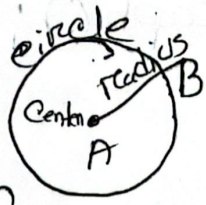
Scan Conversion:

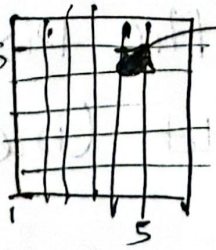
The process of representing continuous graphics object as a group of discrete pixels is called Scan Conversion.

eg: A line is represented by its two end points (P&Q) & line equation ($y = mx+c$).

while circle is represented by radius, centre position & circle equation.



Scan Conversion of point



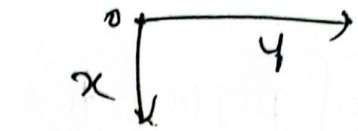
(5,5) is address of pixel

□ each box represented are pixel.

Fig: A pixel Display are of 8 x 7.

In graphics we us command as: put Pixel (int x, int y). Normally we use

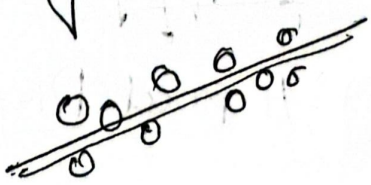
Right handed cartesian coordinate system.



Program to draw a pixel in graphics (c):

