# Mohanad Elbehairy 231000095

# Eyad Abd El Kareem 231000644

# Mariam Shahinn 231001745

# Ahmed Fahmy 231000587

Project Report: Sorting Visualizer

# 1. Project Overview

The Sorting Visualizer is a graphical tool developed in Python using libraries like tkinter and random, which demonstrates the working of various sorting algorithms through real-time visualizations. It is an educational project aimed at helping users understand the internal mechanics and performance of different sorting techniques.

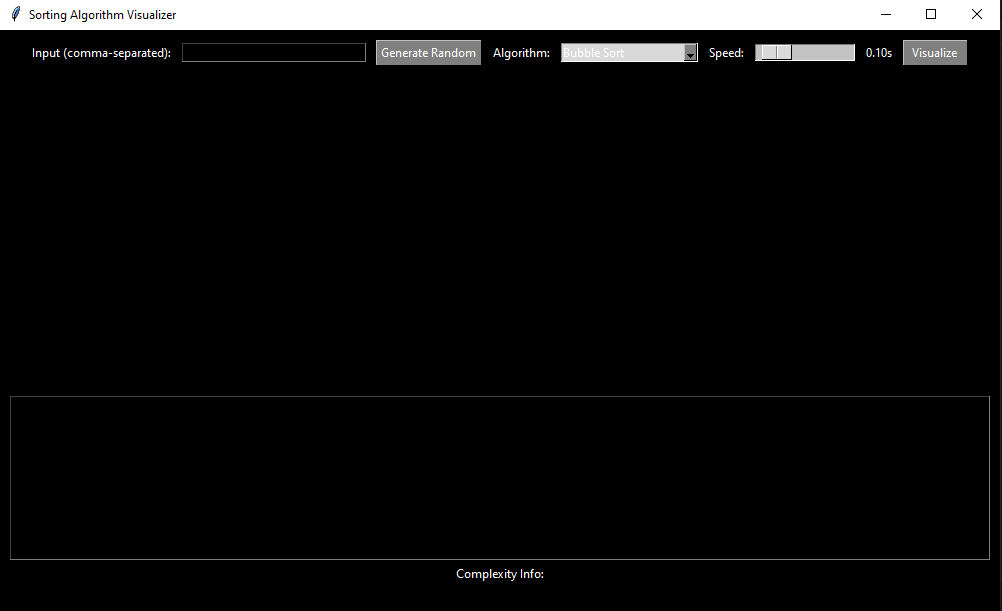
# 2. Implemented Algorithms

The visualizer supports multiple sorting algorithms:  
- Bubble Sort  
- Selection Sort  
- Insertion Sort  
- Merge Sort  
- Quick Sort  
  
Each algorithm is animated to show the process of sorting a list of numbers.

# 3. Visualization Method

- Bars represent array elements; the height of a bar corresponds to the element’s value.  
- Colors highlight comparisons and swaps to emphasize changes during the sorting process.  
- Users can control the speed of sorting and generate new random arrays.

# 4. How It Is Working



1- You can the numbers that you choose or can let the program generate random numbers with using “random” library in Python.

2- You choose the algorithm you want from the menu.

3- You can control the speed of the program you can choose to sort it in 0.01 sec up to 1.00 sec.

4- Final step is to press on visualize button to see the sorting in visualization mode.

