Selection Sort

Introduction

Selection Sort is a simple sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted part of the list and putting it into its correct position in the sorted part of the list.

Logic

1. Find the minimum (or maximum) element from the unsorted part of the list.
2. Swap the found minimum (or maximum) element with the first element of the unsorted part.
3. Move the boundary between the sorted and unsorted parts one element to the right.
4. Repeat steps 1-3 until the entire list is sorted.

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

Complexity

- Time Complexity:
  - Best Case: O(n^2)
  - Average Case: O(n^2)
  - Worst Case: O(n^2)
- Space Complexity:
  - O(1) (constant extra space)

Advantages

- Simple and easy to implement.
- Works well for small lists or when memory usage is a concern.

Considerations

- Inefficient for larger datasets due to its quadratic time complexity.
- Doesn't perform well on partially sorted lists.

Disadvantages/Limitations

- Performs unnecessary swaps, even if the list is partially sorted.
- There are more efficient sorting algorithms available for larger datasets.

Edge Cases

- Selection Sort's performance remains consistently poor regardless of the initial order of the list.
- It requires the same number of comparisons and swaps for any given input permutation.

External Links

To delve further into Selection Sort, check out these resources: - Selection Sort - Wikipedia - Sorting Algorithms: Selection Sort - GeeksforGeeks - Sorting Algorithm Visualization: Selection Sort - VisuAlgo