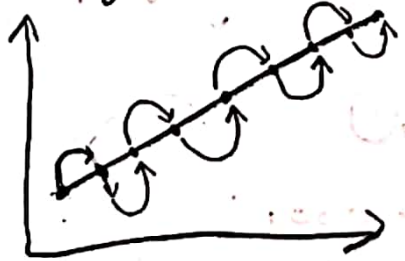
D.D.A Algorithm

→ Digital Differential Analyzer. (Base on technique of num. methods of D.E).

→ The vector generation algorithm in which steps along the line to determine the pixel which should be turned on are sometimes called DDA.

→ It is based on the principle that the two end points (x_1, y_1) & (x_2, y_2) are simultaneously incremented by Δx & corresponding Δy.

पहले एक Question समझते हैं, jisse Algorithm आसानी से समझ आजाएगा!



Q. Consider a line from (0,0) to (4,6). Use the simple DAA algorithm to rastorize the point.

① $(x_1, y_1) = (0,0)$ & $(x_2, y_2) = (4,6)$

② $\Delta x = 4$, $\Delta y = 6$

③ length $= \Delta y = 6$

④ $x_{inc} = \frac{x_2 - x_1}{length} = \frac{4-0}{6} = \frac{2}{3} = 0.667$

$y_{inc} = \frac{y_2 - y_1}{length} = \frac{6-0}{6} = 1$

⑤ $x = x_1 + 0.5 * sign(\Delta x)$

$y = y_1 + 0.5 * sign(\Delta y)$

$i = 1 \rightarrow x = 0.5 \approx 1$

$y = 0.5 \approx 1$

$i = 2 \rightarrow x = 0.5 + 0.667$
$= 1.167 \approx 1$

$y = 0.5 + 1.0$
$= 1.5 \approx 2$

$i = 3 \rightarrow x = 1.167 + 0.667$
$= 1.834 \approx 2$

$y = 1.5 + 1.0$
$= 2.5 \approx 3$

Steps of DDA Algorithm:

1. Read the line and points (x_1, y_1) & (x_2, y_2)
2. $\Delta x = |x_2 - x_1|$
   $\Delta y = |y_2 - y_1|$
3. if $(\Delta x \geq \Delta y)$
   then length $= \Delta x$
   else length $= \Delta y$
4. Select the raster unit, ie.

$$\Delta x_{inc} = \frac{x_2 - x_1}{length} \qquad \Delta y_{inc} = \frac{y_2 - y_1}{length}$$

Note: Either Δx_{inc} or Δy_{inc} will be 1 because length is either $(x_2 - x_1)$ or $(y_2 - y_1)$

5. $x = x_1 + 0.5 * sign(\Delta x)$
   $y = y_1 + 0.5 * sign(\Delta y)$

→ The sign function makes the algorithm work in all 4 quadrants. It returns $-1, 0$, or 1.

→ The factor 0.5 makes it possible to round the value to the nearest integer rather than truncating them.

6. Now plot the points

```
for (i = 1
while (i ≤ length)
{
    plot (Integer(x), Integer(y))
    x = x + Δx_inc
    y = y + Δy_inc
    i = i + 1
}
```

7. STOP

i	x_i	y_i	x_{i+1}	y_{i+1}
1	1	1	1·16	1·5
2	1	2	1·82	2·5
3	2	3	2·48	3·5
4	3	4	3·14	4·5
5	3	5	3·80	5·5
6	4	6	4·46	6·6

∴ pts are → (0,0), (1, 1), (1, 2), (2, 3), (3, 4), (3, 5), (4, 6)

Advantages of DDA Algorithm:

→ It is faster than the direct use of line equation.

→ Since, it calculate points on the line without any floating point multiplication.

→ Simple to implement, does not require any special skills for implementation.

Disadvantage of DDA Algorithm:

→ Accuracy is poor.

→ It is orientation dependent, due to this end point accuracy is poor.

→ A floating point addition is still needed in determining each successive point which is time consuming.

→ Cumulative error due to limited precision in floating point representation.

(1.) Dashed Line : Dashed line can be drawn by using the DDA algorithm by leaving a space after every (4) pixels.

