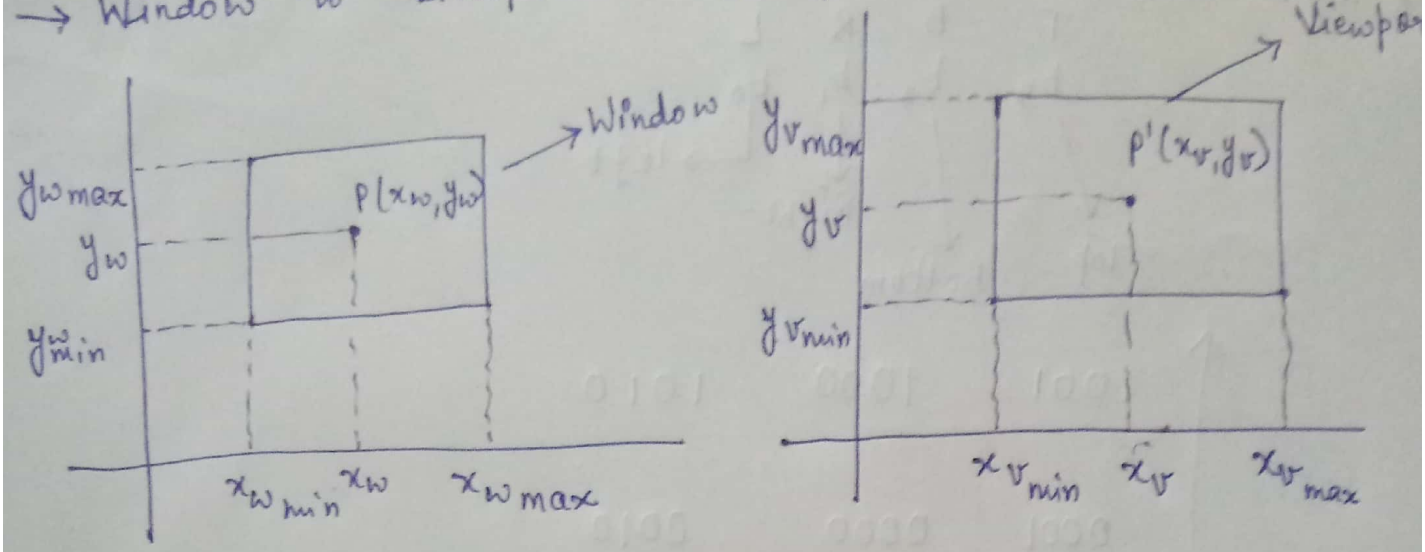
→ Window to Viewport mapping :-



Relative position is same.

$$\frac{x_w - x_{w_{min}}}{x_{w_{max}} - x_{w_{min}}} = \frac{x_v - x_{v_{min}}}{x_{v_{max}} - x_{v_{min}}}$$

Similarly for y.

→ Window → What to display! → clipping

→ Viewport → Where to display! → responsible for scaling.

→ Eg:- youtube ka minimize & maximize

→ Point clipping :-

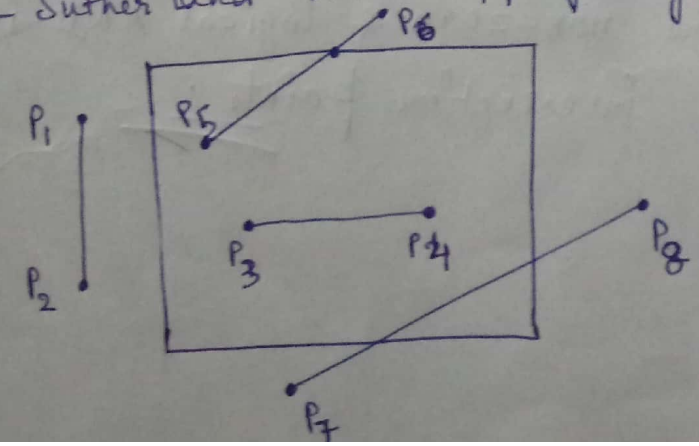
$P(x, y)$

i.e $x_{w_{min}} \leq x \leq x_{w_{max}}$

$y_{w_{min}} \leq y \leq y_{w_{max}}$

→ Line clipping :-

① Cohen-Sutherland Line Clipping Algorithm :-



$P_1 P_2$ → Reject

$P_3 P_4$ → Accept

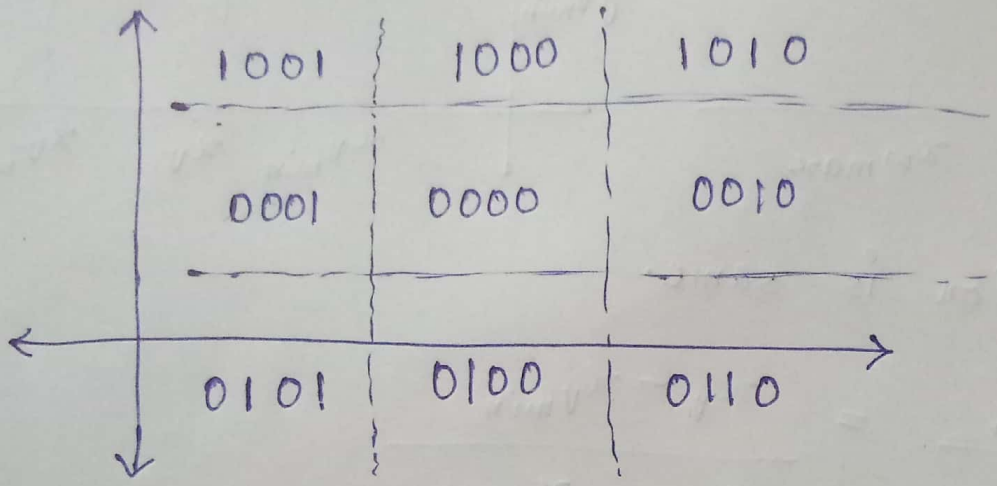
$P_5 P_6$ → Clipping Required

$P_7 P_8$ → Clipping Required

* Region Code :- (4-bit code)

T	B	R	L
b_3	b_2	b_1	b_0

b_0 → left, b_1 → Right, b_3 → Top, b_2 → Bottom



* If region code for both P_1 & P_2 are both 0, then, it is completely visible.

