Scan Conversion

Professor Dr. Md. Ismail Jabiullah
Department of CSE
Daffodil International University

Scan Conversion

Chapter 3:

- Points and Lines
- Line Drawing Algorithm
- DDA Algorithm
- Bresenham's Line Algorithm
  - Parameter Description
  - Algorithm
  - Example
- Circle Generating Algorithm
- Properties of Circle
- Midpoint Circle Algorithm
  - Parameter Description
  - Algorithm
  - Example

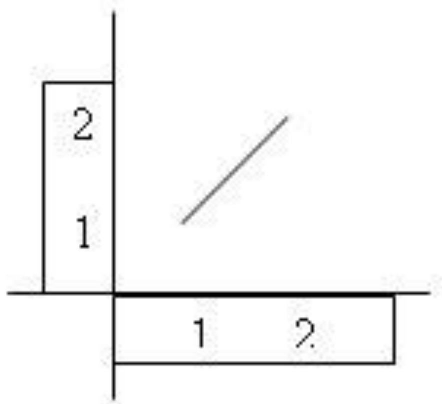
Scan Conversion a using Line Equation

The **Cartesian slope-intercept** equation for a straight line is

$$y = mx + b \quad \text{------} \quad (1)$$

with m->slope, b->y intercept

The 2 end points of a line segment are specified at a position (x_1, y_1) and (x_2, y_2).



Line path between the end point(x1,y1) and(x2,y2)

Plot a Line whose Slope is between 0^0 to 45^0 using Slope-intercept Equation

Algorithm:

Step 1: Compute $dx = x_2 - x_1$

Step 2: Compute $dy = y_2 - y_1$

Step 3: Compute $m = dy / dx$

Step 4: Compute $y = y_1 - mx_1$

Step 5: Set (x, y) equal to the lower left-hand end-point and set x_{end} equal to the largest value of x.

If $dx < 0$, then $x = x_2$, $y = y_2$ and $x_{end} = x_1$.

If $dx > 0$, then $x = x_1$, $y = y_1$ and $x_{end} = x_2$.

Step 6: Test to determine whether the entire line has been drawn. If $x > x_{end}$, Stop.

Step 7: Plot a point at current (x, y) position.

Step 8: Increment x: $x = x+1$.

Step 9: Compute the next value of y from the equation $y = mx + c$.

Step 10: Go to Step 6.

Line Drawing Algorithms

- Digital Differential Analyzer (DDA) Algorithm
- Bresenham’s Line Drawing Algorithm
- Scan Converting a Circle
  - Defining a Circle
  - Eight-way Symmetry of a Circle
  - Bresenham’s Circle Drawing Algorithm
  - Mid-point Circle Drawing Algorithm

DDA Line Algorithm

- The **digital differential analyzer (DDA)** is a scan conversion line algorithm based on calculation either Dy or Dx.
- The line at unit intervals is one coordinate and determine corresponding integer values nearest line for the other coordinate.
- Consider first a line with positive slope.

- **Step 1:** If the slope is less than or equal to 1, the unit x intervals $D_x = 1$ and compute each successive y values.

  $D_x = 1$

  $m = D_y / D_x$

  $m = (y_2 - y_1) / 1$

  $m = (y_{k+1} - y_k) / 1$

  $y_{k+1} = y_k + m$

DDA Line Algorithm

- **Step 2:** If the slope is greater than 1, the roles of x any y at the unit y intervals $D_y = 1$ and compute each successive x values.

  $D_y = 1$
  $m = D_y / D_x$
  $m = 1 / (x_2 - x_1)$
  $m = 1 / (x_{k+1} - x_k)$
  $x_{k+1} = x_k + (1/m)$

- **Step 3:** If the processing is reversed, the starting point at the right

  $D_x = -1$

  $m = D_y / D_x$
  $m = (y_2 - y_1) / -1$
  $y_{k+1} = y_k - m$

- **Step 4:** Here, $D_y = -1$
  $m = D_y / D_x$
  $m = -1 / (x_2 - x_1)$
  $m = -1 / (x_{k+1} - x_k)$
  $x_{k+1} = x_k - (1/m)$

