DDA (Digital Differential Analyzer)

DDA (Digital Differential Analyzer) is a graphics algorithm used to draw lines on a digital display device like a computer monitor. The algorithm is based on linear interpolation between two endpoints of a line segment to determine the positions of pixels that lie along the line.
The derivation of the DDA algorithm is as follows:

Suppose we have two endpoints of a line segment, P1(x1, y1) and P2(x2, y2). We want to draw a line between these two points on a digital display screen. To do this, we need to calculate the position of each pixel that lies on the line segment.
The slope of the line segment can be calculated as:

$m = (y2 - y1) / (x2 - x1)$

If the slope is less than 1, we can start drawing the line from left to right by incrementing the x coordinate by one pixel at a time and calculating the corresponding y coordinate using the slope:

$y = y1 + m(x - x1)$

where x is the current x coordinate of the pixel being drawn.
If the slope is greater than or equal to 1, we can start drawing the line from bottom to top by incrementing the y coordinate by one pixel at a time and calculating the corresponding x coordinate using the inverse slope:

$x = x1 + (y - y1) / m$

where y is the current y coordinate of the pixel being drawn.
We can repeat these calculations for each pixel along the line segment until we reach the second endpoint P2. This process is known as digital differencing because we are incrementing either the x or y coordinate by a fixed amount (1 pixel) and calculating the difference in the other coordinate based on the slope of the line.

Bresenham's Line Drawing Algorithm

Bresenham's line drawing algorithm is an efficient algorithm for drawing lines on a digital display device. It uses only integer arithmetic, making it faster and more efficient than the DDA algorithm. To derive the algorithm, we start with the slope-intercept form of a line equation:
$y = mx + b$,
where m is the slope of the line, b is the y-intercept, and x and y are the coordinates of a point on the line.
Let's assume that we want to draw a line from point (x0, y0) to point (x1, y1) on a digital display device where x0 < x1. We can rearrange the slope-intercept form of the equation to get:
$y - y0 = m(x - x0)$
We can then multiply both sides by a constant scaling factor k to obtain integer values for the coordinates:
$ky - ky0 = kmx - kmx0$
By rearranging the equation, we get:
$ky = kmx - kmx0 + ky0$
This equation tells us that given an x-coordinate, we can calculate the corresponding y-coordinate by evaluating the right-hand side of the equation, which involves only integer arithmetic.
Now we need to determine how to choose the next pixel to draw the line. We know that the x-coordinate increments by 1 for each pixel along the line, but the y-coordinate changes by a non-integer amount. We can use the difference between the actual y-coordinate and the nearest integer as a measure of the error in our calculation.
We start by initializing the error term e to zero. Then for each pixel along the line, we compute the value of the right-hand side of the equation and compare it to the actual y-coordinate. If the value is greater than the actual y-coordinate, we round down to the nearest integer and increment the error term by the difference between the rounded-down value and the actual value. Otherwise, we round up to the nearest integer and increment the error term by the difference between the rounded-up value and the actual value.
To summarize, the Bresenham's line drawing algorithm can be derived as follows:

1. Initialize x and y to the starting point (x0, y0).
2. Calculate the slope of the line m = (y1 - y0) / (x1 - x0).
3. Calculate the constant scaling factor k = max(|y1 - y0|, |x1 - x0|).
4. Initialize the error term e to zero.
5. For each pixel along the line, compute the value of the right-hand side of the equation ky = kmx - kmx0 + ky0 and round it to the nearest integer.
6. Increment x by 1 and update the error term e by subtracting k times the fractional part of the right-hand side of the equation.
7. If e is greater than or equal to k, increment y by 1 and subtract k from e.
8. Repeat steps 5-7 until the endpoint (x1, y1) is reached.