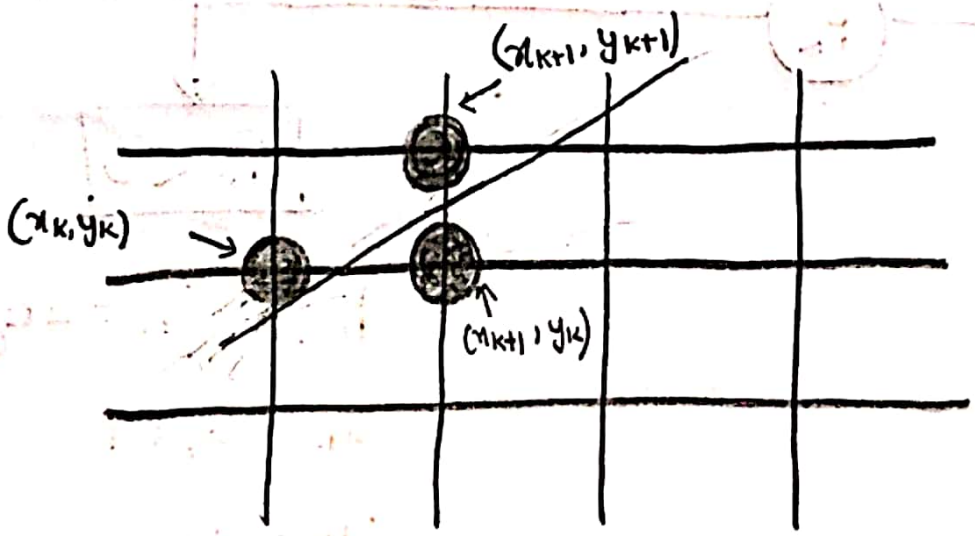
(2.) Dotted Line : Dotted line can be drawn using DDA algorithm by leaving a space after every (1) pixel.

Bresenham's Line Drawing Algorithm

→ It is an accurate & efficient way for rasterising lines.

→ The main principle on which this algorithm works is to find out the distance between the actual line location and the nearest pixel.

→ The distance is known as decision variable/error.



Q. A = (20, 10) B = (30, 18) Illustrate Bresenham's Algorithm.

Given, two end points : A (20,10) B (30,18)

(1) Find initial decision parameter :

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

$= 2 \times 8 - 10 = 6$

(2) Find Δx & Δy

$\Delta x = x_2 - x_1 = 30 - 20 = 10$

$\Delta y = y_2 - y_1 = 18 - 10 = 8$

(3) Find decision parameter

P_k

Case I: $P_k > 0$

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

$= 6 + 2 \times 8 - 2 \times 10$

$= 6 + 16 - 20 = 2$

$x_n = x_1 + 1$

$y_n = y_1 + 1$

Case II: $P_k < 0$

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

$x_n = x_1 + 1$

$y_n = y_1$

Algorithm :

1. Read the line & points (x_1, y_1) and (x_2, y_2) such that they are not equal.
2. Calculate $\Delta x = |x_2 - x_1|$ & $\Delta y = |y_2 - y_1|$
3. Initialize the starting point (x_0, y_0)
4. Calculate the initial decision parameter

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

5. If $P_k < 0$

{

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

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

Plot (x_{k+1}, y_k)

