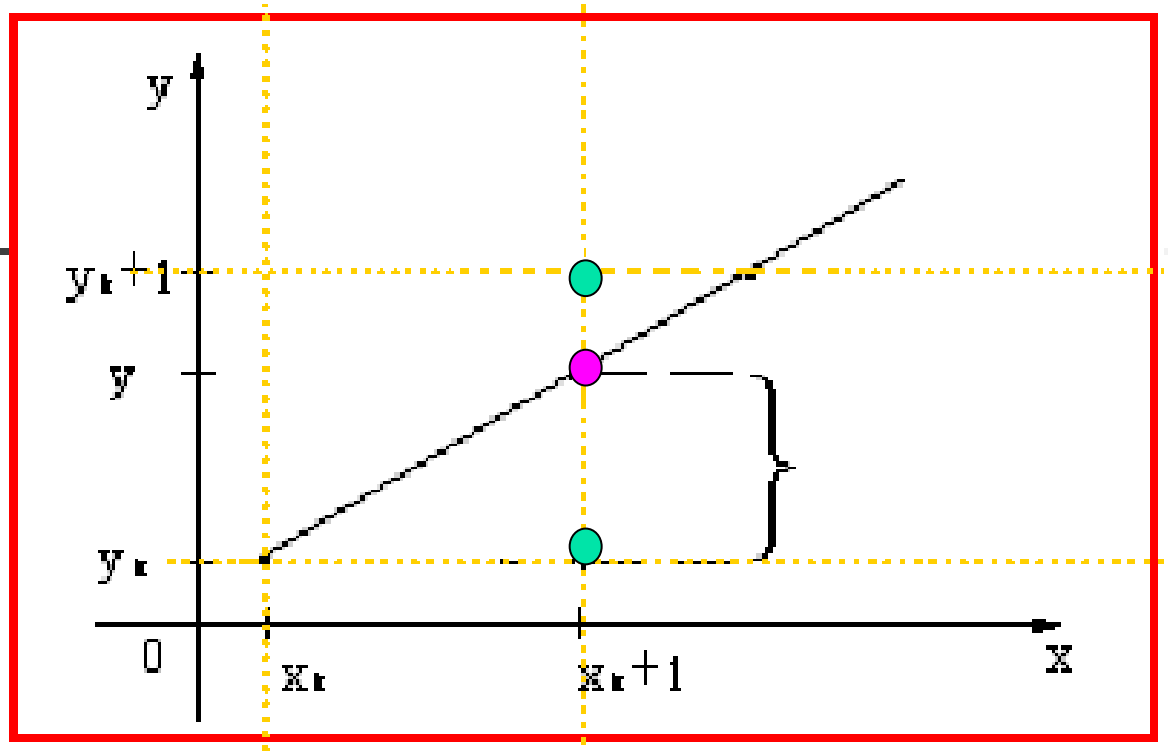
e>0.5

e>0.5



- *X方向：每次走一步* $x_{i+1}=x_i+1$
- y方向：走步与否取决于误差e值的大小

- 如果 e≥0.5时，最接近P2(x_i+1,y_i+1) y方向走一步
- 如果e<0.5时，最接近P1(x_i+1,y_i) y方向不走步

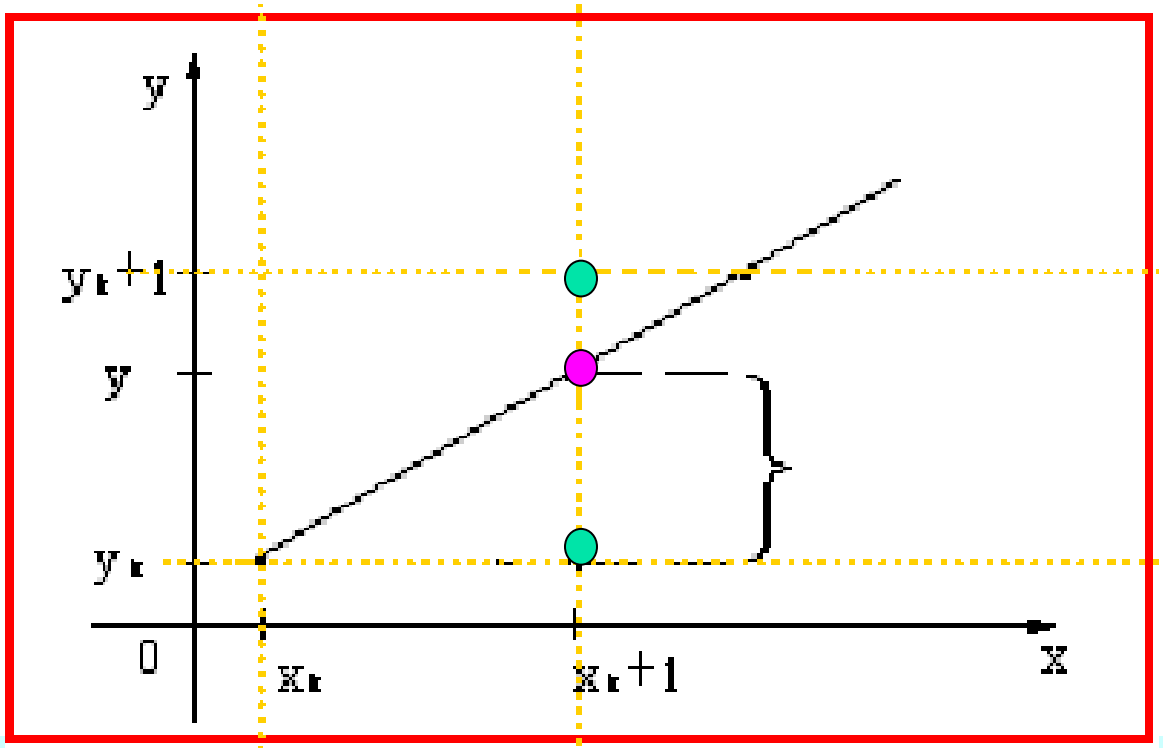


- 误差e计算
- 从第一点开始：

