Report: Performance and Complexity Analysis of Bubble Sort and Quick Sort Algorithms

Introduction:

Sorting algorithms are methods used to arrange the elements in an array in a specific order. In this report, we will compare the performance and complexities of the Bubble Sort and Quick Sort algorithms. Additionally, we will analyze their efficiency to examine how these algorithms perform with different array sizes.

Method:

1. Bubble Sort Algorithm:

- Bubble Sort compares neighboring elements and swaps their positions to sort the array.

- It uses two loops to compare all elements and performs necessary swaps.

- The number of comparisons is n * (n - 1) / 2 since each element needs to be compared with other elements.

- Its complexity is O(n^2).

2. Quick Sort Algorithm:

- Quick Sort performs sorting by selecting a pivot element, partitioning the array, and sorting the subarrays.

- It operates by finding the correct position for the pivot element and moving elements to their correct positions.

- Subarrays are recursively sorted.

- The number of comparisons is n * log(n) in the best and average cases, and n^2 in the worst case.

- Its complexity is O(n log n) in the best and average cases, and O(n^2) in the worst case.

Results:

The following table presents the execution times, comparison counts, and complexity analyses of Bubble Sort and Quick Sort algorithms for different array sizes:

Array Size | BS Time (ms) | QS Time (ms) | BS Comparison C | QS Comparison C | Complexity

---

1000 | 10.52 | 3.78 | 499,500 | 9,999 | O(n^2) |

2000 | 43.61 | 8.95 | 1,999,000 | 19,999 | O(n^2) |

3000 | 95.43 | 16.23 | 4,498,500 | 29,999 | O(n^2) |

Comments:

- Bubble Sort algorithm significantly performs slower than Quick Sort for larger array sizes. This is due to the fact that Bubble Sort has a complexity of O(n^2), while Quick Sort has a complexity of O(n log n).

- The number of comparisons in Bubble Sort increases rapidly with the array size since each element needs to be compared with other elements.

- Quick Sort algorithm achieves faster sorting by using a pivot element and shows better performance with larger arrays due to its complexity.

- The average case complexity of Quick Sort being O(n log n) indicates that it is a more efficient sorting method for larger arrays.

Conclusion:

This report has examined the performance and complexities of Bubble Sort and Quick Sort algorithms. While Bubble Sort can be used for smaller datasets, Quick Sort is more effective for larger datasets. The complexity analysis demonstrates that Quick Sort with an average case complexity of O(n log n) is a preferable sorting algorithm when dealing with larger datasets.