MEDIANS AND ORDER STATISTICS & ELEMENTARY DATA STRUCTURES

Assignment 6

AUGUST 2, 2025

BISHNU SHARMA
ID: 005036007

Part 1: Selection Algorithms

This part of the assignment involved implementing two algorithms to find the k^{th} smallest element in an array.

1. Randomized Quickselect

Quickselect is a randomized algorithm that chooses a pivot randomly and partitions the array recursively. It has excellent average-case performance and is very fast in practice. However, it suffers from a worst-case time complexity of $O(n^2)$ if unlucky pivot choices occur.

- Best/Average Case Time Complexity: O(n)
- Worst Case Time Complexity: $O(n^2)$
- Space Complexity: O(1) (in-place, ignoring recursion)
- Use Case: Efficient when average-case performance is acceptable (e.g., analytics)

2. Deterministic Median of Medians

This method divides the input into groups of 5, finds the median of each, and recursively selects the median of those medians as the pivot. It guarantees worst-case linear time, which makes it useful in time-sensitive or mission-critical applications.

- Worst Case Time Complexity: O(n)
- Space Complexity: O(n) (due to recursion and lists)
- Use Case: Real-time systems or algorithms where predictable performance is critical

Empirical Observation

Both algorithms were tested on the input list [12, 3, 5, 7, 4, 19, 26] to find the 3rd smallest element. Both correctly returned 5, showing that despite different approaches, they arrived at the same result.

Part 2: Elementary Data Structures

This section focused on building and analyzing core data structures from scratch using Python.

Structures Implemented:

- Arrays – with insert, delete, and access
- Stacks – using arrays for LIFO operations
- Queues – using arrays for FIFO processing
- Singly Linked Lists – insert, delete, traverse
- Rooted Trees – node-based hierarchy with multiple children per node

Real-World Use Cases:

- Arrays: For static data with fast index-based access
- Stacks: Undo operations, expression evaluation, recursion tracking
- Queues: Scheduling tasks, managing customer service systems
- Linked Lists: When dynamic memory and frequent insert/delete operations are needed
- Rooted Trees: File system structures, organization charts, syntax trees in compilers

Comparison Summary

Structure	Insert	Delete	Access	Best Use Case
Array	O(1)*	O(n)	O(1)	Fast lookup, Static data
Stack(Array)	O(1)	O(1)	O(1)	Undo, function calls
Queue(Array)	O(1)	O(n)	O(1)	Print queue, customer line
Linked List	O(n)	O(n)	O(n)	Frequent insert/delete
Rooted Tree	O(1)*	O(1)*	O(n)	Hierarchical data (file system)

* Assuming we keep references to the nodes

Conclusion

This assignment provided both practical and theoretical experience with key algorithmic and data structure concepts. I implemented and tested two powerful selection algorithms and several foundational data structures entirely from scratch.

Throughout the project, I also referred to trusted educational resources like GeeksforGeeks to confirm the logic and complexity of certain algorithms, especially for the Median of Medians method.

Overall, this assignment helped me understand time-space trade-offs and when to use each structure in real-world applications.

References

GeeksforGeeks. (n.d.). *Median of Medians Algorithm*.

Retrieved from https://www.geeksforgeeks.org/median-of-medians-algorithm/

GeeksforGeeks. (n.d.). *Quickselect Algorithm*.

Retrieved from https://www.geeksforgeeks.org/quickselect-algorithm/