Introduction to Search and Sorting Algorithms

Linear Search

- Linear search is a simple search algorithm that checks every element in the list until the target is found or the list is exausted.
- Time Complexity : O(n), where n is number of elements in the list or array.
- As the target value is searched, algorithm checks every element until it finds the target or determines the target is not in the list.

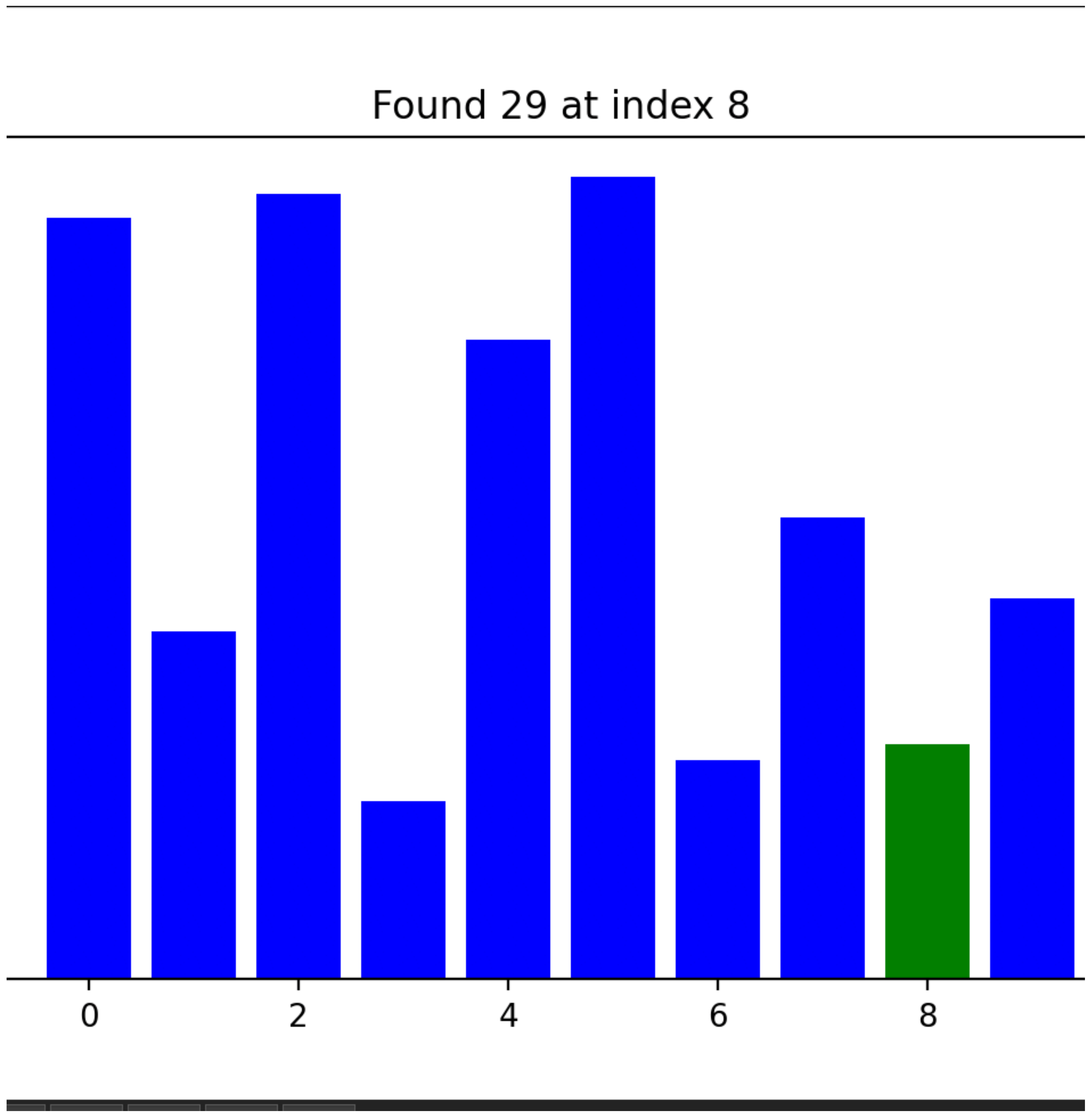
Linear Search Visualization Code

This code visualizes the Linear Search algorithm step by step.

- A random list of numbers is generated.
- The algorithm searches through the list one element at a time.
- The current element being checked is highlighted in yellow.
- If the target is found, it turns green, and the process stops.
- If the target is not found, the entire list is displayed in red , showing that the target is not in the list.

Linear Search Visualization

- This shows how linear search checks each element step by step. The target is highlighted when its found.



