Lab 3

How To Run

This document contains details about various algorithm implementations and comparisons. To execute the provided implementations, ensure the prerequisites are met and follow the commands mentioned in each section.

Prerequisites

1. Python

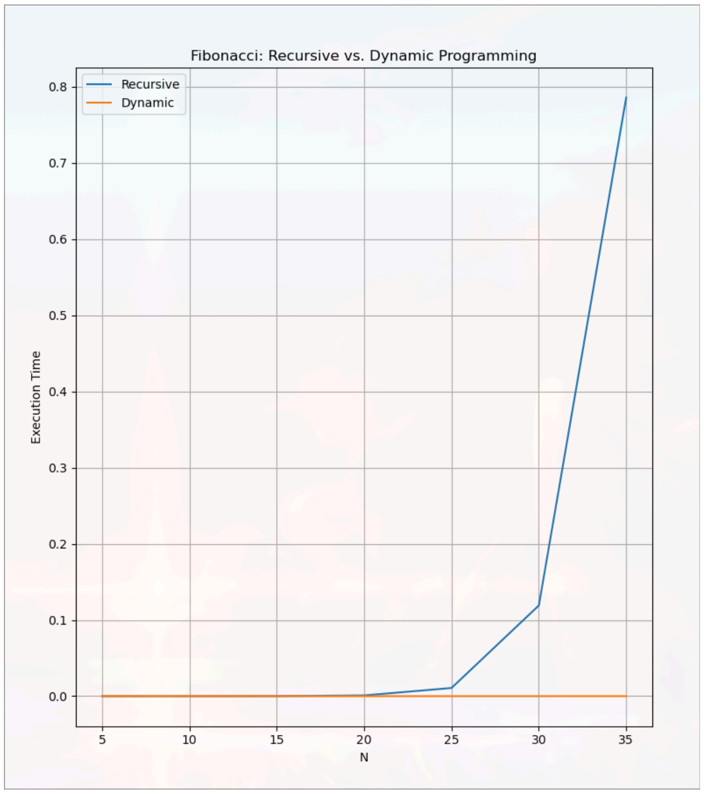
2. Matplotlib

1. Fibonacci Series: Recursive vs Dynamic Programming

This section compares the recursive and dynamic programming approaches to compute the Fibonacci series.

Command to run: python -m lab3.fibonacci

Output



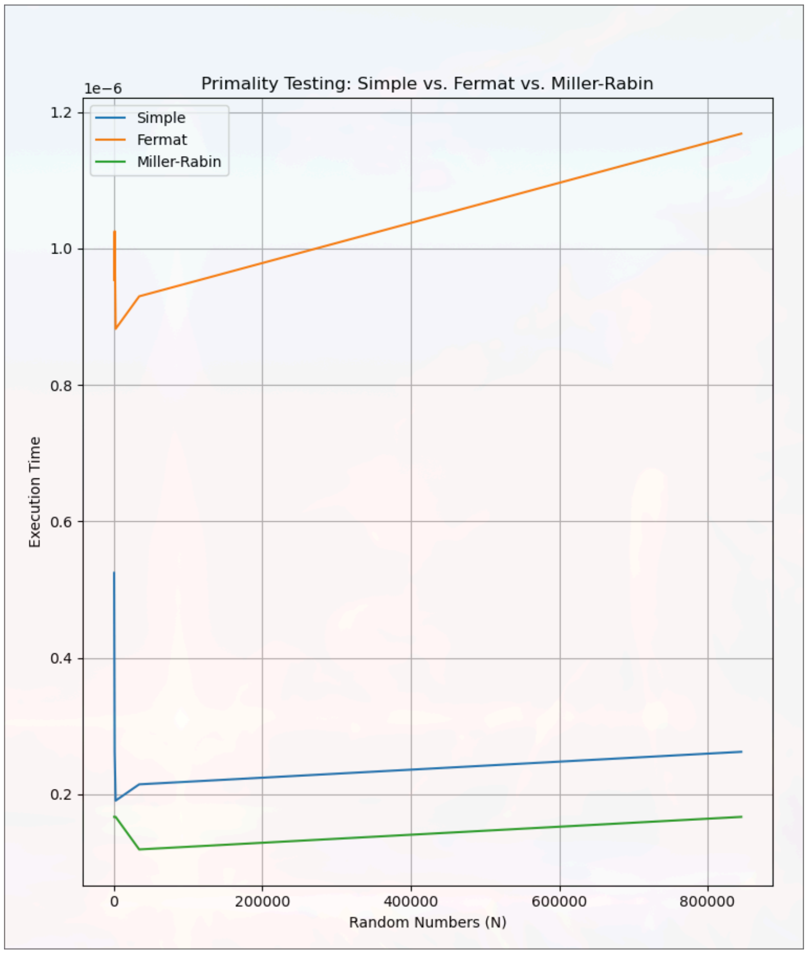
We can see that regular fibonacci problem can not be solved for larger values, and dynamic solution is solving the same problem in infinitesemally less time.

