Bresenham Line Drawing

JAGJIT BHATIA
Dept. of computer Science & IT

Simple Line

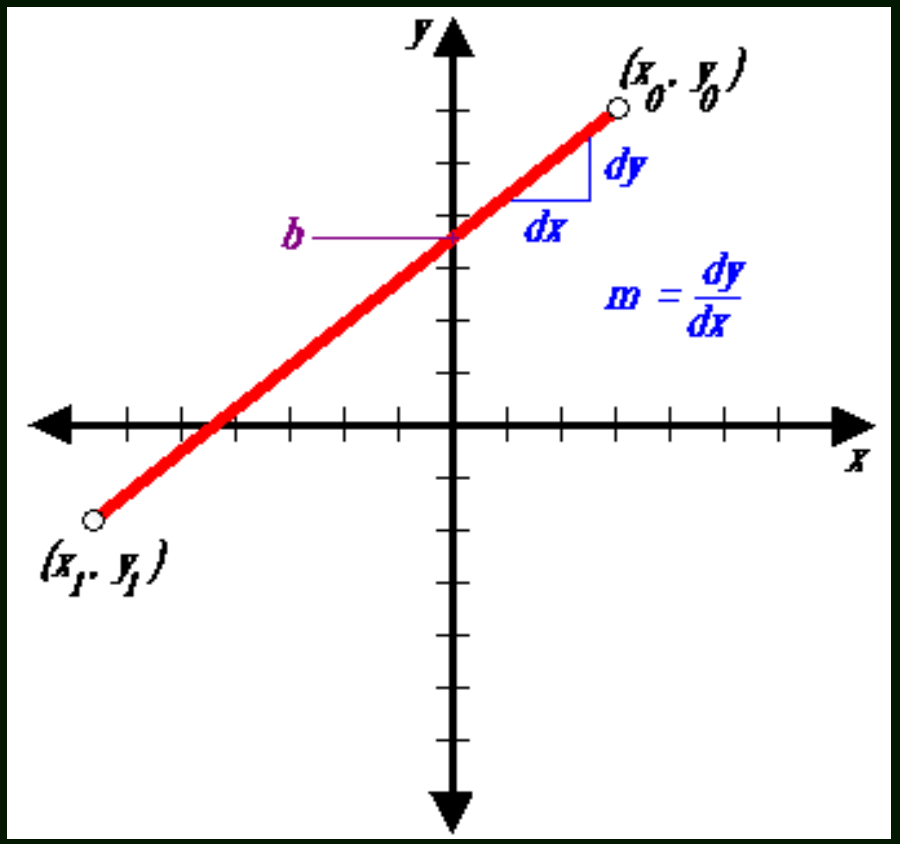
*The **Cartesian slope-intercept equation** for a straight line is :*

$$y = mx + b$$

with *m* as the slope of the line and ***b*** as the ***y*** intercept.

Simple approach:

- increment x, solve for y



Bresenham's line drawing algorithm.

DDA Algorithm

- DDA stands for digital differential analyzer.
- The differential equation for a line is: m = dy / dx

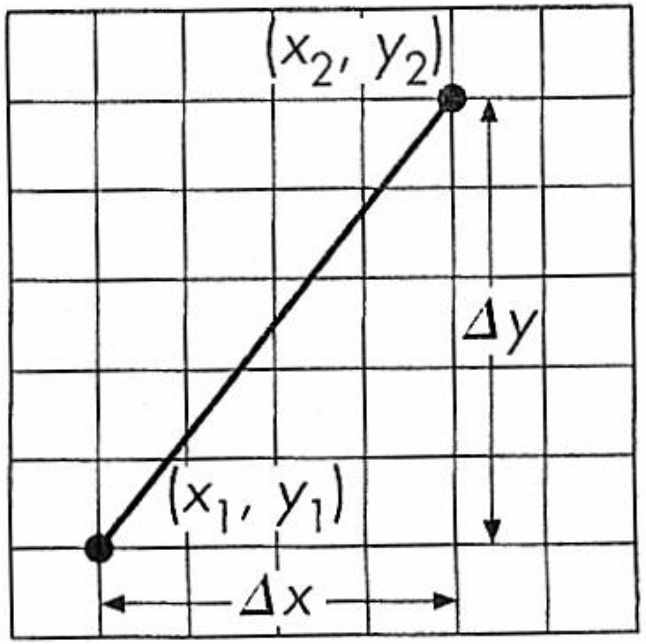


Figure 8.40 Line segment in window coordinates.