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
void main()
    { int gd=DETECT,gm;
  initgraph(&gd,&gm,"C:\\TC\\BGI");
  putpixel(150,150,Green); // putpixel(x1,
                                y1,Color)
  getch();
  closegraph();
  }
```

Scan Conversion line: It's- locate the coordinates of the pixel lie or near an ideal straight line.



DDA (Digital Differential analyzer) algorithm:

The DDA algorithm is an incremental scan conversion method used to draw a straight line on a raster display.

An incremental method means that each point (new) on the line is calculated using the previously computed point.

Working principle: Assume that at step i, the point (x_i, y_i) lies in one line

The next point (x_{i+1}, y_{i+1}) must satisfy the slope eqn:

$$m = \Delta y / \Delta x \cdot \text{ where } \Delta y = y_{i+1} - y_i, \quad \Delta x = x_{i+1} - x_i$$

we have $y_{i+1} = y_i + m\Delta x$

$$x_{i+1} = x_i + \Delta y / m$$

Case 1: When $|m| < 1$ (gentle slope)

- The line is more horizontal than vertical
- Increment x by 1 unit each step:

$$x_{i+1} = x_i + 1$$

Calculate y using: $y_{i+1} = y_i + m$

- Start from the left endpoint (x_1, y_1)
- Continue until x reaches x_2

Case 2: When $|m| > 1$ (steep slope)

- The line is more vertical than horizontal
- Increment y by 1 unit each step:

$$y_{i+1} = y_i + 1$$

- Calculate x using:

$$x_{i+1} = x_i + 1/m$$

- Start from the lower endpoint (x_1, y_1)
- Continue until y reaches y_2

Example: Draw a line from (2,2) to (8,6)

i. $dx = x_2 - x_1 = 8 - 2 = 6$

$dy = y_2 - y_1 = 6 - 2 = 4$

ii Steps = $\max(|dx|, |dy|) = 6$

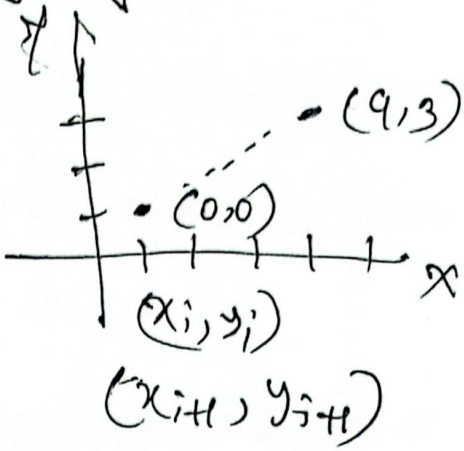
iii. $x_{inc} = \frac{dx}{steps} = 1$ $y_{inc} = \frac{dy}{steps} = 0.67$

$x = x + x_{inc}$

$y = y + y_{inc}$

plot(round(x), round(y))

Bresenham's Line Drawing algorithm:



1. $\Delta x = x_2 - x_1$
   $\Delta y = y_2 - y_1$
2. Decission Parameter P.
   $= 2\Delta y - \Delta x$
3. Case I: $P < 0$

   $x_{i+1} = x_i + 1$

   $y_{i+1} = y_i$

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

   Case II: $P \geq 0$

   $x_{i+1} = x_i + 1$

   $y_{i+1} = y_i + 1$

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

Until we reach end point

3.7. Indicate which raster locations would be chosen by Bresenham's algorithm when scan converting a line from pixel coordinate (1,1) to pixel coordinate (8,5).

Sol: First the starting values must be found.

In this case: $dx = x_2 - x_1 = 8-1 = 7$

$dy = y_2 - y_1 = 5-1 = 4$

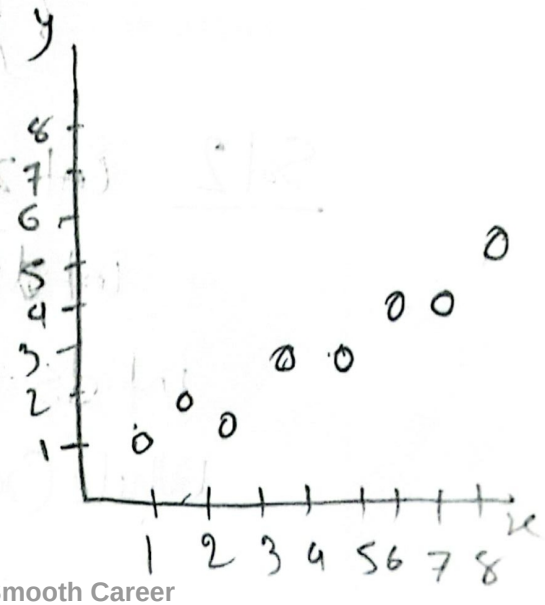
The Cons: $\ln c_1 = 2dy = 2 \times 4 = 8$

$\ln c_2 = 2(dy - dx) = 2 \times (4-7) = -6$

$d = \ln c_1 - dx = 8 - 7 = 1$

The following table indicates the values computed by the alg.

d	x	y
1	1	1
$1 + \ln c_2 = -5$	2	2
$-5 + \ln c_1 = 3$	3	2
$3 + \ln c_2 = -3$	4	3
$-3 + \ln c_1 = 5$	5	3
$5 + \ln c_2 = -1$	6	4
$-1 + \ln c_1 = 7$	7	4
$7 + \ln c_2 = 1$	8	5



3.9 Modify the description of Bresenham's Line algo in the text to set all pixels from inside the loop structure:

Sol:
```
int x=x1', y=y1';
int dx=x2'-x1', dy=y2'-y1', dT=2(dy-dx);
    ds=2dy;
int d=2dy-dx;
while (x<=x2') {
        Set_Pixel(x,y);
        x++;
    if (d<0)
      d=d+ds;
    else {
        y++;
        d=d+dT;
    }}
```

Sol 2:
```
int x=x1'-1, y=y1';
int dx=x2'-x1', dy=y2'-y1', dT=2(dy-dx);
int ds=2dy; d=-dx;
while (x<x2') {
        x++;
```