Bubble Sort

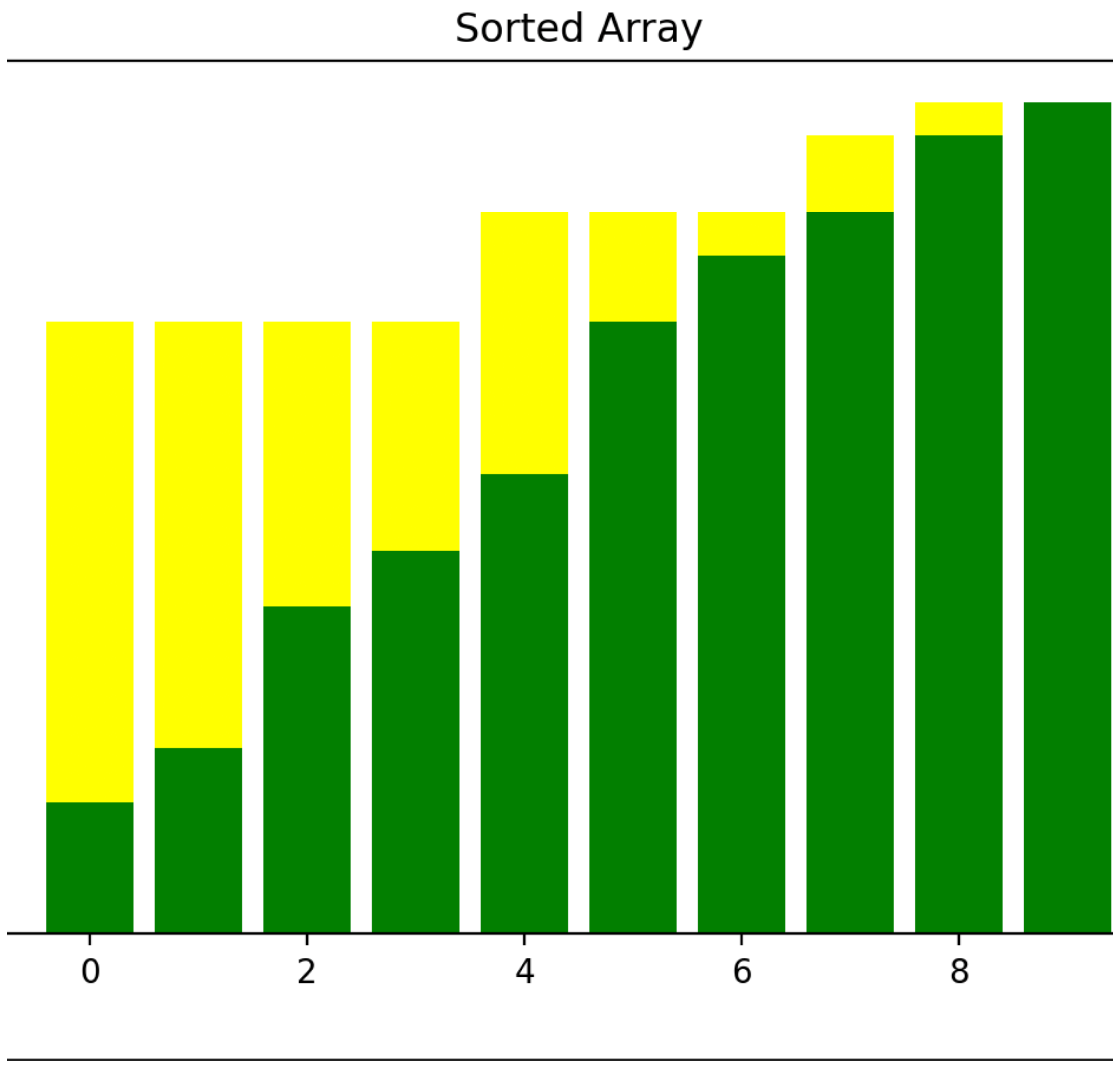
- Bubble sort is a simple algorithm that goes through the list multiple times, compares two elements at a time and swaps them if they are in the wrong order.
- Time complexity : O(n^2), where n is the number of elements in the list.
- In bubble sort the largest unsorted element ‘bubbles up’ to its correct position in each pass.

Bubble Sort Visualization Code

- A random list of numbers generated.
- The unsorted array is visualized in blue color.
- The algorithm compares two numbers at a time.
- The numbers being compared are shown in yellow.
- If the numbers are out of order, they are swapped.
- The array is updated and shown after each comparison.
- When array is sorted, it is shown in green.
- The Visualization helps to understand how the numbers are compared, swapped and sorted.