2. Monte Carlo Primality Tests: Fermat vs Miller-Rabin

This section implements and compares two probabilistic primality tests: Fermat's test and the Miller-Rabin test.

Command to run: python -m lab3.primality

Output



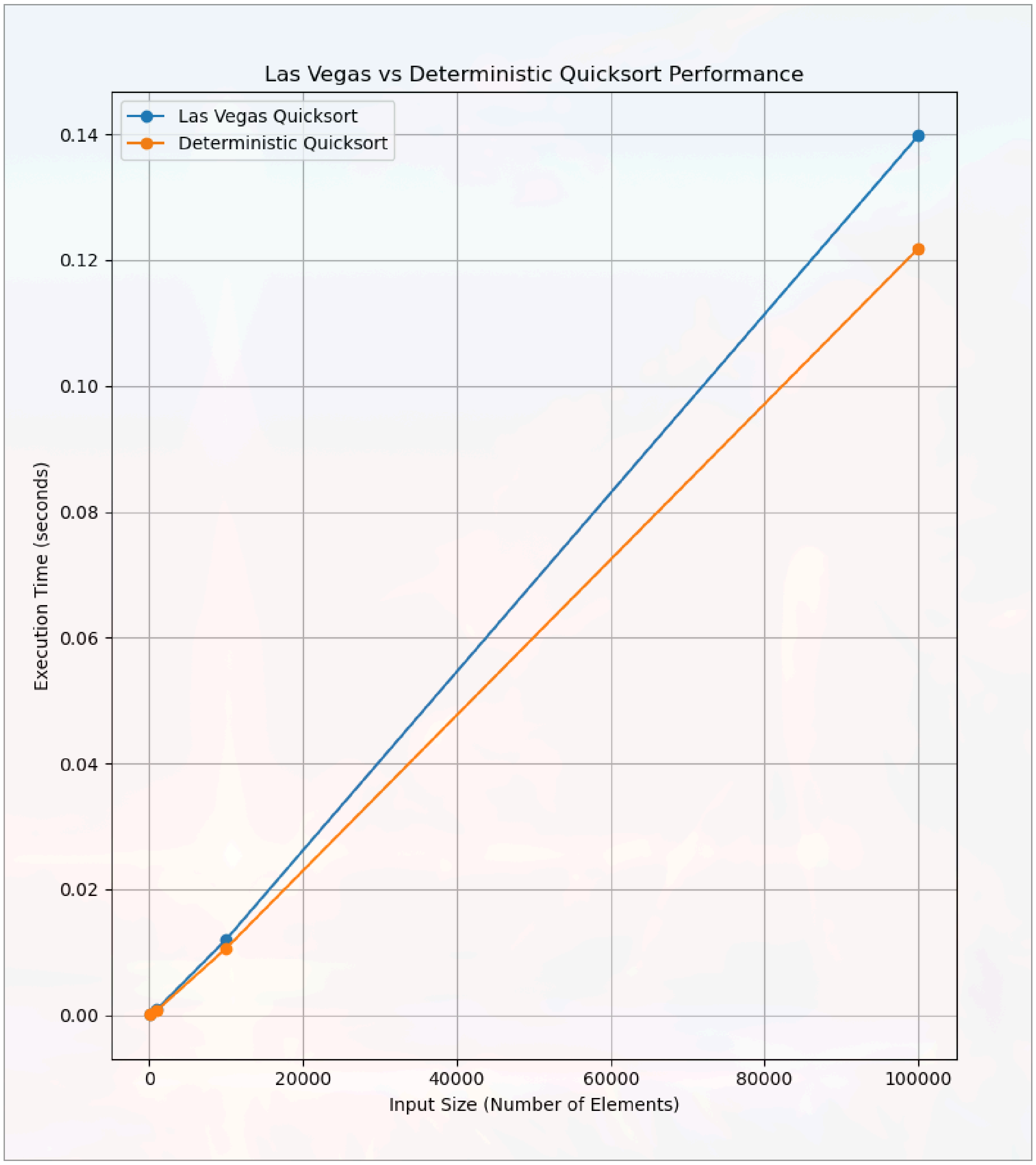
This shows that Miller-Rabin probabilistic algorithm is a lot faster then its counter parts.

