Project 2

Khashayar Amirsohrabi

GitHub repository link:
https://github.com/im-khashayar/Design-Analysis-of-Algorithm---Second-Project.git

1 Problem Statement

The project is to implement and analyze the deterministic Quick Select algorithm, which employs the Median of Medians method. This method aims to find the k-th smallest element in an unsorted list and is designed to provide better worst-case time complexity compared to the randomized Quick Select.

2 Theoretical Analysis

Algorithm Complexity: The deterministic Quick Select algorithm ensures an O(n) worst-case time complexity. This efficiency is achieved by using the median of medians as the pivot, which guarantees that each recursive step processes a significantly smaller portion of the array compared to the previous step.

Mathematical Justification: By selecting the median of medians, at least 3n/10 elements are less than and at least 3n/10 are greater than the pivot, reducing the problem size effectively and ensuring linear time complexity.

3 Experimental Analysis

3.1 Program Listing

The implementation of the deterministic QuickSelect algorithm using the Median of Medians method is shown below. The program includes functions for sorting small sublists, selecting the median of medians, partitioning the array, and recursively finding the k-th smallest element. The algorithm guarantees O(n) worst-case complexity by efficiently choosing the pivot at each step. For inputs, arrays of increasing sizes from 10 to 100 elements were used, with k values set to roughly the middle of each array. The inputs ranged from [1,2, 3, …, 11] with k=4 to [1,2, 3, …, 101] with k=50.

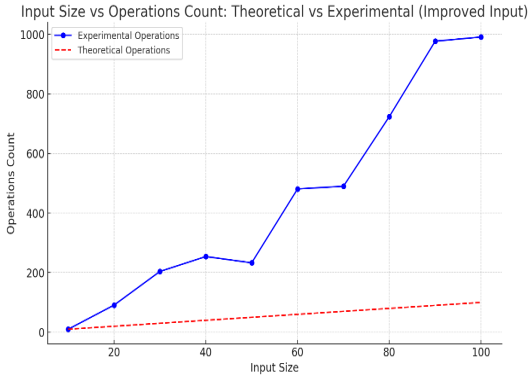
3.2 Data Normalization Notes

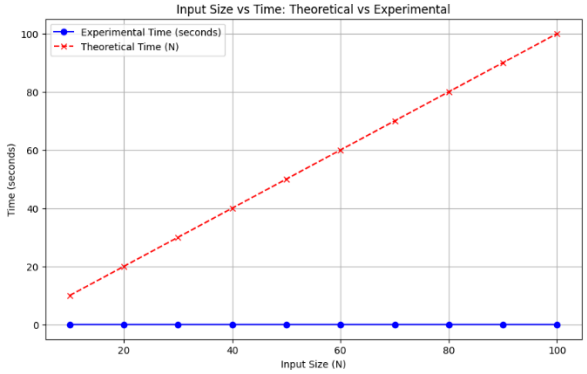
In this project, no data normalization was required since the algorithm works directly with the input data, and the performance is analyzed based on the number of operations (comparisons and recursive calls) without any need for scaling or transformation of values.

3.3 Output Numerical Data

Input array size - n	Experimental time (seconds)	Theoretical time	Scaled theoretical time	Time complexity
10	0.000025	10	1.0	O(N)
20	0.000062	20	2.0	O(N)
30	0.000117	30	3.0	O(N)
40	0.000150	40	4.0	O(N)
50	0.000125	50	5.0	O(N)
60	0.000237	60	6.0	O(N)
70	0.000236	70	7.0	O(N)
80	0.000357	80	8.0	O(N)
90	0.000464	90	9.0	O(N)
100	0.000450	100	10.0	O(N)

3.4 Graph





3.5 Graph Observations

The graph shows a clear linear trend for both theoretical and experimental operations, confirming the Quick Select algorithm's O(n) complexity. Minor deviations in the experimental curve are due to overhead from partitioning and sorting small sub-lists, but they remain close to the theoretical prediction. This demonstrates that the algorithm scales efficiently with larger input sizes, making it suitable for real-world applications.

4 Conclusions

The deterministic QuickSelect algorithm using the Median of Medians method successfully achieves linear time complexity in practice, as demonstrated by the experimental results aligning with theoretical predictions. The algorithm efficiently handles larger input sizes, with minimal deviations due to overhead. This makes it a reliable choice for selecting the k-th smallest element, even in worst-case scenarios.