****

**DAA PROJECT**

**ALGORITHM VISUALIZER**

# 

**GROUP MEMBERS:**

**Syed Muhammad Hassan Naqvi [22K-4393]**

**Mishaal Bheraiya [22K-4393]**

#### 

#### 

#### **ABSTRACT**

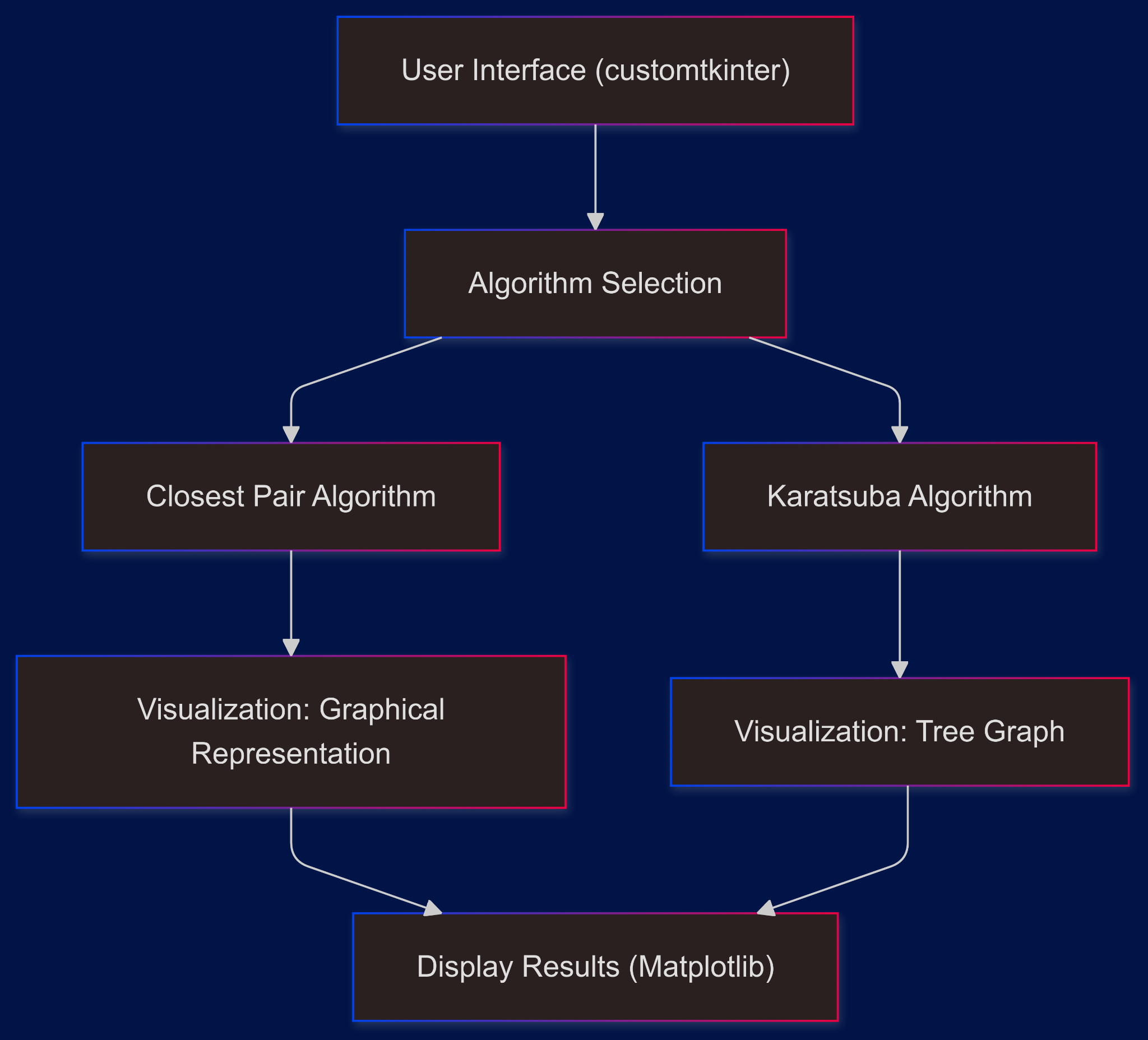
This report presents an Algorithm Visualizer system designed to aid users in understanding and interacting with computational algorithms through a graphical interface. The system supports two algorithms: Closest Pair of Points and Karatsuba Integer Multiplication. By leveraging Python and visualization tools such as matplotlib and networkx, the application allows users to observe algorithmic processes dynamically. This report outlines the system's architecture, experimental setup, results from benchmark tests, and key insights.