Stepping in X direction

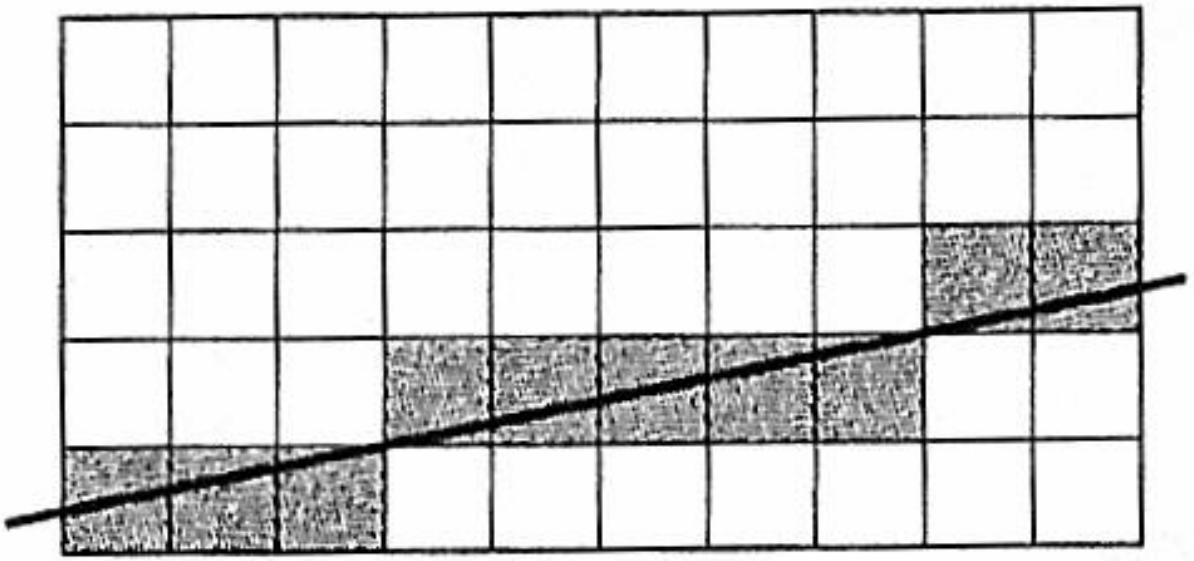


Figure 8.41 Pixels generated by DDA algorithm.

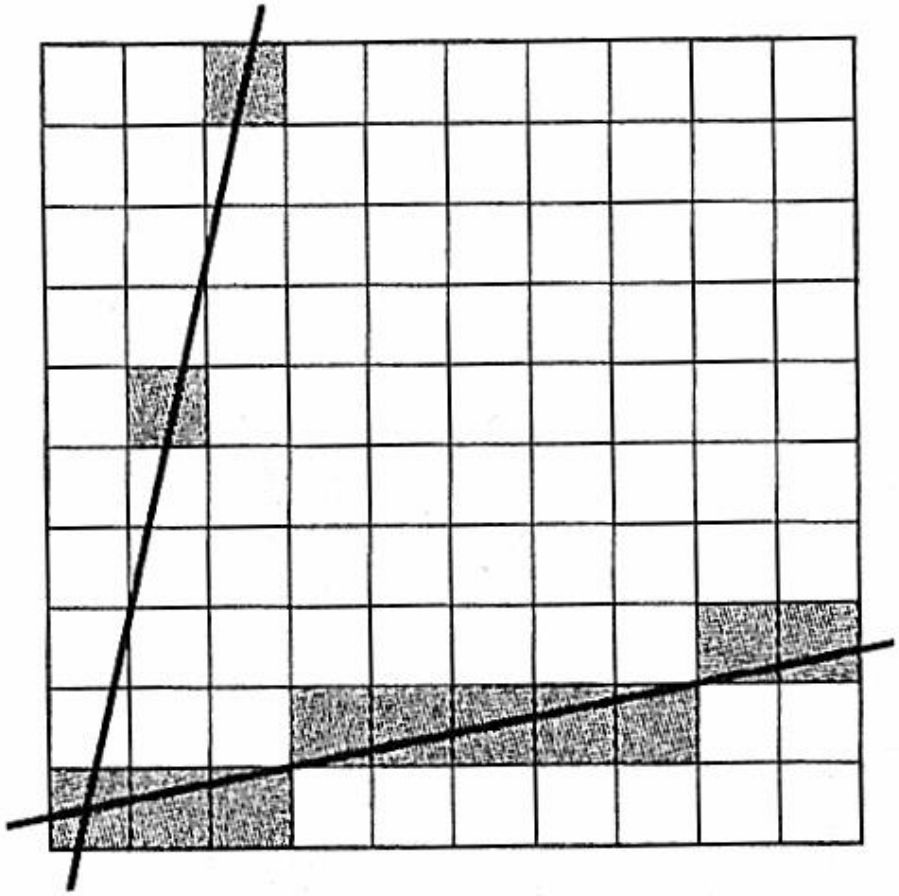


Figure 8.42 Pixels generated by high- and low-slope lines.

WHY Bresenham's Algorithm

- Improve upon DDA algorithm to use integer arithmetic only.
- Applicable to circle drawing too. We discuss only the line drawing here.

