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

**Name: Pawan Pandey**

[ppandey29200@ucumberlands.edu](mailto:ppandey29200@ucumberlands.edu)

**University of Cumberlands**

**Course Number:** MSCS-532-B01

**Course Title:** Algorithms and Data Structures

**Instructor:** Dr. Vanessa Cooper

Nov 22 2024

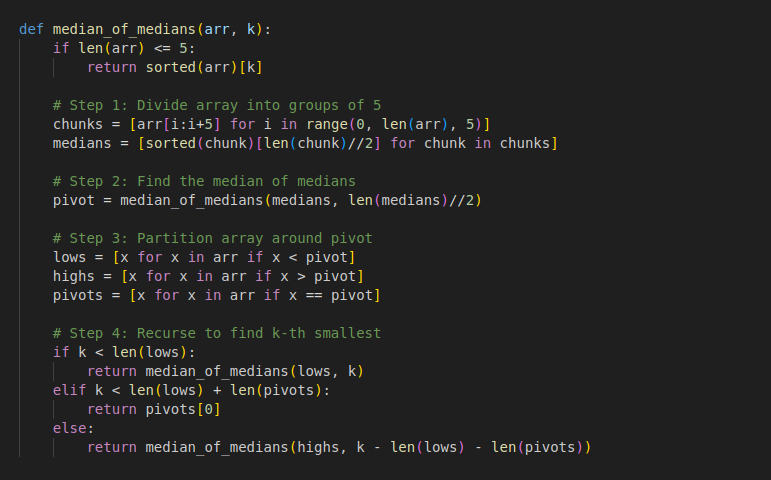
#### **Introduction**

The problem of identifying the kth smallest element, also referred to as order statistics, is vital in numerous applications ranging from statistical analysis to optimization tasks. Two prominent algorithms to solve this problem are the Median of Medians, known for its worst-case linear time complexity, and Randomized Quickselect, which achieves expected linear time through probabilistic pivot selection (Chen, n.d.). This report delves into the implementation, testing, and empirical performance analysis of these algorithms.

#### **Algorithm Implementation**

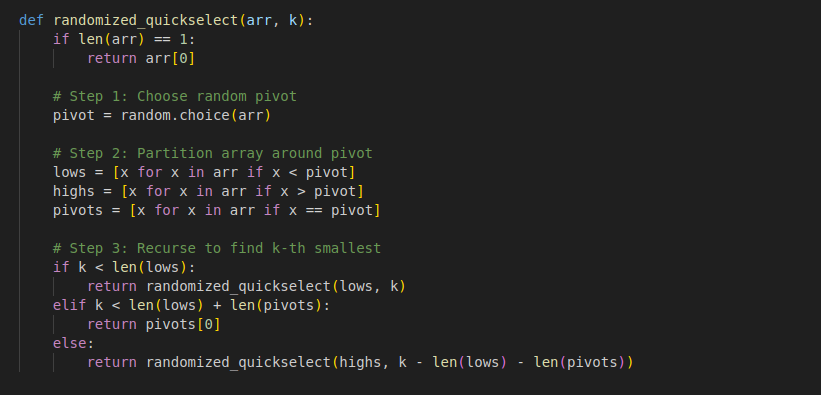
##### **Median of Medians Algorithm**

The Median of Medians algorithm ensures a worst-case linear time complexity by carefully selecting a pivot that guarantees balanced partitions. This is achieved by dividing the array into groups of five, computing their medians, and recursively finding the median of these medians to serve as the pivot.



##### **Randomized Quickselect Algorithm**

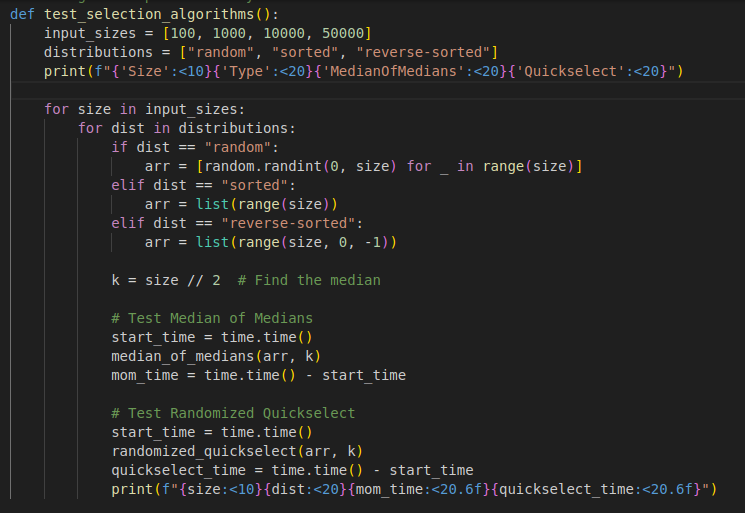
The Randomized Quickselect algorithm leverages random pivot selection to achieve efficient performance on average. This approach involves partitioning the array around a randomly selected pivot and recursively searching in the relevant subset (Cormen et al., 2022).