}

else if $P_k > 0$

{

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

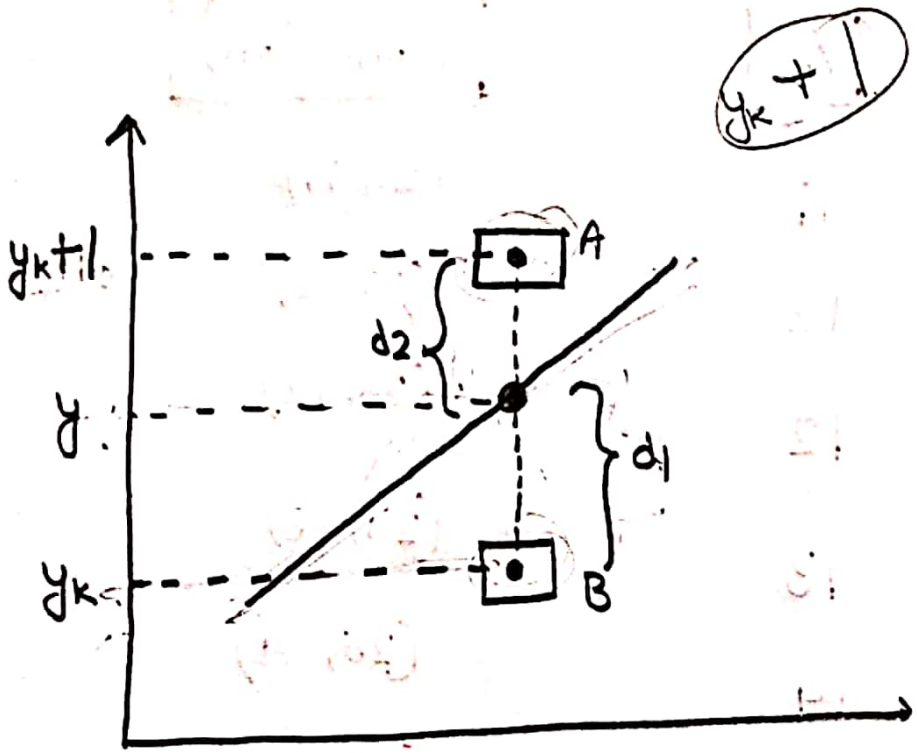
$y_{k+1} = y_k + 1$

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

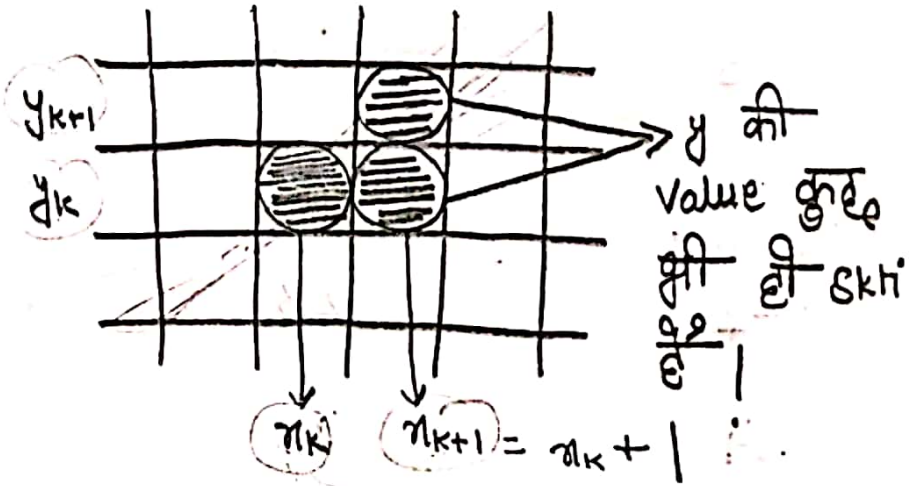
Plot (x_{k+1}, y_{k+1})

}

6. Repeat 5. Δx times
7. STOP



y_{k+1}



Case I: $0 < m < 1$

$0 < \frac{\Delta y}{\Delta x} < 1$

$\boxed{0 < \Delta y < \Delta x}$

* जो भी ज्यादा rate से increase hoga usme unit increment कर दो!

और Dusre का Nikalna Padega

P_k	x_n	y_n	(x_{n+1}, y_{n+1})
6	21	11	(21, 11)
2	22	12	(22, 12)
-2	23	12	(23, 12)
14	24	13	(24, 13)
10	25	14	(25, 14)
6	26	15	(26, 15)
2	27	16	(27, 16)
-2	28	16	(28, 16)
14	29	17	(29, 17)
10	30	18	(30, 18)

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

↓

Ye jo term है isme floating point आ सकता है तो कुछ करना पड़ेगा !

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

(x_1, y_1) (x_2, y_2)

$\Delta x = x_2 - x_1$

$\Delta y = y_2 - y_1$

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

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

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

P_k → Decision parameter : Ye bhai Apni मदद्त करेगा konsa Pixel lena , konsa नहीं लेना !

$$P_k = 2\Delta y\, x_k - 2\Delta x\, y_k + d$$

[∵ Δy, Δx → constant ques. में $(x_2 - x_1)$ दिया Rehta है! $(y_2 - y_1)$]

$$y = mx + c$$

at x_k, $y_k = mx_k + c$.