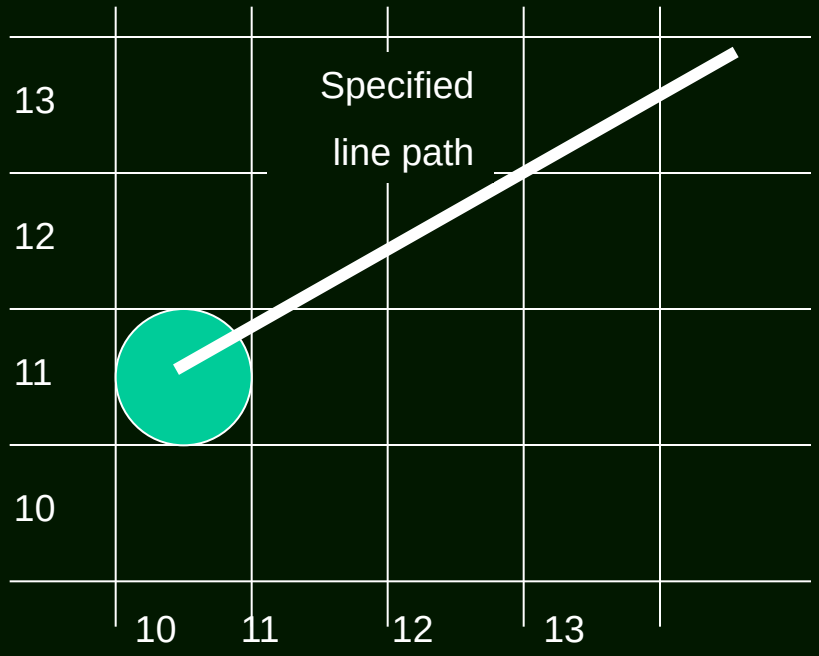
Bresenham's line algorithm

- Accurate and efficient
- Uses only incremental integer calculations

The method is described for a line segment with a positive slope less than one

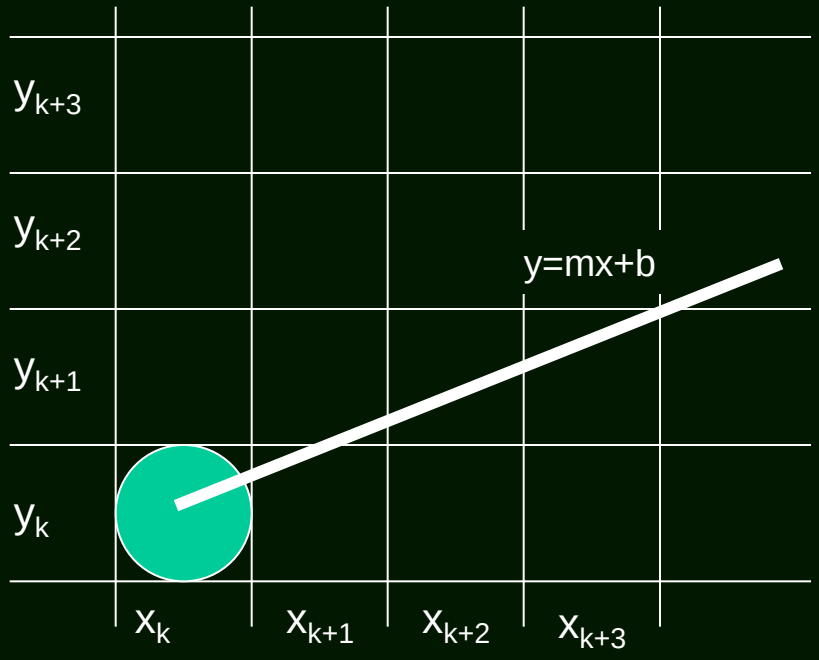
The method generalizes to line segments of other slopes by considering the symmetry between the various octants and quadrants of the xy plane

Bresenham's line algorithm



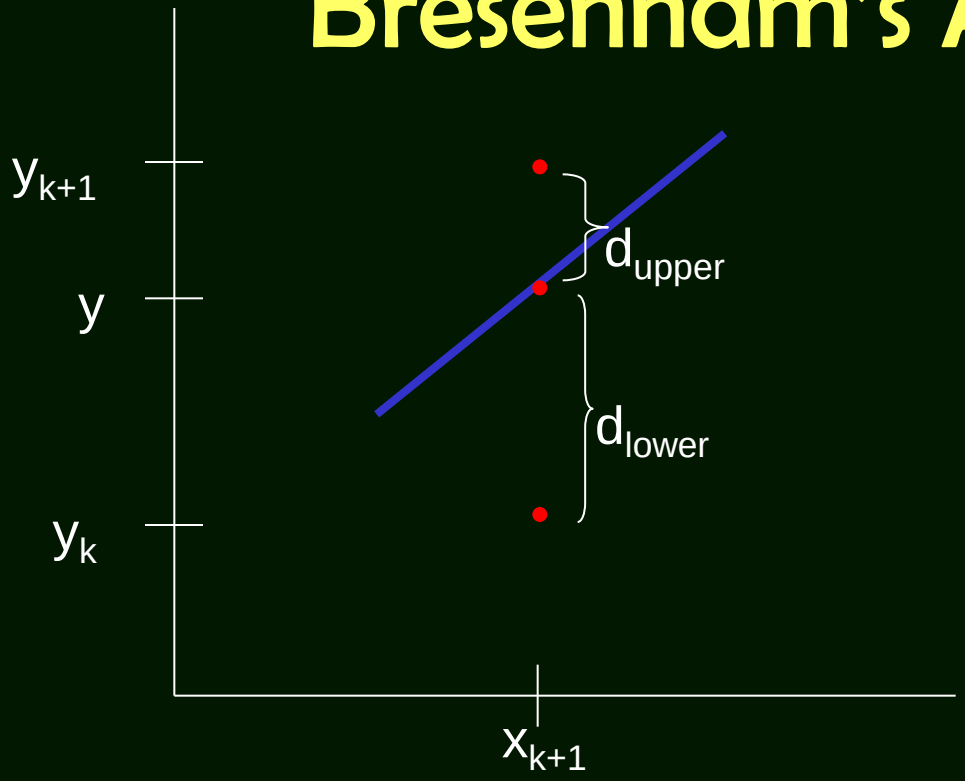
- Decide what is the next pixel position
  - (11,11) or (11,12)

Illustrating Bresenham's Approach



- For the pixel position $x_{k+1}=x_k+1$, which one we should choose:
- (x_{k+1},y_k) or (x_{k+1}, y_{k+1})

