Lab Title: Implementation of Bresenham Line Drawing Algorithm Using Python

Objective

The objective of this lab experiment is to understand and implement the **Bresenham Line Drawing Algorithm** using Python. This algorithm is used in computer graphics to draw a straight line between two points efficiently using only integer calculations.

Theory

The **Bresenham Line Drawing Algorithm** is an efficient algorithm used for drawing straight lines on a raster display. Unlike the DDA algorithm, Bresenham's algorithm uses only integer arithmetic, making it faster and more suitable for real-time graphics applications.

The algorithm works by determining which pixel is closest to the theoretical line at each step. It uses a **decision parameter (p)** to decide whether to increment the y-coordinate while moving along the x-axis.

This implementation assumes:

- The slope of the line is between 0 and 1
- The line is drawn from left to right (x1 < x2)

Mathematical Background

Given two endpoints:

- (x_1, y_1)
- (x_2, y_2)

We calculate:

- $dx = x_2 - x_1$
- $dy = y_2 - y_1$

Initial decision parameter:

- $p_0 = 2dy - dx$

Decision rule:

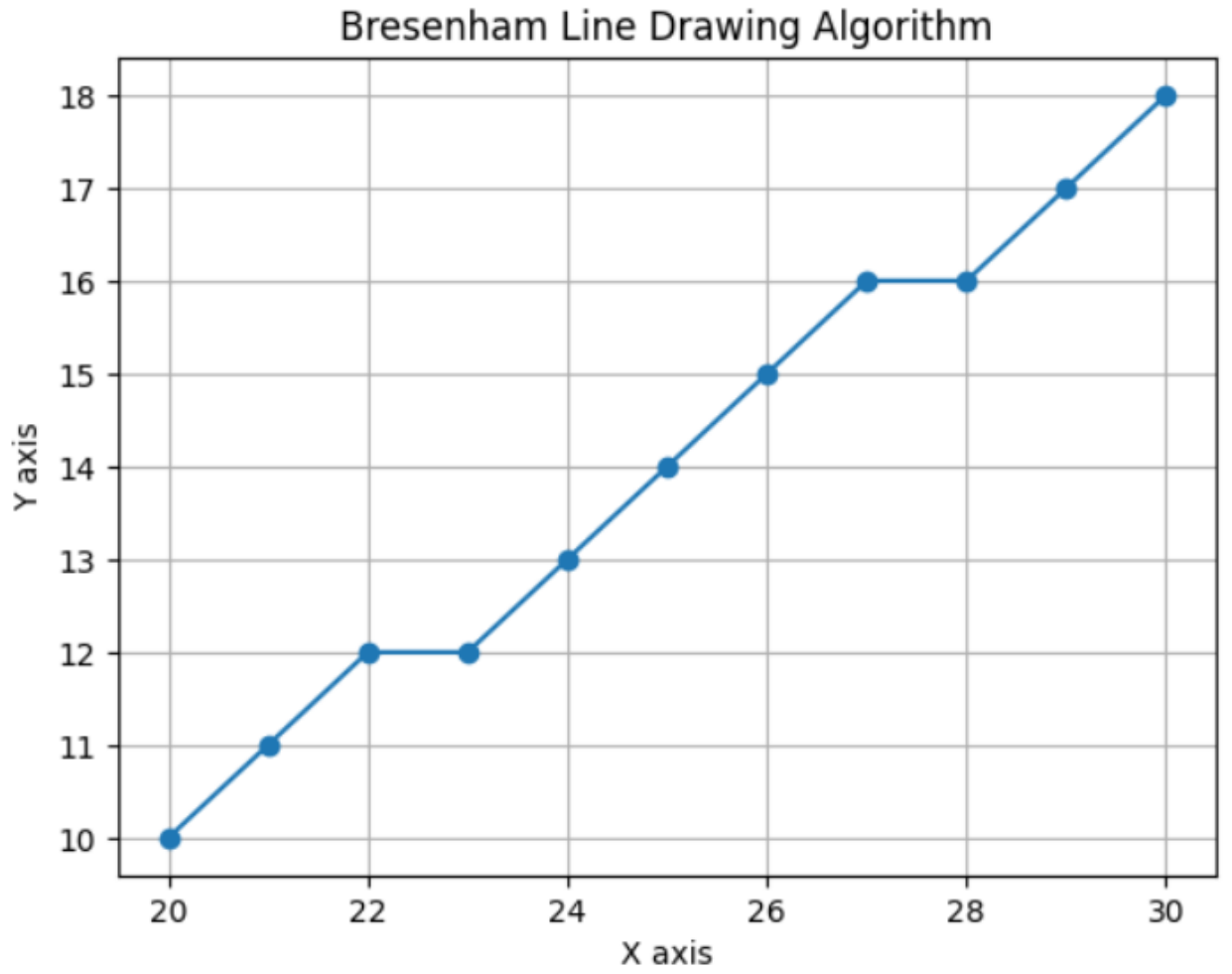
- If $p \geq 0 \rightarrow$ increment y and update $p = p + 2dy - 2dx$
- If $p < 0 \rightarrow$ y remains same and update $p = p + 2dy$

Source Code

```
import matplotlib.pyplot as plt

x1 = int(input("Enter x1: "))
y1 = int(input("Enter y1: "))
x2 = int(input("Enter x2: "))
y2 = int(input("Enter y2: "))

dx = x2 - x1
dy = y2 - y1

x = x1
y = y1

X = []
Y = []

p = 2*dy - dx

while x <= x2:
X.append(x)
Y.append(y)

if p >= 0:
y = y + 1
p = p + 2*dy - 2*dx
else:
p = p + 2*dy

x = x + 1
plt.plot(X, Y, marker='o')
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("Bresenham Line Drawing Algorithm")
plt.grid(True)
plt.show()
```

Output

The program successfully draws a straight line between the given coordinates using discrete pixels. The plotted graph visually represents how the Bresenham algorithm selects the nearest pixel positions.



Result

The Bresenham Line Drawing Algorithm was successfully implemented using Python. The output graph confirms that the algorithm accurately draws a straight line between two given points using integer arithmetic.

Advantages

1. Uses only integer arithmetic, which makes it faster than floating-point based algorithms.
2. Very efficient and suitable for real-time graphics systems.
3. Produces accurate and smooth straight lines on raster displays.
4. Requires less computation and memory.
5. Widely used in low-level graphics hardware implementations.

Disadvantages

1. Basic implementation works only for lines with slope between 0 and 1.
2. Additional logic is required to handle negative slopes and steep lines.
3. Less flexible compared to floating-point based algorithms.
4. Implementation becomes complex when extended for all octants.
5. Not suitable for drawing curves directly.

Discussion

This algorithm is more efficient than floating-point based methods like DDA because it avoids costly floating-point operations. However, this implementation works correctly only for lines with slope between 0 and 1. Additional logic is required to support other slopes and directions.

Conclusion

The experiment helped in understanding the working principle of the Bresenham Line Drawing Algorithm. It demonstrated how decision parameters can be used to efficiently draw lines in raster graphics systems.