Algorithm Efficiency and Scalability Analysis

Understanding Algorithm Efficiency and Scalability through Randomized and Deterministic Quicksort

Project Description

This project investigates the empirical performance differences between two sorting algorithms: Randomized Quicksort and Deterministic Quicksort. By analyzing their execution times on datasets of various sizes and characteristics, we assess their efficiency and scalability in practical scenarios.

Project Structure

- quicksort_analysis.py: Python script implementing both quicksort variants, dataset generation, timing, and plotting logic.

- results_plots/: Directory created automatically to save runtime comparison plots.

- quicksort_comparison.png: Visualization of runtime vs. input size for both algorithms.

How to Run the Code

1. Prerequisites

- Python 3.7+

- Required libraries: matplotlib, random, os, time

2. Installation

Install required libraries using pip:

```
pip install matplotlib
```

3. Execution

Run the analysis script:

```
python quicksort_analysis.py
```

This will generate datasets of sizes 1000, 2000, 4000, 8000, and 16000; run both algorithms; record execution time; and save a plot in the results_plots folder.

Summary of Findings

- Randomized Quicksort outperforms Deterministic Quicksort, especially as data size increases.

- Deterministic Quicksort performance deteriorates on already sorted or reverse-sorted inputs, due to poor pivot choice.

- Random pivot selection mitigates worst-case scenarios, maintaining expected average time complexity of O(n log n).

- Empirical results align with theoretical expectations, confirming the benefits of randomized pivoting.

Contact

For questions or suggestions, please contact:

Sudhansu Sekhar Dash

Git: https://github.com/SDash07/Algorithm-Efficiency-and-Scalability