Assignment 5: Quicksort Algorithm: Implementation, Analysis, and Randomization

# **1. Introduction**

This assignment explores two versions of the Quicksort algorithm—deterministic and randomized. Quicksort is a divide-and-conquer sorting algorithm that recursively partitions arrays around a pivot. While it has excellent average-case performance, its worst-case scenario can be problematic without randomization.

# **2. Implementation**

The deterministic version selects the last element as the pivot. The randomized version selects a pivot at random, reducing the chance of worst-case behavior for sorted or nearly sorted data.

# **3. Theoretical Time Complexity**

Best Case: O(n log n)  
Average Case: O(n log n)  
Worst Case: O(n^2) for deterministic when input is already sorted  
Space Complexity: O(log n) for both versions (due to recursion stack)

# **4. Empirical Analysis**

The performance of both algorithms was tested using arrays of size 1000, 5000, and 10000 with three distributions:

- Random  
- Sorted  
- Reverse Sorted

The following charts show the timing results:

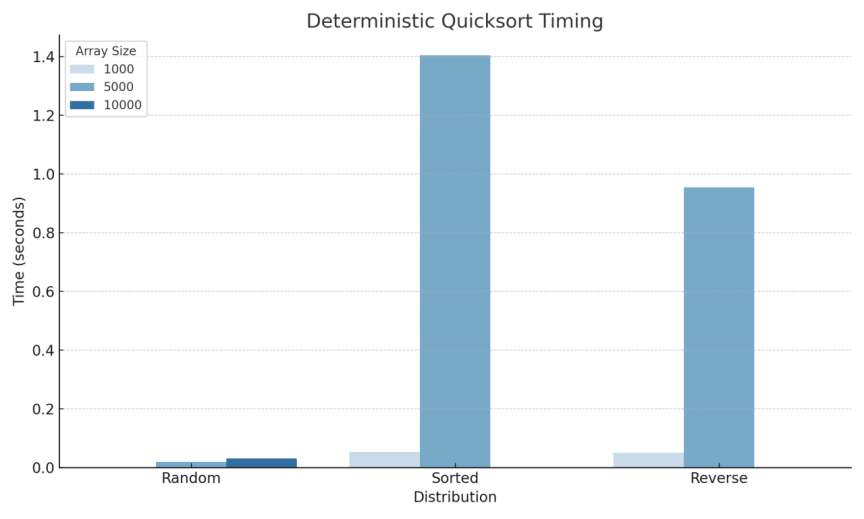


Figure 1: Timing for Deterministic Quicksort

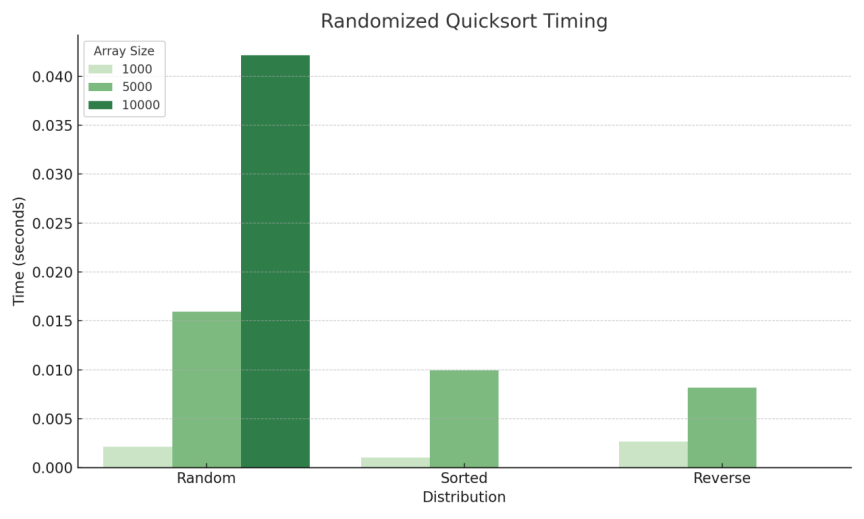


Figure 2: Timing for Randomized Quicksort

## **Timing Results Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Array Size | Distribution | Deterministic (s) | Randomized (s) |
| 1000 | Random | 0.00093 | 0.00211 |
| 1000 | Sorted | 0.05385 | 0.00104 |
| 1000 | Reverse | 0.04992 | 0.00266 |
| 5000 | Random | 0.01895 | 0.01592 |
| 5000 | Sorted | 1.40422 | 0.00994 |
| 5000 | Reverse | 0.95466 | 0.00815 |
| 10000 | Random | 0.03167 | 0.04213 |

# **5. Discussion**

The deterministic version performed well on random arrays but failed with sorted and reverse-sorted arrays of size 10000 due to maximum recursion depth errors. The randomized version handled all cases efficiently, demonstrating its effectiveness in avoiding worst-case partitioning.

# **6. Conclusion**

Randomization significantly improves Quicksort’s robustness. It prevents degradation on nearly sorted datasets, making it preferable for general-purpose sorting in modern applications.