# Medians and Order Statistics & Elementary Data Structures

Name

Institution Affiliation

# Medians and Order Statistics & Elementary Data Structures

Part 1: Selection Algorithm Performance Analysis

The two algorithms the Median of Medians and Quickselect algorithms were developed and tested on arrays of different sizes to understand their performance. The following is the output we got after running the model

=== Selection Algorithm Performance Analysis ===

Array Size: 100 | Median of Medians: 0.001006s | Quickselect: 0.000000s

Array Size: 500 | Median of Medians: 0.002997s | Quickselect: 0.000981s

Array Size: 1000 | Median of Medians: 0.000000s | Quickselect: 0.000000s

Array Size: 5000 | Median of Medians: 0.015604s | Quickselect: 0.015626s

=== Data Structure Operations Analysis ===

Array[2]: 100 (Expected: 100)

Stack Top: 20 (Expected: 20)

Stack Pop: 20 (Expected: 20)

Queue Front: 5 (Expected: 5)

Queue Dequeue: 5 (Expected: 5)

Linked List (Before Deletion): 1 -> 2 -> 3 -> None

Linked List (After Deletion): 1 -> 3 -> None

According to the results above, the quickselect l algorithm runs much faster most of the time. The results also indicate that in some instances, the model executed in a negligible time of 0 seconds which means that it is very fast. On the other hand, the median of medians provides more stable runtime even though it is slower than quickselect because it is a deterministic algorithm.

The size of the array had a significant impact on how the two models performed. For example, the quickselect model was still the most efficient in terms of execution time even when the array size increased. However, when the size of the arrays was increased to about 5,000 the two algorithms had almost equal execution time. The results above alliance with the theoretical expectations where Quickselect performs well at O(n) expected time. On the other hand, the Median of Medians is better in terms of guaranteeing worst-case O(n) complexity even though it has a higher overhead.

Part 2: Data Structure Operations Analysis

The operations on Arrays, Stack, Queue, and Linked List all performed as expected. Insert and access auto operations were executed as expected. The model was also able to perform push, pop and peek and maintained last in first out LIFO behavior. Queue operations like queue, dequeue, and front followed First-In-First-Out behavior as expected according to the results above. Linked List operations which include insert, delete, traverse were able to manipulate the nodes correctly including deleting operations.

Conclusion

This results shows the expected time complexity O(1) for stack/queue operations and O(n) for linked list deletion in the worst case. The results confirm that Quickselect is faster in practice, while Median of Medians on the other hand is more stable and reliable even though at the cost of speed.