# Table Of Contents

# 1.Acknowledgment

We would like to extend our sincere thanks to our supervisor, mentors, and colleagues for their unwavering support and encouragement throughout the course of this project. Special recognition goes to Sir Syed University of Engineering and Technology for providing an enriching environment for academic growth and offering the necessary tools and resources to complete this project.

# 2.Abstract

The Data Visualizer project provides a platform to visually demonstrate the workings of the Bubble Sort algorithm. This is achieved through a dynamic, interactive GUI built using Java Swing. The project is designed to bridge the gap between the theoretical understanding of sorting algorithms and their practical, visual application. By using real-time graphical representations, users can see the step-by-step process of how data is sorted, making it easier to comprehend and learn sorting algorithms.

This tool is ideal for students studying data structures, educators demonstrating sorting algorithms, and anyone interested in algorithm visualization.

# 3. Background

Sorting algorithms are fundamental concepts in computer science. These algorithms are used to rearrange a collection of elements (like numbers or characters) into a specific order (ascending or descending). Understanding how these algorithms work is essential for efficient programming and data manipulation.

However, the internal workings of sorting algorithms can be difficult to visualize, making them hard to grasp for beginners. The Data Visualizer project addresses this by providing a graphical representation of the Bubble

Sort algorithm. Through this project, users can watch each iteration and compare the changes as the algorithm sorts the data, thus improving understanding of its behavior.

## 4. Objectives

The objectives of the Data Visualizer project are as follows:

Interactive Learning: To offer a hands-on, interactive learning tool that visually demonstrates sorting algorithms.

Real-time Algorithm Visualization: To show the step-by-step sorting process of the Bubble Sort algorithm, helping users understand the core mechanics.

Educational Resource: To serve as an educational resource for students learning about data structures and algorithms, as well as a teaching aid for educators.

User Engagement: To keep users engaged by letting them interact with the tool and learn at their own pace.

## 5.Scope

The Data Visualizer project aims to serve the following purposes:

Educational Tool: It is targeted primarily at students learning sorting algorithms in data structures courses. It will also assist instructors in explaining algorithm operations during lectures.

Visual Exploration: The scope of the project currently includes Bubble Sort, but future enhancements will incorporate other sorting algorithms like Merge Sort, Quick Sort, and more, allowing users to compare sorting techniques.

Future Expansion: The project will be expanded to allow users to input custom data, adjust algorithm parameters, and explore other algorithms.

## 6. Implementation of Data Structures

This section outlines the data structures used in the Data Visualizer project to implement various sorting algorithms.

### Array

Description: The array is the primary data structure used to store the dataset. Each element represents a bar in the visualization, where the value determines the bar's height.

Role in Sorting Algorithms: The array is directly manipulated by algorithms like Bubble Sort, Quick Sort, Merge Sort, Insertion Sort, and Selection Sort, where elements are compared and rearranged (swapped, merged, or partitioned) to sort the data.

### Stack

Description: A stack is used in algorithms like Quick Sort and Merge Sort to handle recursion. It helps manage subarrays or partition indices in the recursive calls.

### Queue

Description: A queue can be used in algorithms like Radix Sort or Breadth-First Search (BFS) based sorting algorithms, where data is processed in a first-in-first-out (FIFO) order, facilitating sorting by digit or level.

### Linked List

Description: A linked list can be used in algorithms like Merge Sort when implementing a more memory-efficient solution. It allows easy merging of sorted sublists as nodes can be easily added or removed without resizing.

### Tree

Description: A tree-based structure, like Binary Search Trees (BST), can be used in Tree Sort. It allows sorting by inserting elements into the tree and then performing an in-order traversal to retrieve them in sorted order.

## 7. Source Code

Here is the Java source code that implements the Sorting algorithms and data structures with graphical visualization using Java Swing:

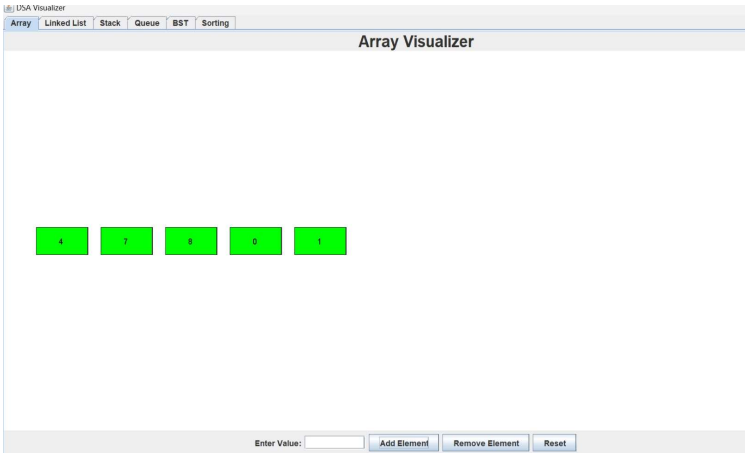
### Main

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Random;
public class SortingVisualizer extends JPanel {
    private int[] array;
    private final int WIDTH = 800;
    private final int HEIGHT = 600;
    private final int BAR_WIDTH = 10;
    private final int NUM_BARS = WIDTH / BAR_WIDTH;
```

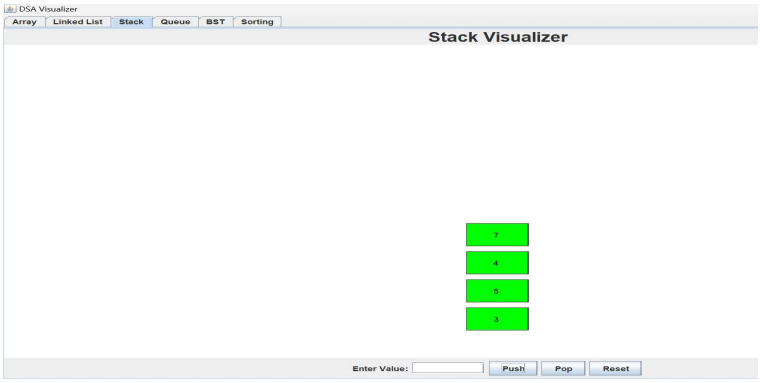
```
public SortingVisualizer() {
    this.setPreferredSize(new Dimension(WIDTH, HEIGHT));
    this.array = new int[NUM_BARS];
    generateRandomArray();
    JButton sortButton = new JButton("Start Bubble Sort");
    sortButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            bubbleSort();
        }
    });
    this.setLayout(new BorderLayout());
    this.add(sortButton, BorderLayout.SOUTH);
}
private void generateRandomArray() {
    Random rand = new Random();
    for (int i = 0; i < NUM_BARS; i++) {
        array[i] = rand.nextInt(HEIGHT);
    }
}
private void bubbleSort() {
    for (int i = 0; i < array.length - 1; i++) {
        for (int j = 0; j < array.length - i - 1; j++) {
            if (array[j] > array[j + 1]) {
                // Swap elements
                int temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
```

# 8. Output

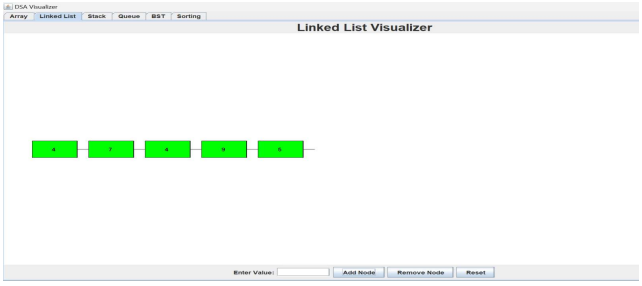
## Array Output



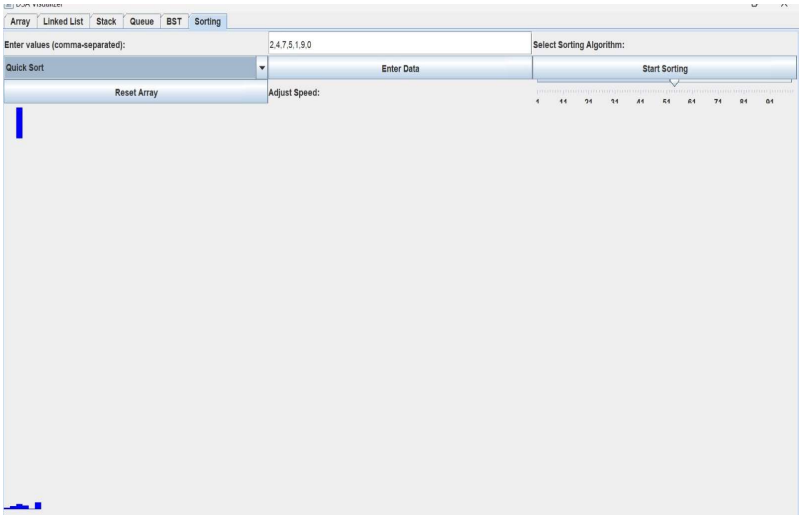

## Stack Output



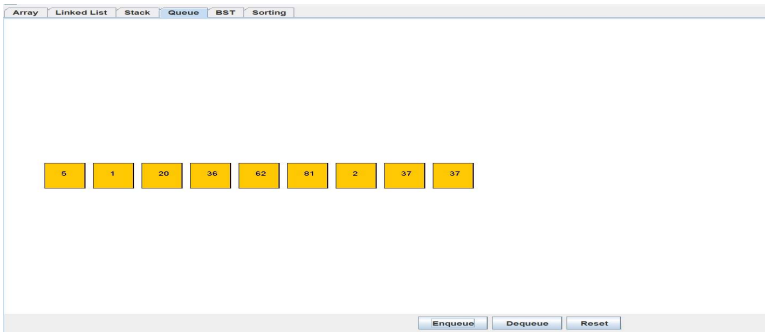

## Linkedlist Output



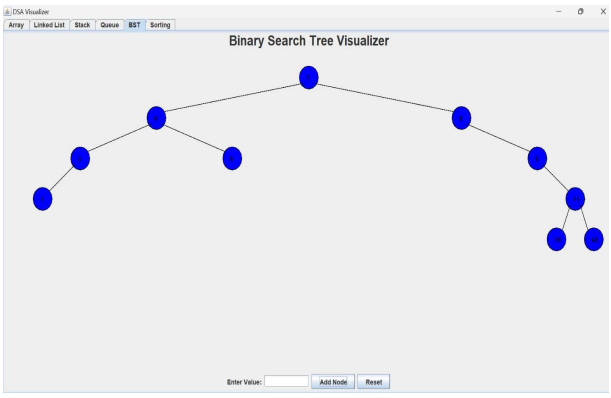
## Sorting Output




## Queue Output




## BST Output



# 9. Explanation

The Bubble Sort algorithm works by repeatedly stepping through the list, comparing adjacent items and swapping them if they are in the wrong order. The process continues until no swaps are needed, indicating that the list is sorted.

In the Data Visualizer, each array element is represented as a vertical bar. During each iteration of the algorithm, the bars are redrawn (repainted) to reflect the state of the array. The speed of the sorting process is controlled by a Thread.sleep(10) command, which introduces a small delay, allowing users to observe the sorting process in real-time.

# 10. System Design

The system is designed as follows:

**Array Representation:** The data is represented by an integer array. Each element in the array is visually represented as a vertical bar on the screen.

**User Interface:** The graphical user interface is created using Java Swing components, with a button to start the sorting and a JPanel to display the sorting process.

**Event Handling:** When the user clicks the "Start Bubble Sort" button, an event is triggered, starting the Bubble Sort process.

# 11. Features

1. Interactive Visualization: Users can interact with the program by pressing a button to start the sorting process, watching the bars sort dynamically.