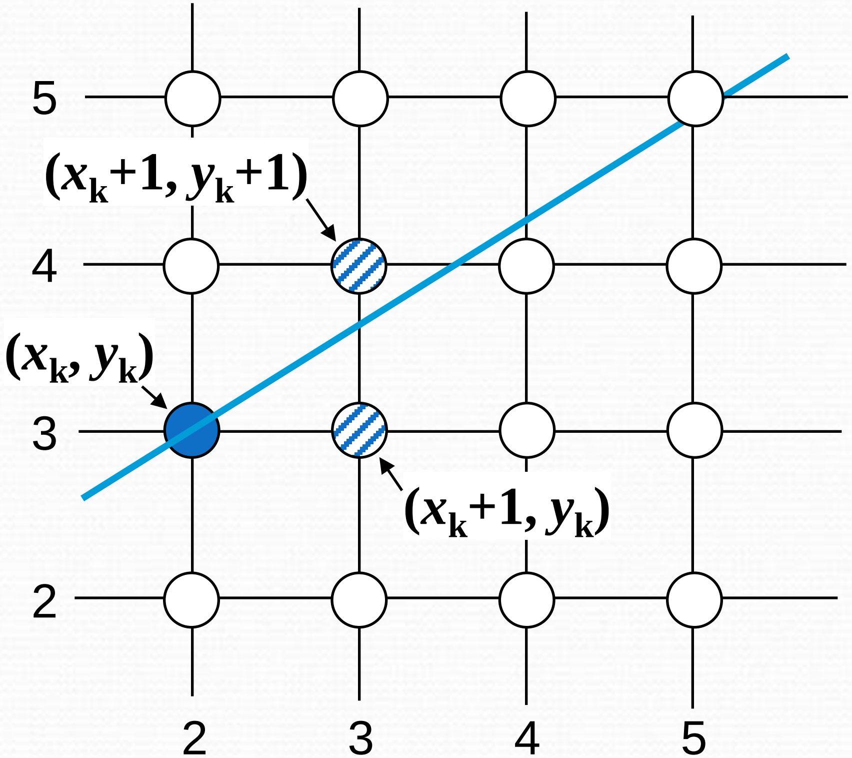
The Bresenham Line Algorithm

- The **Bresenham algorithm** is another **incremental scan conversion algorithm**.
- The big advantage of this algorithm is that it uses only **integer calculations**.

- **Jack Bresenham** worked for **27 years** at **IBM** before entering **Academia**.
- **Bresenham** developed his famous algorithms at IBM in the early **1960s.**

The Big Idea

Move across the *x* axis in unit intervals and at each step choose between two different *y* coordinates

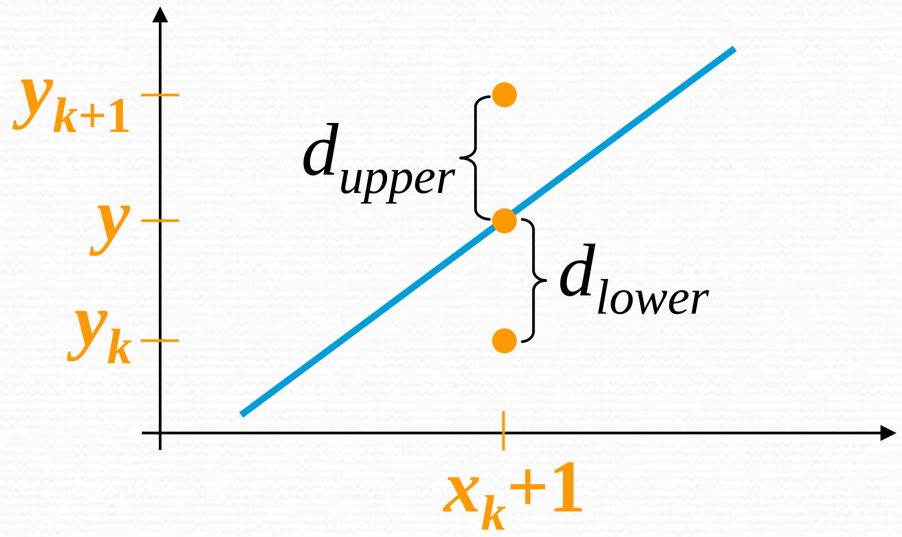


For example,

- from position **(2, 3)** we have to choose between **(3, 3)** and **(3, 4)**
- We would like the point that is closer to the original line

Deriving The Bresenham Line Algorithm

At sample position $\boldsymbol{x_k}$**+1** the vertical separations from the mathematical line are labelled $\boldsymbol{d_{upper}}$ and $\boldsymbol{d_{lower}}$



The $\boldsymbol{y}$ coordinate on the mathematical line at $\boldsymbol{x_k}$**+1** is:

$$y = m(x_k + 1) + b$$

Deriving The Bresenham Line Algorithm (cont…)

So, d_{upper} and d_{lower} are given as follows:

$$d_{lower} = y - y_k$$

and:

$$= m(x_k + 1) + b - y_k$$

$$d_{upper} = (y_k + 1) - y$$

$$= y_k + 1 - m(x_k + 1) - b$$

- We can use these **to make a simple decision** about
  - which pixel is **closer to the mathematical line**.

