# NEPAL COLLEGE OF INFORMATION TECHNOLOGY

Balkumari, Lalitpur

*Affiliated to Pokhara University*

## ASSIGNMENT FOR COMPUTER GRAPHICS

## ASSIGNMENT 2


Submitted by:

Name: Arpan Adhikari

Roll No.: 231309

BE-CE (3rd SEM)

Submitted to:

Saroj Shakya

Name : Arpan Adhikari
Roll No. 231309

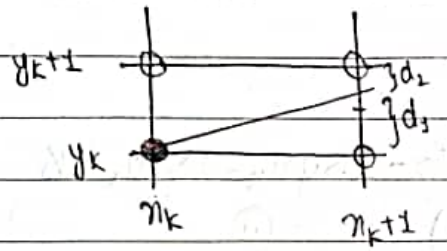
# Assignment 2

1) How is the decision parameter calculated in case of Bresenham's line drawing and Mid point circle algorithm? What is its role?

→

## Bresenham's Line drawing Algorithm:

Let us consider the line to be drawn, has slope less than 1.




Let us consider the distance of pixel as shown in figure,
$d_1$ → distance of lower pixel from ideal location
$d_2$ → distance of upper pixel from ideal location

~~The next~~ For any $K^{th}$ step, suppose $(x_k, y_k)$ has been selected in previous step. Then the next pixel to be selected will be $(x_{k+1}, y_k)$ or $(x_{k+1}, y_{k+1})$. Lets consider it as $(x_{k+1}, y_{k+1})$.

Eqⁿ of line is $y = mx + c$.
The eqⁿ of line at ideal location will be $y = m(x_k + 1) + c$.

$$d_1 = y - y_k = m(x_k + 1) + c - y_k$$

$$d_2 = y_k + 1 - y = y_k + 1 - m(x_k + 1) - c$$

$$d_1 - d_2 = m(x_k+1) + c - y_k - y_k - 1 + m(x_k+1) + c$$
$$= 2m(x_k+1) - 2y_k + 2c - 1$$

Substituting $m = \frac{\Delta y}{\Delta x}$,

$$\Delta x(d_1 - d_2) = 2\Delta y(x_k+1) - 2\Delta x y_k + 2\Delta x c - \Delta x$$
$$= 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x c - \Delta x$$
$$\Delta x(d_1 - d_2) = 2\Delta y x_k - 2\Delta x y_k + 2\Delta y + 2\Delta x c - \Delta x$$

Suppose, $b = 2\Delta y + 2\Delta x c - \Delta x$ (constant), ~~and $\Delta x(d_1 - d_2) = p_k$~~

$$p_k = \Delta x(d_1 - d_2) = 2\Delta y x_k - 2\Delta x y_k + b \quad \text{——— (i)}$$

For $(k+1)^{th}$ step,

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

So, $p_{k+1} = p_k + 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + b - 2\Delta y x_k - 2\Delta x y_k + b$

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

So, eqⁿ (i) and (ii) are decision parameters for $k^{th}$ & $(k+1)^{th}$ step.

## Midpoint circle algorithm:

To apply midpoint circle algo, we define a function,

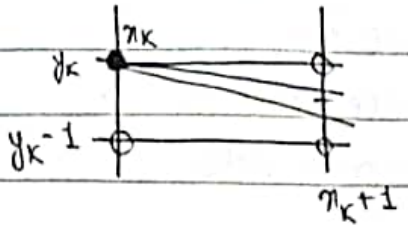
$$f_{circle}(x,y) = x^2 + y^2 - r^2$$

where, $f_{circle}(x,y) = 0$ if point $(x,y)$ is on circle boundary.

$f_{circle}(x,y) < 0$ if point $(x,y)$ lies inside circle boundary.

$> 0$ if point $(x,y)$ lies outside circle boundary.

The circle function serves as a decision parameter.




For drawing circle, start at $(0, r)$, take unit step in 'x'. (sample in x direction i.e. $x_{k+1} = x_k + 1$). The next pixel chosen will be either $(x_k + 1, y_k)$ or $(x_k + 1, y_k - 1)$.

Midpoint between candidates $= \left( \frac{x_k + 1 + x_k + 1}{2}, \frac{y_k + y_k - 1}{2} \right)$

$= \left( x_k + 1, y_k - \frac{1}{2} \right)$

The decision parameter is,

$P_k = f_{circle}\left( x_k + 1, y_k - \frac{1}{2} \right)$

$= (x_k + 1)^2 + \left( y_k - \frac{1}{2} \right)^2 - r^2$

At $(k+1)^{th}$ step,

$P_{k+1} = f_{circle}\left( x_{k+1} + 1, y_{k+1} - \frac{1}{2} \right)$

$= f_{circle}\left[ (x_k + 1) + 1, y_{k+1} - \frac{1}{2} \right]$

$= \left[ (x_k + 1) + 1 \right]^2 + \left( y_{k+1} - \frac{1}{2} \right)^2 - r^2$

Now,

$P_{k+1} - P_k = (x_k + 1)^2 + 2(x_k + 1) + 1 + \left( y_{k+1} - \frac{1}{2} \right)^2 - r^2 - (x_k + 1)^2 - \left( y_k - \frac{1}{2} \right)^2 + r^2$

$\therefore P_{k+1} = P_k + 2(x_k + 1) + 1 + \left( y_{k+1} - \frac{1}{2} \right)^2 - \left( y_k - \frac{1}{2} \right)^2$ ——— (iii)

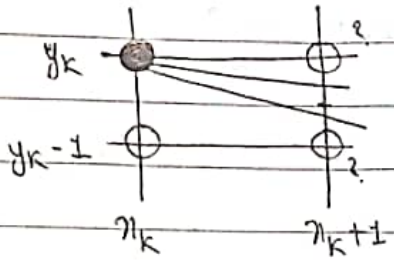
eqn (iii) is the decision parameter for midpoint circle algorithm.

Role :

The role of decision parameter is to select the pixel closest to the ideal line, by evaluation of distance between the pixels and ideal position or through other methods.

Q2) Explain the logic used to draw lines with negative slope with Bresenham's Line Drawing Algorithm.

→




Let us consider a line as shown in figure having negative slope. For line formation, value of $x$ increases, but $y$ decreases.

