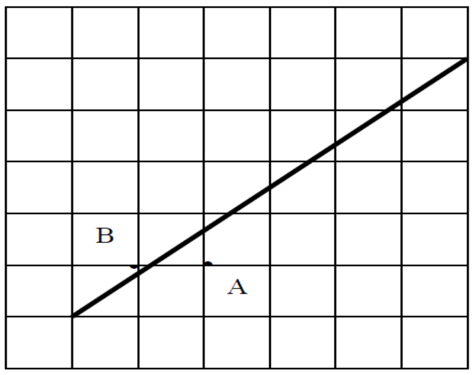
Arbitrary Lines

Drawing lines with arbitrary slope creates several problems, such as:

1- The display screen can be illuminated only at pixels locations; therefore a raster scan display has a staircase effect that only approximates the actual line as shown in figure below:



Although it may not be possible to choose pixels that lie on the actual line, we want to turn on pixels lying closest to it. For example, in previous figure, the pixel at location B is a better choice than the one at location A.

2- Determining the closest (best) pixels is not easy.

Different algorithms calculate different pixels to make up the approximating line.

The choice of algorithm depends on:
1- The speed of line generation.
2- The appearance of the line.
Therefore, to understand these criteria better let's look at several different line generating algorithms.

1- Direct method:

In this method, we learn how to draw a line between two points by drawing a group of pixels using the command plot (x , y , color), with substituting in straight line equation:

$$Y = m \times X + b$$

Where (m) is the slope and (b) is a constant which represents the clipping from y-axis (y-intercept).

$$m = \frac{yend - ystart}{xend - xstart} \quad , b = ystart - m \times xstart$$

Note: start may be 1, and end may be 2.

Direct method for drawing lines can be shown in algorithm (4).

Algorithm (4):

```
Input: Xstart,Ystart, Xend, Yend.
Output: Arbitrary line.
{
    $m = \frac{yend - ystart}{xend - xstart}$
b= ystart – a × xstart ;
for x= Xstart to Xend
{
Y=m × x + b + 0.5;
plot (x , y , color);
}
}
```

H.W.: Write complete program in order to draw arbitrary line using direct method?

2. Simple DDA (Digital Differential Analysis)

One technique for obtaining a straight line is to solve the differential equation for straight line.

DDA Algorithm(5):

Consider one point of the line as (X0,Y0) and the second point of the line as (X1,Y1).

```
1.  calculate dx , dy
    dx =ABS( X1 - X0);
    dy = ABS(Y1 - Y0);

2.  Depending upon absolute value of dx & dy
     choose number of steps (length) to put pixel as
     steps(length) = abs(dx) if abs(dx) > abs(dy)
       Else
       Steps(length) = abs(dy) if abs(dy) > abs(dx)

3.  calculate increment in x & y for each steps
    Xinc = dx / (length) steps;
    Yinc = dy / (length) steps;

4.  Put pixel for each step
    X = X0;
    Y = Y0;
    for (int i = 0; i <= steps; i++)
    {
       plot (X,Y,WHITE);
       X += Xinc;
       Y += Yinc;
    }
```

<u>H.W</u>.: Write complete program in order to draw arbitrary line using DDA algorithm?

Algorithm (5):

```
Input: $x_1$, $y_1$, $x_2$, $y_2$.
        i=1
Output: Arbitrary line.
{
    If (abs($x_2$-$x_1$) $\geq$ abs($y_2$-$y_1$))
         length= abs($x_2$-$x_1$);
    else
         length= abs($y_2$-$y_1$);
    $\Delta x = \frac{(x_2 - x_1)}{length}$;
    $\Delta y = \frac{(y_2 - y_1)}{length}$;
    x=$x_1$+0.5 $\times$ sign ($\Delta x$);
    y=$y_1$+0.5 $\times$ sign ($\Delta y$);
    while (i $\leq$ length)
        {
           plot(round(x),round(y),color);
           x=x+ $\Delta x$;
           y=y+ $\Delta y$;
           i=i+1;
        }
```

Example on DDA algorithm: Consider the line from (3, 2) to (4, 7), use DDA line algorithm to rasterize this line. Evaluate and tabulate all the steps involved.

Solution:

Given data,

1. $(x_1, y_1)= (3, 2)$

   $(x_2, y_2)= (4,7)$

2. To find,

   $\Delta x=x_2-x_1=4-3=1$

   $\Delta y=y_2-y_1=7-2=5$

3. Find out Length by checking condition,

   If (dx>=dy) then

   length=dx

   else

   length=dy

   end if

   ▯ **Length=Δy=5**

4. $\Delta x=\Delta x/\text{Length} =1/5= 0.2$

   $\Delta y=\Delta y/\text{Length}=5/5=1$

5. Initial values,

   $X=x_1+0.5*\text{sign}\ (\Delta x)$

   $=3+0.5*\ (+)$

   ▯ **X=3.5**

   $Y=y_1+0.5*\text{sign}\ (\Delta y)$

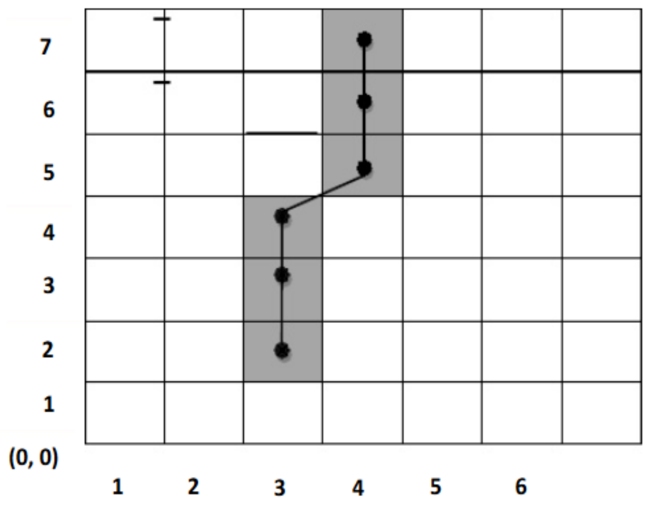
   $=2+0.5*\text{sign}(+)$

   Y=2.5

The result in tabulated form as,

Iteration	Pixel	x	y	R(x)	R(y)
0	**Initially (3,2)**	3.5	2.5	3	2
1	**(3,3)**	3.7	3.5	3	3
2	**(3,4)**	3.9	4.5	3	4
3	**(4,5)**	4.1	5.5	4	5
4	**(4,6)**	4.3	6.5	4	6
Length=5	**(4,7)**	4.5	7.5	4	7

∴ By **pictorial presentation** in graph is as shown below,



NOTE:

The DDA algorithm is faster than the direct use of the line equation since it calculates points on the line without any floating point multiplication. However, a floating point addition is still needed in determining each successive point. Furthermore, cumulative error due to limited precision in the floating point representation may cause calculated points to drift away from their true position when the line relatively long.

3.Bresenham's algorithm

- Developed by Bresenham J.E.
- Designed so that each iteration changes one of the coordinates values by ± 1.
- The other coordinate may or may not change depending on the value of an error term maintained by the algorithm.
- The error term records the distance measured perpendicular to the axis of greatest movement between the exact path of the line and the actual dots generated.

Bresenham's line algorithm is shown in algorithm (6).

Algorithm (6):

Input: x_1, y_1, x_2, y_2.

Output: Arbitrary line.

```
{    $x = x_1$; $y = y_1$; $\Delta x = x_2 - x_1$; $\Delta y = y_2 - y_1$;
     $e = \frac{\Delta y}{\Delta x} - \frac{1}{2}$;
     for i=1 to $\Delta x$
     {   plot(x,y,color);
         while($e \geq 0$)
         { y=y+1; e=e-1; }
         x=x+1;
         $e = e + \frac{\Delta y}{\Delta x}$;     }
}
```

H.W.: Write complete program in order to draw arbitrary line using Bresenham's algorithm?

Example: Consider the line from (0, 0) to (-8,-4), use general Bresenham's line algorithm to rasterize this line. Evaluate and tabulate all the steps involved.

Solution:

Given data,

$(x_1, y_1) = (0, 0)$

$(x_2, y_2) = (-8,-4)$

$\Delta x = x_2 - x_1 = -8-0 = 8$

$\therefore S_1 = -1$

$\Delta y = y_2 - y_1 = -4-0 = 4$

$\therefore S_2 = -1$

Decision Variable=$e = 2*(\Delta y)-(\Delta x)$

$\therefore e = 2*(4)-(8)$

$= 8-8 = 0$

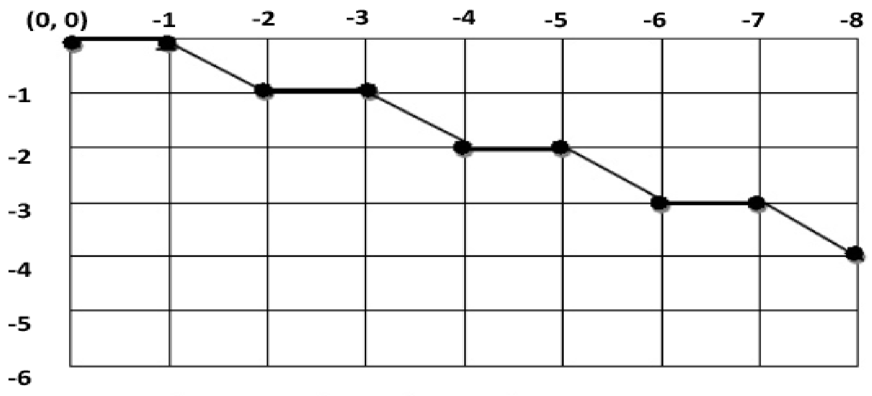
$\boxed{\therefore\ e=0}$

∴ **By using general Bresenham's algorithm,**

The result in tabulated form as,

Pixel	e	x	y
Initially (0,0)	0	0	0
(-1,0)	+8	-1	0
(-2,-1)	0	-2	-1
(-3,-1)	-8	-3	-1
(-4,-2)	0	-4	-2
(-5,-2)	+8	-5	-2
(-6,-3)	0	-6	-3
(-7,-3)	+8	-7	-3
(-8,-4)	0	-8	-4

∴ By **pictorial presentation** in graph is as shown below,



--- This is required solution for the given line using Bresenham's algorith