Deriving The Bresenham Line Algorithm (cont…)

This simple decision is based on the difference between the two pixel positions:

$$d_{lower} - d_{upper} = 2m(x_k + 1) - 2y_k + 2b - 1$$

Let's substitute ***m*** with Δ***y***/Δ***x*** where Δ***x*** and Δ***y*** are the differences between the end-points:

$$\Delta x(d_{lower} - d_{upper}) = \Delta x(2\frac{\Delta y}{\Delta x}(x_k + 1) - 2y_k + 2b - 1)$$

$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + 2\Delta y + \Delta x(2b - 1)$$

$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

Deriving The Bresenham Line Algorithm (cont…)

So, a decision parameter p_k for the kth step along a line is given by:

$$p_k = \Delta x(d_{lower} - d_{upper})$$
$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

- The sign of the **decision parameter** $\boldsymbol{p_k}$ is the same as that of $d_{lower} - d_{upper}$
- If $\boldsymbol{p_k}$ **is negative**, then we choose the **lower pixel**, otherwise we choose the **upper pixel**.

Deriving The Bresenham Line Algorithm (cont…)

Remember that, coordinate changes occur along the ***x* axis** in unit steps so we can do everything with integer calculations.

At step k+1 the decision parameter is given as:

Subtracting $\boldsymbol{p_k}$ from this we get:

$$p_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c$$

$$p_{k+1} - p_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$

Deriving The Bresenham Line Algorithm (cont…)

But, $\boldsymbol{x_{k+1}}$ is the same as $\boldsymbol{x_k}$**+1** so:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$$

where $\boldsymbol{y_{k+1} - y_k}$ is either **0** or **1** depending on the sign of $\boldsymbol{p_k}$

The first decision parameter $\mathbf{p_0}$ is evaluated at $\mathbf{(x_0, y_0)}$ is given as:

$$p_0 = 2\Delta y - \Delta x$$

The Bresenham Line Algorithm

Bresenham's Line Drawing Algorithm

(for $|m| < 1.0$)

Step 1: Input the two line end-points, storing the left end-point in **(x_0, y_0)**

Step 2: Plot the point **(x_0, y_0)**

Step 3: Calculate the constants **Δx, Δy, $2\Delta y$**, and **$(2\Delta y - 2\Delta x)$** and get the first value for the decision parameter as:

$$p_0 = 2\Delta y - \Delta x$$

Step 4: At each **x_k** along the line, starting at $k = 0$, perform the following test. If $p_k < 0$, the next point to plot is (x_k+1, y_k) and:

$$p_{k+1} = p_k + 2\Delta y$$

The Bresenham Line Algorithm (cont…)

Otherwise, the next point to plot is (x_k+*1*, y_k+*1*) and:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

Step 5: Repeat **Step 4** (Δx – 1) times

Attention!

- The algorithm and derivation above assumes slopes are less than 1.
- For other slopes we need to adjust the algorithm slightly.

Bresenham Line Algorithm: Example

Let's have a go at this

Let's plot the line from (20, 10) to (30, 18)