$e = \Delta y / \Delta x$

- 为方便与0比较，设 **S=e-0.5**

S=Δy/ Δx-0.5



当s≥0时， 最接近P2(xi+1,yi+1)y方向走一步

当s<0时， 最接近P1(xi+1,yi)y方向不走步

缺点：有除法，不宜硬件实现

改进

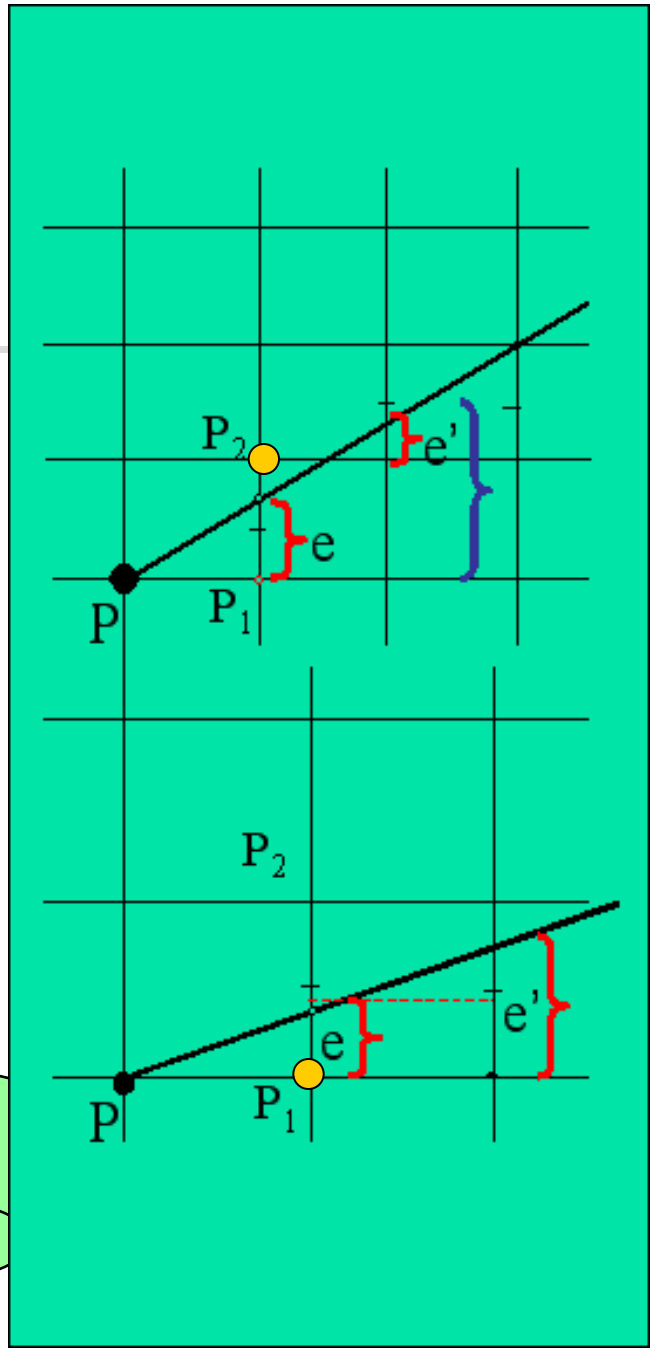
- $s=\Delta y/\Delta x-0.5$
- $=(2\Delta y-\Delta x)/(2\Delta x)$
- 与0比较大小

