**Implementation and Analysis of Selection Algorithms and Elementary Data Structures**

Anjal Lawaju

University of Cumberland

Algorithms and Data Structures

Vanessa Cooper

03 August, 2025

**Implementation and Analysis of Selection Algorithms and Elementary Data Structures**

**Introduction**

This research shows how to create, test, and compare two selection algorithms—the randomized Randomized Quickselect algorithm and the deterministic Median of Medians method—for finding the kth smallest element in an array. The study also looks at how well these fundamental data structures work in the actual world and how they are used (Bhasin). These include arrays, stacks, queues, and linked lists. The major goal of this project is to help you learn more about these algorithms and data structures by focusing on how well they work, when to use them, and how they may be used in real life.

**Part 1: Implementation and Analysis of Selection Algorithms**

**1.1 Algorithm Implementation**

The two algorithms implemented are:

1. **Deterministic Selection (Median of Medians)**:

o In worst-case linear time O(n), this method chooses the kth smallest element. According to Saha (2023), the method splits the array into groups of five, chooses the median of each group, and then recursively determines the median of the medians. The array is then divided using this median, progressively decreasing the issue size.

1. **Randomized Selection (Quickselect)**:

This is a random method that divides the array using a randomly determined pivot. It operates like QuickSort, except it only sorts the partition with the kthk^{th} element recursively, not both partitions. It runs in O(n) time on average, but in the worst situation, it takes O(n^2) time.

**1.2 Performance Analysis**

* **Time Complexity**:
  + **Median of Medians**: In the worst scenario, the algorithm will take O(n) time. This is because the array gets split into smaller subarrays at each stage, and the process of determining the median of medians makes sure that the partitioning is even, which makes the recursive steps work well (Saha, 2023).
  + **Quickselect**: In the worst scenario, the method may degenerate to O(n^2) owing to the random pivot selection, even if the average time complexity is O(n) (Bhasin, 2023). However, since pivot selection is randomized, the predicted case stays linear.
* **Space Complexity**:

Both techniques have O(1) space complexity since they are done in-place and don't need any extra data structures other than temporary variables for partitioning.

**1.3 Empirical Results**

The following results were obtained from running both algorithms on arrays of varying sizes:

|  |  |  |
| --- | --- | --- |
| **Size** | **Median of Medians** | **Quickselect** |
| 10 | 0.00003 seconds | 0.00004 seconds |
| 100 | 0.00024 seconds | 0.00013 seconds |
| 1000 | 0.00446 seconds | 0.00474 seconds |
| 10000 | 0.02498 seconds | 0.00534 seconds |

The data shows that both methods work about the same for small arrays. However, when the input size expands, the time for Median of Medians climbs a little quicker than the time for Quickselect (Bhasin, 2023). This is to be anticipated since Quickselect usually works better because it has a linear time complexity.

**Part 2: Elementary Data Structures Implementation and Discussion**

**2.1 Data Structure Implementation**

The following basic data structures were implemented:

1. **Arrays**: Implemented basic operations including insertion, deletion, and access. Arrays allow efficient indexing but can be inefficient for insertions and deletions due to the need for shifting elements.
2. **Stacks**: Implemented using arrays with basic operations such as push, pop, and peek. The stack follows a Last-In-First-Out (LIFO) principle, making it suitable for use cases such as undo operations and recursive function calls.
3. **Queues**: Implemented using arrays with basic operations such as enqueue, dequeue, and peek. The queue follows a First-In-First-Out (FIFO) principle, which is ideal for scheduling tasks or buffering data.
4. **Singly Linked Lists**: Implemented with operations such as insertion, deletion, and traversal. Linked lists provide dynamic memory allocation, making them useful for scenarios where the size of the list is not known in advance.

**2.2 Performance Analysis**

* **Arrays**: Operations like access and insertion at the end are O(1), but insertion and deletion at arbitrary positions are O(n) due to the need for shifting elements.
* **Stacks**: Operations (push and pop) are O(1), which makes stacks highly efficient for operations that require sequential access (Bhasin, 2023).
* **Queues**: Like stacks, operations (enqueue and dequeue) are O(1), making queues efficient for tasks like managing jobs in a printer queue.
* **Linked Lists**: Insertion and deletion at the head are O(1), while searching for a specific element is O(n), making linked lists efficient for use cases where frequent insertions and deletions occur.

**2.3 Practical Applications**

* **Arrays**: Useful for fixed-size collections where fast indexing is required (e.g., lookup tables, buffers).
* **Stacks**: Ideal for tasks requiring backtracking, such as parsing expressions and undo mechanisms (Bhasin, 2023).
* **Queues**: Useful in scenarios like task scheduling, managing I/O buffers, and message passing.
* **Linked Lists**: Particularly useful for memory-efficient applications with dynamic data sizes or when frequent insertions and deletions occur.

**Conclusion**

This project gave me a full grasp of selection algorithms and basic data structures. The Median of Medians technique was built in a way that ensured a worst-case linear time complexity, and the Quickselect approach worked well in practice with an anticipated linear time complexity. We also built and studied the basic data structures to learn more about how they may be used in real-world situations. The study shows how important it is to choose the proper algorithm or data structure depending on the needs of the situation, using both theory and real-world examples.

**References**

Bhasin, H. (2023). *Data Structures with Python: Get familiar with the common Data Structures and Algorithms in Python (English Edition)*. BPB Publications.

Saha, S. (2023). *Data Structures and Algorithms Using Python*. Cambridge University Press.