at x_{k+1}, $y_{k+1} = mx_{k+1} + c$

$$y_{k+1} = m(x_k + 1) + c \quad \text{(1)}$$

$\Delta x > \Delta y$

$0 < m < 1$

$0 < \frac{\Delta y}{\Delta x} < 1$

$\Delta y < \Delta x$

→ अब ध्यान देना हर एक step में :-

$$d_1 = y - y_k$$

$$= [m(x_k + 1) + c] - y_k$$

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

$$d_2 = y_{k+1} - y$$

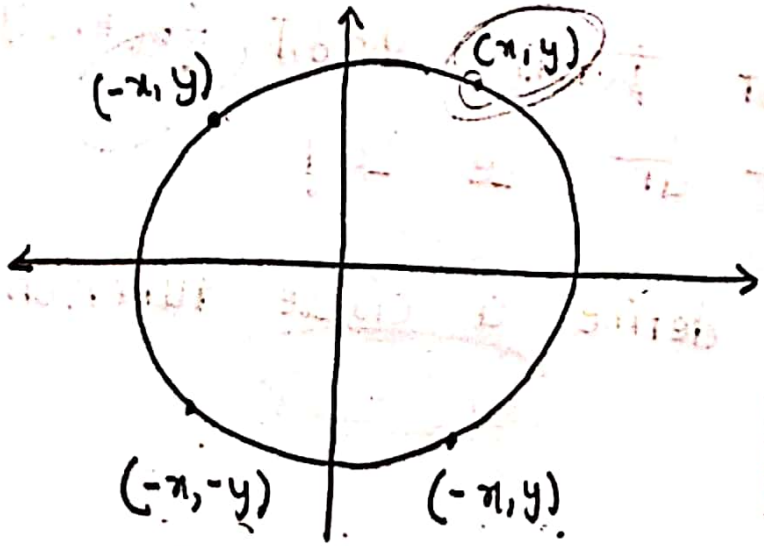
$$= (y_k + 1) - [m(x_{k+1}) + c]$$

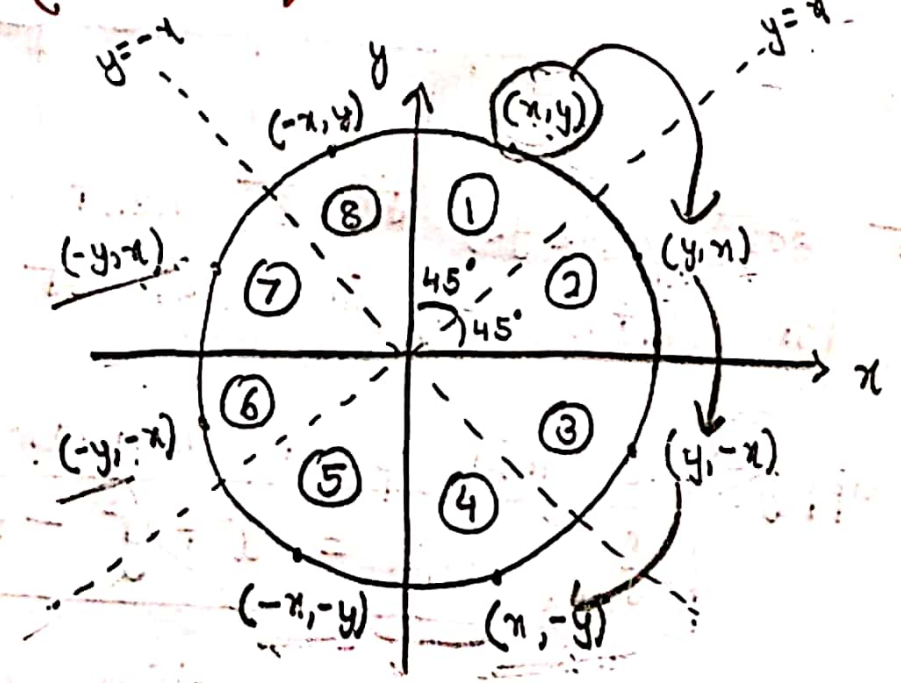
$$d_2 = (y_{k+1}) - [m(x_k + 1) + c]$$

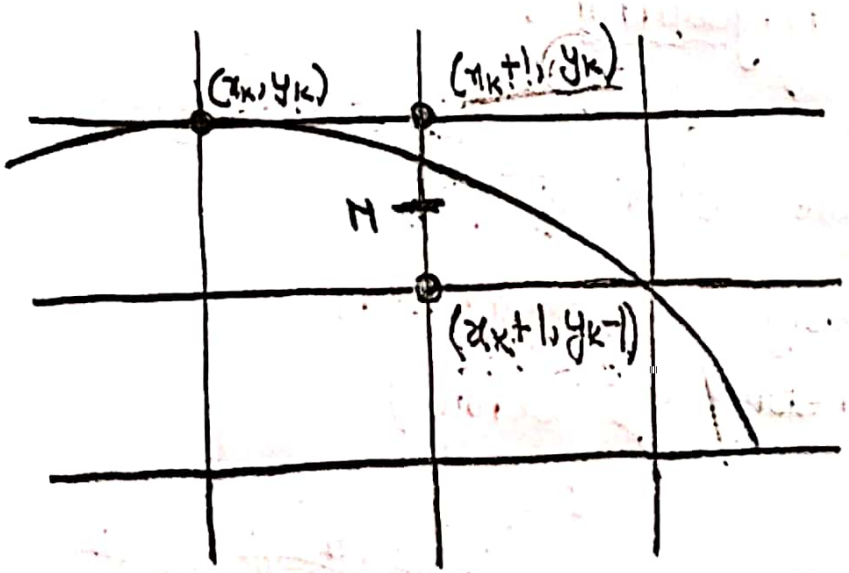
Mid Point Circle Drawing Algorithm :

Circle एक Symmetric figure होता है।

- 4 way Symmetry (4 Quadrants)
- 8 way Symmetry (8 octants)







अभी हम (x_k, y_k) में है, next point देखना पड़ेगा (x_k+1, y_k) या Fir (x_k+1, y_k-1). जो पास में हो वो Le Lo!