Bresenham's Approach



- $y=m(x_k + 1)+b$
- $d_{lower}=y-y_k$
  $=m(x_k + 1)+b-y_k$
- $d_{upper}=(y_k+1)-y$
  $= y_k+1 -m(x_k + 1)-b$
- $d_{lower}- d_{upper}= 2m(x_k + 1)+2y_k+2b-1$
- Rearrange it to have integer calculations:

  $m=\Delta y/\Delta x$

  Decision parameter: $p_k= \Delta x(d_{lower}- d_{upper})=2\Delta y.x_k - 2\Delta x. y_k + c$

The Decision Parameter

Decision parameter: $p_k = \Delta x(d_{lower} - d_{upper}) = 2\Delta y.x_k - 2\Delta x.\ y_k + c$

- p_k has the same sign with $d_{lower} - d_{upper}$ since $\Delta x > 0$.
- c is constant and has the value $c = 2\Delta y + \Delta x(2b-1)$
  - c is independent of the pixel positions and is eliminated from decision parameter p_k.

- If $d_{lower} < d_{upper}$ then p_k is negative.
  - Plot the lower pixel (East)
- Otherwise
  - Plot the upper pixel (North East)

Succesive decision parameter

- At step k+1

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

- Subtracting two subsequent decision parameters yields:

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

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

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

  - $y_{k+1}-y_k$ is either 0 or 1 depending on the sign of p_k
- First parameter p_0
  - $p_0=2\Delta y - \Delta x$

Bresenham's Line-Drawing Algorithm for | m | < 1

1. Input the twoline endpoints and store the left endpoint in (x_0, y_0).

2. Load (x_0, y_0) into the frame buffer; that is, plot the first point.

3. Calculate constants Δx, Δy, $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain the starting value for the decision parameter as

   $p_0 = 2\,\Delta y - \Delta x$

4. At each x_k along the line, starting at k = 0, perform the following test:

   If $p_k < 0$, the next point to plot is (x_{k+1}, y_k) and

   $p_{k+1} = p_k + 2\Delta y$

   Otherwise, the next point to plot is (x_{k+1}, y_{k+1}) and

   $p_{k+1} = p_k + 2\Delta y - 2\Delta x$

5. Repeat step 4 Δx -1 times.

Trivial Situations: Do not need Bresenham

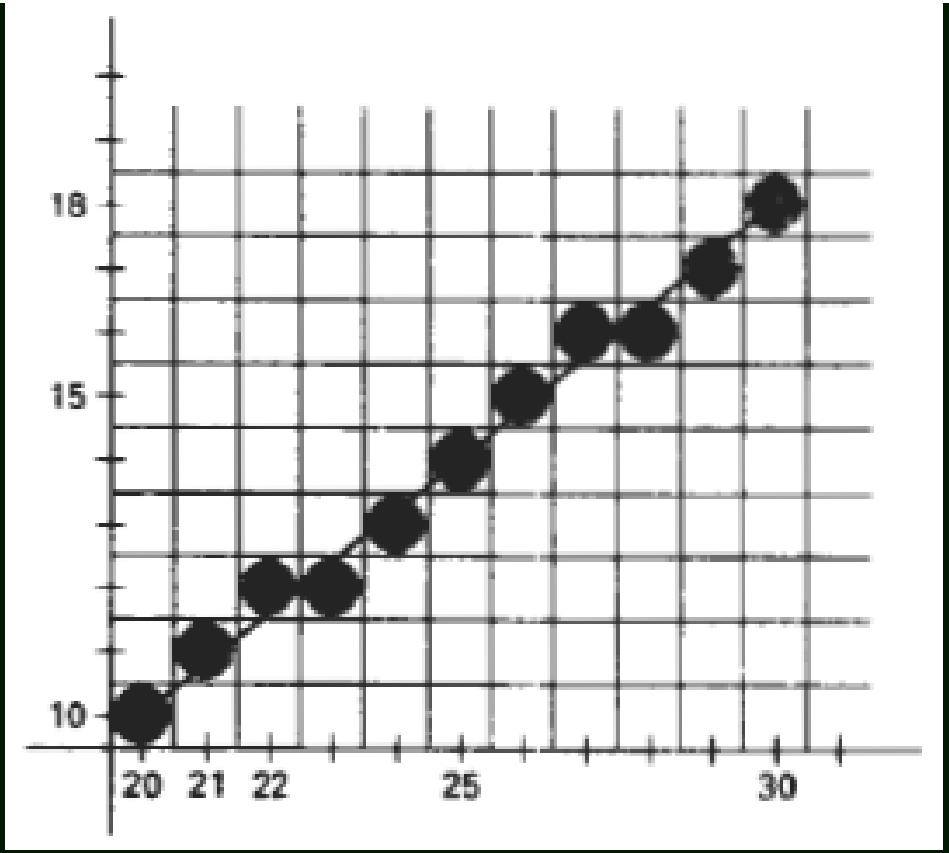
- $m = 0 \quad \Rightarrow \quad \text{horizontal line}$
- $m = \pm 1 \quad \Rightarrow \quad \text{line} \;\; y = \pm x$
- $m = \infty \quad \Rightarrow \quad \text{vertical line}$