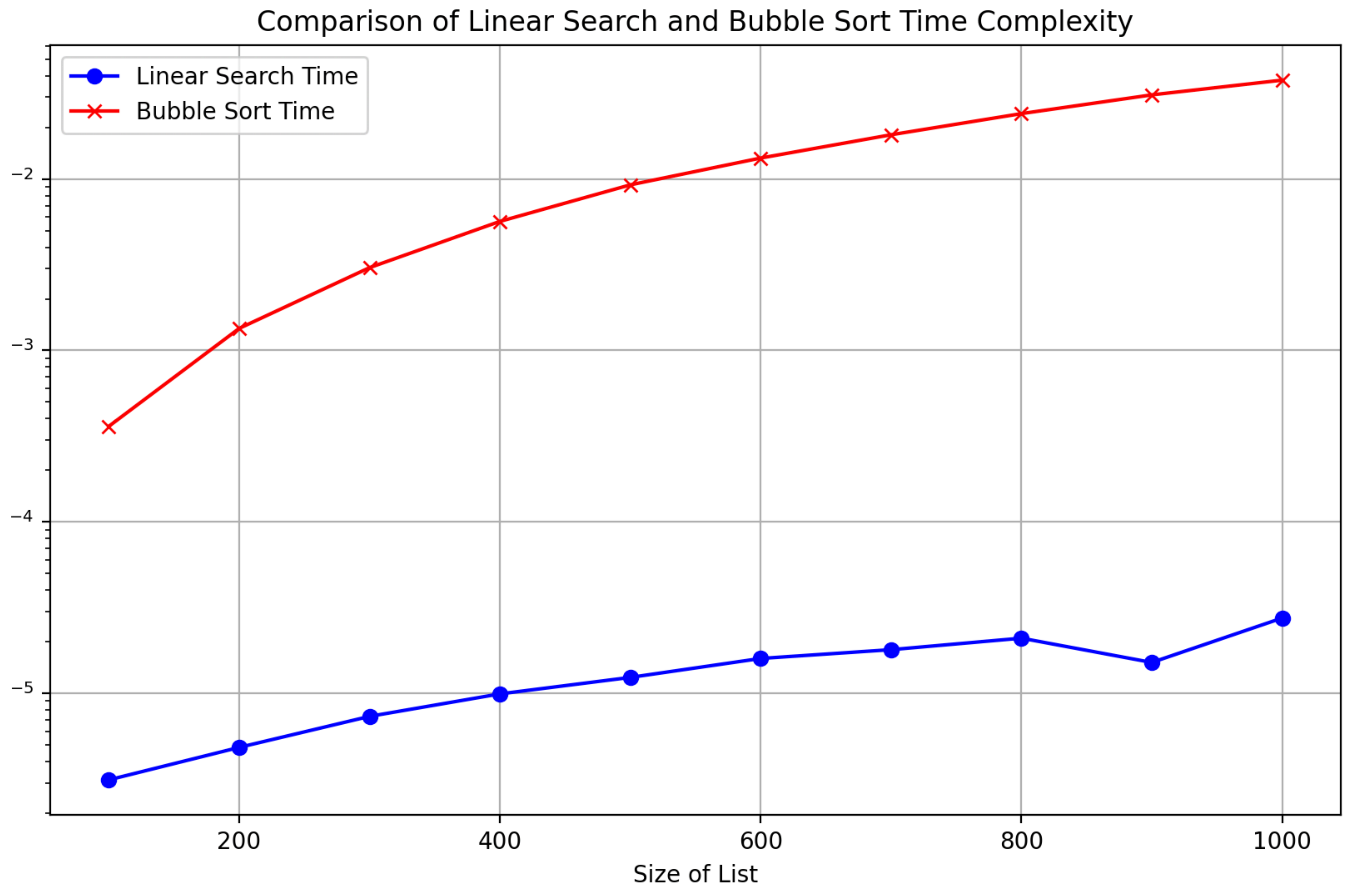
Bubble Sort Visualization

- This shows how bubble sort compares and swaps elements step by step. The list sorted gradually.



Time Complexity Analysis

- This graph compares the time of Bubble sort and Linear search. It shows how input size affects the time required time for each algorithm.



Advantages and Disadvantages of the Algorithms

Linear Search

- Advantages:
- It is easy to write and understand.
- It doesn’t require the list to be sorted, which makes it more flexible.
- Disadvantages:
- As the size of dataset grows, the time it takes to find an element increases significantly.(Inefficient in large datasets)
- Time complexity is O(n) which means that in worst case, it will have to look at every element in the list.

Bubble Sort

- Advantages:
- The algorithm is straightforward and easy to code.
- It performs better when the list is small or nearly sorted.
- Disadvantages:
- For large lists bubble sort is really slow.
- Time complexity is O(n^2) which means It makes repeated comparisons and swaps so it can be slow for large lists.

Code Improvements and Errors

- Challenges with large datasets: Algorithms may struggle with performance when dealing with larger input sizes, leading to longer execution times.
- Alternative algorithms for better efficiency: Algorithms like Quick sort and Merge sort provide faster performance with larger datasets due to their better time complexity.

Tools and Environment Used

- IDE: Pycharm for Python
- Libraries: Matplotlib, Random, etc

Results

- Linear Search: Not efficient in large datasets as it checks every element one by one, making it slower for larger lists. It works well in small datasets but becomes inefficient as the dataset grows.
    - Bubble Sort: Works fine for small datasets but becomes very slow with larger lists due to its quadratic time complexity O(n^2).
    - For larger datasets, Linear Search is generally more efficient than Bubble Sort.