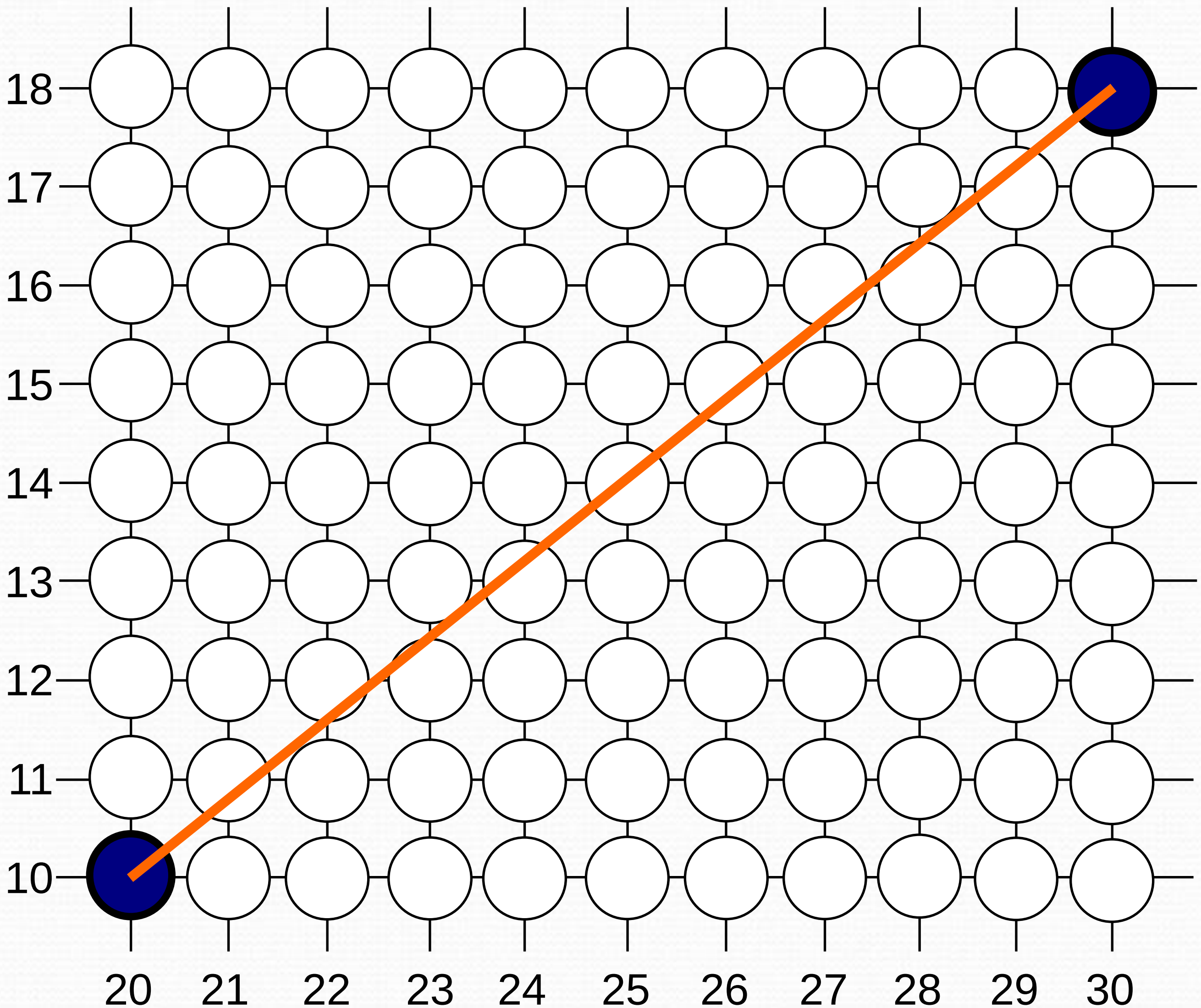
First off calculate all of the constants:

- Δx: $(x_2 - x_1) = (30\text{-}20) = 10$
- Δy: $(y_2 - y_1) = (18\text{-}10) = 8$
- $2\Delta y$: $(2 \text{ x } 8) = 16$
- $2\Delta y - 2\Delta x$: $(2\text{x}8 - 2\text{x}10) = (16 - 20) = -4$

Calculate the initial decision parameter p_0:

- $p_0 = 2\Delta y - \Delta x = (2\text{x}8 - 10) = (16 - 10) = 6$

Bresenham Line Algorithm: Example (cont…)

