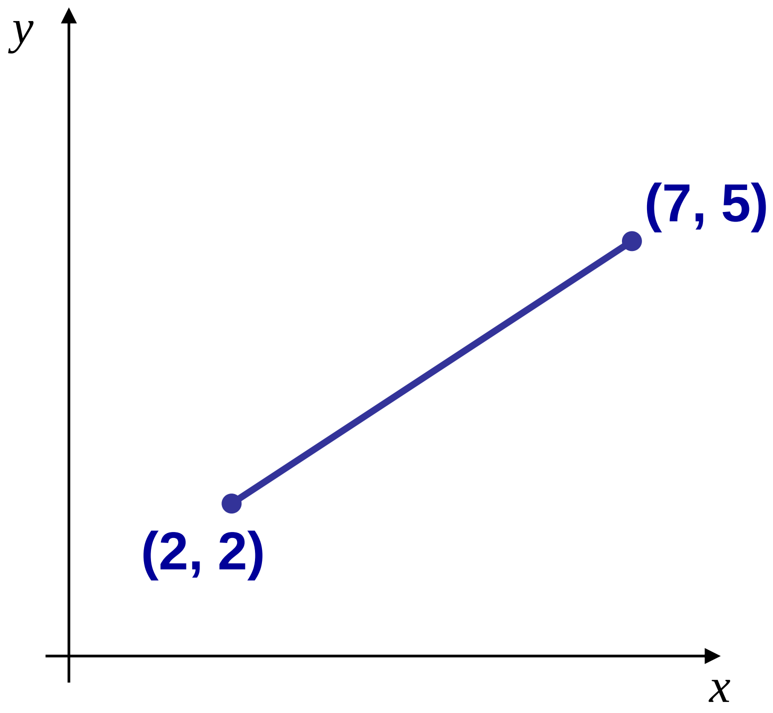
Computer Graphics:
Line Drawing Algorithms

Scan Conversion Algorithms

Contents

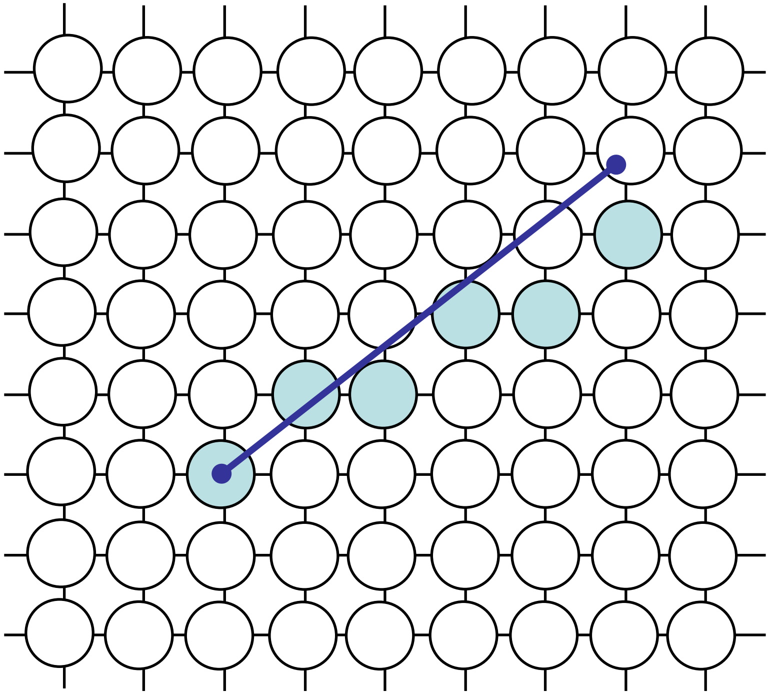
The Problem Of Scan Conversion

A line segment in a scene is defined by the coordinate positions of the line end-points



The Problem (cont…)

But what happens when we try to draw this on a pixel based display?



How do we choose which pixels to turn on?

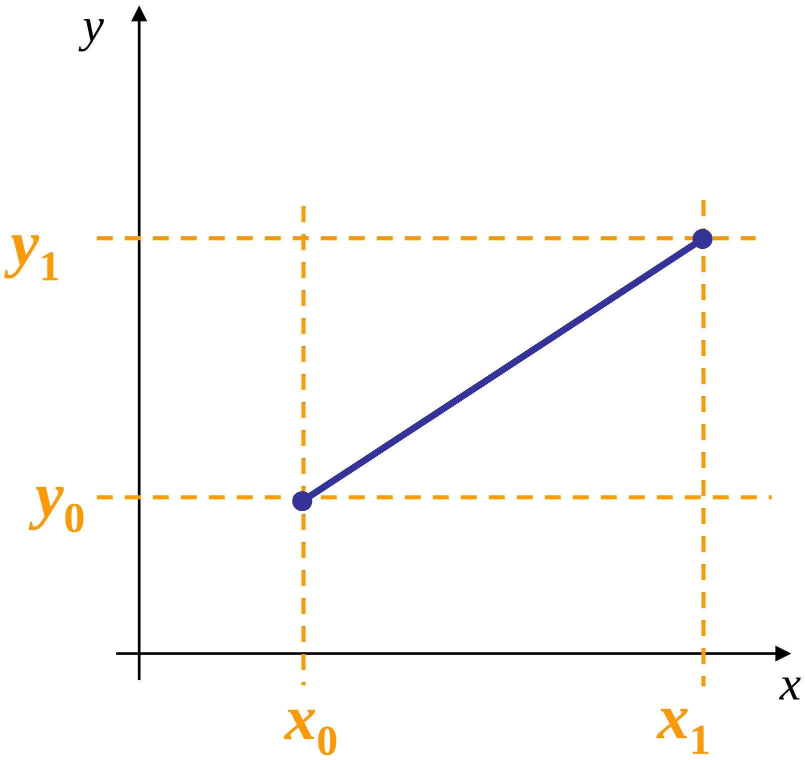
Considerations

Considerations to keep in mind:

- The line has to look good
  - Avoid *jaggies*
- It has to be lightening fast!
  - How many lines need to be drawn in a typical scene?
  - This is going to come back to bite us again and again

