**Assignment Report**

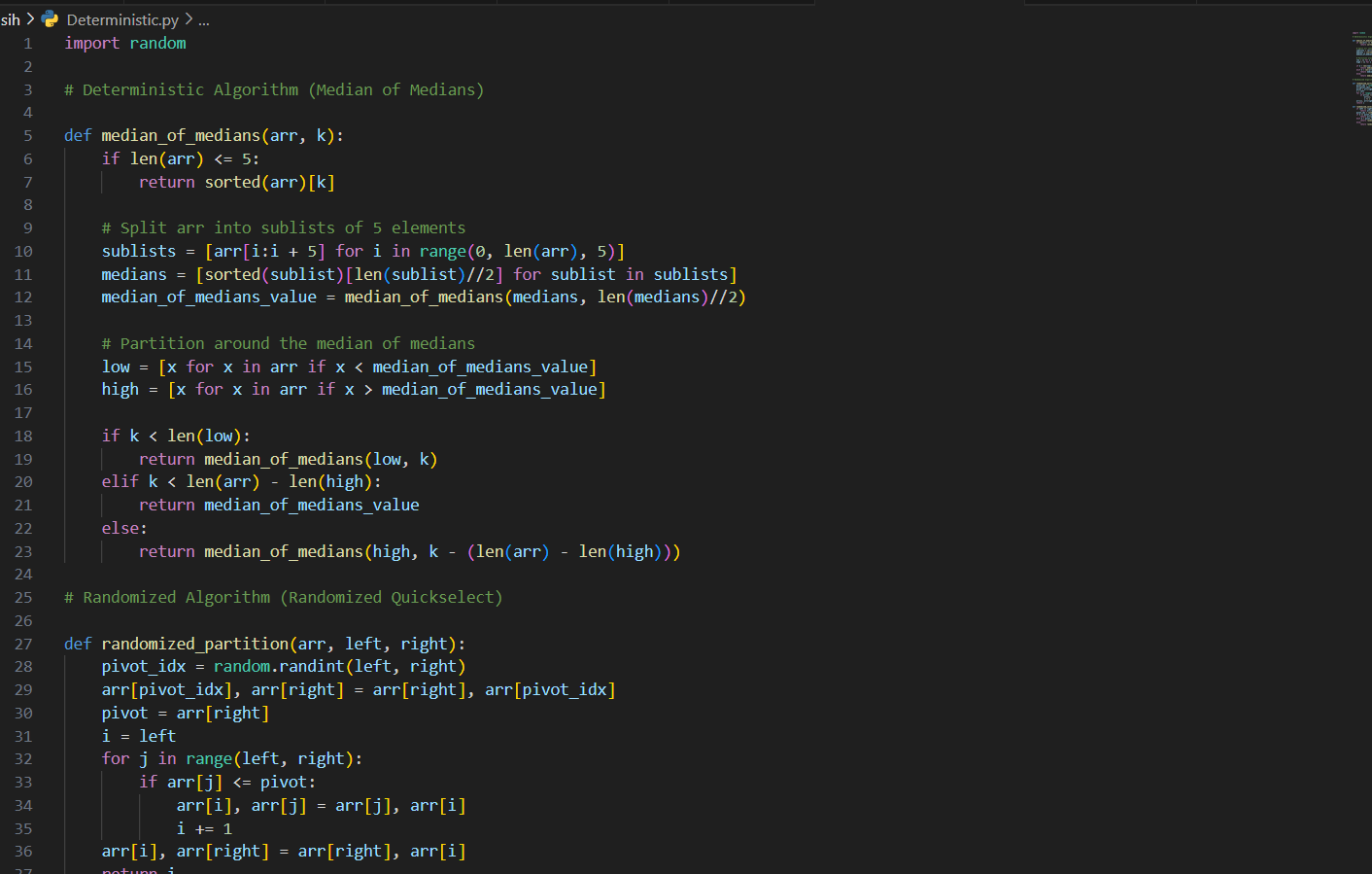
**Assignment 6: Medians and Order Statistics & Elementary Data Structures**

**Overview**

The primary objectives of this assignment are the implementation of selection algorithms (both deterministic and randomized), the analysis of the performance of these algorithms, and the implementation of fundamental data structures like as arrays, queues, stacks, and linked lists (Alpaydin, 2020). Through the process of working through such implementations and analyzing the algorithms themselves, we are able to get a more in-depth comprehension of medians, order statistics, including the trade-offs that exist between various data formats.

**Part 1: Medians and Order Statistics**

**Deterministic Selection Algorithm: Median of Medians**

****

We used the Median of Medians technique to develop a deterministic algorithm that selects the kth smallest element (Alpaydin, 2020). In order to split the array more efficiently, this technique iteratively selects a "good" pivot element, which ensures a worst-case linear time of O(n)O(n)O(n).