```
if (d<0)
    d = d+ds;
else{ y++;
    d = d+dT; }
SetPixel(x,y); }
```

3.11 What steps are required to scan convert a circle using the trigonometric method?

i. Set the initial variables: r = Circle radius; (h,k) = coordinates of the circle center; i = step size; $\theta_{end} = \pi/4$ rad $= 45°$; $\theta = 0$.

ii. Test to determine whether the entire circle has been scan converted. If $\theta > \theta_{end}$, Stop.

iii. Compute the value of the x and y coordinates: $x = r\cos(\theta)$, $y = r\sin(\theta)$

iv. 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)

v. Increment θ:

$\theta = \theta + i$

vi. Go to Step 2.

3.14] Is overstrike harmful besides wasting time?

It is often harmless since resetting a pixel with the same value does not really change the image in the frame buffer. However, if pixel values are sent out directly, for example, to control the exposure of a photographic medium such as a slide or a negative, then

Overstrike amounts to double exposure at locations where it occurred.

Furthermore, if we set pixels using their complementary colors, then overstrike would leave them unchanged, since complementing a color twice simply yields the color itself.

3.17 In the derivation of Bresenham's circle algorithm we have used a decision variable $d = D(T) + D(S)$ to help choose between pixels S & T. However, function D as defined in the text is not a true measure of the distance from the center of a pixel to the true circle.

Sol: Let d_S be the actual distance from S to the true circle and d_T be the actual distance from T

to the true circle. Also substitute x for x_i+1 and y for y_i in the formula for d_i to make the following proof easier to read. $d_i = 2x^2 + y^2 + (y-1)^2 - 2r^2 = 0$

Since $(r+dT)^2 = x^2 + y^2$ and $(r-dS)^2 = x^2 + (y-1)^2$

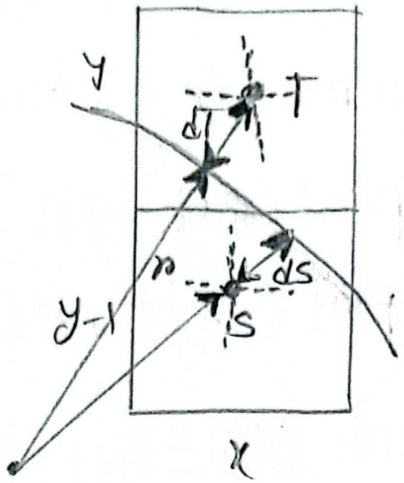
we have $2rdT + dT^2 = x^2 + y^2 - r^2$

and $-2rdS + dS^2 = x^2 + (y-1)^2 - r^2$

Hence $2r\,dT + dT^2 - 2rdS + dS^2 = 0$

$dT(2r + dT) = dS(2r - dS)$

since $dT/dS = (2r - dS)/(2r + dT) < 1$, we have $dT < dS$. This means that, when $d_i = 0$, pixel T actually closer to the true circle than pixel S.



3.21 What steps are required to scan convert an ellipse using the trigonometric method?