To apply the mid point method, we define a circle function:

$x^2+y^2 = r^2$

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

The value of this function can be -

$$f_{circle}(x,y) = \begin{cases} < 0, & \text{inside point} \\ = 0, & \text{surface में है} \\ > 0, & \text{outside point} \end{cases}$$

Calculating this function at the mid point :

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

This, P_k is the decision parameter for this algorithm.

$$P_k = x_k^2 + 2x_k + 1 + y_k^2 - y_k + \frac{1}{4} - r^2$$

$$= x_k^2 + y_k^2 + 2x_k - y_k - r^2 + \frac{5}{4} \quad \text{— (2)}$$

Case I: $P_k < 0 \Rightarrow$ the mid-point is inside the circle boundary

मतलब $\boxed{y_{k+1} = y_k}$

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

$y_{k+1} = y_k$

For next iteration -

$$P_{k+1} = \left(x_{k+1} + 1\right)^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2 \quad [\text{From eq}^n \text{ (1)}]$$

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

$$= x_k^2 + 4 + 4x_k + y_k^2 + \frac{1}{4} - y_k - r^2 \quad —$$

$$= x_k^2 + y_k^2 + 4x_k - y_k - r^2 + \frac{17}{4} \quad \text{(3)}$$

Eq^n (3) − (2)

$$P_{k+1} - P_k = 2(x_k + 1) + (y_{k+1}) \left(x_k^2 + y_k^2 + 4x_k - y_k - r^2 + \frac{17}{4} \right) -$$

$$\left(x_k^2 + y_k^2 + 2x_k - y_k - r^2 + \frac{5}{4} \right)$$

$$= 2x_k + 3$$

$$\boxed{P_{k+1} = P_k + 2x_k + 3}$$

Case II : $P_k > 0 \rightarrow$ $y_{k+1} = y_k - 1$.

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

$$= (x_k + 2)^2 + \left(y_k - \frac{3}{2}\right)^2 - r^2$$

$$= x_k^2 + 4 + 4x_k + y_k^2 + \frac{9}{4} - r^2 - 3y_k$$

$$= x_k^2 + y_k^2 + 4x_k - 3y_k - r^2 + \frac{25}{4} \quad \text{(4)}$$

Plot

Algorithm :

1. Input radius r and centre (x_c, y_c) and obtain the first point on the circumference of a circle centered at the origin $(0,0)$, the initial starting coord. (x_0, y_0) is equal to $(0, r)$.

2. Calculate the initial value of the decision parameter as

$$P_0 = \frac{5}{4} - r$$

3. do {

```
Plot(x, y)
if (p<=0)
  {
    x = x+1
    y = y
    p = p+2x+1
  }
else
  {
    x = x+1
    y = y-1
    p = p+2x-2y+1
  }
} while (x<y)
```

4. Determine the symm. points in other seven octants.

5. STOP

Eq^n (4) − (2)

$$P_{k+1} - P_k = \left(x_k^2 + y_k^2 + 4x_k - 3y_k - r^2 + \frac{25}{4}\right) - \left(x_k^2 + y_k^2 + 2x_k - y_k - r^2 + \frac{5}{4}\right)$$

$$= 2x_k - 2y_k + 5$$

$$\boxed{P_{k+1} = P_k + 2x_k - 2y_k + 5}$$

Now, Initial value for P_k:

$k=0$ $\quad x_k = 0 \;,\; y_k = r$

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

$$P_0 = (0+1)^2 + \left(r - \frac{1}{2}\right)^2 - r^2$$

$$= 1 + r^2 + \frac{1}{4} - r - r^2$$

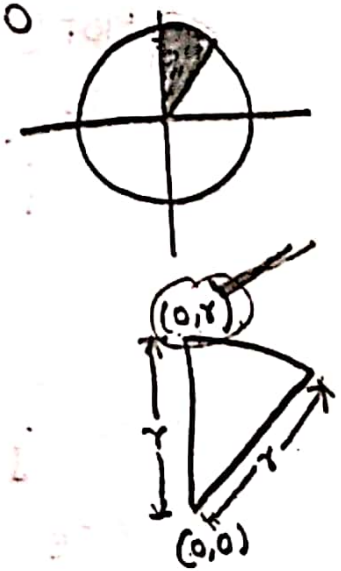
$$\boxed{P_0 = \frac{5}{4} - r}$$

P_0

P_{k+1}

$P_k > 0$

$P_k < 0$



P	X	Y	Plot
-8·75	1	10	(1,10)
-5·75	2	10	(2,10)
-0·75	3	10	(3,10)
6·25	4	9	(4,9)
-2·75	5	9	(5,9)
8·25	6	8	(6,8)
5·25	7	7	(7,7)

