**LAB-2**

**Bresenham's line drawing algorithm**

**Objective: -**

* To determine the points to select in a raster to form a close approximation of a straight line between two points
* Minimizes the error in pixel approximation to produce visually appealing lines.

**Theory: -**

Bresenham's line algorithm efficiently draws lines on a raster display using only integer addition, subtraction, and bit shifting. It determines the next pixel to plot by comparing a decision parameter to zero, ensuring the closest pixel to the ideal line is chosen. This incremental approach minimizes calculations and makes it suitable for hardware implementation.

**Algorithm: -**

**Input**

* Start point and end point .

**Steps**

1. **Calculate Differences:**
2. **Determine the Dominant Axis:**
   * If Δx≥Δy, the line has a shallow slope (∣m∣ ≤ 1|).
   * If Δx<Δy, the line has a steep slope (∣m∣ > 1).
3. **Set Step Directions:**
   * , otherwise
   * , otherwise
4. **Initialize Decision Parameter:**
   * For **shallow slopes** (∣m∣≤1|)
   * For **steep slopes** (∣m∣>1|)  
     Swap roles of x and y, and set:
5. **Plot the Initial Point:**
   * Plot .
6. **Iterate Until the Endpoint is Reached:**

* For k = 0 to the dominant axis length :
  + **For shallow slopes** (∣m∣≤1|)
    - Increment .
    - If
      * Increment .
      * Update
    - Else:
      * Update .
  + **For steep slopes (∣m∣>1|m| > 1):**
    - Increment .
    - If
      * Increment .
      * Update .
    - Else:
      * Update .
    - Plot the new point .

1. **Terminate:**

* Stop when the endpoint is plotted.

**Code: -**

import matplotlib.pyplot as plt

def BA():

    x1=int(input('Enter the value of x1: '))

    y1=int(input('Enter the value of y1: '))

    x2=int(input('Enter the value of x2: '))

    y2=int(input('Enter the value of y2: '))

    dx=abs(x2-x1)

    dy=abs(y2-y1)

    xes=[]

    yes=[]

    x,y = x1,y1

    if(dx>=dy):

        if(x2>x1):

            sx=1

        else:

            sx=-1

        if(y2>y1):

            sy=1

        else:

            sy=-1

        k=0

        Po=(2\*dy)-dx

        Pk=Po

        while(x<x2):

            xes.append(x)

            yes.append(y)

            x=x+sx

            if(Pk>=0):

                y=y+sy

                pkk=Pk+2\*dy-2\*dx

            else:

                pkk=Pk+2\*dy

    if(dy>=dx):

        if(x2>x1):

            sx=1

        else:

            sx=-1

        if(y2>y1):

            sy=1

        else:

            sy=-1

        k=0

        Po=(2\*dx)-dy

        Pk=Po

        while(x<x2):

            xes.append(x)

            yes.append(y)

            y=y+sy

            if(Pk>=0):

                x=x+sx

                pkk=Pk+2\*dx-2\*dy

            else:

                pkk=Pk+2\*dx

    plt.plot(xes, yes, marker='x')

    plt.show()

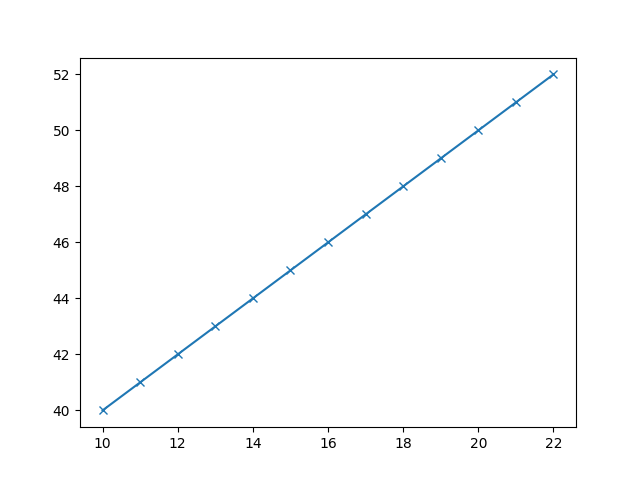
BA()

**Output: -**

Enter the value of x1: 10

Enter the value of y1: 40

Enter the value of x2: 23

Enter the value of y2: 56

**Discussion**

The Bresenham's Line Drawing Algorithm is an efficient method for plotting lines on a raster grid using integer arithmetic, avoiding floating-point operations. It handles both shallow (∣m∣≤1|m|) and steep (∣m∣>1) slopes by updating the decision parameter incrementally. Challenges like handling negative slopes and switching dominant axes for steep lines were addressed during implementation. The experiment confirmed that the algorithm is accurate and efficient, even for extreme cases like vertical and horizontal lines.

**Conclusion**

The Bresenham's Algorithm is a reliable and efficient approach to rasterizing lines, making it ideal for computer graphics. Its simplicity, accuracy, and computational efficiency ensure its continued relevance in real-world applications.