Sol: i. Set the initial variables: a = length of major axis; b = length of minor axis; (h, k) = coordinates of ellipse center; i = counter step size; $\theta_{end} = \pi/2$; $\theta = 0$.

ii. Test to determine whether the entire ellipse has been scan-converted. If $\theta > \theta_{end}$, stop.

iii. Compute the values of the x and y coordinates: $x = a\cos(\theta)$, $y = b\sin(\theta)$

iv. Plot the four points, found by symmetry, at the current (x, y) coordinates:

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

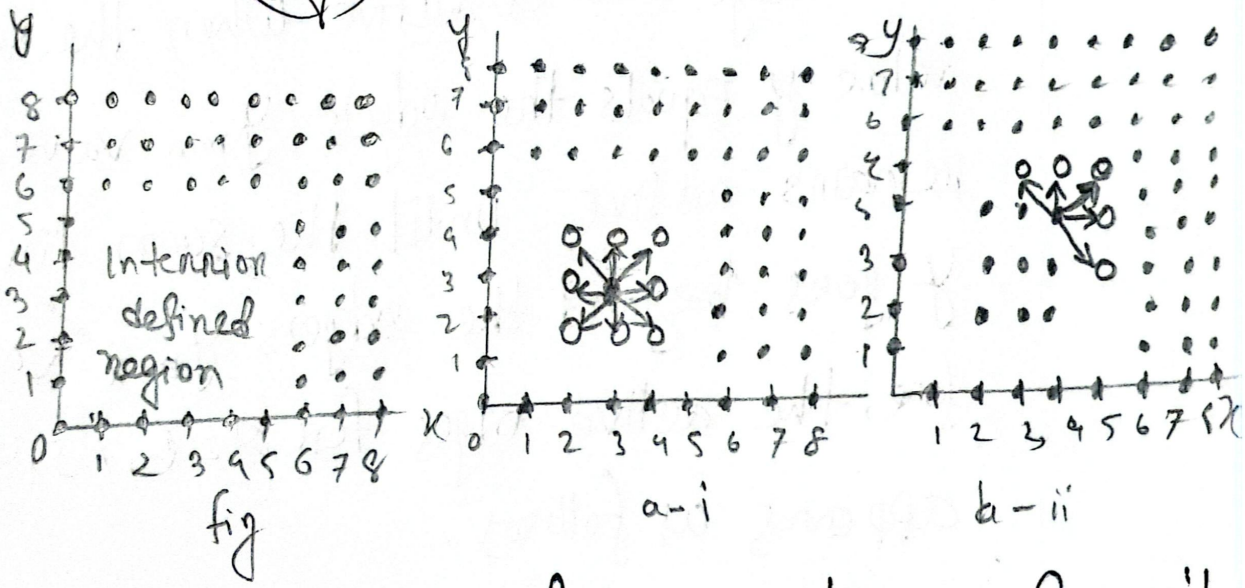
v. Increment θ: θ = θ+i.

vi. Go to Step 2.

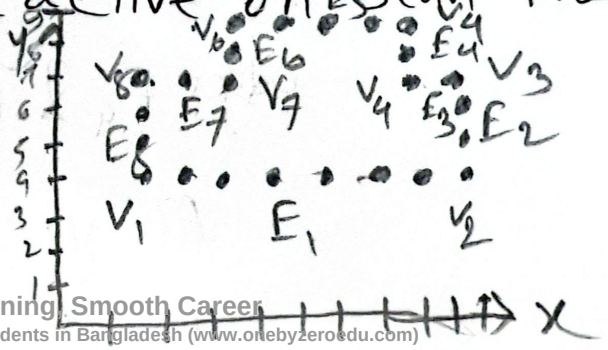
3.27 How would a flood fill algorithm fill the region shown in fig. using the 8 connected definition for region pixels?

Sol: i. Assume that a seed is given at coordinate 3, 3. The flood fill algorithm will inspect the eight points surrounding the seed (9,4; 3,4; 2,4; 2,3; 2,2; 3,2; 4,2; 4,3). Since all the points surrounding the seed have the region's original color, each point will be filled

ii. Each of the eight points found in step 1 becomes a new seed, and the points surrounding each new seed are inspected and filled. This process continues continues until all the points surrounding all the seeds are rid of the region's original color (fig)



3.30] The coordinates of the vertices of a polygon are shown in fig. (a) write the initial edge list for the polygon (b) State which edges will be active on scan lines $y = 6, 7, 8, 9$ & 10.



a) Column x contains the x coordinate of the corresponding edge's lower endpoint. Horizontal edges are not included.

Edge	y_{min}	y_{max}	x	1/m
E_2	4	7	9	0
E_8	4	7	2	0
E_4	7	9	8	0
E_6	7	9	4	0

10,

b) An edge becomes active when the scan line value y equals the edge's y_{min} value. The edge remains active until the scan line value y goes beyond the edge's y_{max} value. Therefore, the active edges for y = 6, 7, 8, 9 & 10 appears as follows.

At y = 6, E_2 & E_8

At y = 7, $y = y_{max}$ for both edges E_2 & E_8 so they remain active. Also at y = 7 edges E_4 & E_6 become active.

At y = 7, $y = y_{max}$ for both edges E_4 & E_6 so they remain active.

At $y=8$, E_2 & E_8 are removed from the edge list, E_4 & E_6 remain active.

$y=9$, the active edges remain the same. At $y=10$, edges E_2 & E_4 are removed from the edge list and the edge list becomes empty.

Supplementary problem:

3.38]