简化s与0的比较，拿分母与0比较

设 $t0=2\Delta y-\Delta x$

当t0≥0时，
最接近P2(xi+1,yi+1)
y方向走一步
当t0<0时，
最接近P1(
y方向不走

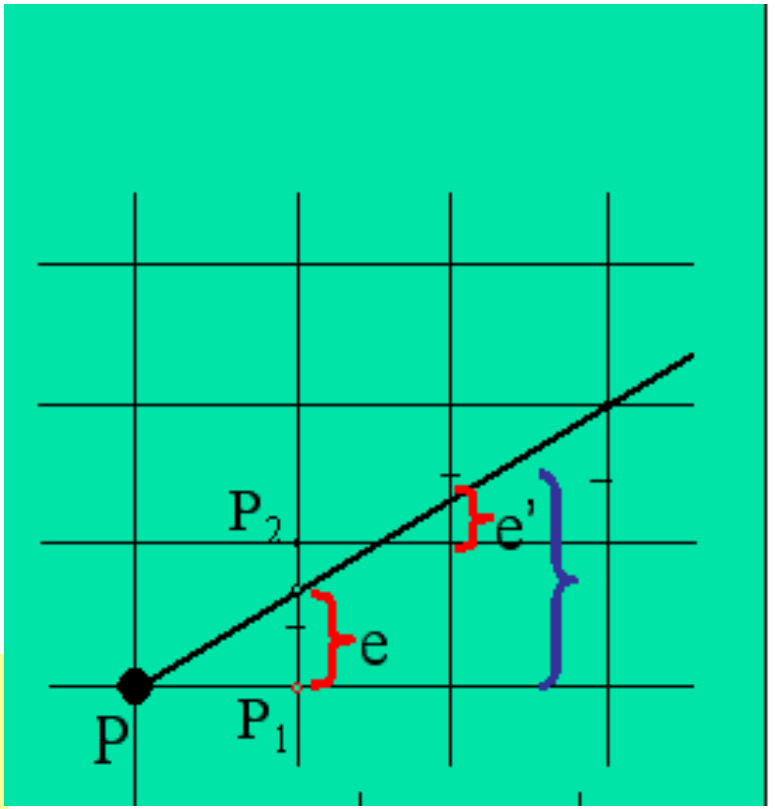
下一个点如何判断呢？



设 t0= 2Δy - Δx >=0

- 上一步误差

- e= Δy/ Δx



- *计算下一步误差的计算*
- 当t_0>=0时，y方向走一步

e'=2e-1 = 2Δy/ Δx - 1

- 同样为了方便与0比较
- e'-0.5=2Δy/ Δx - 1 -0.5
- =(2Δy-Δx+2Δy-2Δx) /(2Δx)