As depicted in figure, consider a pixel is selected at position $(x_k, y_k)$. The next pixel to be selected is either $(x_k + 1, y_k)$ or $(x_k + 1, y_k - 1)$. The pixel to be chosen is decided by the decision parameter based on its magnitude.

The initial decision parameter is,

$$P_0 = 2\Delta y - \Delta x$$

The decision parameter on consecutive step is,

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

case 1 : If $P_k <= 0$, then pixel on scanline '$y_k$' is closer to the line path and $y_{k+1} = y_k$.

so, $P_{k+1} = P_k + 2\Delta y$

case 2 : If $P_k > 0$, then pixel on scanline '$y_k - 1$' is closer to the line path and $y_{k+1} = y_k - 1$.

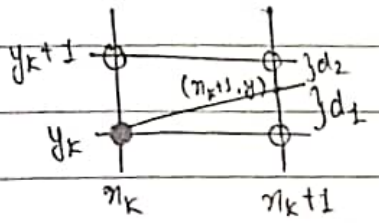
so, $P_{k+1} = P_k + 2\Delta y - 2\Delta x(y_k - 1 - y_k)$

$\therefore P_{k+1} = P_k + 2\Delta y + 2\Delta x$

According to these parameters, the pixel point to be selected is chosen on the base of previous step's decision parameter. In this way, lines are drawn with negative slope with BLA.

Q3) Derive Bresenham's Line drawing Algorithm for $|m| > 1$. How can a line with end points $A(x_1, y_1)$, $B(x_2, y_2)$ and slope less than 1 can be drawn if starting point is taken as $B(x_2, y_2)$ using BLA algorithm.

→




Here, $y = m(x_k + 1) + c$ (At point both candidate pixels)

$d_1 = y - y_k = m(x_k + 1) + c - y_k$ $\quad \frac{y_k - 1 + c}{m} - x_k$

$d_2 = y_{k+1} - y = y_{k+1} - m(x_k + 1) - c$

$x_k - 1 - (y_k - 1 - c)/m$

Now,

$d_2 - d_1 = y_k$

$d_1 - d_2 = m(x_k + 1) + c - y_k - y_k - 1 + m(x_k + 1) + c$

$d_1 - d_2 = 2m(x_k + 1) - 2y_k + 2c - 1$ ——— (i)

Or, $\Delta x(d_1 - d_2) = 2\Delta y(x_k + 1) - 2\Delta x y_k + 2\Delta x c - \Delta x$

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

So, $P_k = \Delta x(d_1 - d_2) = 2\Delta y x_k - 2\Delta x y_k + b$

$P_k = \Delta y(d_1 - d_2) = 2\Delta x y_k - 2\Delta y x_k + b$

where, $b = 2\Delta y + 2\Delta x c - \Delta x$.

At $(k+1)^{th}$ step,

$P_{k+1} = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + b$

Now, $P_{k+1} = P_k + 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + b - 2\Delta y x_k + 2\Delta x y_k - b$

or, $P_{k+1} = P_k + 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$

For slope $< 1$, $y_{k+1} = y_k + 1$.

So, $P_{k+1} = P_k - 2\Delta x + 2\Delta y(x_{k+1} - x_k)$

If $P_k <= 0$, then $y_{k+1} = y_k$

$\therefore P_{k+1} = P_k + 2\Delta y$.

If $P_k > 0$, then $y_{k+1} = y_k + 1$

$\therefore P_{k+1} = P_k + 2\Delta y + 2\Delta y$

The initial decision parameter is calculated as,

If eqn (i) passes through $(x_0, y_0)$,

$d_1 - d_2 = 2m(x_0 + 1) - 2y_0 + 2c - 1$

or, $d_1 - d_2 = 2mx_0 + 2m - 2y_0 + 2c - 1$

or, $d_1 - d_2 = 2m - 1$

or, $d_1 - d_2 = \frac{2\Delta y}{\Delta x} - 1$

$\therefore P_k = \Delta x(d_1 - d_2) = 2\Delta y - \Delta x$,

These are reqd derivation for BLA.

To draw the given line $A(x_1, y_1)$ to $B(x_2, y_2)$.

Given, slope < 1.

starting point = $B(x_2, y_2)$

At first, the initial decision parameter is calculated as,

$$P_0 = 2\Delta y - \Delta x$$

$$\therefore P_0 = 2(y_2 - y_1) - (x_2 - x_1)$$

Based on the initial decision parameter, next pixel (point) is chosen after $B(x_2, y_2)$.

case 1 : If $P_0 <= 0$, then $x_{k+1} = x_k + 1$, $y_{k+1} = y_k$.

The decision parameter for next step will be $P_{k+1} = P_k + 2\Delta y$

case 2 : If $P_0 > 0$, then $x_{k+1} = x_k + 1$, $y_{k+1} = y_k - 1$,

The decision parameter for next step will be $P_{k+1} = P_k + 2\Delta y + 2\Delta x$

This step will be repeated until we find $A(x_1, y_1)$ point.

Q4) Derive necessary equations for drawing a circle using Midpoint circle Algorithm? How can you use this algorithm to draw a circle if the starting point is $(-r, 0)$ moving in clockwise direction?

→

To draw circle using midpoint circle algorithm, we define a circle function:

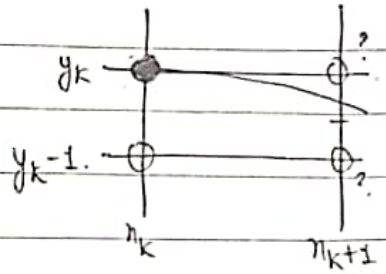
$$f_{circle} = x^2 + y^2 - r^2$$

Here, $f_{circle} = 0$ if point $(x, y)$ lies in circle boundary.

$< 0$ if point lies inside circle boundary.

$> 0$ if point lies outside circle boundary.

This circle function acts as a decision parameter for selecting the next pixel during circle construction.