Q. Using mid point circle algorithm plot a circle whose radius is equal to 10, centered at the origin.

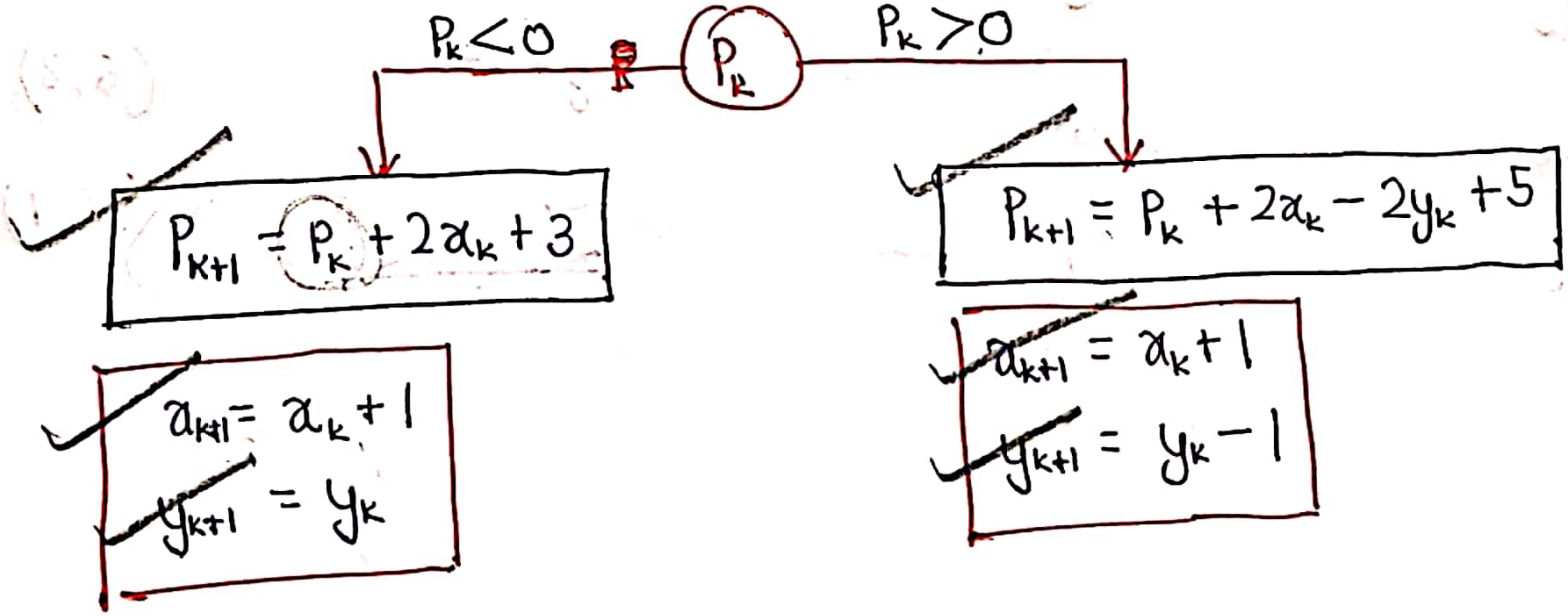
Given $r=10$ $(x_0, y_0) = (0,10)$

↓

[Plot (10,0), (0,-10), (-10,0)]

→ Calculate the initial decision Parameter:

$$P_0 = \frac{5}{4} - r = \frac{5}{4} - 10 = -8.75$$



$$P_k = 2\Delta y\, x_k - 2\Delta x\, y_k + d$$

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

$$P_{k+1} - P_k = \left[2\Delta y\, x_{k+1} - 2\Delta x\, y_{k+1} + d\right] - \left[2\Delta y\, x_k - 2\Delta x\, y_k + d\right]$$

$$= 2\Delta y\, x_{k+1} - 2\Delta x\, y_{k+1} + d - 2\Delta y\, x_k + 2\Delta x\, y_k - d$$

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

$$= 2\Delta y\, x_k + 2\Delta y - 2\Delta x\, y_{k+1} - 2\Delta y\, x_k + 2\Delta x\, y_k$$

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

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

If $P_k \rightarrow$ -ve then $y_{k+1} = y_k$

$P_k \rightarrow$ +ve then $y_k = y_k + 1$

∴ When P_k is -ve

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

When P_k is +ve

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

The initial value of decision parameter P_0 can be cal. as:

$$P_k = 2\Delta y\, x_k - 2\Delta x\, y_k + d$$

$$P_0 = 2\Delta y\, x_0 - 2\Delta x\, y_0 + d$$

$k=0$,

Also, the initial line equation: $y_0 = m x_0 + d$

$$y_0 = \frac{\Delta y}{\Delta x} x_0 + d \Rightarrow \boxed{d = y_0 - \frac{\Delta y}{\Delta x} x_0}$$

The initial value of decision parameter can be calculated as:

$$d1 - d2 = 2\frac{\Delta y}{\Delta x}(x_k + 1) - 2y_k + 2c - 1$$

$$\Delta x(d1 - d2) = 2\Delta y(x_k + 1) - 2\Delta x y_k + 2\Delta x c - \Delta x$$