Line Equations

Let's quickly review the equations involved in drawing lines



Slope-intercept line equation:

$$y = m \cdot x + b$$

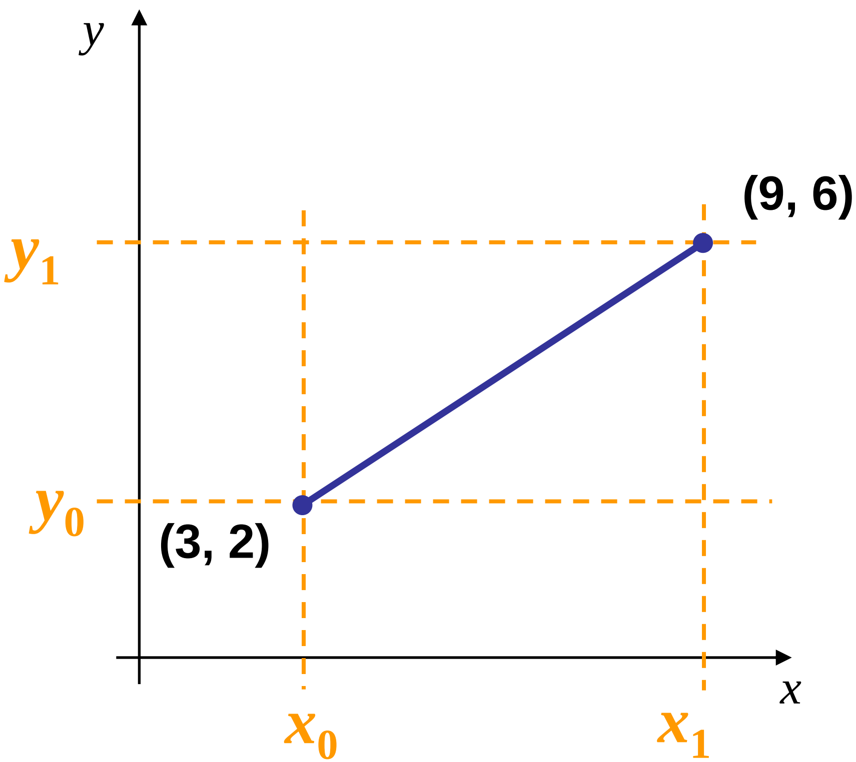
where:

$$m = \frac{y_1 - y_0}{x_1 - x_0}$$

$$b = y_0 - m \cdot x_0$$

Line Equations

Let's quickly review the equations involved in drawing lines



Slope-intercept line equation:

$$y = m \cdot x + b$$

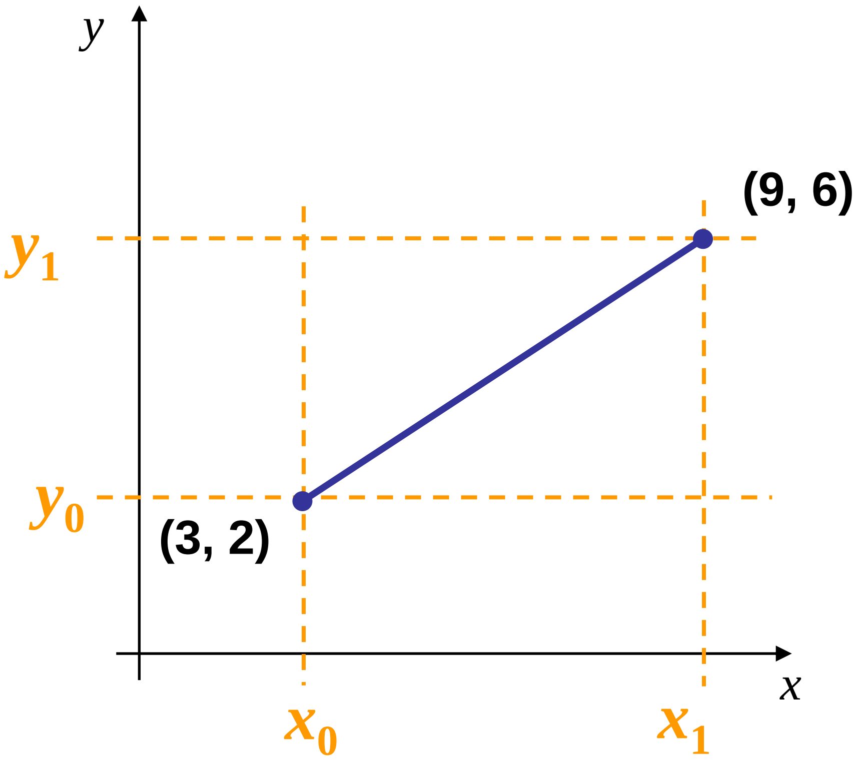
where:

$$m = \frac{6-2}{9-3} = \frac{4}{6} = \frac{2}{3}$$

$$b = 3 - \frac{2}{3} \cdot 3 = 3 - 2 = 1$$

Line Equations

Let's quickly review the equations involved in drawing lines



Slope-intercept line equation:

$$y = \frac{2}{3} \cdot x + 1$$

where:

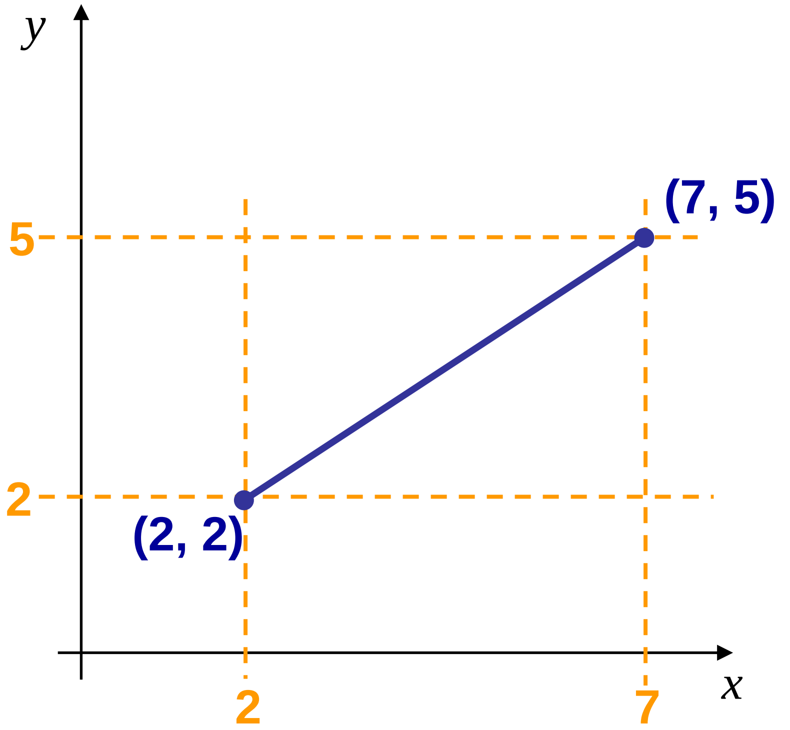
$$m = \frac{6-2}{9-3} = \frac{4}{6} = \frac{2}{3}$$