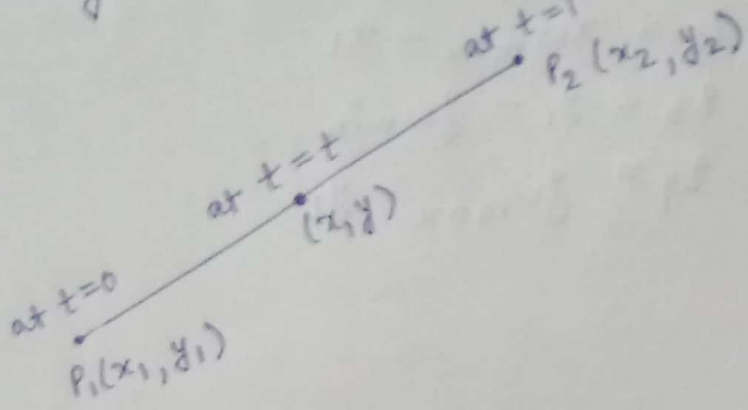
* If P_1 & P_2 are non-zero, logical AND operation of region code of P_1 & P_2 is also non-zero, then, line is exterior to the clipping window.

Eg :- $P_1 \rightarrow 0001 \longrightarrow$ Non-Zero
$P_2 \rightarrow 0001 \longrightarrow$ Non-Zero

AND $\rightarrow 0001 \longrightarrow$ Non-Zero.

* 3rd Case :-
a) One of P_1 & P_2 is 0.
b) Both are non-zero & logical AND is 0.
find the intersection points.

C → Liang - Barsky line clipping Algorithm :-



$$t_1 \le t \le t_2$$

Hence, $0 \le t \le 1$

$$x = t \cdot x_2 + (1-t)x_1$$

$$\Rightarrow x = t \cdot x_2 + x_1 - x_1 \cdot t$$

$$\Rightarrow \boxed{x = x_1 + t(x_2 - x_1)} \quad \text{i.e} \quad \boxed{x = x_1 + t \cdot \Delta x}$$

$$y = t \cdot y_2 + (1-t)y_1$$

$$\Rightarrow y = y_1 + t(y_2 - y_1) \quad \text{i.e} \quad \boxed{y = y_1 + t \cdot \Delta y}$$

Conditions :- for (x, y) to be an intersection point.

$$x_{w\,min} \le x_1 + t \cdot \Delta x \le x_{w\,max}$$

$$y_{w\,min} \le y_1 + t \cdot \Delta y \le y_{w\,max}$$

Now,

$x_1 + t\Delta x \ge x_{w\,min}$

$x_1 + t\Delta x \le x_{w\,max}$

$y_1 + t\Delta y \ge y_{w\,min}$

$y_1 + t \cdot \Delta y \le y_{w\,max}$

Derive the general eqn

$$t \cdot p_k \le q_k$$

where $k = 1, 2, 3, 4$.

$\rightarrow t \cdot \Delta x \ge x_{w\,min} - x_1$ — (I) $\rightarrow -t \cdot \Delta x \le x_1 - x_{w\,min}$

$\rightarrow t \cdot \Delta x \le x_{w\,max} - x_1$ — (II)

$\rightarrow t \cdot \Delta y \ge y_{w\,min} - y_1$ — (III) $\rightarrow -t \cdot \Delta y \le y_1 - y_{w\,min}$

$\rightarrow t \cdot \Delta y \le y_{w\,max} - y_1$ — (IV)

$p_1 = -\Delta x$	$q_1 = x_1 - x_{w\min}$
$p_2 = \Delta x$	$q_2 = x_{w\max} - x_1$
$p_3 = -\Delta y$	$q_3 = y_1 - y_{w\min}$
$p_4 = \Delta y$	$q_4 = y_{w\max} - y_1$

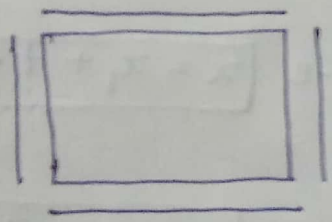
Case - ① :-

If $p_k = 0$ i.e $p_1, p_2, p_3, p_4 = 0$;

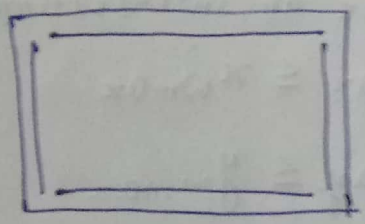
↓ line is ∥ to the clipping window.

Δx or $\Delta y = 0$

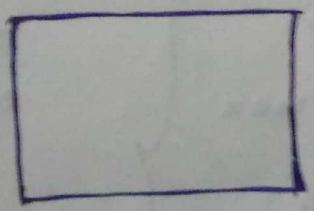
a) If $q_k < 0$, then, line is outside.



b) If $q_k > 0$, then, line may be inside or partially inside.



c) If $q_k = 0$, within the boundary or partial.



Case - ② :-

If $p_k < 0$, we need to find $t_1(x,y)$ → intersection point

& $t_1 = \max\left(0, \frac{q_k}{p_k}\right)$

$$t_1 \begin{cases} x = x_1 + t_1 \Delta x \\ y = y_1 + t_1 \Delta y \end{cases}$$

Case - ③ :-

If $p_k > 0$; we need to find $t_2(x, y)$ → due to intersection

&

$$t_2 = \min\left(1, \frac{q_k}{p_k}\right)$$

$$t_2 \begin{cases} x = x_1 + t_2 \Delta x \\ y_0 = y_1 + t_2 \Delta y \end{cases}$$

---

* Note :-

If $t_1 = 0$, then, no change & initial point is inside.

$t_1 \neq 0$, initial point is outside the window.

$t_2 = 1$, then, end point is inside.

$t_2 \neq 1$, then, end point is outside.

Q) $x_{w_{min}} = 5$ $\quad y_{w_{min}} = 5$

$x_{w_{max}} = 9$ $\quad y_{w_{max}} = 9$

Line $P_1(4, 12)$

$P_2(8, 8)$

A) $\Delta x = x_2 - x_1 = 4$

$\Delta y = -4$

$p_1 = -4$ $\quad q_1 = -1$

$p_2 = 4$ $\quad q_2 = 5$

$p_3 = 4$ $\quad q_3 = 7$

$p_4 = -4$ $\quad q_4 = -3$

$p_k < 0$ (P_1, P_4)

$$t_1 = \max\left(0, \frac{q_1}{p_1}, \frac{q_4}{p_4}\right)$$

$$= \max\left(0, \frac{1}{4}, \frac{3}{4}\right)$$

$$t_1 = \frac{3}{4}$$

$p_k > 0$ (P_2, P_3)

$$t_2 = \min\left(1, \frac{q_2}{p_2}, \frac{q_3}{p_3}\right)$$

$$= \min(1, 5/4, 7/4)$$

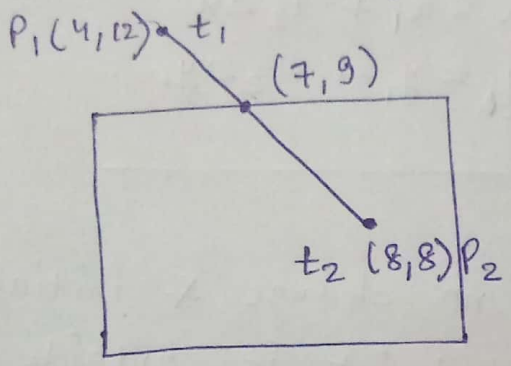
$$= 1$$

$\Rightarrow t = 1$

$\because t_2 = 1$, ending point is inside the window.