$k = 0 \rightarrow$ $$P_0 = 2\Delta y(x_0 + 1) - 2\Delta x y_0 + 2\Delta x c - \Delta x$$

$$y = mx + c$$

$$y_0 = m x_0 + c \Rightarrow y_0 = \frac{\Delta y}{\Delta x} x_0 + c$$

$$c = y_0 - \frac{\Delta y}{\Delta x} x_0$$

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

Substituting the value of d in above equation:

$$P_0 = 2\Delta y x_0 - 2\Delta x y_0 + d$$

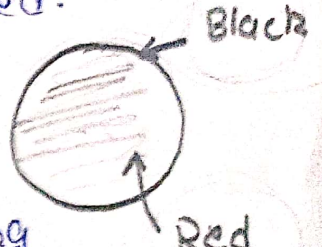
$$= 2\Delta y x_0 - 2\Delta x y_0 + y_0 - \frac{\Delta y}{\Delta x} x_0$$

$$=$$

Boundary Fill Algorithm:

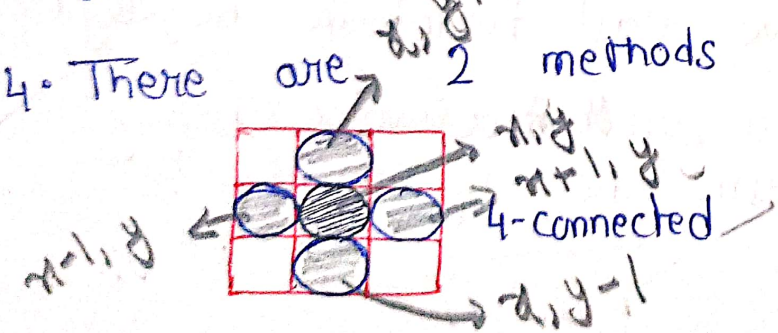
→ It is used where we have to do an interactive painting in Computer Graphics and the interior point can be easily selected.

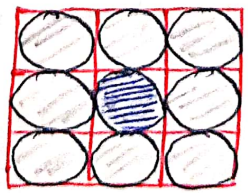
→ If we have a specified boundary in a single color, then the fill algorithm proceeds pixel by pixel until boundary color is encountered.



Steps for Boundary Fill Algorithm:

1. Start from any interior pixel (x, y) & test the neighbouring pixels to determine whether they are of Boundary colour.
2. If not, then point the pixel with fill colour and test their neighbouring pixels.
3. This process continues until all the pixels upto the boundary has been tested.
4. There are 2 methods for proceeding to the next pixel:





8-connected

Algorithm :

```
Boundary_fill (int x, int y, int fillcolor, int bcolor)
{
    if (getpixel(x,y) != bcolor && getpixel(x,y) != fillcolor)
    {
        putpixel(x,y, fillcolor);
        Boundary_fill (x+1, y, fillcolor, bcolor);
        Boundary_fill (x, y+1, fillcolor, bcolor);
        Boundary_fill (x-1, y, fillcolor, bcolor);
        Boundary_fill (x, y-1, fillcolor, bcolor);
    }
}
```

Drawbacks :—

1. अगर Pehle se hi कुछ interior point (pixels) fillcolor se fill है तो usse recursive call terminate ho jaega!
2. 4-connected & 8-connected methods involve heavy recursion which may consume memory & time!

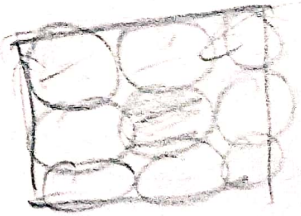
Flood Fill Algorithm :

→ It is used to fill an area where the boundary is not specified with a single color.

→ We start from a specified interior pixel (x,y) and reassign all pixel values that are currently set to a given interior color with the desired fill color.

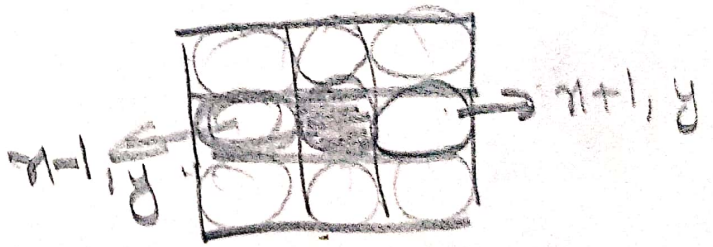
→ If the area has more than one interior color we can first reassign pixel values so that all interior pixels have the same color.

→ Using either 4-connected or 8-connected approach, we then step through pixel positions until all interior pixels have been repainted.