$$b = 3 - \frac{2}{3} \cdot 3 = 3 - 2 = 1$$

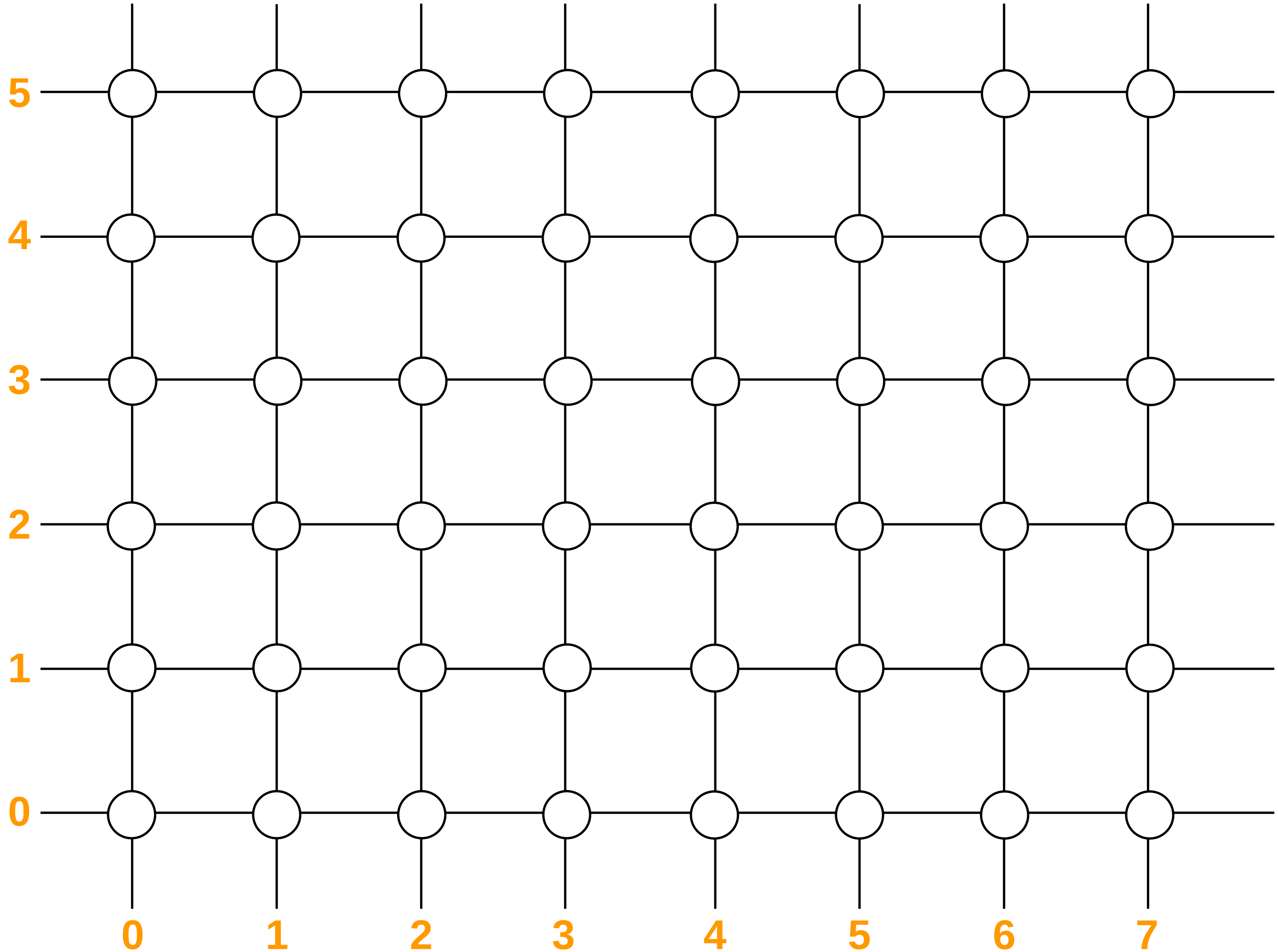
A Very Simple Solution

We could simply work out the corresponding y coordinate for each unit x coordinate

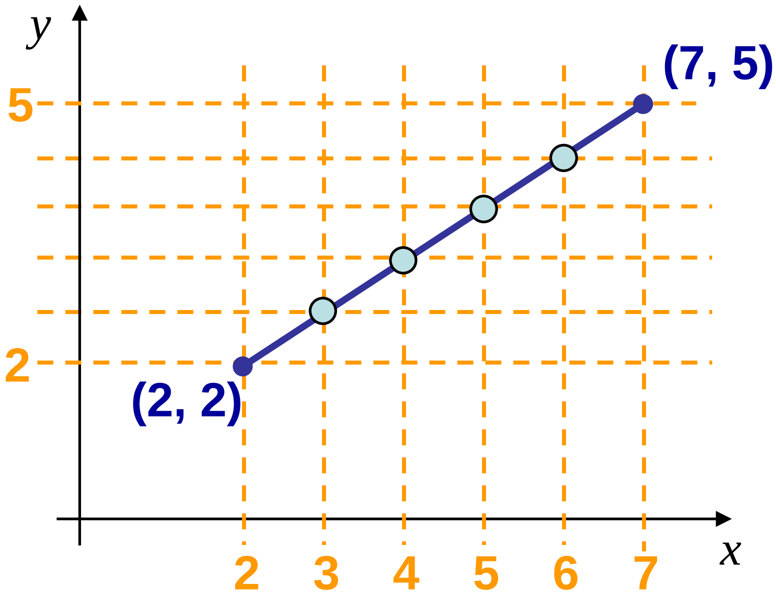
Let's consider the following example:



A Very Simple Solution (cont…)



A Very Simple Solution (cont…)



First work out m and b:

$$m = \frac{5-2}{7-2} = \frac{3}{5}$$

$$b = 2 - \frac{3}{5} * 2 = \frac{4}{5}$$

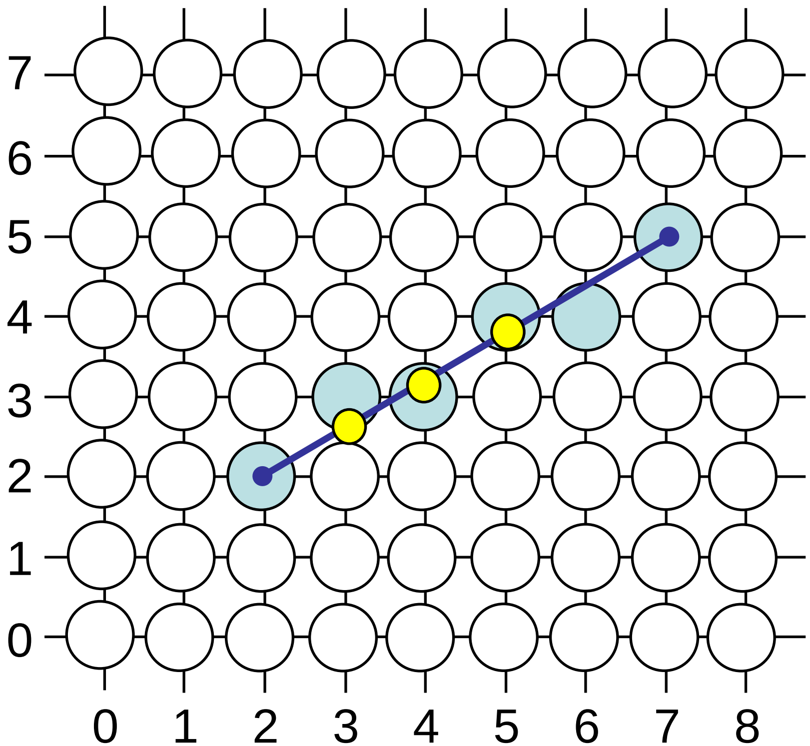
Now for each x value work out the y value:

$$y(3) = \frac{3}{5} \cdot 3 + \frac{4}{5} = 2\frac{3}{5} \qquad y(4) = \frac{3}{5} \cdot 4 + \frac{4}{5} = 3\frac{1}{5}$$

$$y(5) = \frac{3}{5} \cdot 5 + \frac{4}{5} = 3\frac{4}{5} \qquad y(6) = \frac{3}{5} \cdot 6 + \frac{4}{5} = 4\frac{2}{5}$$

A Very Simple Solution (cont…)

Now just round off the results and turn on these pixels to draw our line



$$y(3) = 2\frac{3}{5} = 2.6 \approx 3$$

$$y(4) = 3\frac{1}{5} = 3.2 \approx 3$$

$$y(5) = 3\frac{4}{5} = 3.8 \approx 4$$

$$y(6) = 4\frac{2}{5} = 4.4 \approx 4$$

Pixel
(3, 3)
(4,3)
(5,4)
(6,4)

A Very Simple Solution (cont…)

However, this approach is just way too slow

In particular look out for:

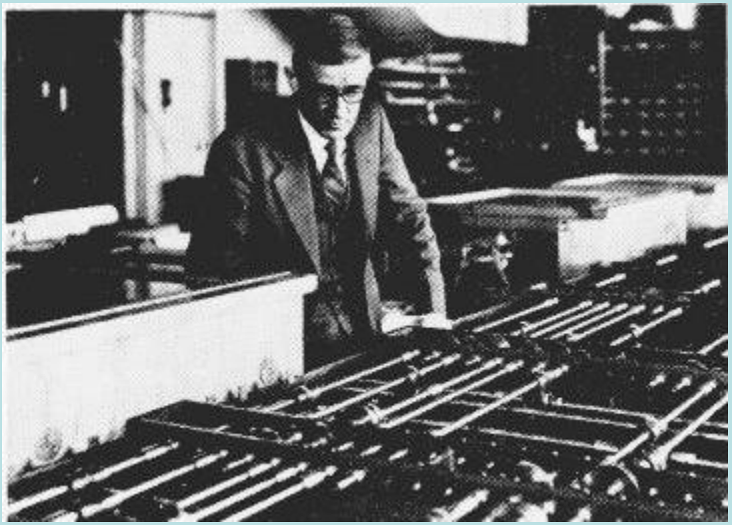
- The equation $y = mx + b$ requires the multiplication of m by x
- Rounding off the resulting y coordinates

We need a faster solution

The DDA Algorithm

The *digital differential analyzer* (DDA) algorithm takes an incremental approach in order to speed up scan conversion

Simply calculate y_{k+1} based on y_k



The original differential analyzer was a physical machine developed by Vannevar Bush at MIT in the 1930's in order to solve ordinary differential equations.

More information here.

The DDA Algorithm (cont…)

Consider the list of points that we determined for the line in our previous example:

$(2, 2), (3, 2^3/_5), (4, 3^1/_5), (5, 3^4/_5), (6, 4^2/_5), (7, 5)$

Notice that as the x coordinates go up by one, the y coordinates simply go up by the slope of the line

This is the key insight in the DDA algorithm

The DDA Algorithm (cont…)

When the slope of the line is between -1 and 1 begin at the first point in the line and, by incrementing the x coordinate by 1, calculate the corresponding y coordinates as follows:

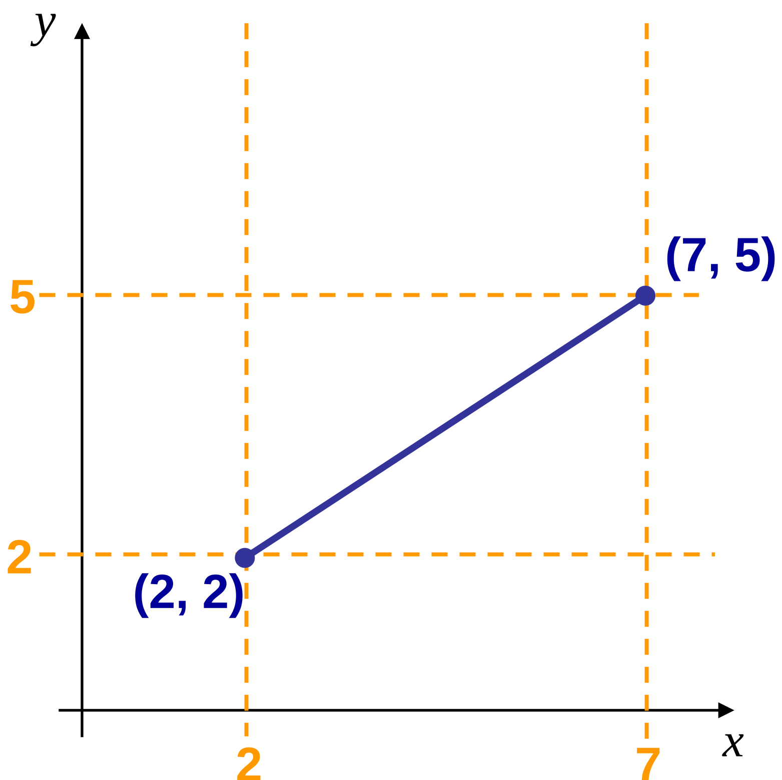
$$y_{k+1} = y_k + m$$

When the slope is outside these limits, increment the y coordinate by 1 and calculate the corresponding x coordinates as follows:

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

DDA Algorithm Example

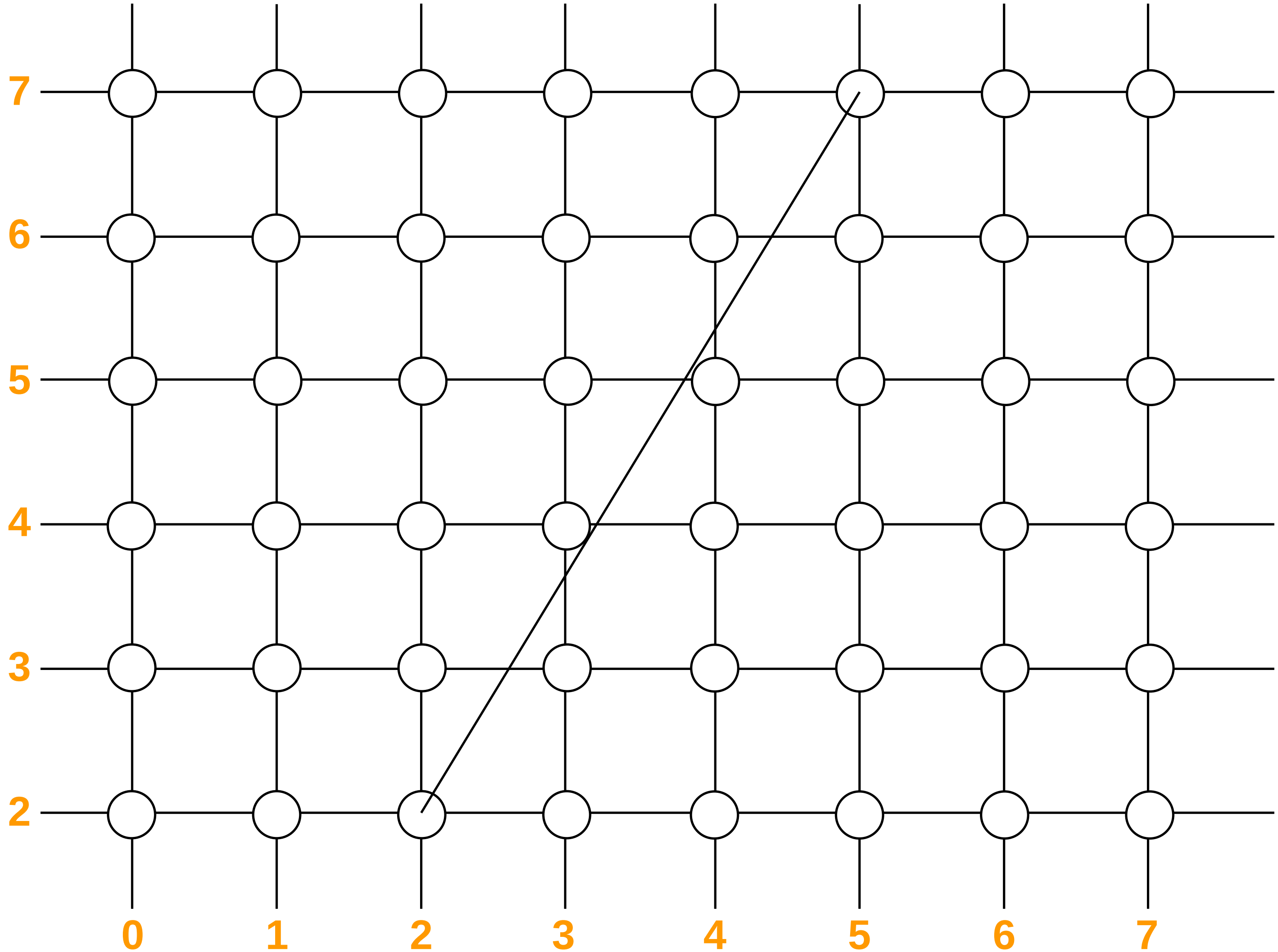
Let's try out the following examples:



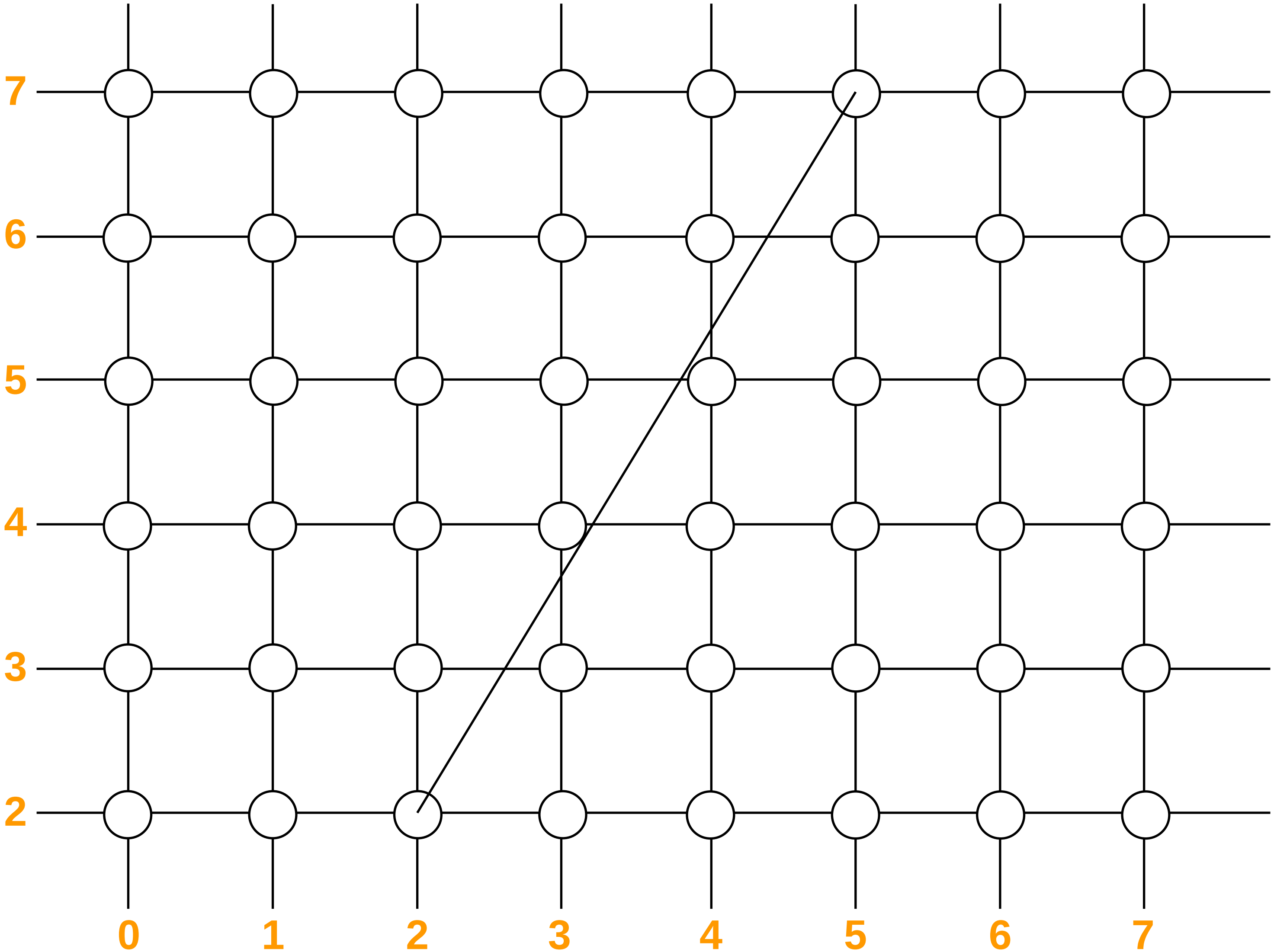
$$m = \frac{5-2}{7-2} = \frac{3}{5} = 0.6$$