Now,

$$x = x_1 + t_1 \Delta x = 4 + (3/4) \cdot 4 = 7$$
$$y = y_1 + t_1 \Delta y = 12 + (3/4) \cdot (-4) = 9$$



→ Text clipping :-

TEXT CLIPPING

There are several techniques that can be used to provide text clipping in a graphics package. The clipping technique used will depend on the methods used to generate characters and the requirements of a particular application.

The simplest method for processing character strings relative to a window boundary is to use the *all-or-none string-clipping* strategy shown in Fig. 6-28. If all of the string is inside a clip window, we keep it. Otherwise, the string is discarded. This procedure is implemented by considering a bounding rectangle around the text pattern. The boundary positions of the rectangle are then compared to the window boundaries, and the string is rejected if there is any overlap. This method produces the fastest text clipping.

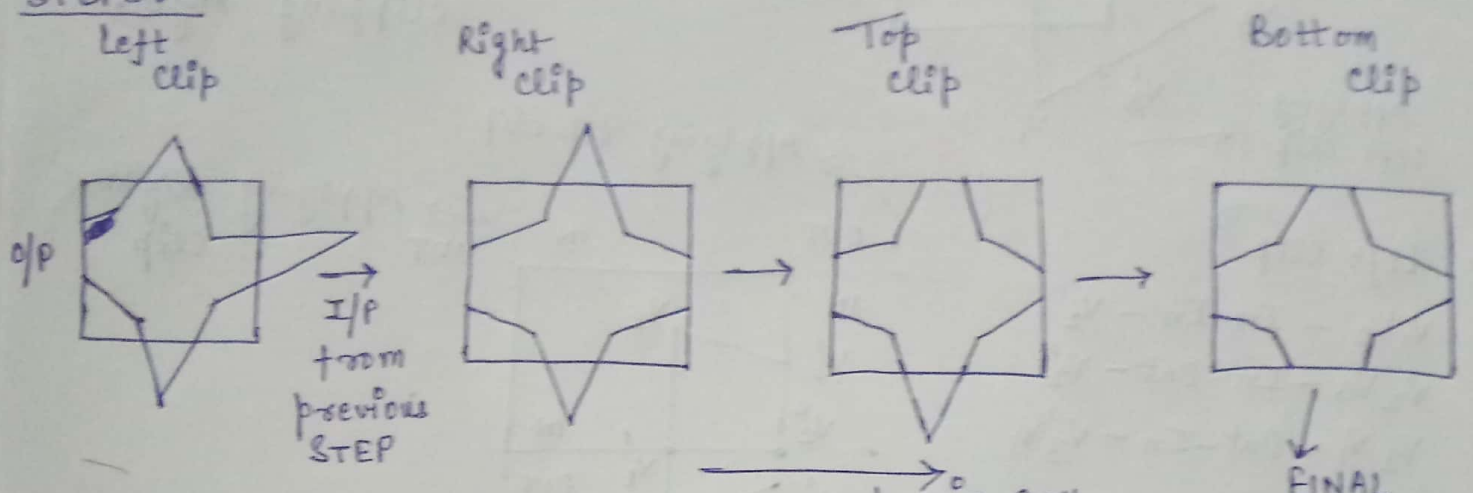
An alternative to rejecting an entire character string that overlaps a window boundary is to use the *all-or-none character-clipping* strategy. Here we discard only those characters that are not completely inside the window (Fig. 6-29). In this case, the boundary limits of individual characters are compared to the window. Any character that either overlaps or is outside a window boundary is clipped.

A final method for handling text clipping is to clip the components of individual characters. We now treat characters in much the same way that we treated lines. If an individual character overlaps a clip window boundary, we clip off the parts of the character that are outside the window (Fig. 6-30). Outline character fonts formed with line segments can be processed in this way using a line-clipping algorithm. Characters defined with bit maps would be clipped by comparing the relative position of the individual pixels in the character grid patterns to the clipping boundaries.

→ Polygon Clipping Algorithm :-

Sutherland Hodgeman Polygon Clipping Algorithm :-

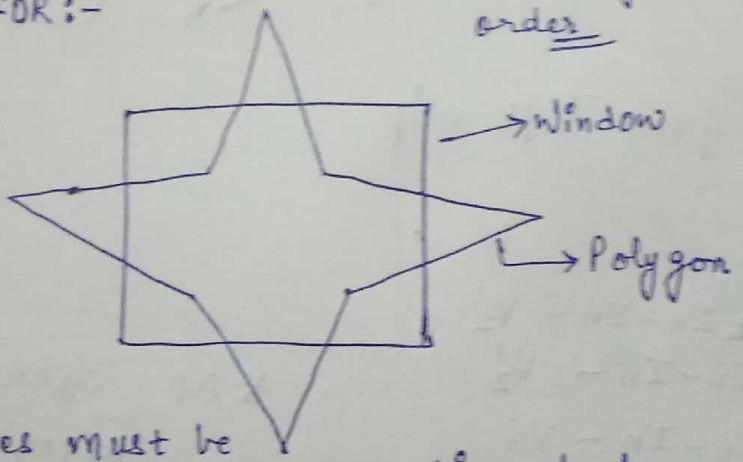
STEPS :-



DRAWBACK :-

*Best Suited for Convex Polygons & do not give accurate result in case of concave polygons

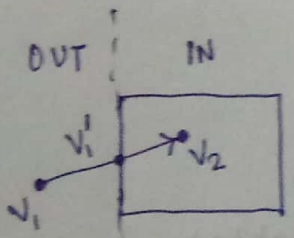
* FOR :-



DRAWBACK :-
Inclusion of extraneous lines

→ These cases must be tested for each case mentioned above.

*4 Cases to be Tested :-

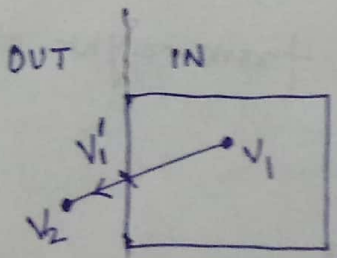


①

Output

(Out-in) → $V_1'V_2$

(must be saved)

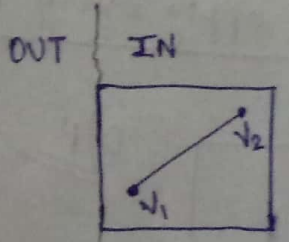


②

Output

(In-Out) → V_1' → Next Point

(must be saved)

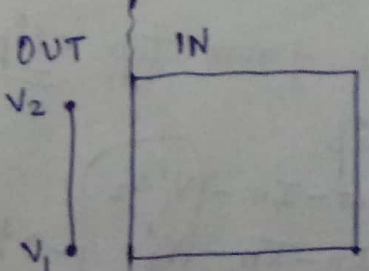


③

Output

(IN-IN) → V_2 → Next point is saved to show progress in our traversal.