# 5. Theoretical Analysis

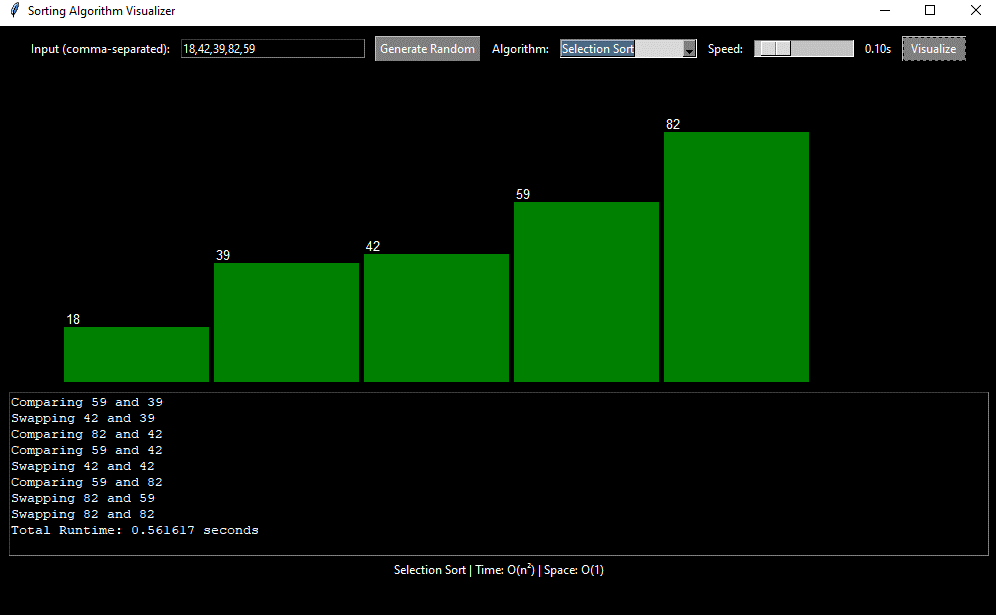
Algorithm | Best Time | Avg Time | Worst Time | Space   
-------------- | --------- | -------- | ----------- | ------   
Bubble Sort | O(n) | O(n²) | O(n²) | O(1)   
Selection Sort | O(n²) | O(n²) | O(n²) | O(1)   
Insertion Sort | O(n) | O(n²) | O(n²) | O(1)   
Merge Sort | O(n log n)| O(n log n) | O(n log n) | O(n)   
Quick Sort | O(n log n)| O(n log n) | O(n²) | O(log n)

# 6. Empirical Analysis

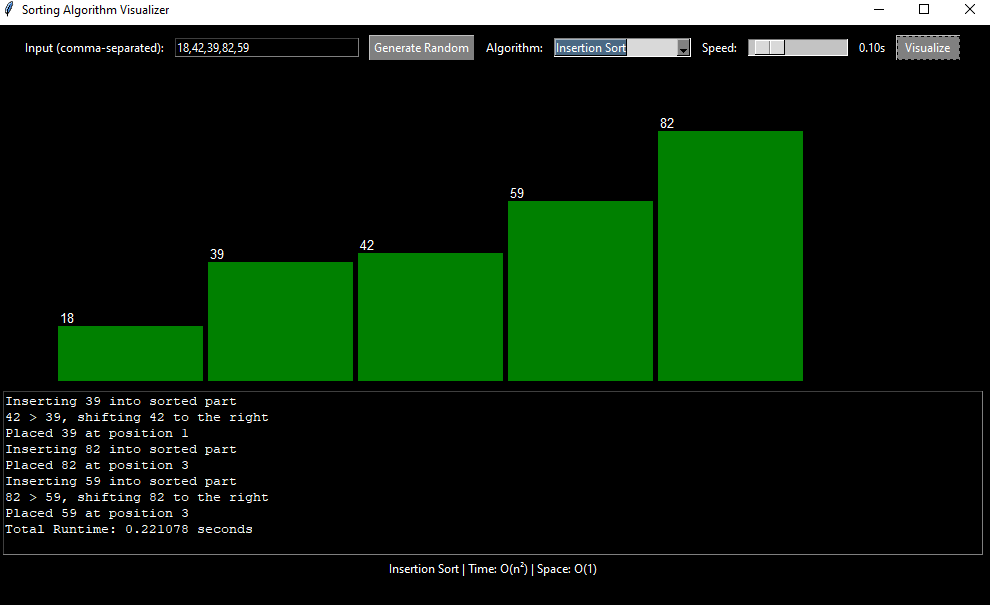
Testing was done with different array sizes (e.g., 10, 50, 100) and visualized execution speed.  
Key observations:  
- Bubble, Selection and Insertion Sort perform significantly slower with larger arrays.