x(+1)	y(+m)	y(round off)	pixel
2	2		
3	2.6	3	(3, 3)
4	3.2	3	(4, 3)
5	3.8	4	(5, 4)
6	4.4	4	(6, 4)

DDA Algorithm Example (cont…)

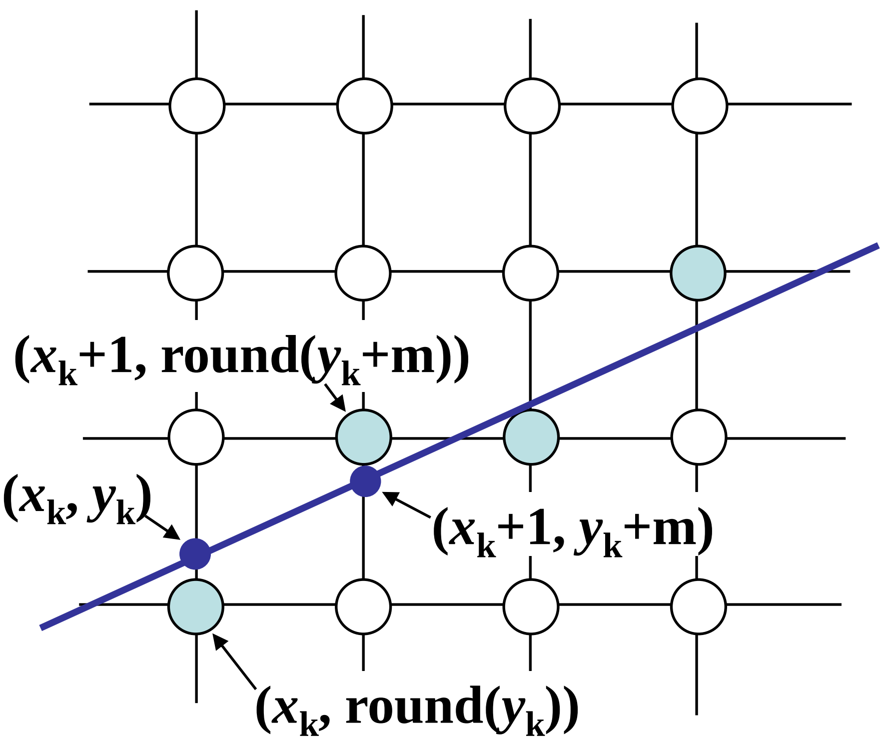


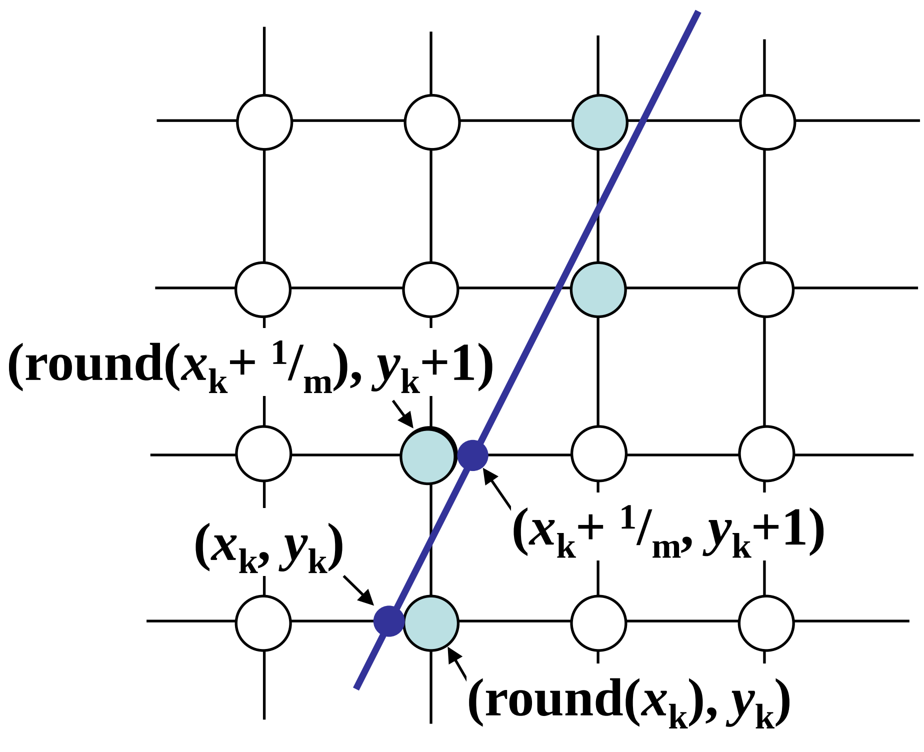
DDA Algorithm Example (cont…)



The DDA Algorithm (cont…)

Again the values calculated by the equations used by the DDA algorithm must be rounded to match pixel values





The DDA Algorithm (cont…)

If -1<m<1 then

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

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

Then roundoff y.