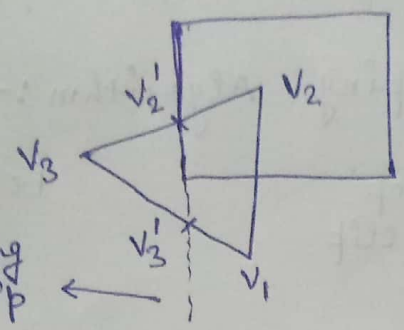
(must be saved)



Output (Out-Out)

NIL

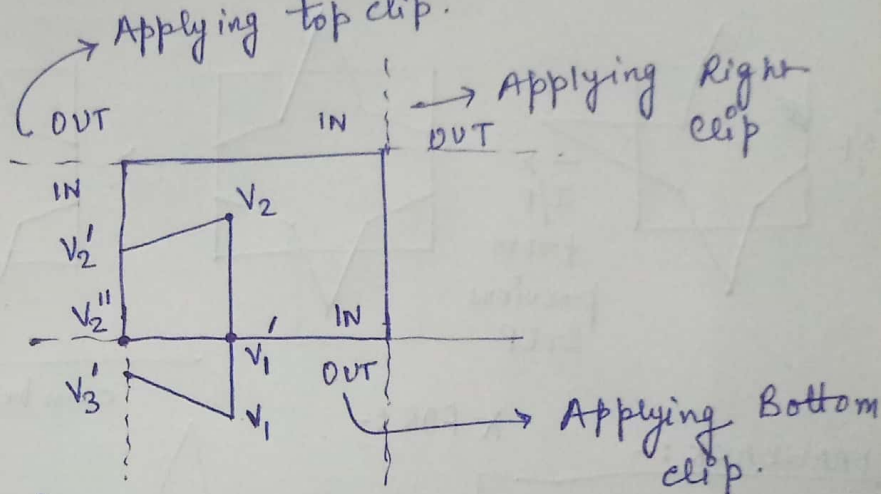
Q)



Apply Sutherland Hodgeman Polygon clipping Algo :-

A) Left clip

V_1V_2 – In-In – V_2
V_2V_3 – In-Out – V_2'
V_3V_1 – Out-In – $V_3'V_1$



⇓ Input from left clip

Right clip

V_1V_2 – In-In – V_2
V_2V_2' – In-In – V_2'
$V_2'V_3'$ – In-In – V_3'
$V_3'V_1$ – In-In – V_1

* No change after right clip.

⇓ Input from right clip.

Top clip

V_1V_2 – V_2
V_2V_2' – V_2'
$V_2'V_3'$ – V_3'
$V_3'V_1$ – V_1

* No change after top clip.

*Note :- Even if no change, we need to apply every clip.

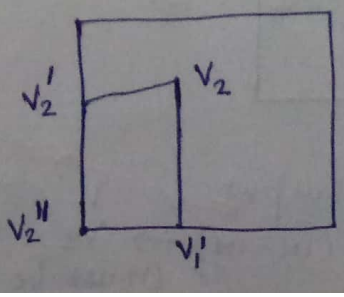
⇓ Input from top clip

Bottom clip

V_1V_2 – Out-In – $V_1'V_2$
V_2V_2' – In-In – V_2'
$V_2'V_3'$ – In-Out – V_2''
$V_3'V_1$ – Out-Out – Nil

→ final O/P

O/P

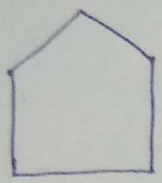


Weiler-Atherton Polygon Clipping Algo :-

Convex Polygon

→ Interior angle $< 180°$

Eg:-

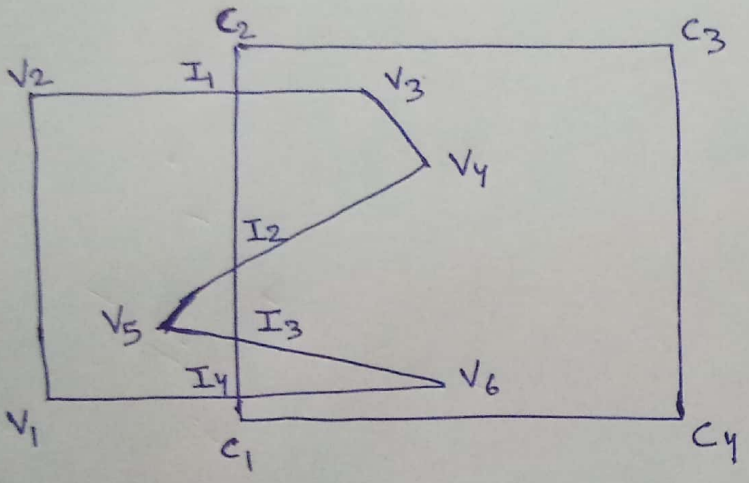


Concave Polygon

* One or more interior angle $> 180°$.

Eg:-

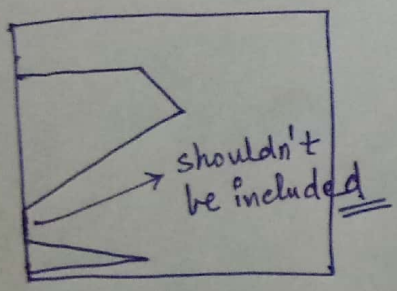
Eg:-



V_i – Vertices
C_i – clip polygon window
I_i – Intersection

*We will move in the clockwise dirⁿ.

Sutherland-Hodgeman Approach



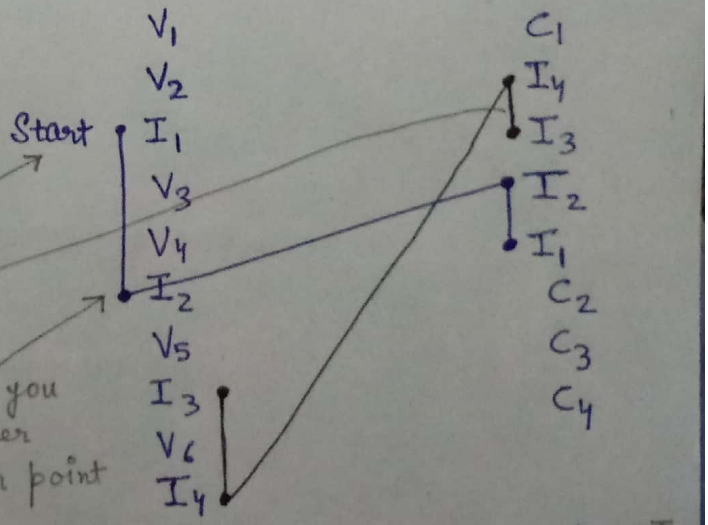
Weiler-Atherton Approach

Subject Polygon (with intersections) → Clipping Window

① → find 1st edge which is going from outside to inside. (while moving in clockwise dirⁿ)

