# Assignment 6 Part 1: Implementation of Selection Algorithms

This work aims to use two well-known methods for choosing the kth-smallest element in an array, hence referred to as "order statistics". The work consists on using a deterministic and a random selection technique respectively. Whereas the randomized technique is predicted to execute in linear time on average but may deteriorate to quadratic time in the worst case, the deterministic algorithm guarantees worst-case linear time performance. This response clarifies the issue, the used techniques, their performance and empirical investigation.

**Deterministic Algorithm**

Based on the Median of Medians method, which ensures O(n) worst-case time complexity for choosing the kth-smallest element, the deterministic algorithm used in this project assures The method divides the array into sublists of at most five items, computes the median of each sublist, and then use a recursive technique to identify the median of these medians. The array is split into items less than, equal to, or greater than the median using it as a pivot. The method looks in the suitable partition recursively depending on the value of k to identify the kth-smallest element. This method has the benefit in that, even in the worst-case situations, its constant and predictable performance. The method starts by seeing if the input array's length falls either less than five or equal five. If so, it just arranges the array and produces the kth-smallest element. Otherwise the array is divided into five-element sublists from which the median is calculated. The method then divides the array based on the median of the medians, choosing the pivot in turn recursively. At last, it calls recursively on the partition housing the kth element.

A screenshot of a computer program

Description automatically generated

**Randomized Algorithm**

Usually running in linear time, the well-known Quickselect method used is randomized selection algorithm with worst-case time complexity of O(n2). Under this method, the array is split into items less than, equal to, or larger than the random pivot chosen. Recursively searches in the suitable partition depending on the value of k. Though it lacks worst-case performance guarantees, the simplified and smaller constant factors of the randomized method make it frequently quicker in reality than the deterministic version. Random selection of a pivot from the array, splitting the array depending on the pivot, and thereafter recursively invoking the function on the suitable partition define the Quickselect code. The method examines the smaller partition if the kth smallest element occurs there; otherwise, it searches the bigger partition. The method produces the pivot if the kth smallest element equals the pivot.

A screenshot of a computer program

Description automatically generated

# Performance Analysis

In the worst situation the deterministic Median of Medians method has O(n) time complexity. This is so because the method ensures that the pivot splits the array into properly balanced segments, hence guiding the operation of the recursive calls in every step. The need to save sublists and perform recursive calls determines O(n) the space complexity. Although obtaining medians and partitions involves very substantial overhead, it guarantees constant linear-time performance.

A screenshot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

On the other hand, while the randomized Quickselect method has an estimated time complexity of O(n), if the random pivot often produces imbalanced partitions it may decline to O(n2). O(n) is also the space complexity as the technique calls for extra memory for partitioning and recursive calls. Though in theory Quickselect is quicker than the deterministic method, its randomized character makes it less predictable. When worst-case assurances are needed—that is, in real-time systems where performance cannot afford to deteriorate—the deterministic approach is recommended. When simplicity and average-case performance are more valued than worst-case assurances, as in non-critical applications or situations involving significant volumes where performance is anticipated to be near to linear, the randomized approach is recommended.

# Empirical Analysis

Running the algorithms on many kinds of input arrays—random, sorted, reverse-sorted—allows one to empirically evaluate both methods. Measuring the time needed by every method to identify the kth-smallest element in arrays of different diameters and distributions, the benchmarking tool Usually for random inputs specifically, the randomized Quickselect method ran quicker on average. In certain situations, including reverse-sorted arrays, the deterministic method did, however, provide more consistent performance. With its predicted linear time plainly shown, the Quickselect method proved outstanding performance for random arrays. Because it carefully chooses pivots to ensure linear time, the deterministic method takes somewhat more time than Quickselect for sorted arrays. On reverse-sorted arrays, where the random pivot can often provide inadequate partitions, the deterministic method was more stable than Quickselect. Theoretical temporal complexity analysis was validated by actual data. Though its performance was less predictable, particularly for structured input arrays such as sorted and reverse-sorted arrays, the randomized approach did well on average. With no appreciable performance loss in any input type, the deterministic method routinely ran in linear time.

A screenshot of a computer

Description automatically generated

On three kinds of input arrays—random, sorted, and reverse-sorted—the results of the benchmarking script provide insights into the performance of the Randomized Quickselect and Deterministic Median of Medians algorithms. Tasked with determining the 500th smallest element in a list of 1000 items, these algorithms ran and their runtimes were noted to evaluate efficiency. Beginning with the Randomized Quickselect method, it did very well across all three input kinds. The method proved efficient and successful for processing random inputs as it took around 0.00050 seconds to find the intended element in the random array. With a runtime of 0.00028 seconds, the performance became better yet on the sorted array, implying that sorted data could help to find the kth-smallest element faster. With 0.00047 seconds, the time was still quite fast even on the reverse-sorted array, suggesting that the average performance of the method stays strong independent of the array organization. This result shows how well the Quickselect method preserves predicted linear-time performance in typical scenarios; if badly picked pivots are routinely used, it may degenerate to O(n2).

On the other hand, the Deterministic Median of Medians method showed much slower but more consistent performance. It took 0.00093 seconds for the random array, a longer period than Quickselect could explain given the method of more intentional pivot selection in the algorithm. Although it guarantees a worst-case linear-time guarantee, this extra cost causes usually worse performance. Though quicker than the random array, the 0.00068 seconds needed for the sorted array still behind Quickselect's speed. Taking 0.00073 seconds, the deterministic method behaved on the reverse-sorted array exactly like the sorted array. The deterministic method's main advantages are its predictability and worst-case performance guarantee, which guarantees linear time even in cases when input structures can compromise the performance of other methods. These findings underline the main trade-offs between the two techniques. Particularly for random or sorted data, the Randomized Quickselect method is much quicker on average and a useful option if worst-case performance is less important. Its simplicity and quickness enable it to shine in average-case situations, so it is a sensible option in many real-world uses. The Deterministic Median of Medians method is more dependable, while slower, in settings where worst-case performance can result in significant inefficiencies because of its meticulous pivot selection procedure. It offers a linear-time guarantee that Quickselect cannot provide in the worst scenario and performs constantly across all input kinds.

In essence, the particular application needs determine which of the two algorithms to use even if both are efficient in their own right and provide the proper answer. Randomized Quickselect is the best choice if the main issue is average-case speed. Nevertheless, despite its much longer running time, the Deterministic Median of Medians method becomes a more appropriate alternative when consistency and worst-case performance guarantees are required. With Quickselect shining in typical situations and the deterministic technique providing consistent performance across all kinds of input data, the practical findings support the theoretical assumptions of both algorithms.

# Conclusion

This work shows the theoretical and practical sides of two significant selection methods, one deterministic and one randomized. When consistency is crucial, the Median of Medians method offers worst-case performance guarantees, so it is a trustworthy option. Though its simplicity makes the Randomized Quickselect method superior on average, it lacks worst-case assurances. With the randomized approach usually running quicker but being less predictable, the practical study confirmed the theoretical assumptions. Generally, the particular application needs determine the method of choosing; the deterministic algorithm provides dependability and the randomized approach provides quicker average performance.