Example

- Draw the line with endpoints (20,10) and (30, 18).
  - **$\Delta x=30-20=10$, $\Delta y=18-10=8$,**
  - **$p_0 = 2\Delta y - \Delta x=16-10=6$**
  - ***$2\Delta y=16$,*** and ***$2\Delta y$*** **$- 2\Delta x=-4$**
-

k	p_k	(x_{k+1}, y_{k+1})	k	p_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)	5	6	(26, 15)
1	2	(22, 12)	6	2	(27, 16)
2	−2	(23, 12)	7	−2	(28, 16)
3	14	(24, 13)	8	14	(29, 17)
4	10	(25, 14)	9	10	(30, 18)

k	p_k	(x_{k+1}, y_{k+1})	k	p_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)	5	6	(26, 15)
1	2	(22, 12)	6	2	(27, 16)
2	−2	(23, 12)	7	−2	(28, 16)
3	14	(24, 13)	8	14	(29, 17)
4	10	(25, 14)	9	10	(30, 18)



Example

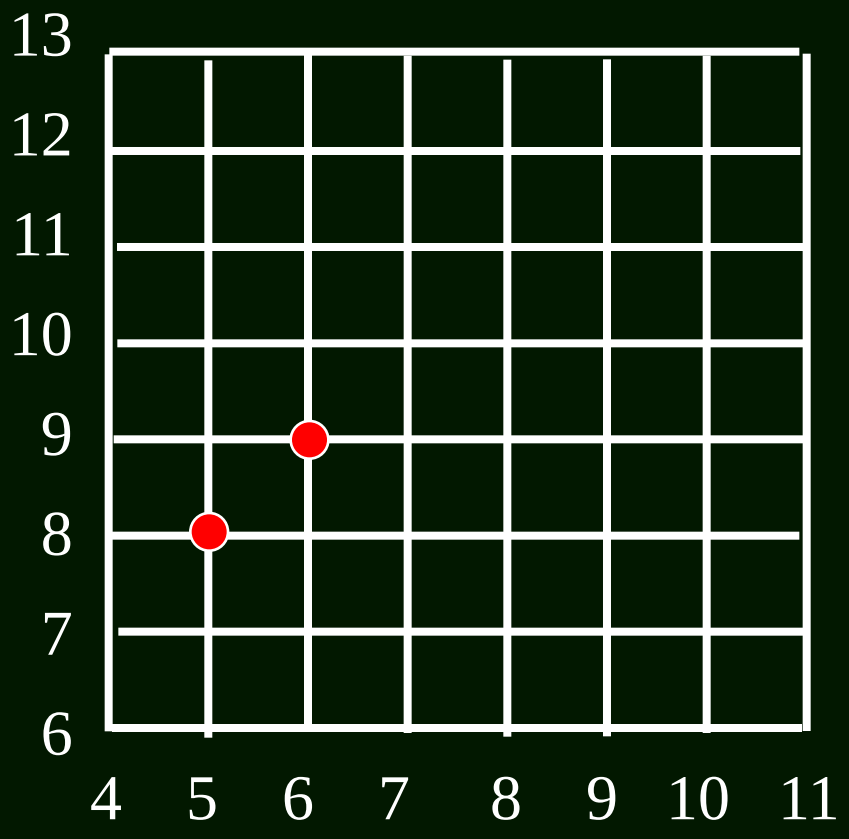
- Line end points: $(x_0, y_0) = (5,8); \quad (x_1, y_1) = (9,11)$
- Deltas: $dx = 4; dy = 3$

