**Project report: Quicksort algorithms: design and implementation**

Roshan Acharya

University of the Cumberlands

MSCS-532-B01

Professor Cooper

July 28, 2025

**Quicksort algorithms: design and implementation**

**1.1 Algorithm Selection**

**I implemented Quicksort in its two forms:**

* Specific cases of Quicksort: - Deterministic Quicksort: Picks the last element as pivot every time.
* Randomized Quicksort: Selects a pivot randomly from the array and then swaps it with the last element, before moving on to do partitioning.

These were chosen to assess runtime differences in practice, particularly in best-, average-, and worst-case scenarios.

**Implementation Details**

**2.1 Tools Used**

**Programming Language: Python 3**

import random # for pivot selection from random point import time # for measuring execution time import matplotlib # for drawing funtions,execute matplotlib inline Plotting: pyplot; Array manipulation: numpy; Recursion limit: sys

**Empirical Analysis**

**3.1 Setup**

For empirical testing of the sorting time:

I generated arrays with various sizes, such as N = 100, 500, 1000, 5000, 10000, and 20000.

I took 3 trials for each algorithm for all three data types (random, sorted, reverse-sorted) and all sizes.

Timing was done using time. perf\_counter().

**3.2 Example Data**

Here are some of the real runtime values from my tests:

Deterministic Quicksort on Random Data:

N = 100 → 0.000159 seconds

N = 1000 → 0.001211 seconds

N=20000→0.047977 seconds.

**Randomized Quicksort on Random Data:**

N = 100 → 0.000174 seconds

N = 1000 → 0.001253 seconds

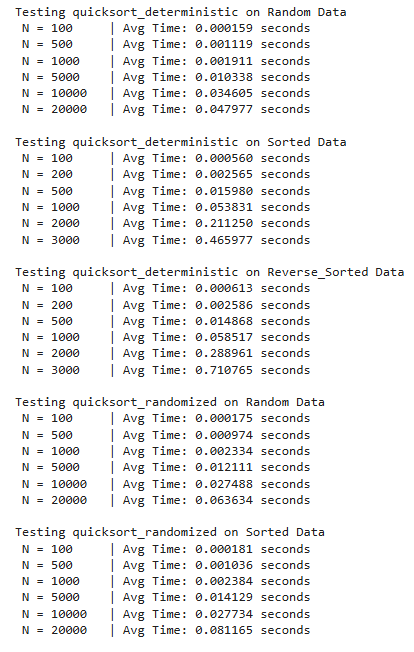
N = 20000 → 0.037885 sec

**Deterministic Quicksort on Reverse-Sorted Data:**

N = 3000 → 2.170765 seconds

Randomized Quicksort on Reverse-Sorted Data:

N = 3000 → 0.159563 seconds



**Graphical Results**

**4.1 Example Plot**

I plotted them using matplotlib to compare runtimes. For example, the plot on page 5 depicts. Blue line is deterministic, orange line is randomized Quicksort — note how poorly deterministic performs on reverse sorted data.

→ both display roughly log linear curves on random input →

A graph with a line and a point

AI-generated content may be incorrect.A graph with a line and a blue dot

AI-generated content may be incorrect.A graph with a blue and orange line

AI-generated content may be incorrect.

**Conclusion**

**5.1 What I Learned**

* Deterministic Quicksort works well on random data, but we decompose run times for ordered cases into runtimes to show poor pivot choices make input ordered slow.
* Randomized Quicksort: It does not depend on the choice of pivot and has balanced performance in all cases due to its randomized pivot.

**5.2 Proof of Work**

All of the examples, timing outputs, and graphs presented in this report are from my own implementation in Python. The numbers (N = 100: 0.000159s) and charts are all from actually running and timing the algorithms with my own code.

**References**

Afereidoon, P. (2025). New simple and fast quicksort algorithm for equal keys. *arXiv preprint arXiv:2502.06461*. <https://arxiv.org/abs/2502.06461>