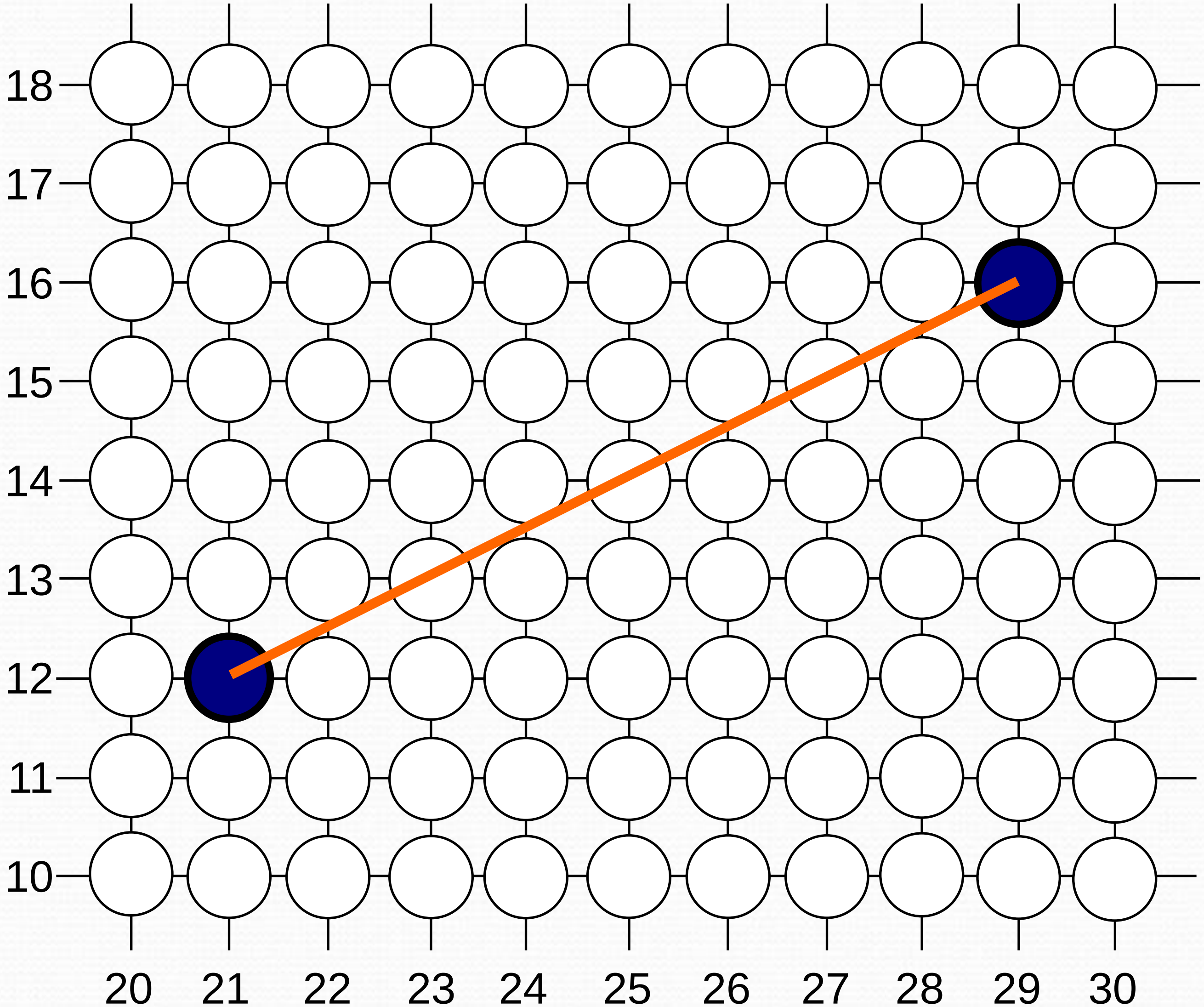


k	p_k	(x_{k+1},y_{k+1})
0	P_0	$(x_1,y_1)=(20,10)$
1	P_1	
2		
3		
4		
5		
6		
7		
8		
9		

Bresenham Line Algorithm: Exercise

Go through the **Step 1** to **Step 5** of the Bresenham line drawing algorithm for a line going from (21,12) to (29,16)

Bresenham Exercise (cont…)



k	p_k	(x_{k+1},y_{k+1})
0		
1		
2		
3		
4		
5		
6		
7		
8		

Bresenham Line Algorithm: Summary Advantages and Problems

The Bresenham line algorithm has the following **advantages**:

- A fast incremental algorithm
- Uses only integer calculations

Comparing this to the DDA algorithm, DDA has the following **problems**:

- Accumulation of round-off errors can make the pixelated line drift away from what was intended
- The rounding operations and floating point arithmetic involved are time consuming

Plot a Line whose Slope is between 0^0 to 45^0 using Bresenham's Line Algorithm

Algorithm:

Step 1: Compute the initial values:

$dx = x_2 – x_1$, $Inc_2 = 2(dy – dx)$

$dy = y_2 – y_1$, $d = Inc_1 – dx$

$Inc_1 = 2dy$

Step 2: Set (x, y) equal to the lower left-hand end-point and set x_{end} equal to the largest value of x.

If $dx < 0$, then $x = x_2$, $y = y_2$ and $x_{end} = x_1$.

If $dx > 0$, then $x = x_1$, $y = y_1$ and $x_{end} = x_2$.