initially $p(5,8) = 2(dy)\text{-}(dx)$

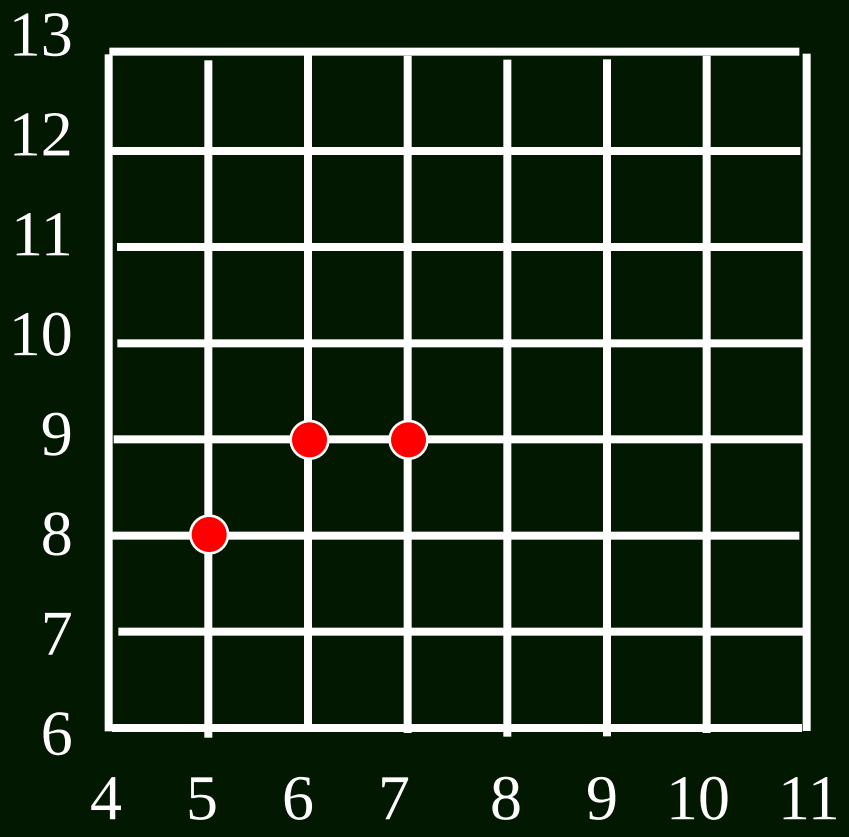
$$= 6 - 4 = 2 > 0$$

$$p = 2 \quad \Rightarrow \quad NE$$

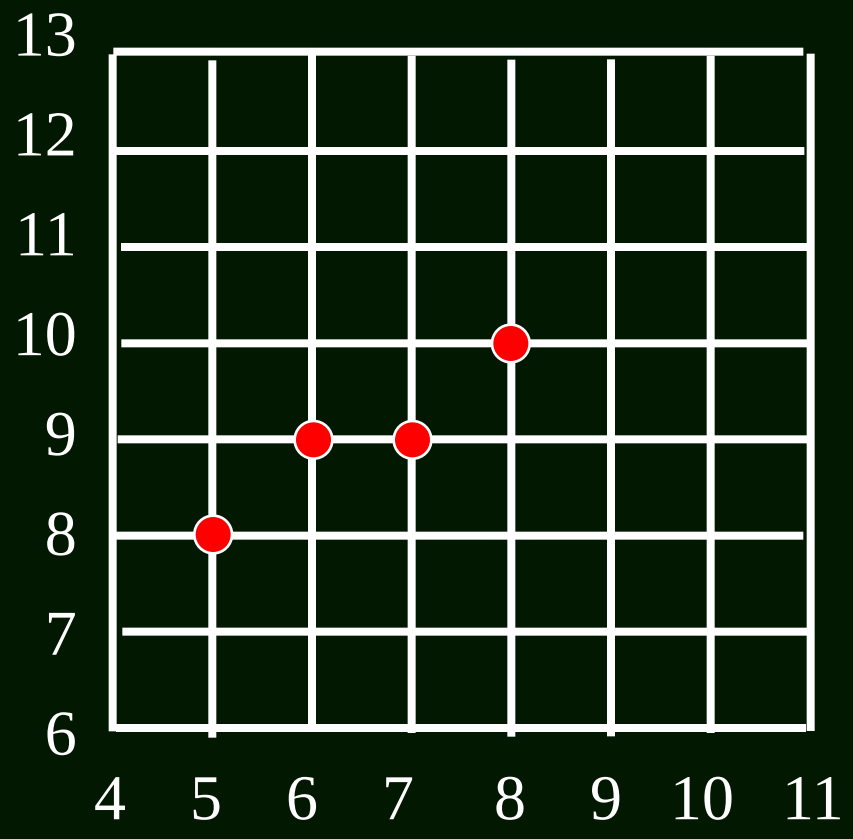
Graph



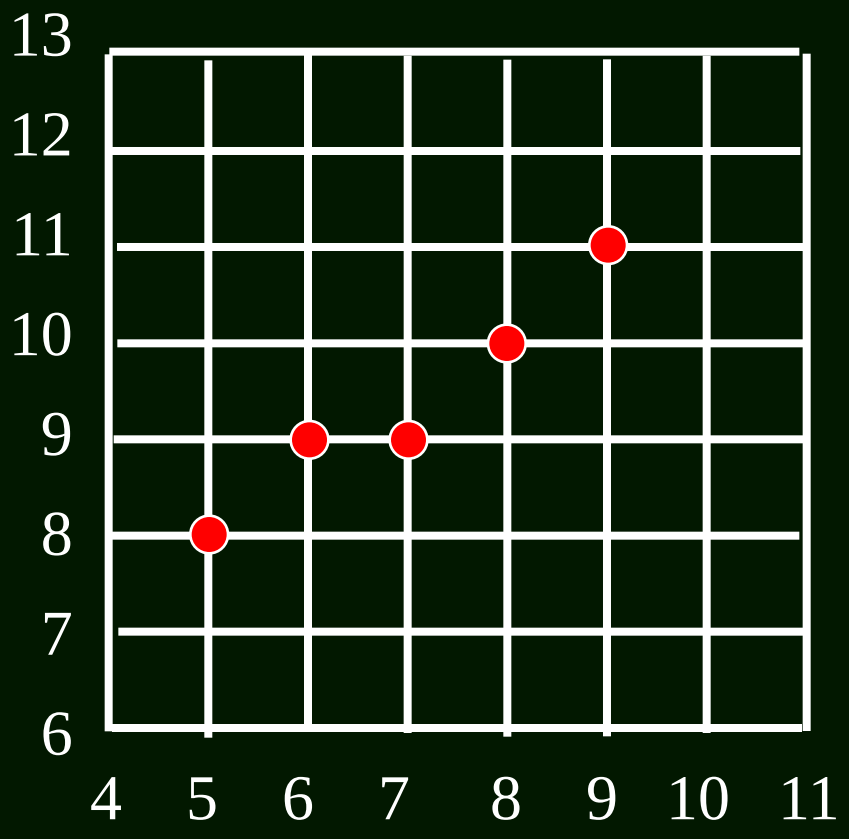
Continue the process...