- 令分母为 t1= t0+ 2Δy−2Δx

设 t0= $2\Delta y - \Delta x$<0

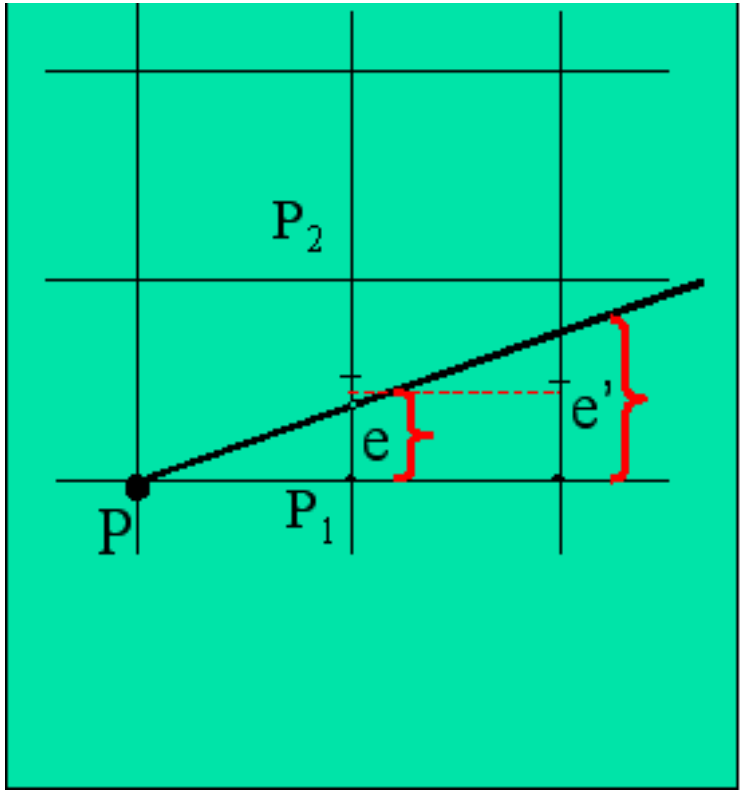
- 当t_0<0时，y方向不走步

- $e' = 2e = 2\Delta y / \Delta x$

- 同样为了方便与0比较
- $e' - 0.5 = 2\Delta y / \Delta x - 0.5$

$= (2\Delta y - \Delta x + 2\Delta y)/(2\Delta x)$

令分母为t1=t0+ $+2\Delta y$



- 综上　$\Delta x=1$　　$\Delta y=k$
- 　　$t0=2\Delta y-\Delta x=2K-1$

当$t_i \geq 0$时，y方向走一步

$$t_{i+1}=t_i+2\Delta y-2\Delta x$$
$$=t_i+2k-2$$

当$t_i<0$时，y方向不走步

$$t_{i+1}=t_i+2\Delta y$$
$$=t_i+2k$$

- **例：** 利用**Bresenham**算法画出**p0(0,0)**和**p′(6,2)**之间的一条直线。

  解答 **K=(2-0)/(6-0)=1/3<=1**

- **p0(0,0)**

t0= *2k -1*

当$t_i \geqslant 0$时$t_{i+1}=t_i+2k-2$

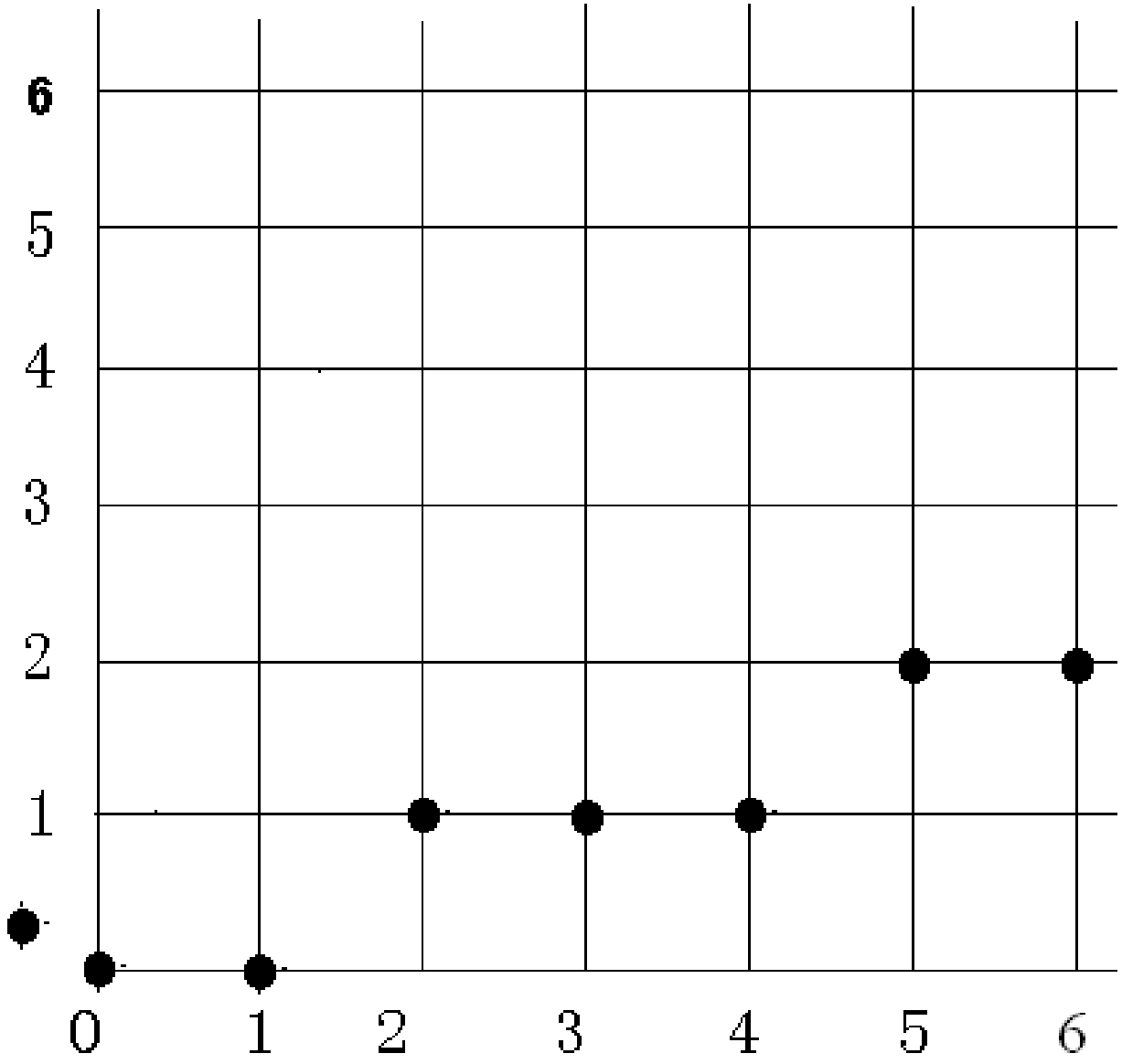