3. Las Vegas Approach for Quicksort

This section demonstrates the Las Vegas algorithm applied to Quicksort, which uses randomness to ensure optimal performance.

Command to run: python -m lab3.las_vegas

Output



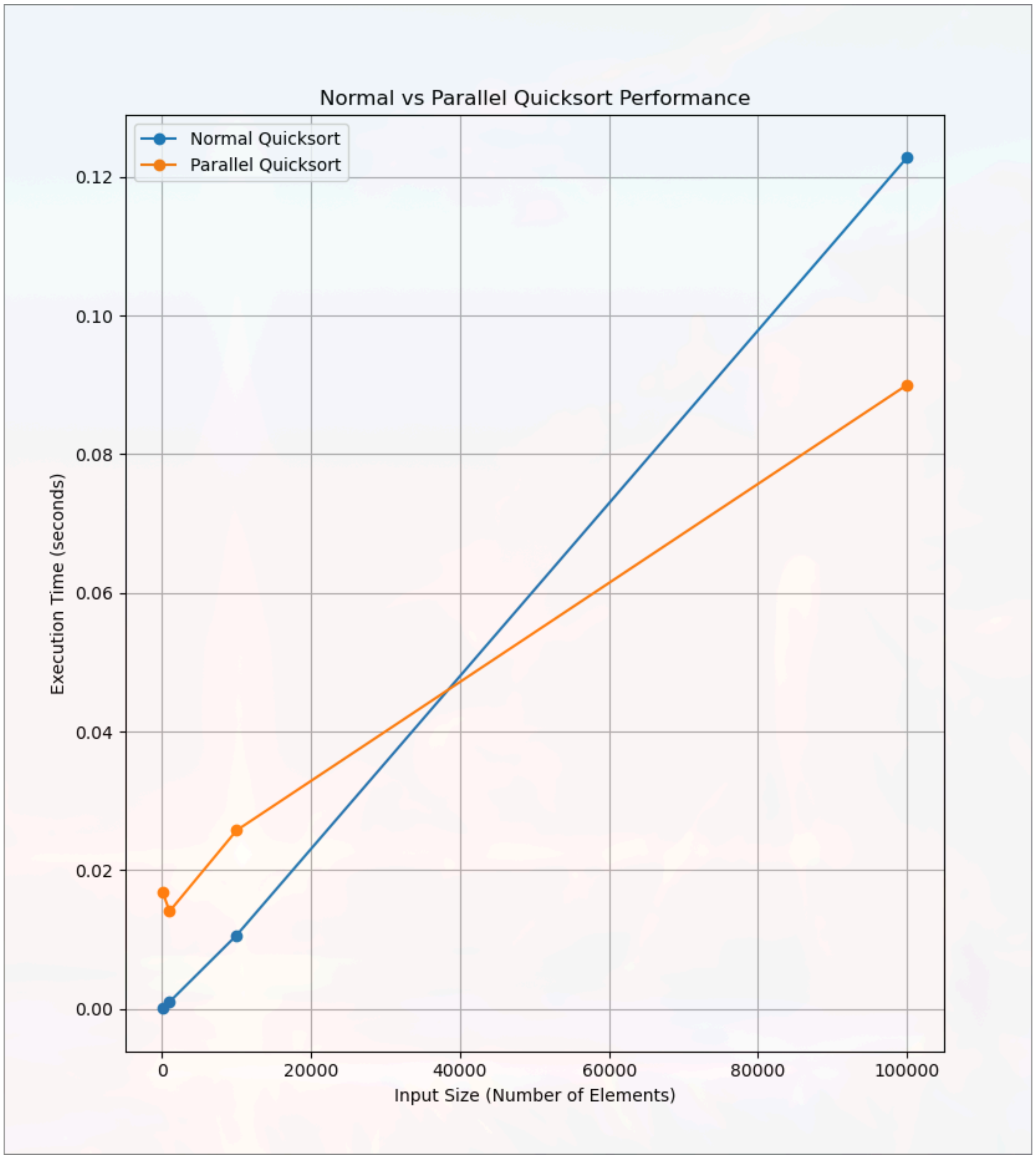
This shows that the Las-Vegas quicksort approach takes longer than traditional method, but this depends on the input, and its type. It increases the probablitity that the worst case won't occur.

4. Quicksort: Normal vs Parallel

This section compares the performance of the standard Quicksort algorithm with its parallelized version.

