Medians and Order Statistics & Elementary Data Structures

Algorithms and Data Structures (MSCS-532-B01)

Shreya Sapkota

005026644

11/24/2024

Introduction

Introduction

Median of Medians

This algorithm carefully chooses a pivot element to ensure O(n) time complexity in the worst scenario. The input array is divided into smaller groups, their medians are determined, and the pivot is recursively chosen from the median of these medians. By doing this, the number of recursive calls is reduced and balanced partitions are guaranteed. It is beneficial for real-time systems, including embedded systems or computational geometry problems when worst-case guarantees are crucial.

Quickselect

Quickselect partitions the array using a random pivot and chooses the partition that contains the kth smallest entry. Even while O(n) is its predicted time complexity, in the worst situation such as with very unbalanced partitions, it may decrease to O(n2). This is suitable for general-purpose applications like statistical sampling where average performance is more significant than guarantees.

Performance Analysis

Median of Medians

Time Complexity: The Median of Medians achieves O(n) by dividing the input into uniformed groups and balancing the partition size using a properly chosen pivot (Blum et al., 1973).

Space Complexity: The space complexity is O(logn) as recursion depth is proportional to logn, however there is extra overhead from creating and sorting subgroups.

Quickselect

Time Complexity: Quick select achieves O(n) performance by using randomness to balance partitions (Hoare, 1961). In the worst-case scenario, O(n2) arises when partitions are very unbalanced, such as with adversarial inputs.

Space Complexity: The depth of a recursive stack is equal to logn, similar to the Median of Medians which makes the space complexity O(logn).

Empirical Analysis

Size	Median of Medians (s)	Quickselect (s)
500	0.00022	0.000119
1000	0.000435	0.000236
5000	0.002259	0.000965
10000	0.003805	0.001397

Evaluation of runtime results reveals significant variations between the Median of Medians and Quickselect algorithms. For lesser input sizes (e.g., 500 or 1000 elements), the performance difference is small, with Quickselect marginally faster. However, as the input size increases, Quickselect becomes significantly more efficient, with runtimes that are 2-3 times faster than Median of Medians for arrays of 5000 or 10000 entries. Quickselect has less overhead in selecting random pivots than the Median of Medians, which requires deterministic grouping and median calculations.

In practice, both algorithms display linear growth consistent with their O(n) time complexity, but Median of Medians ensures a consistently balanced partition, making it reliable even in worst-case scenarios. In contrast, Quickselect uses probabilistic balance, which is quicker on average but can degrade to O(n2) in unusual cases, such as adversarially ordered inputs. These trade-offs make Median of Medians more stable and suitable for important systems that require predictable performance, but Quickselect is best suited for general-purpose activities where average-case efficiency is sufficient.

In conclusion, Quickselect is the recommended option for the majority of practical circumstances due to its quickness and simplicity. However, the Median of Medians is useful in contexts where worst-case performance guarantees are critical, such as real-time systems or applications that require robustness against skewed data distributions. The algorithm used is determined by the task's specific needs as well as the allowable trade-offs between speed and stability.

References

Blum, M., Floyd, R. W., Pratt, V. R., Rivest, R. L., & Tarjan, R. E. (1973). *Time bounds for selection*. Journal of Computer and System Sciences, 7(4), 448-461. https://doi.org/10.1016/S0022-0000(73)80033-9

Hoare, C. A. R. (1961). *Algorithm 65: Find*. Communications of the ACM, 4(7), 321–322. https://doi.org/10.1145/366622.366647