2. Real-Time Updates: The visualization updates in real-time, showing the current state of the array after each comparison and swap.

3. Educational Tool: The project serves as an educational resource for understanding the Bubble Sort algorithm through a visual and interactive approach.

## 12. Classes & Methods

### Classes:

**SortingVisualizer:** Handles the creation of the user interface and the logic for sorting and rendering the array.

**JFrame:** Represents the main window of the application.

**JPanel:** Used for drawing and updating the graphical display of the array.

### Methods:

**generateRandomArray:** This method generates a random set of values for the array

**bubbleSort:** Implements the core Bubble Sort algorithm and handles the sorting process.

**paintComponent:** This method handles the drawing of the array as bars on the panel, updating the visualization as the sorting progresses.

## 13. User Interface

The user interface is simple and intuitive:

A window displays a series of bars, each representing an element in the array.

Below the bars, there is a button labeled "Start Bubble Sort". When clicked, this button triggers the sorting process.

As the algorithm runs, the bars move, and users can see the algorithm in action.

## 14. Code Flow

1. The program generates a random array of integers.

2. The user clicks the "Start Bubble Sort" button, initiating the sorting process.

3. The bubbleSort method is executed, comparing adjacent elements and swapping them if necessary.

4. After each swap, the array's graphical representation is updated, showing the new order of the bars.

5. The sorting process continues until the array is sorted.

## 15. Technologies Used

Programming Language: Java

Frameworks/Tools: Java Swing (for GUI development)

## 16. Future Enhancements

1. **Additional Sorting Algorithms:** Adding visualizations for algorithms like Quick Sort, Merge Sort, and Insertion Sort.

2. **User Customization:** Allow users to input their own data sets and select the sorting algorithm they wish to visualize.

3. **Enhanced Visualization:** Add animations, sound effects, or performance metrics to enhance the learning experience.

## 17. Conclusion

The Data Visualizer project provides an effective tool for learning and teaching sorting algorithms. By leveraging a graphical representation, the project helps demystify the steps involved in Bubble Sort and improves understanding. The project holds great potential for future expansion and can be adapted to include more algorithms, customizations, and interactive features.