② Move till you see another intersection point

Trace back the previous intersection point in the clipping window



O/P :- $I_1 \rightarrow V_3 \rightarrow V_4 \rightarrow I_2 \rightarrow I_1$

& $I_3 \rightarrow V_6 \rightarrow I_4 \rightarrow I_3$

DDA Algo

(x_1, y_1) (x_2, y_2)

(2, 2) (9, 2)

$m = 0$

$\Delta x = 7$

$\Delta y = 0$

$\Delta x > \Delta y$

$$m = \frac{\Delta y}{\Delta x} = \frac{0}{7} = 0$$

No. of steps = 7 ($\because \Delta x > \Delta y$)

$$x_{in} = \frac{\Delta x}{\text{No. of steps}} = \frac{7}{7} = 1$$

$$y_{in} = \frac{\Delta y}{\text{No. of steps}} = \frac{0}{7} = 0.$$

x	y
2	2
3	2
4	2
5	2
6	2
7	2
8	2
9	2

(x_1, y_1) (x_2, y_2)

$(2, 5)$ $(2, 12)$

$m = \infty$

$\Delta x = 2 - 2 = 0$

$\Delta y = 7$

$m = \frac{\Delta y}{\Delta x} = \frac{7}{0} = \infty$

Steps = 7 ($\because \Delta y > \Delta x$)

$x_{in} = \frac{0}{7} = 0$

$y_{in} = \frac{7}{7} = 1$

x	y
2	5
2	6
2	7
2	8
⋮	
2	12

→ Algorithm :-

```
DDA (x1, y1, x2, y2)
{
  dx = x2 - x1;
  dy = y2 - y1;
  if (abs(dx) > abs(dy))
  {
    step = abs(dx);
  }
  else
  {
    step = abs(dy);
  }
  xin = dx/step;
  yin = dy/step;
for (i=1; i<= step; i++)
{
  putpixel(x1, y1);
  x1 = x1 + xin;
  y1 = y1 + yin;
}.
```

(x_1, y_1) (x_2, y_2)

(5, 4) (12, 7)

$m < 1$

$\Delta x = 12 - 5 = 7$

$\Delta y = 7 - 4 = 3$

$m = \frac{\Delta y}{\Delta x} = \frac{3}{7}$

No. of steps = 7 ($\because \Delta x > \Delta y$).

$x_{in} = \frac{7}{7} = 1$

$y_{in} = \frac{3}{7} \approx 0.4$

*observation:-

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

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

x	y	Round (y)
5	4	4
6	4.4	4
7	4.8	5
8	5.2	5
9	5.6	6
10	6.0	6
11	6.4	6
12	6.8	7

(x_1, y_1) (x_2, y_2)

$(5, 7)$ $(10, 15)$

$m > 1$

$\Delta x = 5$

$\Delta y = 8$

$$m = \frac{\Delta y}{\Delta x} = \frac{8}{5}$$

Take absolute value in case of -ve.

Now, Steps = 8 ($\because \Delta y > \Delta x$).

$$x_{in} = \frac{5}{8} \approx 0.6$$

→ $1/m$.

$$y_{in} = \frac{8}{8} = 1.$$

Observation.

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

$$x_{k+1} = x_k + \frac{1}{m}$$

Round(x)	x	y
5	5	7
6	5.6	8
6	6.2	9
7	6.8	10
7	7.4	11
8	8.0	12
9	8.6	13
9	9.2	14
10	9.8	15

Rasters due to this rounding as a result of discrete pixel values.

Calculation of float takes extra time.

* When $m = 1$:-

diff. in x & y by 1

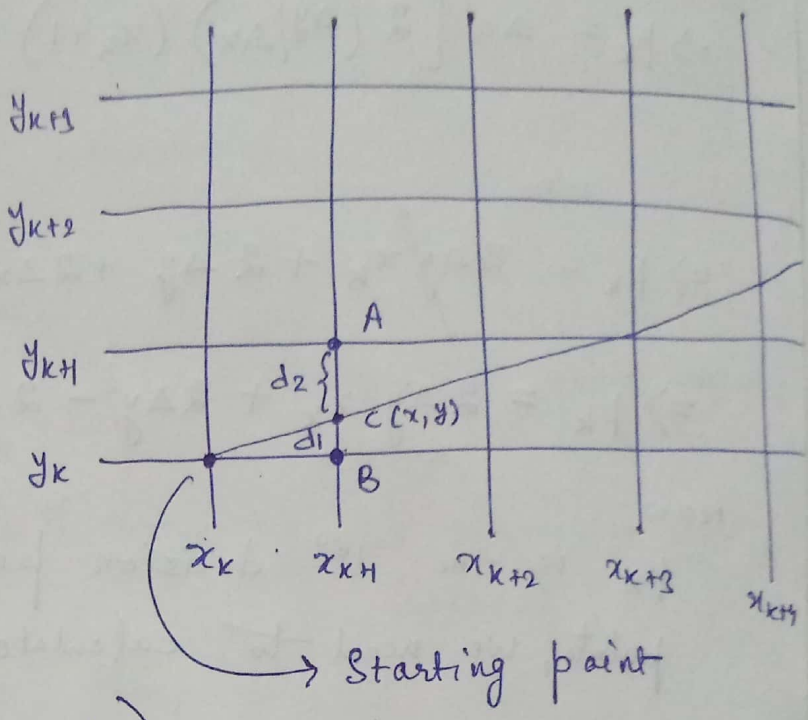
$\because \Delta x = \Delta y$.

→ Bresenham's line Drawing Algorithm :-

(x_s, y_s) (x_e, y_e)

$$m = \frac{y_e - y_s}{x_e - x_s}$$

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



* **Considering $m < 1$:-**

taking $y = mx + c$,

Now, $x = x_k + 1$

$\left(\because \frac{\Delta y}{\Delta x} < 1 \Rightarrow \Delta x > \Delta y\right)$.

Now, for value of y,

$$y = m(x_k + 1) + c. \quad \text{——— (1)}$$

A new term is introduced, the decision parameter (p_k). This parameter will take the decision, which grid line will be selected, i.e y_k or y_{k+1}

$$p_k = \Delta x (d_1 - d_2) \quad \text{——— (II)}$$

Now, $d_1 = y - y_k$

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

(∵ pixels are discrete ∴ $y_{k+1} = y_k + 1$).