#### **INTRODUCTION**

Algorithm visualization is a crucial tool in computational education and research, enabling users to better understand complex operations through graphical representations. The presented system combines functionality with interactivity, allowing users to input datasets, execute algorithms, and visualize their results. The application is built using Python with customtkinter for the graphical user interface, and matplotlib and networkx for plotting. The system supports two algorithms:

1. Closest Pair of Points: A divide-and-conquer approach to find the two closest points in a plane.
2. Karatsuba Integer Multiplication: A recursive algorithm for fast multiplication of large integers.

#### **PROPOSED SYSTEM**



**User Interface**:

* Built using customtkinter to provide an interactive GUI.
* Users can select algorithms and visualize their operation.

**Algorithm Selection**:

* The user chooses one of the implemented algorithms:
  + **Closest Pair of Points**: Finds the closest pair in a set of points.
  + **Karatsuba Integer Multiplication**: Efficiently multiplies large integers.

**Algorithms**:

* Each algorithm is implemented separately and processes input data provided by the user.
* Outputs are computed and sent to the visualization component.

**Visualization**:

* Uses **matplotlib** to plot graphs and **networkx** for visualizing trees.
* Provides a graphical representation of the algorithm’s process and results.

**Multithreading**:

* Ensures the interface remains responsive during heavy computation.

#### **EXPERIMENTAL SETUP**

**Input Dataset**

The datasets for the experimental evaluation were generated using the [Number Generator](https://numbergenerator.org/) website. The datasets differ in size, range, and type, ensuring diverse scenarios for performance evaluation. Below is a summary of the datasets used:

1. **Closest Pair of Points**:
   * Text files containing 2D points (x y format).

| **Dataset Name** | **Size (Number of Points)** | **Range of Coordinates (X and Y)** | **Dataset Name** | **Size (Number of Points)** | **Range of Coordinates (X and Y)** |
| --- | --- | --- | --- | --- | --- |
| file1 110 bw | 110 | -10 to 70 | file6 160 bw | 160 | -50 to -10 |
| file2 150 bw | 150 | -50 to 50 | file7 120 bw | 120 | -20 to 60 |
| file3 200 bw | 200 | 30 to 35 | file8 150 bw | 150 | 10 to 80 |
| file4 150 bw | 150 | 20 to 80 | file9 200 bw | 200 | -90 to -10 |
| file5 150 bw | 150 | 1 to 99 | file10 200 bw | 200 | -100 to 100 |

1. **Integer Multiplication**:
   * Text files containing pairs of integers (x y format).

| **Dataset Name** | **Type** | **Dataset Name** | **Type** |
| --- | --- | --- | --- |
| file1-3 digit pos | 3-digit positive integers | file6-4 digit mix | Mixed 4-digit integers |
| file2-3 digit neg | 3-digit negative integers | file7-5 digit pos | 5-digit positive integers |
| file3-3 digit mix | Mixed 3-digit integers | file8-5 digit neg | 5-digit negative integers |
| file4-4 digit pos | 4-digit positive integers | file9-5 digit mix | Mixed 5-digit integers |
| file5-4 digit neg | 4-digit negative integers | file10-6 digit | 6-digit integers |

#### **RESULTS AND DISCUSSION**

1. **Closest Pair of Points**:
   * Input: 150 random points between 20 and 80.
   * Output: Closest pair and their city block distance.
     + Example: Closest Pair: ((75.0, 50.0),(75.0, 49.0)), Distance: 1.00
   * Visualization: Points plotted on a graph with the closest pair highlighted.
2. **Karatsuba Integer Multiplication**:
   * Input: Pair -831 -429
   * Output: Product: 356499
   * Visualization: Recursive steps visualized as a tree graph, showing each multiplication step.

The algorithm consistently produced the correct closest pair and computed distances accurately. The results also verified the algorithm's ability to handle both positive and negative coordinate ranges.

The Karatsuba algorithm correctly computed all results and matched the standard multiplication results. Handling mixed, positive, and negative integers demonstrated the robustness of the implementation.

#### **CONCLUSION**

The Algorithm Visualizer system provides a user-friendly and interactive platform for understanding computational algorithms. By enabling step-by-step visualization, the application bridges the gap between theoretical concepts and practical understanding. Future improvements could include support for additional algorithms and real-time performance metrics.

#### **REFERENCES**

1. Closest Pair of Points Algorithm -Based on *Algorithm Design* by Kleinberg & Tardos (CMSC 451 Lecture Slides, Carl Kingsford, University of Maryland).
2. Karatsuba Algorithm -Implementation referenced from [GeeksforGeeks](https://www.geeksforgeeks.org/karatsuba-algorithm-for-fast-multiplication-using-divide-and-conquer-algorithm/).
3. User Interface Design -Inspired by [Python Hub](https://python-hub.com/frames-and-radio-buttons-python-gui/).