Algorithm :

```
void flood_fill (int x, int y, int oldcolor, int newcolor)
{       if (getpixel (x,y) == oldcolor)
        {
```



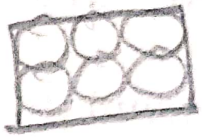
```
putpixel (x, y, newcolor);
floodfill (x+1, y, oldcolor, newcolor);
flood_fill (x-1, y, oldcolor, newcolor);
flood_fill (x, y-1, oldcolor, newcolor);
flood_fill (x, y+1, oldcolor, newcolor);
flood_fill (x+1, y+1, oldcolor, newcolor);
flood_fill (x+1, y-1, oldcolor, newcolor);
flood_fill (x-1, y+1, oldcolor, newcolor);
flood_fill (x-1, y-1, oldcolor, newcolor);
}
```

Polygon filling Algorithm :

→ Filling a polygon is a process of colouring each & every pixel of that comes inside the polygon region.

→ There are two basic approaches used for polygon filling :

- Seed fill algorithm
- Scan # line Algorithm



① Seed fill algorithm : (Seed मतलब कोई भी figure ka bich का point)

- Boundary fill algorithm
- Flood fill algorithm

~~→ There are two methods of filling a polygon~~

→ The initial step to fill a polygon is to test whether the point is inside or outside the polygon.

→ For testing two methods are used

- Inside out test
- Non-Zero winding number rule

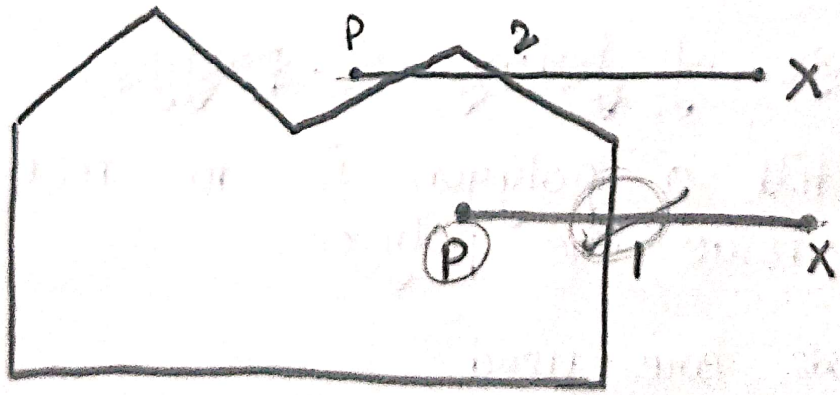
#Inside Out Test :

→ This is a simple method to test the point in this method we draw a line segment between the point P to be tested and a point outside the polygon. Now, count how many intersections of the line with the polygon boundary occurs.

if the number is even
then point P is exterior point

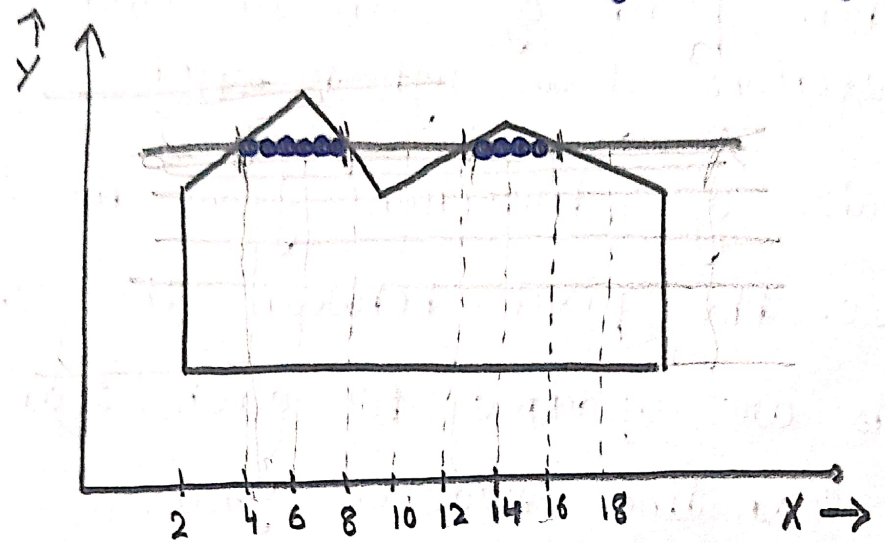
else
it is an interior point

→ It is also called "ODD PARITY CHECK".



Scan Line Polygon Filling Algorithm :

→ It is used for solid color filling in ~~algo~~ polygons.



→ Intersections = 4, 8, 12, 16

→ Sort करो = 4, 8, 12, 16

→ Pair बनाओ = (4,8) & (12,16)

Inke Bich jitne भी pixel हैं use fill कर दो!

Steps for Algorithm:

1. For each scan line, do
   (a) Obtain intersection point of scan line with polygon edges.
   (b) Sort the intersections from left to right.
2. Form the pairs of intersection from the list.
3. Fill within pairs, the pixels between the coordinates x and y.
4. Intersection points are updated for each scan line.
5. Stop when the scan line reaches y_{max}.

Data Structure used:

- Edge Table (ET)
- Active Edge Table (AET)