Assuming $(x_k, y_k)$ has been selected in previous step, we select the next pixel $(x_{k+1}, y_{k+1})$ as either $(x_{k+1}, y_k)$ or $(x_k + 1, y_k - 1)$.
[∵sample in x direction $x_{k+1} = x_k + 1$]

Calculate midpoint as,

$$\left(\frac{x_{k+1} + x_k + 1}{2}, \frac{y_k + y_k - 1}{2}\right)$$

$$\left(x_k + 1, y_k - \frac{1}{2}\right)$$

So, The decision parameter is,

$$P_k = f_{circle}\left(x_k + 1, y_k - \frac{1}{2}\right)$$

$$P_k = (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2 \quad \text{——— (i)}$$

For $k+1^{th}$ step,

$$P_{k+1} = (x_{k+1} + 1)^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2 \quad \text{——— (ii)}$$

Now, $P_{k+1} = P_k + \left((x_k + 1) + 1\right)^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2 - (x_k + 1)^2 - \left(y_k - \frac{1}{2}\right)^2 + r^2$

or, $P_{k+1} = P_k + 2(x_k + 1) + 1 + \left(y_{k+1} - \frac{1}{2}\right)^2 - \left(y_k - \frac{1}{2}\right)^2 \quad \text{——— (iii)}$

This is the decision parameter for midpoint circle algorithm.

If $P_k <= 0$, then, $y_{k+1} = y_k$.
so, (iii) becomes, $P_{k+1} = P_k + 2(x_k+1) + 1$

If $P_k >= 0$, then $y_{k+1} = y_k - 1$,
so, (iii) becomes $P_{k+1} = P_k + 2(x_k+1) + 1 + \left(y_k - \frac{3}{2}\right)^2 - \left(y_k - \frac{1}{2}\right)^2$

starting point = (-r, 0) & clockwise direction

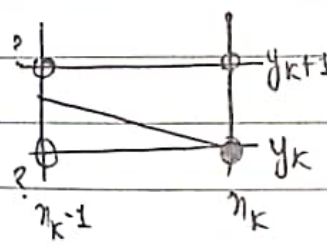
The decision parameter at starting position $(x_0, y_0) = (0, r)$ is,
The next pixel to plot is either (1, r) or (1, r-1).
Midpoint is, $\left(1, r - \frac{1}{2}\right)$

$P_0 = f_{circle}\left(1, r - \frac{1}{2}\right)$
$= 1^2 + r^2 - r + \frac{1}{4} - r^2$

$= 5/4 - r$ $\quad \therefore P_0 = 1 - r.$

starting point = (-r, 0) and cw direction:

For drawing circle with given conditions, we set the initial point (-r, 0). Using this, the initial decision parameter is calculated as $P_0 = 1 - r$.




In clockwise direction, suppose $(x_k, y_k)$ has been chosen in the previous step, now the next pixel $(x_{k+1}, y_{k+1})$ will be either $(x_k - 1, y_k)$ or $(x_k - 1, y_k + 1)$.

The initial decision parameter $P_0$ will be chosen as, $P_0 = 1 - r$.

If $P_K < 0$, the next point will be $(x-1, y)$ and the decision parameter is updated as, $P_{K+1} = P_K + 2y_K + 1$

If $P_K >= 0$, the next point will be $(x-1, y+1)$ and the decision parameter is updated as,

$$P_{K+1} = P_K + 2y_K - 2x_K + 1$$

Q5) Digitize a line with end point A(-2, -4) and B(-6, -9) using BLA and DDA as well.

Soln

Using DDA,

$$\text{Slope}(m) = \frac{\Delta y}{\Delta x} = \frac{-9+4}{-6+2} = \frac{-5}{-4} = \frac{5}{4}$$

Here, it is positive slope (left to right).

~~slope < 1. ($\Delta y < \Delta x$)~~ slope > 1 ($\Delta y > \Delta x$)

So, $x_{K+1} = x_K$

Using DDA,

$$\text{Slope}(m) = \frac{\Delta y}{\Delta x} = \frac{-9+4}{-6+2} = \frac{-5}{-4} = \frac{5}{4}$$

↳ Positive slope.

(Q5) Digitize a line with end point A(-2,-4) & B(-6,-9) using BLA and DDA as well.