Otherwise,

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

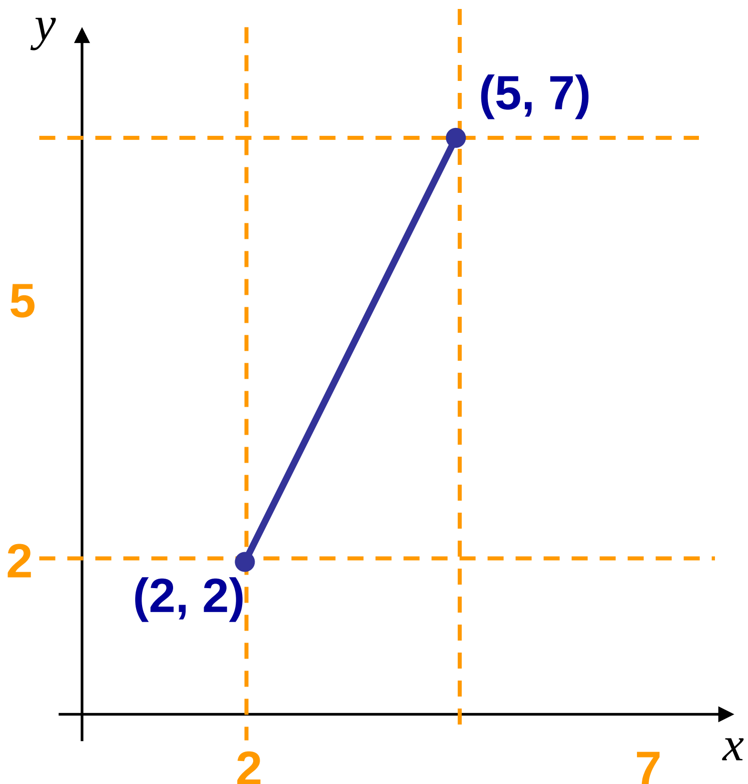
$$x_{k+1} = x_k + \frac{1}{m}$$

Then roundoff x.

DDA Algorithm Example

Let's try out the following examples:

$$\frac{1}{m} = \frac{3}{5} = 0.6$$

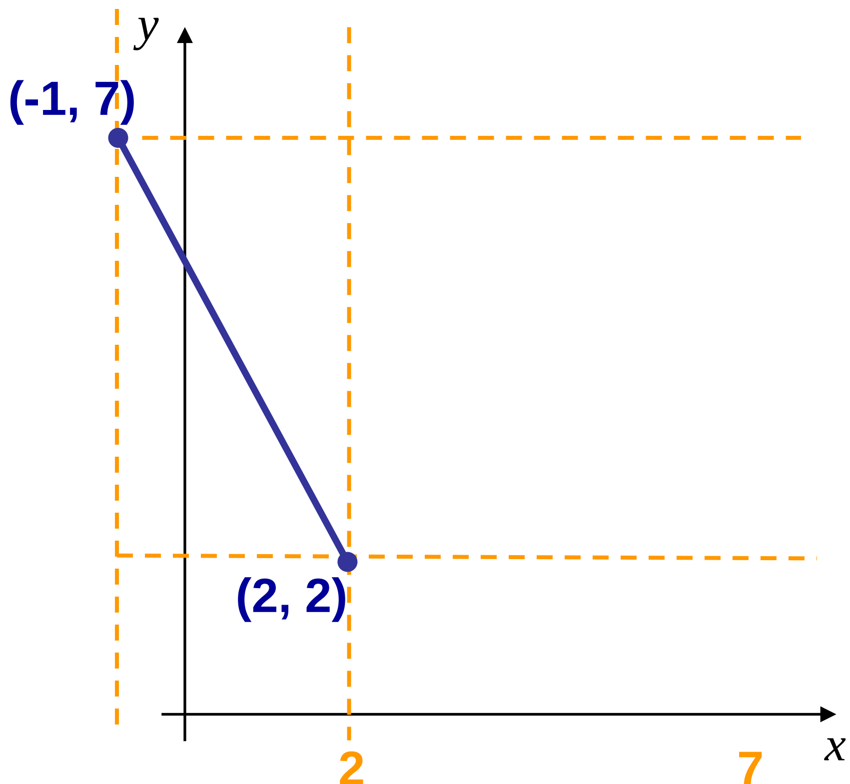


$$m = \frac{7-2}{5-2} = \frac{5}{3}$$

y(+1)	x(+1/m)	x(round off)	pixel
2	2		
3	2.6	3	(3, 3)
4	3.2	3	(3, 4)
5	3.8	4	(4, 5)
6	4.4	4	(4, 6)

DDA Algorithm Example

Let's try out the following examples: $\frac{1}{m} = -\frac{3}{5} = -0.6$



$$m = \frac{7-2}{-1-2} = -\frac{5}{3}$$

y(+1)	x(+1/m)	x(round off)	pixel
2	2		
3	1.4	1	(1, 3)
4	0.8	1	(1, 4)
5	0.2	0	(0, 5)
6	-0.4	0	(0, 6)

The DDA Algorithm Summary

The DDA algorithm is much faster than our previous attempt

- In particular, there are no longer any multiplications involved

However, there are still two big issues:

- Accumulation of round-off errors can make the pixelated line drift away from what was intended
- The rounding operations and floating point arithmetic involved are time consuming

The DDA Algorithm (cont…)

If -1<m<1 then

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

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

Then roundoff y.

Otherwise,

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

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

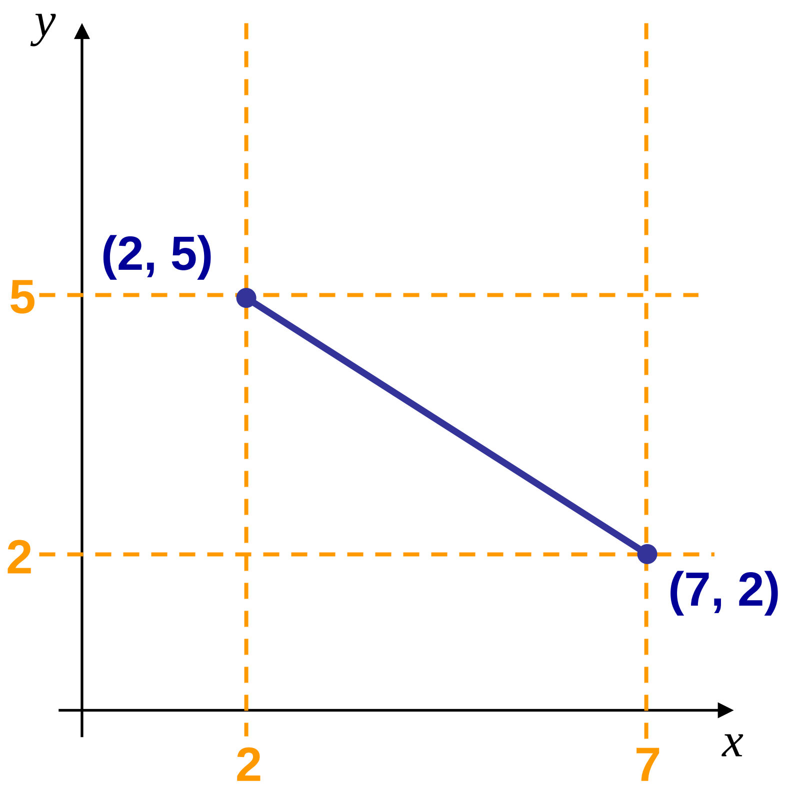
Then roundoff x.

