Aim: To implement Bresenham's algorithms for drawing a line segment between two given end points.

Objective:
Draw a line using Bresenham's line algorithm that determines the points of an n-dimensional raster that should be selected to form a close approximation to a straight line between two points

Theory:
In Bresenham's line algorithm pixel positions along the line path are obtained by determining the pixels i.e. nearer the line path at each step.

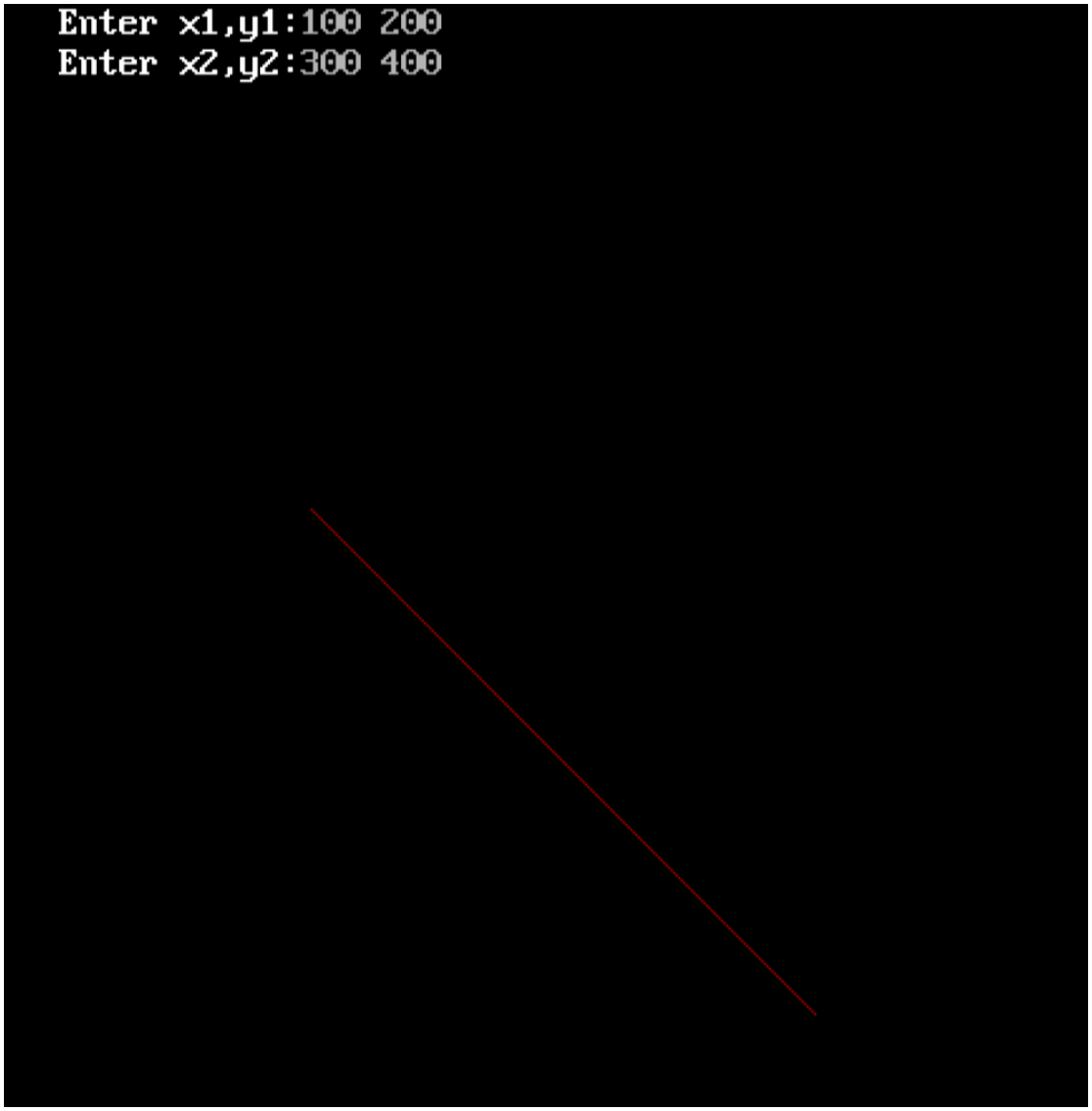
Algorithm –(x1,y1,x0,y0)
dx=x1-x0
dy=y1-y0
p0=2dy-dx
for k=0 to dx do
if pk<0 then
putpixel(xi+1,yi)
pn=pk+2dy
else
putpixel(xi+1,yi+1)
pn=pk+(2dy-2dx)
end
end

Program -

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>

void Bresenham(int x1, int y1, int x2, int y2) {
   int dx, dy, x, y, p, end;
   dx = abs(x1 - x2);
   dy = abs(y1 - y2);
   p = 2 * dy - dx;
   if (x1 > x2) {
      x = x2;
      y = y2;
      end = x1;
   } else {
```

```
        x = x1;
        y = y1;
        end = x2;
    }
    putpixel(x, y, 7);
    while (x < end) {
        x = x + 1;
        if (p < 0) {
            p = p + 2 * dy;
        } else {
            y = y + 1;
            p = p + 2 * (dy - dx);
        }
        putpixel(x, y, 7);
    }
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, RED);

    int x1, y1, x2, y2;
    printf("Enter the coordinates of the first point (x1 y1): ");
    scanf("%d %d", &x1, &y1);
    printf("Enter the coordinates of the second point (x2 y2): ");
    scanf("%d %d", &x2, &y2);

    Bresenham(x1, y1, x2, y2);


    getch();

    closegraph();
    return 0;
}
```

Output:



Conclusion: Comment on -

1. Pixel:-Bresenham's algorithm does not perform any rounding operation
2. Equation for line:-y=mx+c
3. Need of line drawing algorithm:-Involves cheaper operation like addition and subtraction
4. Slow or fast:-It is faster than DDA