Using BLA,

$\Delta x = -6+2 = -4$

$\Delta y = -9+4 = -5$

slope $= \frac{5}{4}$ i.e. slope > 1

If $P_K \leq 0$, $y_{K+1} = y_K + 1$

$x_{K+1} = x_K$

$P_{K+1} = P_K + 2\Delta x$

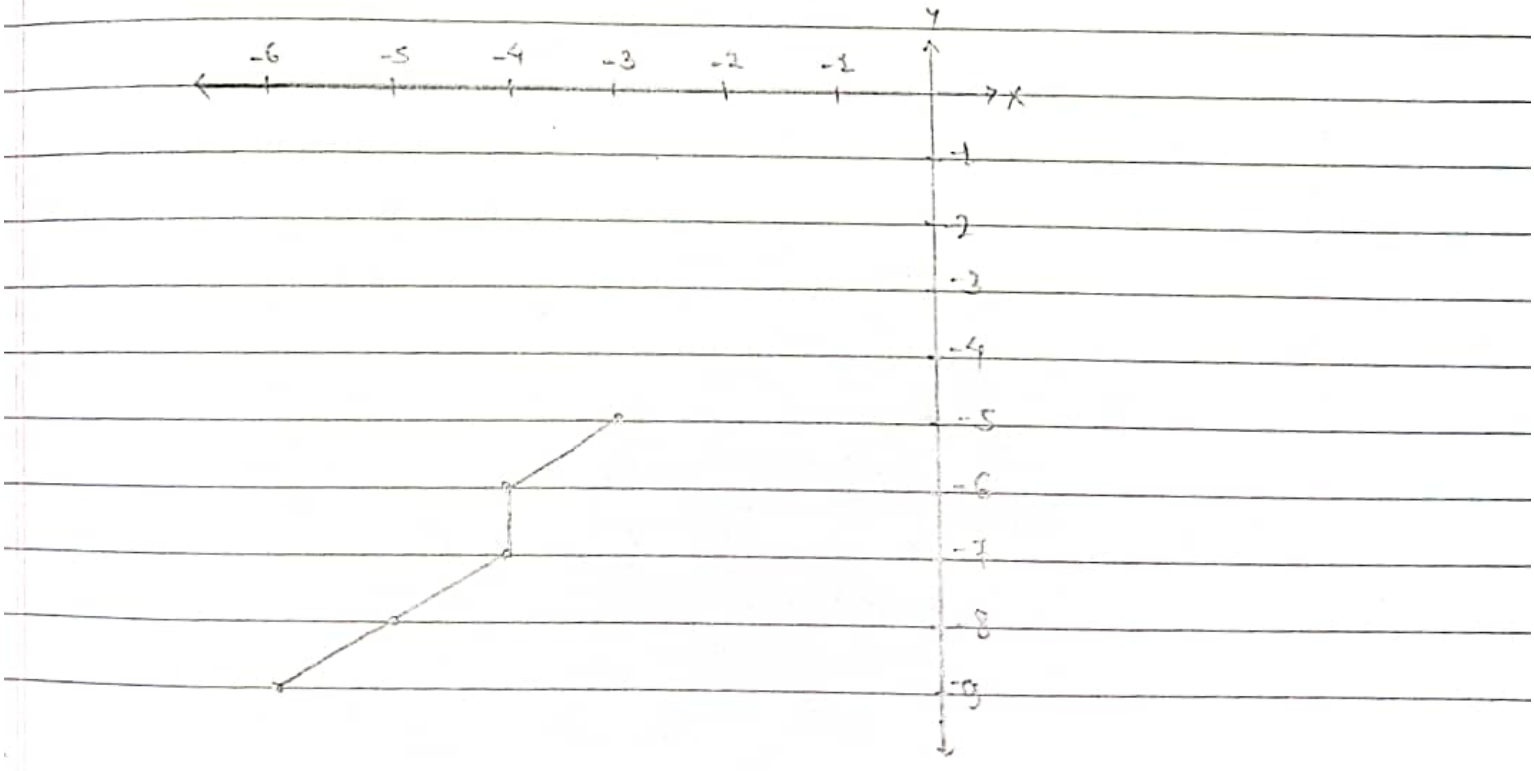
If $P_K > 0$, $y_{K+1} = y_K + 1$

$x_{K+1} = x_K + 1$

$P_{K+1} = P_K + 2\Delta x + 2\Delta y$

starting pixel = A(-2,-4). $P_0 = -2\Delta x + \Delta y = -2\times(-4) - 5 = 3$

| K | $P_K$ | $x_{K+1}, y_{K+1}$ |
|---|---|---|
| 0 | $P_0 = 3$ | (-3, -5) |
| 1 | $P_1 = 3 - 2\times(-4) + 2\times(-5) = 1$ | (-4, -6) |
| 2 | $P_2 = 1 - 2\times(-4) + 2\times(-5) = -1$ | (-4, -7) |
| 3 | $P_3 = -1 + 2\times(-4) + 2\times(-5) = 7$ | (-5, -8) |
| 4 | $P_4 = 7 - 2\times(-4) + 2\times(-5) = 5$ | (-6, -9) |



Using DDA,

$\Delta y = -9+4 = -5$ , $\Delta x = -6+2 = -4$

Here, steps = $\max(|x|, |y|)$

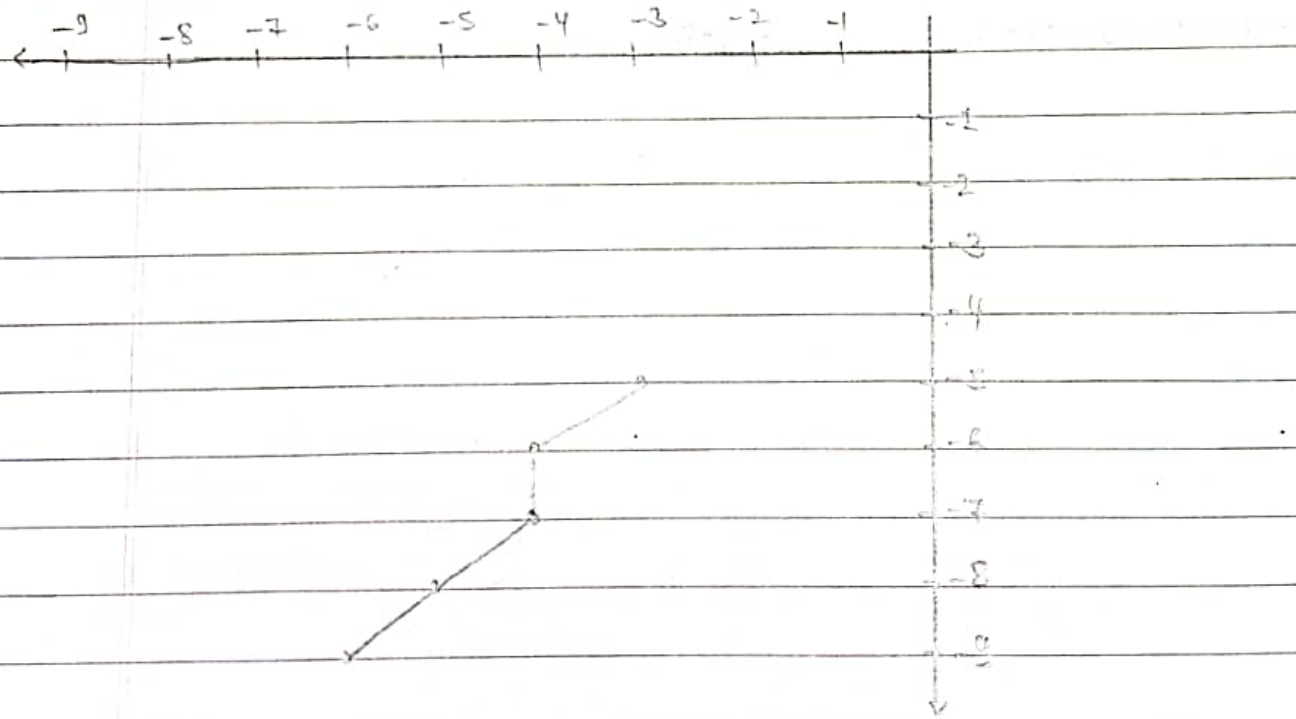
$= 5$

So, $x\text{Inc} = \frac{\Delta x}{\text{steps}} = \frac{-4}{5}$ , $y\text{Inc} = \frac{\Delta y}{\text{steps}} = \frac{-5}{5} = -1$