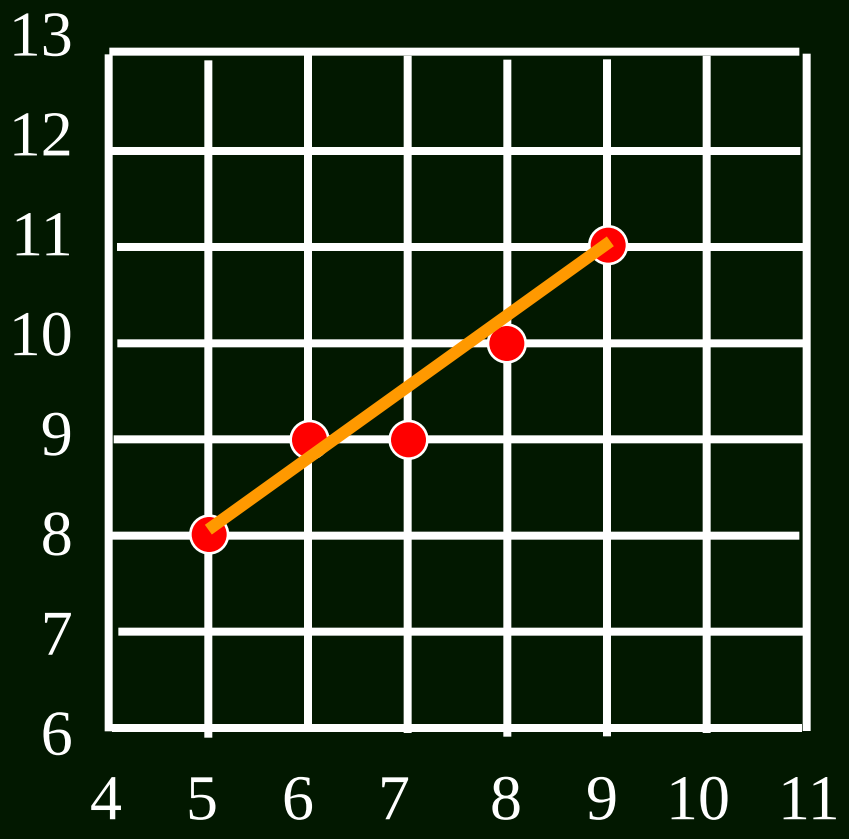
Graph



Graph



Graph

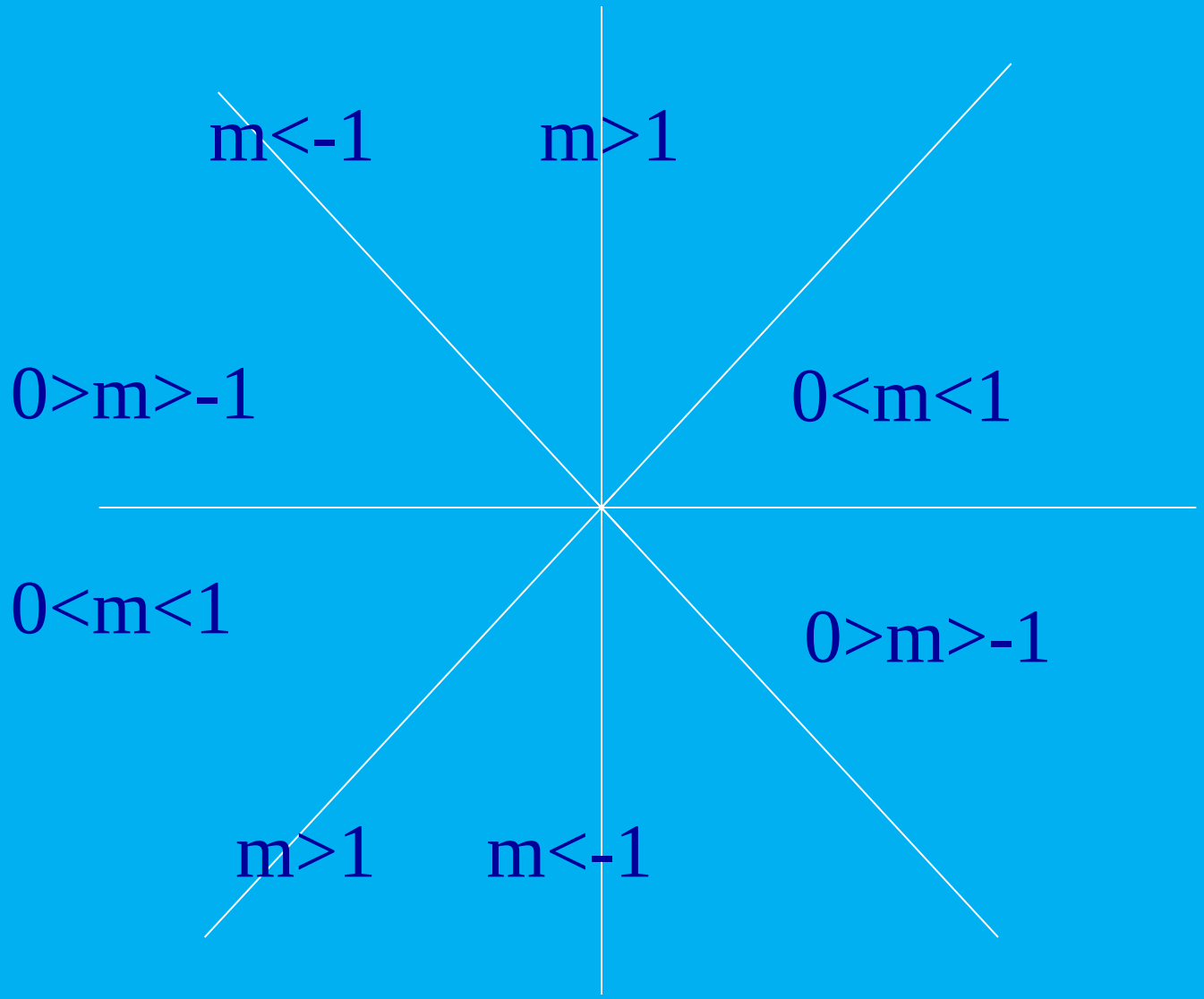


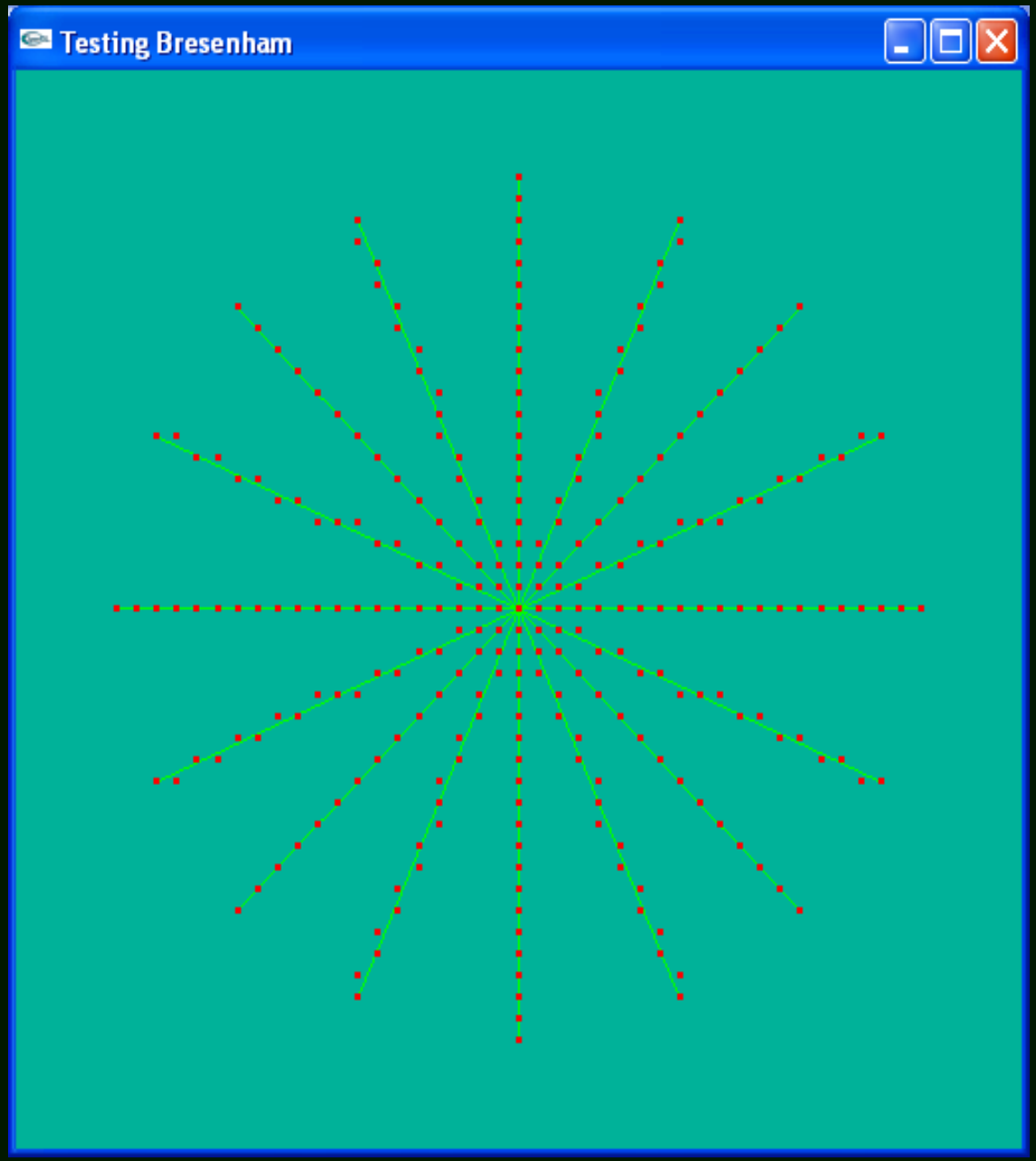
```
/*  Bresenham line-drawing procedure for |m| < 1.0.  */
   void lineBres (int x0, int y0, int xEnd, int yEnd)
   {
      int dx = fabs (xEnd - x0),  dy = fabs(yEnd - y0);
      int x, y, p = 2 * dy - dx;
      int twoDy = 2 * dy,  twoDyMinusDx = 2 * (dy - dx);

      /* Determine which endpoint to use as start position.  */
      if (x0 > xEnd) {
         x = xEnd;    y = yEnd;   xEnd = x0;
      }
      else {
         x = x0;    y = y0;
      }
      setPixel (x, y);

      while (x < xEnd) {
         x++;
         if (p < 0)
            p += twoDy;
         else {
            y++;
            p += twoDyMinusDx;
         }
         setPixel (x, y);
      }
   }
```

Line-drawing algorithm should work in every octant, and special cases





Simulating the Bresenham algorithm in drawing 8 radii on a circle of radius 20

Horizontal, vertical and ±45° radii handled as special cases