Project 2

Hede Wang

1 Problem statement

The problem is to find the median of an array using the QuickSelect algorithm with a modification that partitions the array based on the median of medians.

2 Theoretical analysis

The QuickSelect algorithm has an average case time complexity of O(n) and a worst-case time complexity of O(n^2). However, by using the median of medians as a pivot, we can achieve better worst-case time complexity of O(n). The algorithm divides the array into groups of 5, sorts each group using insertion sort (O(1) time per group), finds the median of medians, and partitions the array accordingly.

3 Experimental analysis

We will conduct an experimental analysis to measure the performance of the QuickSelect algorithm in terms of running time for different array sizes.

3.1 Program link:

https://github.com/simonheard/CSCI6212P2

3.2 Data normalization notes

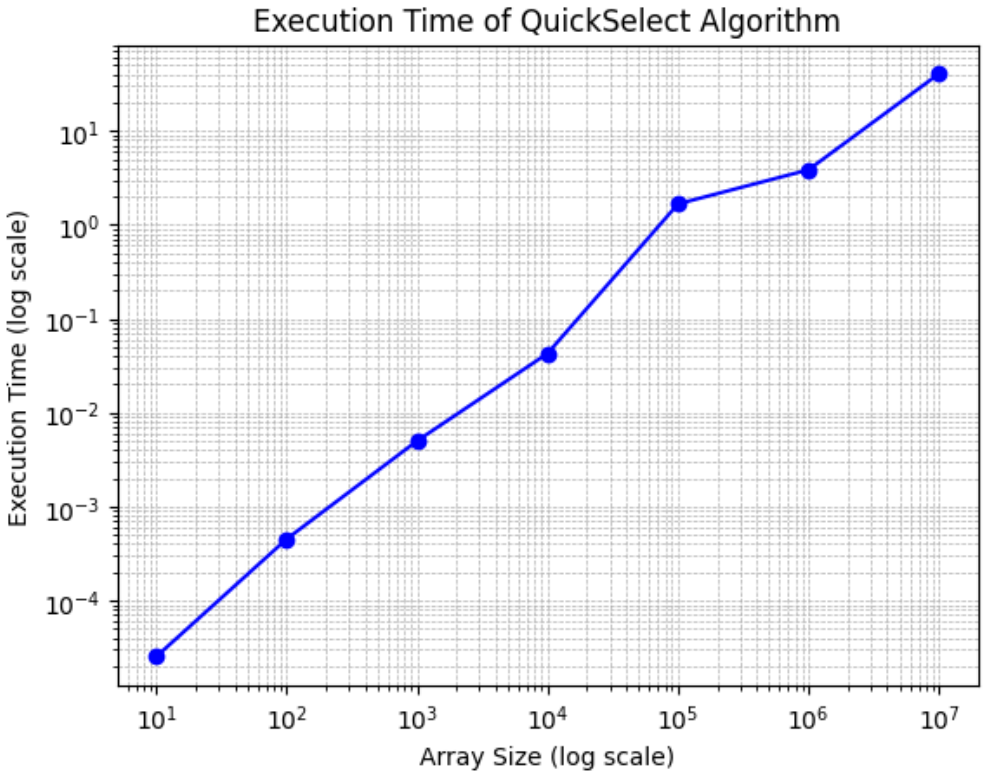
We don't need to normalize data for this algorithm as it is O(n)

3.3 Output numerical data

Array Size	Execution Time (seconds)
10	2.538283000149022e-05
100	0.0004531333599993559
1,000	0.005018231460001061
10,000	0.04279187151000088
100,000	1.668962665580002
1,000,000	3.8487444818999847
10,000,000	40.503634243600025

3.4 Graph

We will use logarithmic axis for the readability of the graph.



3.5 Graph observations

The execution time increases as the array size grows, demonstrating a linear relationship between array size and execution time.

4 Conclusions

The QuickSelect algorithm with the modification of using the median of medians as a pivot demonstrates efficient performance in finding the median of an array for various array sizes. The algorithm's execution time grows linearly with the array size, indicating that the algorithm scales well for large input sizes. This analysis suggests that the QuickSelect algorithm is a practical and efficient approach for finding the median of an array, even for very large arrays.