Hence, $x_{k+1} = x_k - 0.8$

$y_{k+1} = y_k - 1$ , starting pixel = A(-2, -4)

| K | $x_{k+1}$ | $y_{k+1}$ | Rounded pixel |
|---|---|---|---|
| 1 | -2-0.8 = -2.8 | -4-1 = -5 | (-3, -5) |
| 2 | -2.8-0.8 = -3.6 | -5-1 = -6 | (-4, -6) |
| 3 | -3.6-0.8 = -4.4 | -6-1 = -7 | (-4, -7) |
| 4 | -4.4-0.8 = -5.2 | -7-1 = -8 | (-5, -8) |
| 5 | -5.2-0.8 = -6 | -8-1 = -9 | (-6, -9) |



Q6) Digitize a line with end points A(19,9) and B(17,11) using Bresenham's Line drawing Algorithm and DDA as well.

→

Using DDA,

$\Delta x = 17 - 19 = -2$

$\Delta y = 11 - 9 = 2$

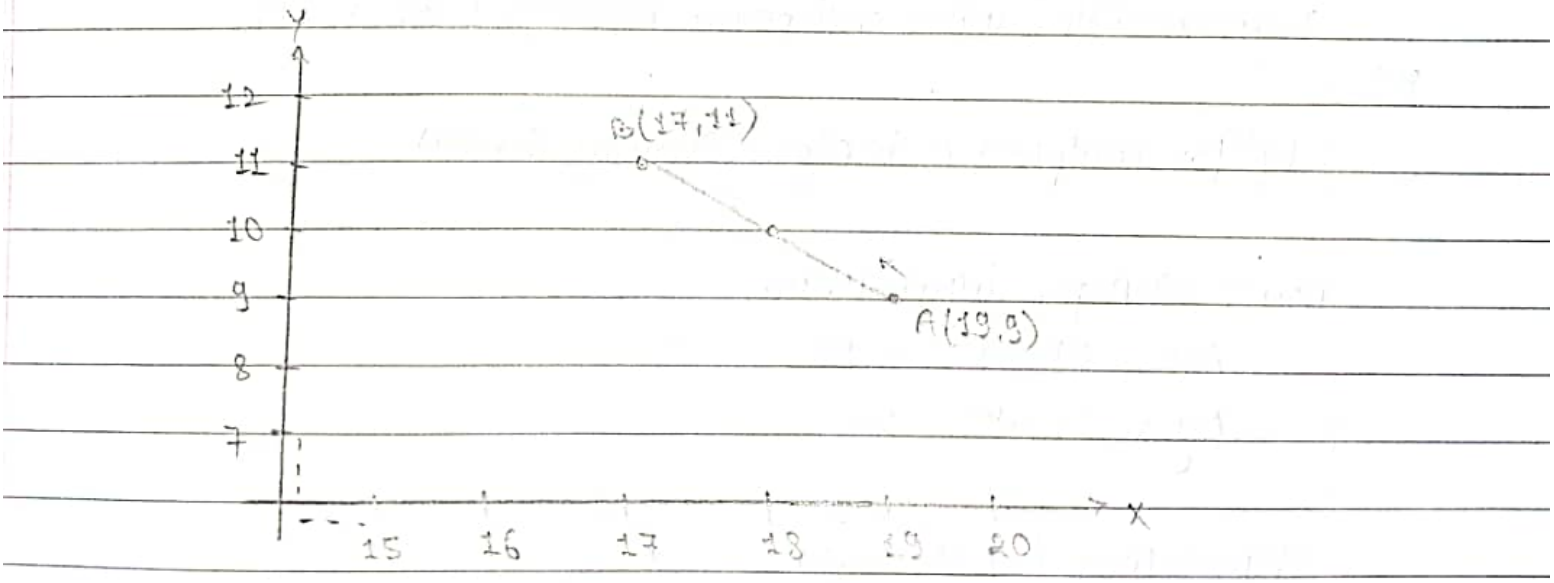
Here, steps = $\text{Max}(|\Delta x|, |\Delta y|) = 2$.

So, $x\text{Inc} = \frac{\Delta x}{\text{steps}} = \frac{-2}{2} = -1$ $\quad y\text{Inc} = \frac{\Delta y}{\text{steps}} = \frac{2}{2} = 1.$

So, $x_{k+1} = x_k - 1$

$y_{k+1} = y_k + 1$ , starting pixel = A(19,9)

| K | $x_{k+1}$ | $y_{k+1}$ | Rounded pixel |
|---|---|---|---|
| 1 | 19-1 = 18 | 9+1 = 10 | (18,10) |
| 2 | 18-1 = 17 | 10+1 = 11 | (17,11) |



Using BLA,

$\Delta x = -2, \quad \Delta y = 2$

We slope $= \frac{\Delta y}{\Delta x} = \frac{2}{-2} = -1.$

Here, $P_0 = 2\Delta y - \Delta x = 2 \times 2 - (-2) = 6$

If $P_k <= 0$, $x_{k+1} = x_k - 1$, $y_{k+1} = y_k$

$P_{k+1} = P_k + 2\Delta y$

If $P_k > 0$, $x_{k+1} = x_k - 1$, $y_{k+1} = y_k + 1$,

$P_{k+1} = P_k + 2\Delta y - 2\Delta x$

So, starting pixel = A(19, 9)

| K | $P_K$ | $x_{k+1}, y_{k+1}$ |
|---|---|---|
| 0 | $P_0 = 6$ | (18, 10) |
| 1 | $P_1 = 6 + 2 \times 2 - 2 \times (-2) = 14$ | (17, 11) |

Hence, points are A(19,9), (18,10) and B(17,11). Ans

Q7) Derive necessary equations for midpoint circle algorithm. Digitize a line with end points A(11,9) & B(29,17).

Soln

Eq$^n$ for midpoint circle algo: same as Qsn (4).

