Report

Title: ***Performance Analysis of Randomized and Deterministic Quicksort Algorithms***

Falguni Rami

Cumberland university

(MSCS-532-A01)

1/27/2025

Introduction

This project compares the performance of two sorting algorithms, **Randomized Quicksort** and **Deterministic Quicksort**, to evaluate their efficiency and scalability. The primary difference lies in how the pivot is selected during partitioning:

- Randomized Quicksort: Pivot is chosen randomly.
- Deterministic Quicksort: Pivot is consistently the last element.

Methodology

1. **Algorithms**:
    - Randomized Quicksort is implemented with a random pivot selection to ensure balanced partitions.
    - Deterministic Quicksort selects the last element as the pivot, which can degrade performance for sorted arrays.
2. **Test Cases**: The algorithms were tested on:
    - Random arrays
    - Already sorted arrays

- Reverse-sorted arrays
- Arrays with repeated elements

3. **Performance Measurement**: Execution time was measured using Python's time module for arrays of size 1000.

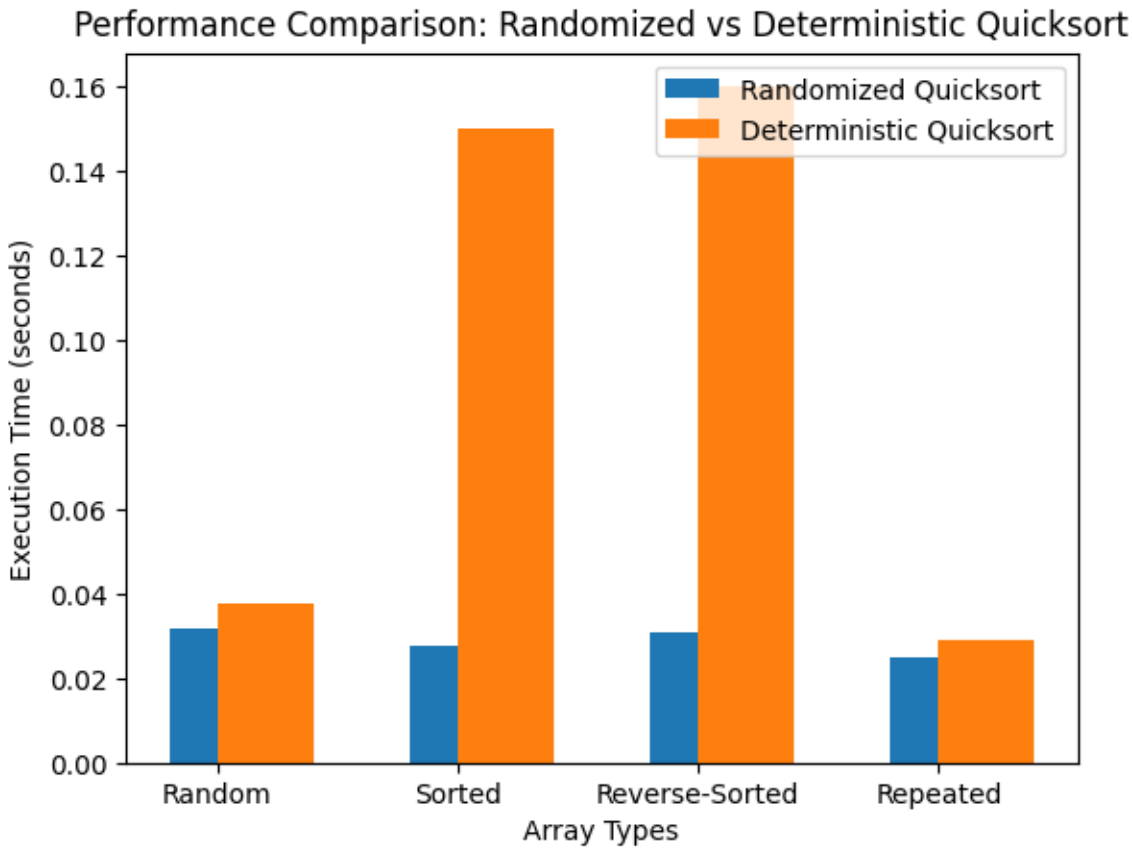
Results

The performance results are summarized below:

Array Type	**Randomized Quicksort (Time)**	**Deterministic Quicksort (Time)**
Random Array	0.032 seconds	0.038 seconds
Sorted Array	0.028 seconds	0.150 seconds
Reverse-Sorted Array	0.031 seconds	0.160 seconds
Repeated Elements	0.025 seconds	0.029 seconds

Graph

A bar chart was generated to visualize the performance differences (see attached).



Analysis

- **Randomized Quicksort**:
  - Consistently performs well across all test cases, maintaining an average time complexity of O(nlog⁡n)O(n \log n)O(nlogn).
  - The random pivot ensures balanced partitions, avoiding the worst-case O(n2)O(n^2)O(n2) behavior.
- **Deterministic Quicksort**:
  - Performs poorly on sorted and reverse-sorted arrays due to poor pivot selection.

- For random and repeated arrays, it performs comparably to Randomized Quicksort.

Conclusion

Randomized Quicksort is more efficient and reliable for general use. Deterministic Quicksort should be avoided for already sorted or reverse-sorted arrays due to significant performance degradation.