Step 3: Plot a point at current (x, y) position.

Step 4: Test to see whether the entire line has been drawn. If $x>x_{end}$, Stop.

Step 5: Compute the location of the next pixel. If $d<0$, then $d = d+ inc_1$. If $d>=0$, then $d = d + Inc_2$, and $y = y + 1$.

Step 6: Increment x: $x = x+1$.

Step 7: Plot a point at current (x, y) position.

Step 8: Go to Step 4.

Bresenham's Line Algorithm: Scan-conerting a Line from (1, 1) to (8, 5)

Algorithm:

Step 1: Find the starting values.

Step 2: In this case, dx = $x_2 - x_1 = 8 - 1 = 7$, dy = y2 – y1 = 5 – 1 = 4.

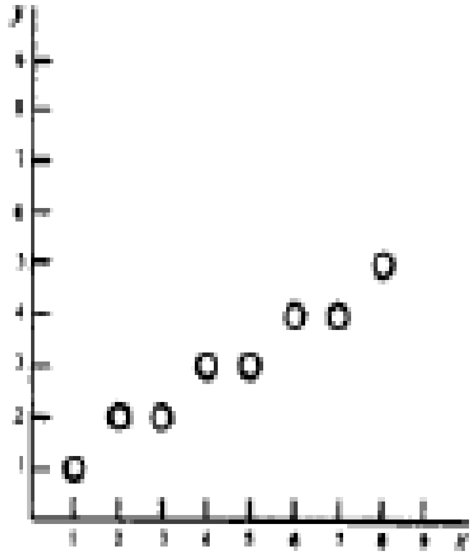
Step 3: Therefore,

$Inc_1 = 2$, dy = 2 x 4 = 8.

Inc2 = 2(dy –dx) = 2(4 - 7) = - 6, d= $Inc_1 - dx = 8 - 7 = 1$.

The following table indicates the values computed by the algorithm:

d	x	y
1	1	1
$1 + Inc_2 = -5$	2	2
$-5 + Inc_1 = 3$	3	2
$3 + Inc_2 = -3$	4	3
$-3 + Inc_1 = 5$	5	3
$5 + Inc_2 = -1$	6	4
$-1 + Inc_1 = 7$	7	4
$7 + Inc_2 = 1$	8	5



Circle Drawing Algorithm using Polynomial Method

Algorithm:

Step 1: Set the initial variables r = circle radius, (h, k) = co-ordinates of the circle center, $x = 0$, $x_{end} = r/\sqrt{2}$.

Step 2: Test to determine whether the entire circle has been scan-converted.

If $x = x_{end}$, stop.

Step 3: Compute the value of the y-co-ordinate, where $y = \sqrt{(r^2 - x^2)}$.

Step 4: Plot the eight points, found by symmetry with respect to the center (h, k), at the current (x, y) coordinates:

Plot (x+h, y+k) Plot (-x+h, -y+k)

Plot (y+h, x+k) Plot (-y+h, -x+k)

Plot (-y+h, x+k) Plot (y+h, -x+k)

Plot (-x+h, y+k) Plot (x+h, -y+k)

Step 5: Increment x: $x = x+1$.

Step 6: Go to Step 2.

Scan-Converting a Circle using Bresenham's Algorithm

Algorithm:

Step 1: Set the initial value of the variables (h, k) = co-ordinates of the circle center, x = 0, y = circle radius r, and d= 3 -2r.

Step 2: Test to determine whether the entire circle has been scan-converted. If x>y, stop.

Step 3: Plot the eight points, found by symmetry with respect to the center (h, k), at the current (x, y) coordinates:

Plot (x+h, y+k)	Plot (-x+h, -y+k)
Plot (y+h, x+k)	Plot (-y+h, -x+k)
Plot (-y+h, x+k)	Plot (y+h, -x+k)
Plot (-x+h, y+k)	Plot (x+h, -y+k)

Step 4: Compute the location of the next pixel.

If d<0, then d = d + 4x + 6, and x = x+1.

If d>= 0, then d= d + 4(x-y) +10, x= x + 1 and y = y -1.

Step 5: Go to Step 2.

We have Learnt:

- Points and Lines
- Line Drawing Algorithm
- DDA Algorithm
- Bresenham's Line Algorithm
  - Parameter Description
  - Algorithm
  - Example
- Circle Generating Algorithm
- Properties of Circle
- Midpoint Circle Algorithm
  - Parameter Description
  - Algorithm
  - Example