Using DDA for digitization,

$\Delta x = 29 - 11 = 18$

$\Delta y = 17 - 9 = 8$

steps = max ($|\Delta x|, |\Delta y|$)

= 18

$x\text{Inc} = \frac{\Delta x}{\text{steps}} = 1$ $\quad$ $y\text{Inc} = \frac{\Delta y}{\text{steps}} = \frac{8}{18} = 0.44$

So, $x_{k+1} = x_k + 1$ and $y_{k+1} = y_k + 0.44$ St. pixel = (11, 9)

Now,

| Steps | $x_{k+1}$ | $y_{k+1}$ | Rounded pixels |
|---|---|---|---|
| 1 | 12 | 9+0.44 = 9.44 | (12, 9) |
| 2 | 13 | 9.88 | (13, 10) |
| 3 | 14 | 10.32 | (14, 10) |
| 4 | 15 | 10.76 | (15, 11) |
| 5 | 16 | 11.2 | (16, 11) |
| 6 | 17 | 11.64 | (17, 12) |
| 7 | 18 | 12.08 | (18, 12) |
| 8 | 19 | 12.52 | (19, 13) |
| 9 | 20 | 12.96 | (20, 13) |
| 10 | 21 | 13.4 | (21, 13) |
| 11 | 22 | 13.84 | (22, 14) |
| 12 | 23 | 14.28 | (23, 14) |
| 13 | 24 | 14.72 | (24, 15) |
| 14 | 25 | 15.16 | (25, 15) |
| 15 | 26 | 15.6 | (26, 16) |
| 16 | 27 | 16.04 | (27, 16) |
| 17 | 28 | 16.48 | (28, 16) |
| 18 | 29 | 16.92 | (29, 17) |

The starting pixel = A(11, 9)
Ending pixel = B (29, 17)

18) Digitize a circle with a radius of 14 pixels and centered at (-10, -12).

Sol^n

Here, starting pixel = ~~(-10, r+(-12))~~ (0, 14)
= ~~(-10, 2)~~

Initial decision parameter ($P_0$) = 1 - r = 1 - 14 = -13

Here, For next steps,

If $P_k < 0$, $x_{k+1} = x_k + 1$
$y_{k+1} = y_k$
$P_{k+1} = P_k + 2(x_{k+1}) + 1$

If $P_k \geq 0$, $x_{k+1} = x_k + 1$
$y_{k+1} = y_k - 1$
$P_{k+1} = P_k + 2(x_{k+1}) - 2y_{k+1} + 1$

| Step | $P_k$ | $x_{k+1}$ | $y_{k+1}$ |
|---|---|---|---|
| 1 | -13 | ~~-9~~ -01 | 214 |
| 2 | -13+2x(-9)+1 = -10 | 2 | 14 |
| 3 | -10+2x2+1 = -5 | 3 | 14 |
| 4 | -5+2x3+1 = 2 | 4 | 13 |
| 5 | 2+2x4-2x13+1 = -15 | 5 | 13 |
| 6 | -15+10+1 = -4 | 6 | 13 |
| 7 | -4+12+1 = 9 | 7 | 12 |
| 8 | 9+2x7-2x12+1 = 0 | 8 | 11 |
| 9 | 0+2x8-2x11+1 = -5 | 9 | 11 |
| 10 | -5+2x9-2x11+1 = -8 | 10 | 11 |
| 11 | -8+2x10+1 = 13 | 11 | 10 |

Since the center of circle is (-10, -12),

$x = x - 10$, $y = y - 12$,

(-9, 2) (-6, 1) (-3, 0) (0, -1)
(-8, 2) (-5, 1) (-2, -1) (1, -2)
(-7, 12) (-4, 1) (-1, -1)

If $p_k < 0$, $x_{k+1} = x_k + 1$
$y_{k+1} = y_k$
$p_{k+1} = p_k + 2x_{k+1} + 1$

$p_k \geq 0$, $x_{k+1} = x_k + 1$
$y_{k+1} = y_k - 1$
$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$

Q9) Digitise a circle described by an eqn $(x+5)^2 + (y-3)^2 = 25$.

soln

Here, radius = 5 , starting pixel = $(0, r) = (0, 5)$

$p_0 = 1 - r = -4$

Assuming circle is centered at origin,

| k | $p_k$ | $x_{k+1}$ | $y_{k+1}$ |
|---|---|---|---|
| 0 | $-4$ | 1 | 5 |
| 1 | $-4+2+1 = -1$ | 2 | 5 |
| 2 | $-1+4+1 = 4$ | 3 | 4 |
| 3 | $4+6+1-8 = 3$ | 4 | 3 |

Since the center of circle is at $(-5, 3)$,

$x = x - 5$, $y = y + 3$

So, The points are $(-4, 8)$
$(-3, 8)$
$(-2, 7)$
$(-1, 6)$

Q10) Digitise an ellipse with an equation $\frac{x^2}{1000} + \frac{y^2}{81} = 1$.

soln Given, $r_x = \sqrt{1000} = 10$ , $r_y = 9$

For region 1, we take starting pixel as $(0, 9)$.

For region 1, If $P_1 < 0$,

$x_{k+1} = x_k + 1$

$y_{k+1} = y_k$

$P_{k+1} = P_k + 2r_y^2 x_{k+1} + r_y^2$

If $P_1 \geq 0$

$x_{k+1} = x_k + 1$

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

$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$

Region 1 :

$P1_0 = r_y^2 + \frac{1}{4} r_x^2 - r_x^2 r_y = 9^2 + \frac{1}{4} \times (10\sqrt{10})^2 - (10\sqrt{10})^2 \times 9 = -8669$

| K | $P_K$ | $(x_{k+1}, y_{k+1})$ | $2r_y^2 x_{k+1} \geq 2r_x^2 y_{k+1}$ |
|---|---|---|---|
| 0 | $P1_0 = -8669$ | (1, 9) | $162 \geq 18000$ |
| 1 | $P1_1 = -8426$ | (2, 9) | $324 \geq 18000$ |
| 2 | $P1_2 = -8021$ | (3, 8) | $486 \geq 18000$ |
| 3 | $P1_3 = -7454$ | (4, 8) | |

Region 1 :