Bubble Sort

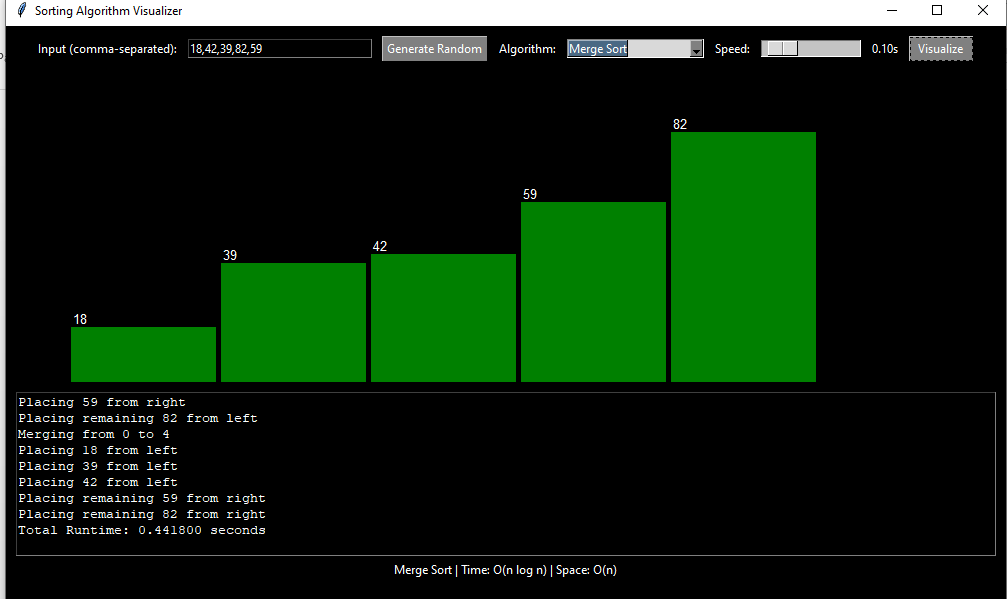


Selection Sort

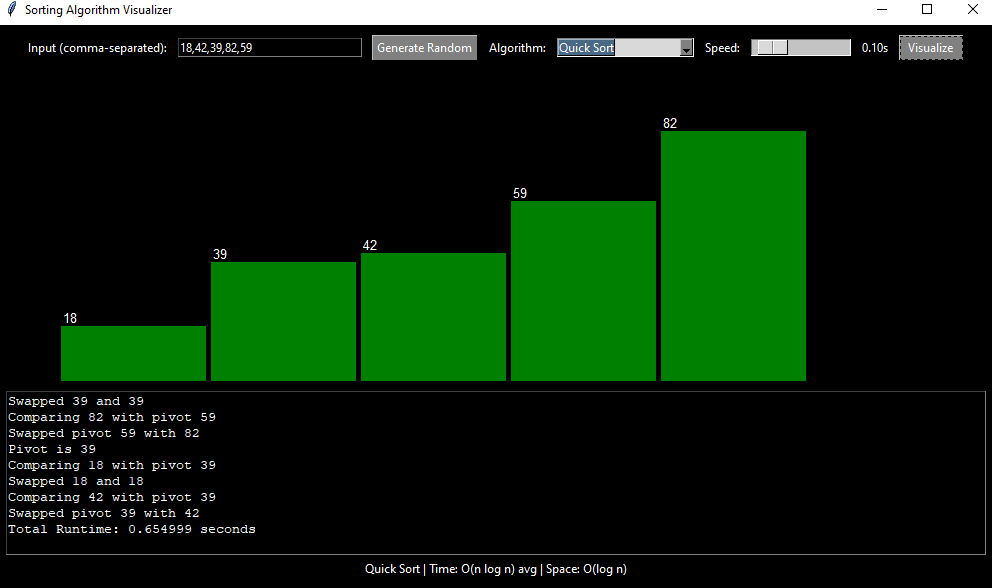


Insertion Sort

- Merge and Quick Sort demonstrate better scalability.



Merge Sort



Quick Sort

- Visual difference in performance is noticeable in sorting time and number of operations.

# 7. Technologies Used

- Python  
- Tkinter for GUI  
- Random module for generating arrays  
- Time for animation control

# 8. Conclusion

This project is a practical educational tool for understanding sorting algorithms. The visual animations help users see what textbook diagrams often fail to demonstrate: the real-time behavior and efficiency of algorithms under different conditions.