Let's look at an example with $x_1 < x_0$.

Suppose we were given two endpoints (7, 2) and (2, 5)

$$m = \frac{5-2}{2-7} = -0.6$$

DDA Algorithm Example



$$m = \frac{5-2}{2-7} = -0.6$$

x(+1)	y(+m)	y(round off)	pixel
7	2		
8	1.4	1	(8, 1)
9	0.8	1	(9, 1)

(8, 1), (9, 1) these points are outside the line segment, so unacceptable.

So, we must decrement here. So we need to update our DDA algorithm for these cases.

The DDA Algorithm revised

If -1<m<1 then

If $x_1 > x_0$,

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

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

Else,

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

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

Then roundoff y.

Otherwise, if -∞<m<-1 or 1<m< ∞ then,

If $y_1 > y_0$,

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

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

Else,

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

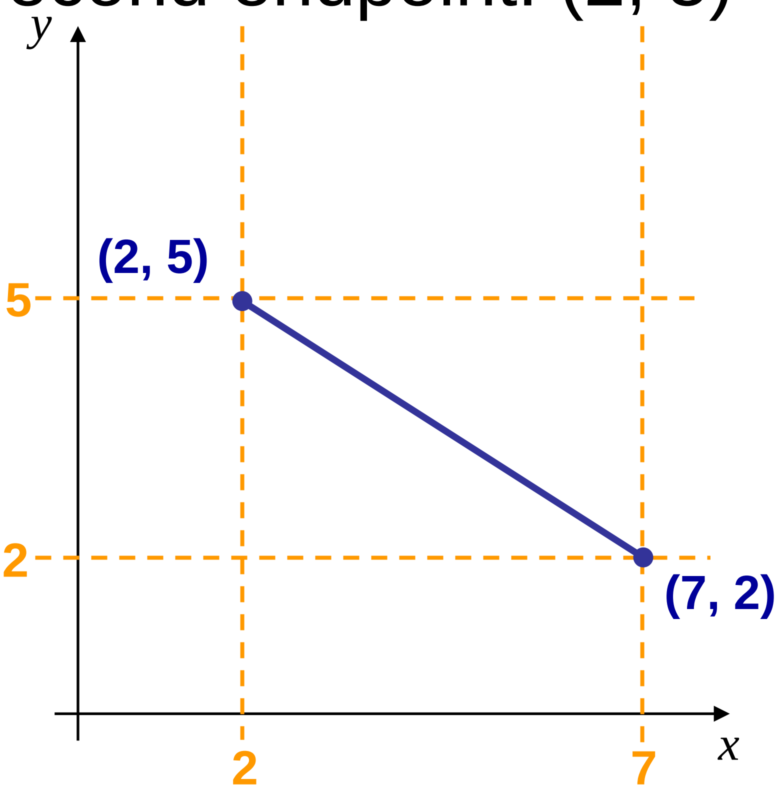
$$x_{k+1} = x_k - \frac{1}{m}$$

Then roundoff x.

DDA Algorithm Example

First endpoint: (7, 2)

Second endpoint: (2, 5)



$$m = \frac{5-2}{2-7} = -0.6$$

$x_1 = 2 < x_0 = 7$

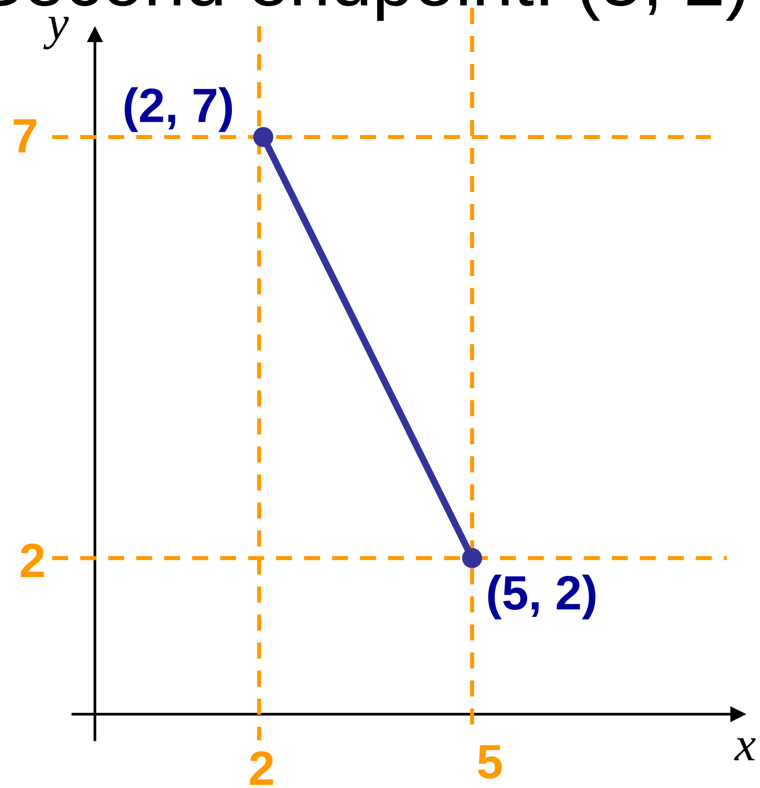
So we decrement

x(-1)	y(-m)	y(round off)	pixel
7	2		
6	2.6	3	(6, 3)
5	3.2	3	(5, 3)
4	3.8	4	(4, 4)
3	4.4	4	(3, 4)

DDA Algorithm Example

First endpoint: (2, 7)

Second endpoint: (5, 2)



$$m = \frac{2-7}{5-2} = -1.6667$$

$y_1 = 2 < y_0 = 7$
So we decrement

$$\frac{1}{m} = -0.6$$

y(-1)	x(-1/m)	x(round off)	pixel
7	2		
6	2.6	3	(3, 6)
5	3.2	3	(3, 5)
4	3.8	4	(4, 4)
3	4.4	4	(4, 3)

Order of 1st and 2nd endpoints make a difference in which case of DDA will be applied. Output will still remain same.

Exercise

1. Find out m and b and the equation of the following lines whose endpoints are given:

(a) (20, 35), (9, 50)

(b) (-5, 50), (-5,0)

(c) (-10, 10), (48, 24)

2. Using DDA algorithm find out the first 5 pixels of the lines whose endpoints are given: (Use the revised DDA if not mentioned)

(a) (20, 5), (19, 50)

(b) (-5, 50), (-15,0)

(c) (-10, -10), (48, 24)