$P1_0 = r_y^2 + \frac{1}{4} r_x^2 - r_x^2 r_y = 81 + \frac{1}{4} \times 100 - 100 \times 9 = -794$

| K | $P_K$ | $(x_{k+1}, y_{k+1})$ | $2r_y^2 x_{k+1} \geq 2r_x^2 y_{k+1}$ |
|---|---|---|---|
| 0 | $P1_0 = -794$ | (1, 9) | $162 \geq 1800$ |
| 1 | $P1_1 = -551$ | (2, 9) | $324 \geq 1800$ |
| 2 | $P1_2 = -146$ | (3, 9) | $486 \geq 1800$ |
| 3 | $P1_3 = 421$ | (4, 8) | $648 \geq 1600$ |
| 4 | $P1_4 = -450$ | (5, 8) | $810 \geq 1600$ |
| 5 | $P1_5 = 441$ | (6, 7) | $972 \geq 1400$ |
| 6 | $P1_6 = 94$ | (7, 6) | $1134 \geq 1200$ |
| 7 | $P1_7 = 109$ | (8, 5) | $1296 \geq 1000$ (TRUE) |

So, we shift to region 2.

Starting pixel for region 2 = (8, 5)

$P2_0 = 81\left(8 + \frac{1}{2}\right)^2 + 100(4)^2 - 100 \times 81 = -647.75$

| K | $P2_k$ | $(x_{k+1}, y_{k+1})$ |
|---|---|---|
| 0 | $P2_0 = -647.75$ | (9, 4) |
| 1 | $P2_1 = 110.25$ | (9, 3) |
| 2 | $P2_2 = -389.75$ | (10, 2) |
| 3 | $P2_3 = 930.25$ | (10, 1) |
| 4 | $P2_4 = 830.25$ | (10, 0) |

AM

Q11) Derive the criteria necessary to identify the condition when we leave region one and enter region two in case of mid-point ellipse algorithm.

→

The equation of an ellipse is given by $\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} = 1$

Or, $f_{ellipse}(x,y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 y^2$ ——— (i)

Here,

$f_{ellipse}(x,y) < 0$ if $P(x,y)$ is inside ellipse boundary.
$= 0$ if $P(x,y)$ is on ellipse boundary
$> 0$ if $P(x,y)$ is outside ellipse boundary

This ellipse function $f_{ellipse}(x,y)$ serves as the decision parameter we select next pixel along ellipse path according to the sign of ellipse function evaluated at midpoint between 2 candidate pixels.

The construction of ellipse has been divided as in region 1 and region 2 for this algorithm.

Region 1 exists where slope of tangent to ellipse satisfies,

$$\left|\frac{dy}{dx}\right| \geq 1$$

Here, the value of $\Delta x > \Delta y$. The algo evaluates whether to move horizontally or diagonally.

Region 2 exists where the slope of the tangent to the ellipse $(dy/dx)$ satisfies,

$$\left|\frac{dy}{dx}\right| < 1$$

Here, the value of $\Delta y > \Delta x$. The algo evaluates whether to move vertically or diagonally.

For testing the shift of region, we calculate slope at each point. Differentiating (i),

$$2r_y^2 x + 2r_x^2 y \frac{dy}{dx} = 0$$

$$\text{or, } \frac{dy}{dx} = -\frac{2r_y^2 x}{2r_x^2 y}$$

The angle at which the transition occurs corresponds to a slope of -1 (135° counter-clockwise from positive axis) i.e. $\frac{dy}{dx} = -1$

$$\therefore \text{or, } 2r_y^2 x = 2r_x^2 y$$

The decision to move out of region 1 is based on whether the increment in x has grown enough relative to y to across the boundary.

So, we move out of region 1 when $2r_y^2 x \geq 2r_x^2 y$.

Q12) How is DDA different from BLA?

| Feature | DDA Algorithm | Bresenham Algorithm |
|---|---|---|
| Approach | Uses floating point arithmetic to calculate pixel positions based on the line equation. | Uses integer arithmetic and a decision parameter to find the closest pixel to the line. |
| Accuracy | May introduce rounding errors due to floating point calculations. | Highly accurate as it avoids floating point operation and uses only integer calculations. |
| Speed | Slower because of float | Faster because of int |
| Implementation | Easier to implement as it directly follows line eqn | More complex to implement due to decision parameter logic but more efficient |
| Applications | Used in simple or educational scenarios where precision is less critical | Widely used in real time graphics rendering and hardware for its speed and accuracy. |

Q13) Write DDA Algorithm code for all eight cases.

→

Code:

```
void lineDDA(int xa, int ya, int xb, int yb)
{
   int dx = xb-xa, dy = yb-ya, steps, k;
   float xInc, yInc, x=xa, y=ya;

   if(dx>dy)
     steps = dx;
   else
     steps = dy;
   xInc = dx/steps;
   yInc = dy/steps;
   SetPixel (ROund(x), Round(y));
   for(k=0; k<steps; k++)
    {
      x = x+xInc;
      y = y+yInc;
      setPixel (Round(x), Round(y))
    }
}
```

Algorithm:

Start at a starting pixel, as given in line, or assume one end point as starting point. The algorithm accepts the two end point (4 values) of the line.

First, we calculate the difference in x and then the difference in 'y'. If change in x is more, we consider steps as dx and if change in y is more, we consider steps as dy. We calculate the change to be done by dividing change by steps. The calculated value is added to the

starting pixel point. The floating point values are rounded off to the nearest integer value. This process is repeated ~~until~~ (process of addition) until we find ending pixel.

Q14) How does Boundary Fill algorithm work? How is it different from Flood Fill algorithm?

→

Boundary Fill Algorithm:→

Start at a point inside a region and paint the interior pixel outward towards the boundary. It accepts as an input, the coordinate of an interior point (x,y), fill color, boundary color.

Starting from (x,y), the procedure tests the neighbouring positions to determine whether they are of the boundary color. If not, they are painted with fill color and their neighbours are tested. The process continues until all pixel upto the boundary color for the area have been tested.

```
void bFill (int x, int y, intbColor, intfColor)
{
   int currentPixel;
   currentPixel = getPixel(x,y);
   if(currentPixel != bColor AND currentPixel != fColor){
      SetPixel(x,y,fColor);
      bFill(x+1, y, bColor, fColor);
      bFill(x-1, y, bColor, fColor);
      bFill(x, y+1, bColor, fColor);
      bFill(x, y-1, bcolor, fColor);
   }
}
```

Differences:

1. Boundary Fill stops at a specified boundary color, while Flood Fill stops when pixel color changes from the start color.
2. Boundary Fill works for enclosed areas with distinct boundaries; Flood Fill works for regions with uniform color.
3. Boundary Fill requires a predefined boundary color; Flood Fill uses the initial pixel's color.
4. Boundary Fill is best for shapes with clear boundaries, while flood fill is ideal for uniform color regions.