Command to run: python -m lab3.parallel

Output

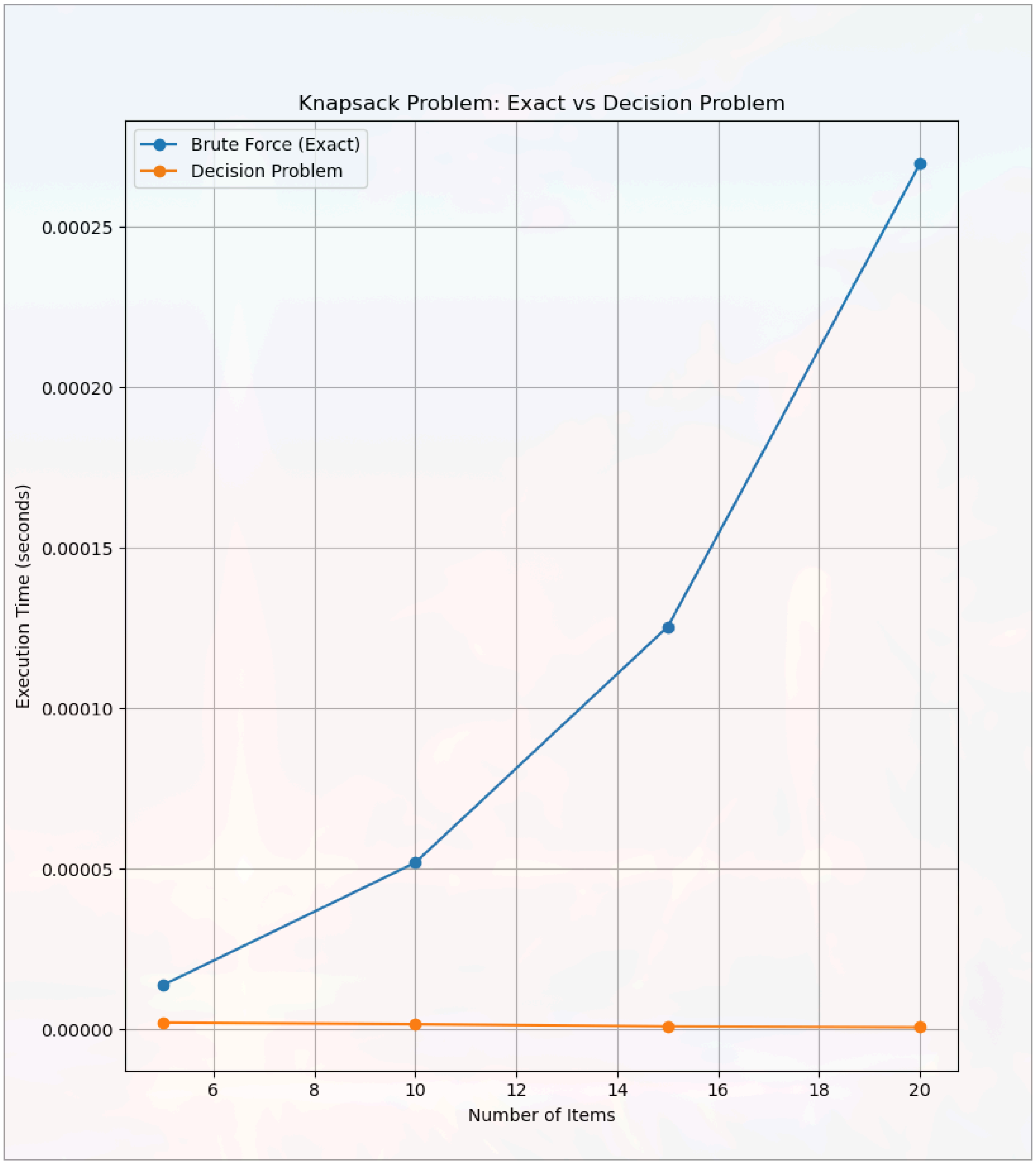


This shows that even though parallel quicksort might be slower than normal quicksort for very few data, in the long run, parallel quick sort occurs faster because of the task division and parallel execution.

4. Knapsack Problem

This section shows that non-polytime nature of Knapsack problem, and the polytime completion of its decision problem, denoting it as an NP-Complete problem.

Command to run: python -m lab3.knapsack



Here we can see that the Knapsack problem (bruteforce) is taking significantly longer for even a small data, and decision problem's execution time is negligible.

Conclusion

This document highlights the differences in implementation, performance, and use cases for various algorithms. The choice of algorithm often depends on the specific requirements, constraints, and data size.