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

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

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

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

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

Now, $p_k = \Delta x (d_1 - d_2)$

$\Rightarrow p_k = \Delta x \left[2 (\Delta y / \Delta x)(x_k + 1) + 2c - 2y_k - 1 \right]$

$\left(\because m = \frac{\Delta y}{\Delta x} \right)$

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

$\Rightarrow p_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + \Delta x (2c - 1)$ ——— (iii)

Now,
p_k is the 1st decision parameter. To find next points, we need to calculate next decision parameters

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

Now,

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

Now, $\because m < 1$

i.e $\frac{\Delta y}{\Delta x} < 1$

$\therefore$ x moves in unit interval).

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

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

We know that the initial point is (x_k, y_k). ii)

$p_k = 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + \Delta x \left[2(y - mx) - 1 \right]$

$= 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + \Delta x \left[2\left(y - \frac{\Delta y}{\Delta x} x\right) - 1 \right]$

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

Now, $\because$ initial point is (x_k, y_k), then, $x = x_k$, $y = y_k$.

then,

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

Now, from the decision parameter, we have,

If $p_k \geq 0$, then,

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

→ x moves in unit interval

→ y also moves in unit interval.

Next point $\equiv (x_{k+1}, y_{k+1})$

But, if $p_k < 0$, then,

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

→ only x moves in unit interval.

Next point $\equiv (x_{k+1}, y_k)$.

Q) (20,10) (30,18)

A) $p_k = 2\Delta y - \Delta x$

$= 16 - 10$

$= 6$

$\Delta x = 10$

$\Delta y = 8$

$m = 0.8$ [<1]

$p_k > 0$,

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

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

k	p_k	x	y
		20	10
0	6	21	11
1	2	22	12
2	-2	23	12
:	:	:	:
:	:	:	1
:	:	:	:
:	:	:	:
		30	18

$p_{k+1} = 2 + 16 - 20(12-11)$

$= 18 - 20$

$= -2$

Again, ~~$p_k = 2\Delta y - \Delta x$~~

~~$= 22 - 21$~~

~~$= 1$~~

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

$= 6 + 16 - 20(11-10) = 2$

* for $m < 1$,

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

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

If $p_k \geq 0 \Rightarrow y_{k+1} = y_k + 1$

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

If $p_k < 0 \Rightarrow y_{k+1} = y_k$

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

* for $m > 1$,

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

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

If $p_k \geq 0 \Rightarrow x_{k+1} = x_k + 1$

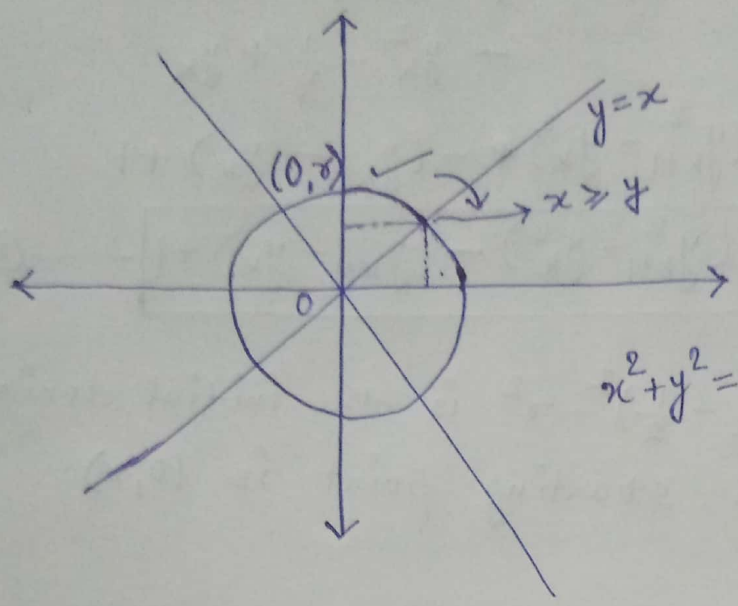
$$y_{k+1} = y_k + 1$$

If $p_k < 0 \Rightarrow x_{k+1} = x_k$

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

If $m = 1$, No calculation needed.
↳ both co-ordinates increase by 1.

→ Mid-point circle drawing algo :-



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

→ y is decreasing.

For 1st octant :-

x - unit intervals → ∵ $\Delta x > \Delta y$

y - ?

$\Rightarrow \frac{\Delta y}{\Delta x} < 1 \Rightarrow m < 1$

Now, We have to select whether y_k or $y_k - 1$ will be the next y-coordinate.

Hence, the next co-ordinate may be (x_k+1, y_k) or (x_k+1, y_k-1) depending on the decision parameter.

Now, Calculating mid-point of (x_k+1, y_k) & (x_k+1, y_k-1)

i.e Mid-point $\equiv \left(x_k+1, y_k - \frac{1}{2}\right)$

$f(x,y) = x^2+y^2-r^2$

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

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

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

$\therefore p_{k+1} - p_k = (x_{k+1}+1)^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - r^2 - (x_k+1)^2 - \left(y_k - \frac{1}{2}\right)^2 + r^2$

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

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

Keeping it same since this is what we need to decide.

$$= (x_k+1)^2 + 1 + 2(x_k+1) + y_{k+1}^2 + \frac{1}{4} - y_{k+1} - (x_k+1)^2 - y_k^2 - \frac{1}{4} + y_k$$

$$\Rightarrow p_{k+1} - p_k = 2(x_k+1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

$$\Rightarrow \boxed{p_{k+1} = p_k + 2(x_k+1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1} \quad \text{——} (1)$$

Now, $\because p_k = (x_k+1)^2 + (y_k - \frac{1}{2})^2 - r^2$ is the initial decision parameter and the starting point is $(0, r)$

$\therefore (0, r)$

$x_k \quad y_k$

$$p_k = (0+1)^2 + (r - \frac{1}{2})^2 - r^2$$

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

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

Now, ignoring the fractional part,

hence, $\boxed{p_k = 1 - r}$

↳ initial decision parameter

Next decision parameter will be calculated from eqⁿ – (1)

Then, after calculating, applying the below conditions :-

i.e $p_k \geq 0$, then, next co-ordinate $\equiv (x_k+1, y_k-1)$

& $p_k < 0$, then, " " $\equiv (x_k+1, y_k)$.

Stop when $x \geq y$, since we have reached the end of Octant 1. Apply the symmetricity property of the circle to calculate the pixel co-ordinate values of other quadrants / Octants.

→ Example on mid-point circle algorithm :-

For $r = 8$, we have,

$p_0 = 1 - r = 1 - 8 = -7$

↳ initial decision parameter.

→ initial point

K	(x_K, y_K)	p_K	(x_{K+1}, y_{K+1})
0	(0, 8)	−7	(1, 8)

$p_{K+1} = -7 + 2(0+1) + 0 - 0 + 1$

$= -7 + 2 + 1 = -4$

1	(1, 8)	−4	(2, 8)

$p_{K+1} = -4 + 2(1+1) + 0 - 0 + 1$

$= 1 > 0$

2	(2, 8)	1	(3, 7)

$p_{K+1} = 1 + 2(2+1) + (49 - 64) - (7 - 8) + 1$

$= 1 + 6 - 15 + 1 + 1$

$= -6 < 0$

3	(3, 7)	−6	(4, 7)

$p_{K+1} = -6 + 2(3+1) + 0 - 0 + 1$

$= 3 > 0$

4	(4, 7)	3	(5, 6)

$p_{K+1} = 3 + 2(4+1) + (36 - 49) - (6 - 7) + 1$

$= 2 > 0$

5	(5, 6)	2	(6, 5)

↳ $\because x \geq y$

Stop for 1st Octant.