Q15) How does scanline Polygon fill algorithm work?

→

The scanline polygon fill algorithm works as follows:

1. Intersection Points
For each scanline crossing the polygon, the algorithm finds the intersection points of the scanline with the polygon's edges.

2. Sorting
These intersection points are sorted from left to right.

3. Filling
Pixels between each pair of intersection points are filled with the specified color.

4. Handling Vertices
If a scanline passes through a verten, the algorithm checks if the intersecting edges are on the same or opposite sides of scanline.

5. Verten Adjustment

To resolve ambiguity at vertices, some polygon edges are shortened so that vertices counted as one intersection are distinguished from those that should be counted as two.

Q16) What is a four and eight connected approach? What are the limitations of Boundary Fill and Flood fill algorithm?

→

Four Connected Approach:

In a four connected approach, the algorithm considers only the four direct neighbours of a pixel : up, down, left and right.

- It is commonly used when connectivity needs to be simple and doesnot involve diagonal pixels.
- This approach may fail to fill diagonal gaps, leaving small unfilled regions in certain shapes.

8-connected Approach:

- In a 8-connected approach, the algorithm considers all eight neighbours of a pixel : up, down, left, right and the four diagonal neighbours.
- Ensures filling of all connected regions, including those connected diagonally.
- Can inadvertently fill areas that should remain unconnected if diagonally adjacent pixels belong to different regions

## Limitations of Boundry Fill Algo:

1. Boundary Dependency : Requires a clearly defined boundary color, which may not always exist.
2. Overlapping Boundaries : May fail if the boundary color overlaps with the fill color.
3. Stack Overflow : Recursive implementations can cause stack overflow for large regions.
4. Complex Boundaries : Inefficient for polygons with highly irregular boundaries.

## Limitations of Flood fill Algo:

1. Uniformity Requirement : The region must have a uniform starting color, or the algorithm may fail.
2. Performance : can be slow for large regions due to repeated checks
3. Memory Overhead : Recursive methods can lead to high memory usage, potentially causing stack overflow.
4. Color Bleed : May unintentionally spread into areas with similar colors if not properly constrained.

Q17) Write :

A) Bresenham's Line Drawing Algorithm for lines with slope less than one.

→

For $|m| <= 1$,

(i) Read $x_a, y_a, x_b, y_b$

(ii) Load $(x_0, y_0)$ into frame buffer (i.e. plot the first point)

(iii) Calculate constants $\Delta y$, $\Delta x$, $2\Delta y$ and $2\Delta y + 2\Delta x$. Obtain the first decision parameter $P_0 = 2\Delta y + \Delta x$.

(iv) At each $x_k$ along the line starting at $k = 0$ perform the following tests:

If $P_k < 0$, then the next point to plot is $(x_{k+1}, y_k)$ and

$$P_{k+1} = P_k + 2\Delta y$$

else the next point to plot is $(x_{k+1}, y_k - 1)$ and

$$P_{k+1} = P_k + 2\Delta y + 2\Delta x$$

(v) Repeat step (iv) $\Delta x$ times.

B) Mid point circle Algorithm

(i) Input radius $r$ and circle center $(x_0, y_0)$ and obtain the first point on the circumferences of a circle centered on the origin as $(x_0, y_0) = (0, r)$

(ii) Calculate the initial value of the decision parameter as,

$$P_0 = 1 - r$$

(iii) At each $x_k$ position, starting at $k = 0$, perform the following test:

If $P_k < 0$, the next point along the circle centered on $(0, 0)$ is $(x_{k+1}, y_k)$ and

$$P_{k+1} = P_k + 2x_{k+1} + 1$$

Otherwise, the next point along the circle is $(x_{k+1}, y_k - 1)$ and,

$$P_{k+1} = P_k + 2x_{k+1} - 2y_{k+1} + 1$$

where,

$$2x_{k+1} = 2x_k + 2 \text{ and } 2y_{k+1} = 2y_k - 2$$

(iv) Determine symmetry points in the other seven octants.

(v) Move each calculated pixel position $(x, y)$ onto the circular path centered on $(x_c, y_c)$ and plot the coordinate values.

$$x = x + x_c$$

$$y = y + y_c$$

(vi) Repeat steps 3 through 5 until $X >= Y$.

c) Midpoint Ellipse Algorithm

(i) Input : Read ~~$x_a, y_a, x_b, y_b$~~ $r_x$ and $r_y$.

(ii) Start at $(0, r_y)$ take unit steps in 'x' direction until we reach boundary region 1 and 2, then switch to unit steps in 'y' direction for remainder of the curve in the first quadrant. There will be change of region from 1 to 2 when $2r_y^2 x \geq 2r_x^2 y$.

(iii) In region 1,

Initial decision parameter = $r_y^2 + \frac{1}{4} r_x^2 - r_x^2 r_y$

If $P1_k < 0$, $y_{k+1} = y_k$

$$x_{k+1} = x_k + 1$$

$$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} + r_y^2$$

If $P1_k >= 0$, $y_{k+1} = y_k - 1$

$$x_{k+1} = x_k + 1$$

$$P1_{k+1} = P1_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

(iv) In region 2,

Initial decision parameter = $r_y^2 \left(x_0 + \frac{1}{2}\right)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$

If $P2_k <= 0$, $x_{k+1} = x_k + 1$

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

$P2_{k+1} = P2_k - 2r_x^2 y_{k+1} + 2r_y^2 x_{k+1} + r_x^2$

If $P2_k > 0$, $x_{k+1} = x_k$

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

$P2_{k+1} = P2_k - 2r_x^2 y_{k+1} + r_x^2$

(v) Repeat process (iii) & (iv) until $y = 0$ in region 2.