**Bikash Shrestha**

**ID: 005032645**

**MSCS-532-A01**

**Assignment 6**

**Medians and Order Statistics & Elementary Data Structures**

**Introduction**:

Medians and order statics are a significant part of solving complex computational problems. Here, I have made a detailed report to explain the implementation of selection algorithms and elementary data structures and analyze their performance and practical applications.

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

**Implementation:**

**1. Deterministic Algorithm: Median of Medians**

The worst-case time complexity of the Median of Medians Algorithm is O(n). It divides the data or array into five groups, sorts them, and finds the pivot (median of medians), which is then used to partition the array for recursion.

The code implementation is shown in the given code snippet:

A screen shot of a computer program

Description automatically generated

**2. Randomized Algorithm: Randomized Quickselect**

Quicksort is the main base of this algorithm. However, it recurses on only one side of the pivot, which makes it faster in practice. This algorithm selects a random pivot, and the array is partitioned. Based on the pivot index, the algorithm determines whether to recurse the left or right of the array.

Below is the code implementation:

A screen shot of a computer program

Description automatically generated

**Performance Analysis:**

**1. Time Complexity**

**Median of Medians:** As mentioned above, in the worst case, this algorithm divides the array into five groups, finds the medians, and processes the groups recursively. So, each recursive call has a time complexity of O(n), and the overall time complexity also becomes the same: O (n).

**Randomized Quickselect:** In this algorithm, only half of the array is processed recursively, either the left or right of the pivot, making its time complexity O(n). In the worst-case scenario, when the pivots are chosen consistently, the time complexity will be O(n^2).

**2. Space Complexity**

In both algorithms, partitions or sub-arrays are stored in the additional space. So, they both have the same space complexity: O(n).

**Empirical Analysis:**

**Part 2: Elementary Data Structures:**

**Arrays**: It is a fundamental data structure storing elements in a contiguous memory location. It implements three operations: Insertion, Deletion, and Access.

**Stacks**: Stacks are also implemented using the arrays. It operates on the Last In, First Out (LIFO) principle. It implements push and pop operations.

**Queues**: This linear data structure uses the First In, First Out (FIFO) principle. It implements enqueue and dequeue operations.

**Linked Lists:** It is a data structure that consists of nodes. Each node contains data and a reference to the next node. It implements: Insertion, Deletion, Traversal

**Performance Analysis:**

The performance analysis of each elementary data structure is given below:

1. **Arrays:**

|  |  |  |
| --- | --- | --- |
| **Operations** | **Time Complexity End** | **Time Complexity Middle** |
| **Insertion** | O(1) | O(n) |
| **Deletion** | O(1) | O(n) |
| **Access** | O(1) | O(n) |

1. **Stacks**:

|  |  |
| --- | --- |
| **Operations** | **Time Complexity** |
| **Push** | O(1) |
| **Pop** | O(1) |
| **Peek** | O(1) |

1. **Queue:**

|  |  |
| --- | --- |
| **Operations** | **Time Complexity** |
| **Enqueue** | O(1) |
| **Dequeue** | O(1) |
| **Peek** | O(1) |

1. **Linked List:**

|  |  |  |
| --- | --- | --- |
| **Operations** | **Time Complexity Beginning** | **Time Complexity Middle/End** |
| **Insertion** | O(1) | O(n) |
| **Deletion** | O(1) | O(n) |
| **Traversal** | O(1) | O(n) |

**Practical Applications:**

All those elementary data structures have their own applications.

* Arrays are used in databases, cache storage, image processing, etc.
* Stacks are used in recursive function calls, as well as in undo-redo mechanisms.
* Queues are the fundamental data structures for request handling, buffering, and scheduling algorithms.
* Linked lists are used for navigation systems and dynamic memory allocations.

**Conclusion:**

As per the problem and requirements, the choice of data structure could vary. For example, arrays are chosen for fast access, queues for request handling, and so on. Choosing the correct data structures and algorithms boost the performance significantly.