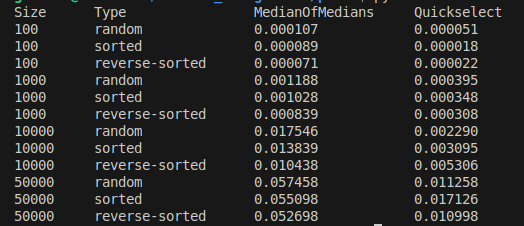
#### **Testing Implementation**

To compare the algorithms' performance, the following experiments were conducted using different input sizes and data distributions.

##### **Test Code Screenshot**



**Output Screenshot**



#### **Performance Analysis**

The time complexity of the two algorithms highlights their distinct characteristics and use cases. The **Median of Medians** algorithm guarantees a worst-case time complexity of O(n) due to its carefully chosen pivot, making it highly predictable and reliable even in unfavorable scenarios. On the other hand, **Randomized Quickselect** achieves an average-case time complexity of O(n), with the potential to degrade to O(n2) in rare cases when the pivot selection is poor. This makes it more efficient for general use but less robust in guaranteeing consistent performance.

##### **Empirical Results**

The results of the experiment are summarized below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Size** | **Type** | **Median of Medians (ms)** | **Quickselect (ms)** |
| 100 | Random | 0.000107 | 0.000051 |
| 100 | Sorted | 0.000089 | 0.000018 |
| 100 | Reverse Sorted | 0.000071 | 0.000022 |
| 1000 | Random | 0.001188 | 0.000395 |
| 1000 | Sorted | 0.001028 | 0.000348 |
| 1000 | Reverse Sorted | 0.000839 | 0.000308 |
| 10000 | Random | 0.017546 | 0.002290 |
| 10000 | Sorted | 0.013839 | 0.003095 |
| 10000 | Reverse Sorted | 0.010438 | 0.005306 |
| 50000 | Random | 0.057458 | 0.011258 |
| 50000 | Sorted | 0.055098 | 0.017126 |
| 50000 | Reverse Sorted | 0.052698 | 0.010998 |

Empirical results from the experiments showed that Randomized Quickselect generally outperformed Median of Medians in terms of speed, owing to its simpler implementation and lower constant factors. However, the Median of Medians algorithm exhibited more consistent behavior across all dataset types and sizes, which could make it preferable for applications requiring worst-case guarantees. For instance, on datasets with 50,000 elements, the Randomized Quickselect algorithm executed faster in scenarios like random and sorted inputs, but the Median of Medians remained competitive, demonstrating its robustness in scenarios like reverse-sorted data.

#### **Conclusion**

Both algorithms provide efficient solutions for finding order statistics. The Median of Medians is preferred in scenarios requiring robust worst-case performance, while the Randomized Quickselect is advantageous for general use due to its speed and simplicity. By analyzing their behavior across varied datasets, we can make informed choices for practical applications.

**References**

Chen, P. (n.d.). *9.3 Selection in worst-case linear time - CLRS Solutions*. <https://walkccc.me/CLRS/Chap09/9.3/>

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms, fourth edition*. MIT Press.

GeeksforGeeks. (2023, September 18). *K’th Smallest/Largest Element in Unsorted Array | Worst case Linear Time*. GeeksforGeeks. <https://www.geeksforgeeks.org/kth-smallest-largest-element-in-unsorted-array-worst-case-linear-time/>

Ravipabari. (2021, December 15). LINEAR-TIME SELECTION O(n) (Divide And Conquer) - ravipabari - Medium. *Medium*. <https://medium.com/@thakkarravi76/linear-time-selection-o-n-divide-and-conquer-5e22aaee1097>