End point for 1st Octant.

Q_1	Q_2	Q_3
(0,8)	(0,8)	(0,8)
(1,8)	(-1,8)	
(2,8)	(-2,8)	
(3,7)	(-3,7)	
(4,7)	(-4,7)	
(5,6)	(-5,6)	
(6,5)	(-6,5)	
(7,4)	(-7,4)	
(7,3)	(-7,3)	
(8,2)	(-8,2)	
(8,1)	(-8,1)	
(8,0)	(-8,0)	

→ Scaling :-

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$= \begin{bmatrix} x S_x \\ y S_y \end{bmatrix}$$

$S_x, S_y \rightarrow$ Scaling factors

If $S_x = S_y \rightarrow$ Uniform Scaling

Differential Scaling ← $S_x \neq S_y \rightarrow$ Non-Uniform Scaling

If $S_x, S_y > 1 \rightarrow$ Object Size increases.

$S_x, S_y < 1 \rightarrow$ Object Size decreases.

Scaling factors with values < 1 move objects closer to the co-ordinate origin while values > 1 move co-ordinate positions farther from the origin.

* Scaling w.r.t to a point :-

We can control the location of a scaled object by choosing a position (called the FIXED POINT) that is to remain unchanged after the scaling transformation.

{ Co-ordinates for the fixed point (x_f, y_f) can be choosen as one of the vertices, the object's centroid or any other arbitrary position.

* A polygon is then scaled relative to the fixed point (by scaling the distance from each vertex to the fixed point)

$$x' = x_f + (x - x_f) S_x \qquad y' = y_f + (y - y_f) S_y$$

$(x - x_f)$ → Distance b/w the 2 points is being scaled.

x_f ↓ due to relative Scaling x_f is added.

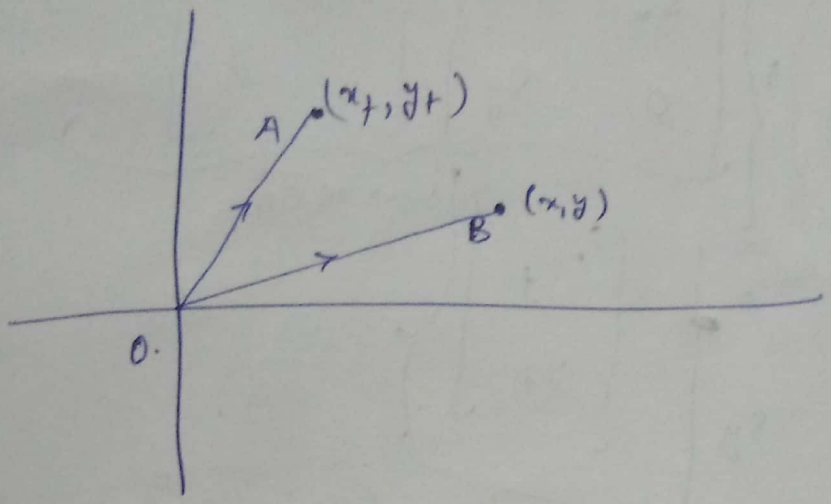
* Intuitive Explanation :-

Also,

$\because \quad x' = x_f + (x - x_f) S_x$

$y' = y_f + (y - y_f) S_y$

from R to L



$\because$ Scaling is done w.r.t origin by default.

Hence,

STEP-1 :- Shift (x_f, y_f) to origin,

Hence,

B changes to $(x - x_f)$

$(y - y_f)$

STEP-2 :- Scale this $\longrightarrow$

$(x - x_f) S_x$

$(y - y_f) S_y$

STEP-3 :- Reverse STEP-1 $\longrightarrow$

$x' = x_f + (x - x_f) S_x$

$y' = y_f + (y - y_f) S_y$

→ Reflection :-

① About x-axis :-

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Make y as -ve of y. (x unchanged)

② About y-axis :-

Make x as -ve of x. (y unchanged)

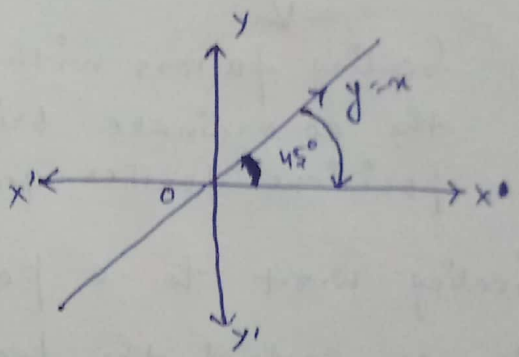
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

③ About origin :-

(i.e opposite quadrant)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

④ Reflection about $y = x$:-



STEP 1 :-

Rotate $y = x$ by 45° in clockwise dirⁿ.

$$\begin{bmatrix} \cos(-45) & -\sin(-45) \\ \sin(-45) & \cos(-45) \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} - (I)$$

STEP 2 :-

Take reflection about the axis. (Here, x-axis).

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} - (II)$$

Step-3 :- (Reverse Step-1)

i.e Rotate $y = x$ by 45° in anti-clockwise direction.

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} - (III)$$

$$FTM = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \text{ (Remember)}$$

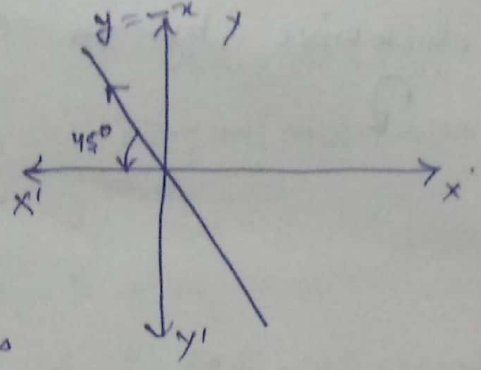
Multiply (I), (II) & (III) in this order to get final transformation matrix.

* NOTE :- Transformations done with the line are done with the points or vertices.

⑤ Reflection about $y = -x$:-

STEP 1 :-

Rotate $y = -x$ by $45°$ in anti-clockwise dirⁿ.