当ti<0时$t_{i+1}=t_i+2k$

• 起点(0,0)	ti	坐标
	t0= 2k-1 =-1/3<0	(1,0)
∵ t0<0	∴ t1=t0 +2k =1/3>0	(2,1)
∵ t1>0	∴ t2=t1+2k-2=-1<0	(3,1)
∵ t2<0	∴ t3=t2+2k=-1/3<0	(4,1)
∵ t3<0	∴ t4=t3+2k=1/3>0	(5,2)
∵ t4>0	∴ t5=t4+2k-2=-1<0	(6,2)

- **例：** 利用**Bresenham**算法画出**p0(0,0)**和**p′(6,2)**之间的一条直线。

解

- **p0(**
  -



例： 利用Bresenham算法画出p0(3, 3)和p′(7, 9)之间的一条直线。 **解答** K=(9-3)/(7-3)=6/5>=1

1/k=4/6

p0(3,3)

t0= *2(1/k) -1=2/6*

当$t_i \geqslant 0$时$t_{i+1}=t_i+2(1/k)-2$

当ti<0时$t_{i+1}=t_i+2(1/k)$

起点(3,3)		ti	坐标
		t0= 2(1/k)-1 =2/6 ⩾0	(4,4)
∵	t0⩾0	∴ t1=t0 +2(1/k)-2 =-2/6<0	(4,5)
∵	t1<0	∴ t2=t1+2(1/k)=1 ⩾0	(5,6)
∵	t2⩾0	∴ t3=t2+2(1/k)-2=2/6⩾0	(6,7)
∵	t3⩾0	∴ t4=t3+2(1/k)-2=-2/6<0	(6,8)
∵	t4<0	∴ t5=t4+2(1/k)=10 ⩾0	(7,9)