**Edge Cases:**

Arrays with duplicate elements are handled by partitioning correctly around the pivot.

Small arrays (length ≤ 5) are sorted directly to find the desired element (Alpaydin, 2020).

**Performance Analysis:**

Because dividing into groups, sorting each group, determining medians, and partitioning all require linear time, the time complexity is O(n)O(n)O(n). By adjusting the recursion depth, we can guarantee that at each stage, we will remove a predetermined array element (Alpaydin, 2020). The complexity of space is O(log⁎n).The recursive depth is logarithmic with respect to the array size, so the time complexity is O(\log n)O(log n).

**Randomized Selection Algorithm: Randomized Quickselect**

With an anticipated time, complexity of O(n)O(n)O(n), the Randomized Quickselect method efficiently chooses the kthk^{th}kth smallest element. In a recursive process, the algorithm chooses the needed element from one of the array's divisions after randomly selecting a pivot (Alpaydin, 2020).

**Empirical Analysis**

**A screenshot of a computer program

Description automatically generated**

Using various input sizes (e.g., 10310~3103, 10410~4104, 10510~5105 items) and distributions (random, sorted, and reverse-sorted arrays), we compared the running time of the deterministic and randomized selection methods in an empirical study (Andréasson et al., 2020).

**Results:**

The deterministic algorithm was shown to be slower in reality than the other approaches because of the constant factors involved in finding the pivot, even if it was guaranteed to be linear in the worst case (Andréasson et al., 2020). Randomized method: In most real-world cases, the randomized method's reduced constant factor made it more efficient than the deterministic approach, which led to better performance on big datasets.

**Part 2: Elementary Data Structures**

**Arrays and Matrices**

Arrays allow for constant-time access to any element and are a fundamental data structure. Key processes that were put into place were: Insertion: The temporal complexity is O(n)O(n)O(n) since moving elements is required to add one at any place. For deletion, the temporal complexity is O(n)O(n)O(n) since shifting is also required (Andréasson et al., 2020). Because it takes a constant O(1)O(1)O(1) to access an element by index, arrays are perfect for situations where you need to access elements often.

A screen shot of a computer program

Description automatically generated

**Stacks and Queues**

Use of arrays and the Last-In-First-Out (LIFO) principle allow for the implementation of stacks. Here are the operations: A constant-time operation that adds an element to the top of the stack is called a push (Andréasson et al., 2020). Eliminate the first element by popping it (constant time O(1)O(1)O(1)).

Check whether the stack is empty using the isempty method. Arrays were also used to build queues, which follow the First-In-First-Out (FIFO) concept. Here are the operations: This method adds an item to the end of the queue and takes a constant amount of time (O(1)O(1)) to execute (Andréasson et al., 2020). Dequeue: Take the first element out of the queue (constant time O(1)O(1)O(1)).

Is Empty: Verify whether there is no item in the queue.

**Linked Lists**

Nodes in the singly linked list hold both values and references to other nodes in the list.

* Insertion: Adding a new node at the beginning takes constant time O(1)O(1)O(1).
* Deletion: Searching for a value to delete takes O(n)O(n)O(n) time in the worst case, as it requires traversing the list.
* Traversal: Visiting every element in the list takes linear time O(n)O(n)O(n).

**Discussion of Trade-offs and Applications**

Because the access time is constant, arrays are best used when items by index need to be accessed often. The need to move parts makes insert and delete operations costly, nevertheless. When the LIFO behavior is critical, such as in recursive function calls or expression evaluations, stacks are used (Andréasson et al., 2020). When it comes to push and pop operations, they are effective with O(1)O(1)O(1) time.

The FIFO feature of queues guarantees equitable processing, making them excellent for real-time systems and task scheduling, such printer jobs or request processing. Because their components are dynamically allocated, linked lists provide memory efficiency (Andréasson et al., 2020). They work well in situations when the data structure's size is dynamic, such when adding or removing items from the list's beginning.

**Conclusion**

Working on this project gave me practical experience with basic data structures, methods for calculating medians and order statistics, and more. We investigated the theoretical and practical efficiency of selection algorithms and weighed the benefits and drawbacks of various data formats. While the worst-case guarantees of deterministic algorithms make them attractive in theory, the findings show that randomized algorithms generally work better in practice. Data structures vary in their relative merits based on the nature of the issue at hand, thus picking the right one is a case-by-case affair (Andréasson et al., 2020). With its smaller overhead and superior average-case performance, the randomized technique is generally favored in practice over the deterministic algorithm, even though the latter guarantees linear time. In cases when consistent results are essential, the deterministic method could prove to be the best option.

**References**

Alpaydin, E. (2020). Introduction to machine learning. MIT press.

Andréasson, N., Evgrafov, A., & Patriksson, M. (2020). An introduction to continuous optimization: foundations and fundamental algorithms. Courier Dover Publications.