Insertion Sort

Introduction

Insertion Sort is a simple sorting algorithm that builds the final sorted list one element at a time. It is efficient for small datasets or partially sorted lists and is often used in combination with more complex algorithms.

Logic

1. Start with the second element (index 1) and compare it with the previous elements.
2. If the current element is smaller (or larger, depending on the desired order) than the previous element, shift the previous element one position to the right.
3. Repeat step 2 until you find the correct position for the current element.
4. Insert the current element into its correct position in the sorted part of the list.

C++ implementation of Insertion sort: Insertion Sort.
Python implementation of Insertion sort: Insertion Sort.
Java implementation of Insertion sort: Insertion Sort.
Javascript implementation of Insertion sort: Insertion Sort.

Complexity

- **Time Complexity**:
  - Best Case: O(n) (when the list is already sorted)
  - Average Case: O(n^2)
  - Worst Case: O(n^2)
- **Space Complexity**:
  - O(1) (constant extra space)

Advantages

- Simple and easy to implement.
- Efficient for small datasets or nearly sorted lists.

Considerations

- Inefficient for larger datasets due to its quadratic time complexity.
- Not suitable for datasets with large inversions (elements that are far from their sorted position).

Disadvantages/Limitations

- The number of comparisons and shifts can be relatively high, especially for larger datasets.
- There are more efficient sorting algorithms available for larger and unsorted datasets.

Edge Cases

- Insertion Sort's performance is relatively better when the initial list is partially sorted.
- It's an adaptive algorithm; its performance improves if the data is almost sorted.

External Links

To learn more about Insertion Sort, explore these resources: - Insertion Sort - Wikipedia - Sorting Algorithms: Insertion Sort - GeeksforGeeks - Insertion Sort Visualization - VisuAlgo