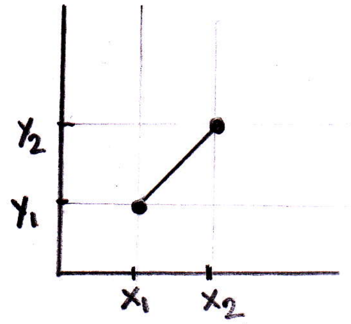
Digital Differential Analyzer (DDA) :- Digital Differential Analyzer (DDA) algorithm is the simple line generation algorithm which is explained here.

The cartesian slop-intercept equation for straight line is ⟶

$$y = mx + c;$$



Line path between end point positions (x_1, y_1) and (x_2, y_2)

$x_1 = x + 1$ And $y_1 = mx_1 + c$

$x_2 = x + 2$ $y_2 = mx_2 + c$

$\therefore \Delta x = x_2 - x_1$
$= x + 2 - x - 1$
$= 1$

$\therefore \Delta y = y_2 - y_1$
$= \cancel{m}mx_2 + \cancel{c} - mx_1 - \cancel{c}$
$= m(x_2 - x_1)$
$= m \Delta x$
$= m.$

Where, $m = \frac{y_2 - y_1}{x_2 - x_1}$ [∵ m representing the slop of the line]

```
void line(int x1, int y1, int x2, int y2, int value)
 {
    double y = y0;
    double m = (y2 - y1)/(x2 - x1);
    for (x = x1; x <= x2; x++){
       writePixel(x, ROUND(y), value);
       y += m;
    }
 }
```

Digital Differential Analyzer (DDA) :- Digital Differential Analyzer (DDA) algorithm is the simple line generation algorithm which is explained step by step here.

step-1 :- Get the input of two end points (x_0, y_0) and (x_1, y_1).

step-2 :- Calculate the difference between two end points.

$dx = x_1 - x_0$;

$dy = y_1 - y_0$;

step-3 :- Based on the calculated difference in step-2, need to identify the numbers of steps to put pixel. If($dx > dy$), then need more steps in x-cordinate; otherwise in y coordinates.

```
if(absolute(dx) > absolute(dy))
    steps = absolute(dx);
else
    steps = absolute(dy);
```

step 4 :- Calculate the increment in x-coordinate and y-coordinate

X increment = dx / (float) steps ;

Y increment = dy / (float) steps ;

step5 : Put the pixel by succenfully incrementing x and y coordinates accordingly and complete the drawing of the line.

```
for ( int k= 0; k< steps ; k++)
 {
        x = x + Xincrement;
        y = y + Yincrement;
        setpixel ( ROUND(x), ROUND(y));
 }
```

```
void drawDDA (int x0, int y0, int x1, int y1){
   int dx = x1-x0, dy = y1-y0, steps, k;
   float Xincrement, Yincrement, x= x0, y= y0;
   if( abs (dx) > abs(dy))
          steps = abs(dx);
     }
   else
        steps = abs (dy);
   Xincrement = dx/(float) steps;
   Yincrement = dy/ (float) steps;
   setPixel(ROUND(x), ROUND(y));
   for( int k=0 ; k< steps ; k++){
     x+= Xincrement;
     y+= Yincrement;
     setPixel (x, y);
  }
}
```