STEP 2 :-

Take reflection about x-axis

STEP 3 :- (Reverse Step-1)

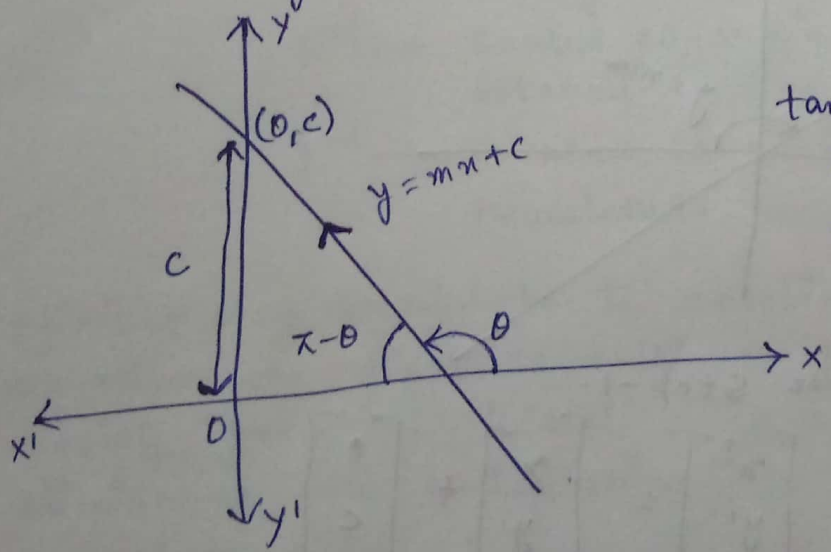
Rotate $y = -x$ by $45°$ in clockwise dirⁿ.

$$FTM \equiv \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{(Remember)}.$$

VVI

⑥ Reflection about $y = mx + c$:-



$\tan\theta = m$

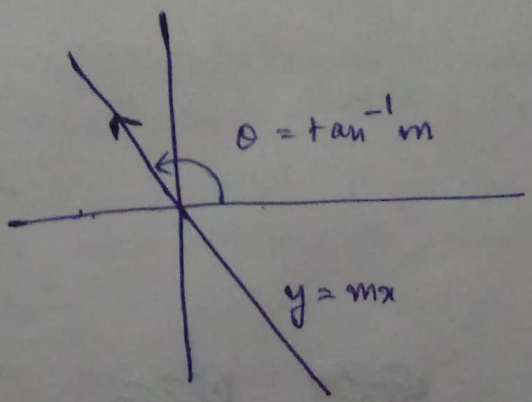
$\theta = \tan^{-1} m$

$\pi - \theta = \pi - \tan^{-1} m$

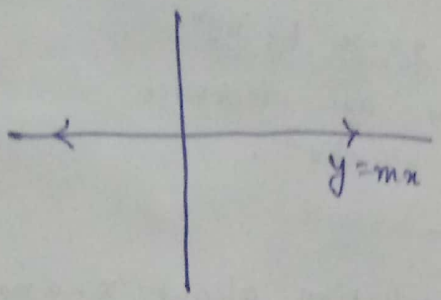
STEP - ① :-

Translate $(0,c)$ to $(0,0)$

i.e $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ -c \end{bmatrix}$



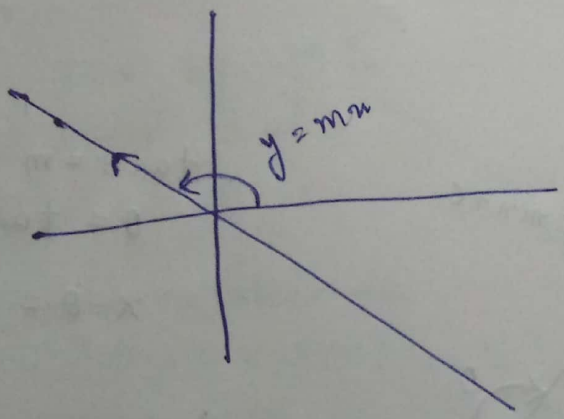
STEP 2 :-
Rotate clockwise by an angle of $\theta = \tan^{-1} m$.



STEP 3 :- Take reflection
↓
Main Step

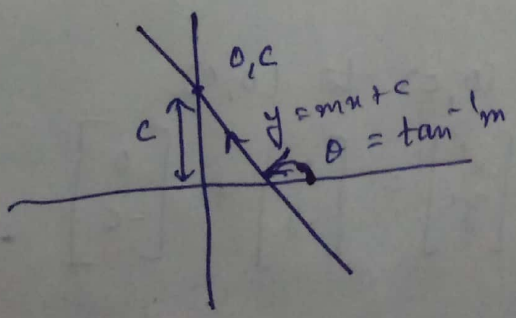
i.e
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

STEP 4 :- Reverse Step 2
i.e rotate anti-clockwise by a an angle of $\theta = \tan^{-1} m$.



STEP 5 :- Reverse Step-1.

i.e
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ c \end{bmatrix}$$



Homogeneous Co-ordinates

→ Each of the basic transformations can be expressed in the general matrix form

$$P' = M_1 \cdot P + M_2$$

with co-ordinate positions P' & P represented as column vectors.

M_1 → 2×2 matrix containing multiplicative factors.

M_2 → 2×1 matrix containing translational terms.

Now, to produce a sequence of transformations such as scaling followed by rotation & then translation, we must calculate the transformed co-ordinates one-step at a time.

i.e 1st → scaling is done

2nd → scaled co-ordinates are rotated.

3rd → Rotated co-ordinates are translated.

A more efficient way would be to combine the transformations so that the final co-ordinate positions are obtained directly from the initial co-ordinate positions, thereby eliminating the calculation of intermediate co-ordinate values.

* To be able to do this, we need to reformulate the above equations to eliminate the matrix addⁿ associated with the translation terms in M_2.

∴ We can combine the multiplicative and translational terms for 2-dimensional geometric transformations into a single matrix representation by expanding the 2×2 matrices to 3×3 matrices. This allows us to express all transformations as matrix product.

i.e $(x, y) \equiv (x_h, y_h, h)$

where, $x = \frac{x_h}{h}$

$y = \frac{y_h}{h}$

Now, $(x, y) \equiv (h \cdot x, h \cdot y, h)$

h is any non-zero value.

Hence, there are an ∞ no. of equivalent homogeneous representations for each co-ordinate point (x, y).

Let $h = 1$.

① For translation :-

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

i.e $P' = T(t_x, t_y) \cdot P$

The inverse of the translation matrix is obtained by replacing the translation parameters t_x & t_y by $-t_x$ & $-t_y$.

② For Rotation :-

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

i.e $P' = R(\theta) \cdot P$

For inverse, put $(-\theta)$ in place of θ.

③ For Scaling :-

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

i.e $P' = S(S_x, S_y) \cdot P$

For inverse put $1/S_x$ & $1/S_y$ in place of S_x & S_y respectively.

④ For reflection about x-axis :-

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

⑤ For reflection about y-axis :-

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

⑥ About Origin :-

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

⑦ About $y = x$:-

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

⑧ About $y = -x$:-

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 01 & 0 \\ 01 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$