Implementation and Analysis of Selection Algorithms

Performance Analysis

Time Complexity

Deterministic Algorithm (Median of Medians)

The Median of Medians algorithm achieves O(n) time complexity in the worst case. This is because array is divided into groups of 5, which takes O(n). Finding the median of these groups takes O(n), since each group has a constant number of elements. Similarly, recursively selecting the median of medians happens on a smaller array, and the partitioning step is linear. Overall, the recurrence relation is $T(n)=T(n/5)+T(7n/10)+O(n)$, leading to a solution of O(n) in the worst case (Cormen et al., 2022).

Space Complexity: The space complexity is O(n) because of the additional storage needed for subsists and partitioned arrays and recursion depth is logarithmic.

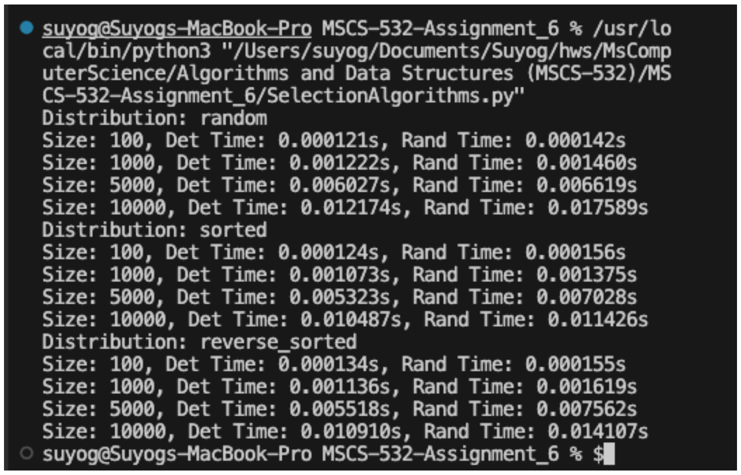
Randomized Algorithm (QuickSelect)

Randomized Quickselect algorithm has an expected time complexity of O(n). On average, the pivot divides the array into two equal halves, and recursive calls reduce the size of the problem logarithmically, similar to QuickSort. However, in the worst case, the complexity is $O(n^2)$ if the pivot is consistently the smallest or largest element.

Space Complexity: The space complexity is O(1) for the in-place partitioning, though the recursive calls incur a stack overhead of O(logn) in the average case.

Empirical Analysis

Functions for both the **deterministic selection** (Median of Medians) and **randomized selection** (Randomized Quickselect) are defined and timeit module is used to capture the execution time. Below is the screenshot of the result after executing the test.



```
● suyog@Suyogs-MacBook-Pro MSCS-532-Assignment_6 % /usr/lo
  cal/bin/python3 "/Users/suyog/Documents/Suyog/hws/MsComp
  uterScience/Algorithms and Data Structures (MSCS-532)/MS
  CS-532-Assignment_6/SelectionAlgorithms.py"
  Distribution: random
  Size: 100, Det Time: 0.000121s, Rand Time: 0.000142s
  Size: 1000, Det Time: 0.001222s, Rand Time: 0.001460s
  Size: 5000, Det Time: 0.006027s, Rand Time: 0.006619s
  Size: 10000, Det Time: 0.012174s, Rand Time: 0.017589s
  Distribution: sorted
  Size: 100, Det Time: 0.000124s, Rand Time: 0.000156s
  Size: 1000, Det Time: 0.001073s, Rand Time: 0.001375s
  Size: 5000, Det Time: 0.005323s, Rand Time: 0.007028s
  Size: 10000, Det Time: 0.010487s, Rand Time: 0.011426s
  Distribution: reverse_sorted
  Size: 100, Det Time: 0.000134s, Rand Time: 0.000155s
  Size: 1000, Det Time: 0.001136s, Rand Time: 0.001619s
  Size: 5000, Det Time: 0.005518s, Rand Time: 0.007562s
  Size: 10000, Det Time: 0.010910s, Rand Time: 0.014107s
○ suyog@Suyogs-MacBook-Pro MSCS-532-Assignment_6 % $
```

Summary of the Result

For smaller input sizes, such as 100 and 1,000 elements, the deterministic selection algorithm shows a slight performance advantage over the randomized one. As the input size grows to 5,000 and 10,000, the deterministic algorithm consistently maintains its lead, performing faster across all larger inputs. This trend is observed with random distributions, where the deterministic algorithm consistently outperforms the randomized approach. A similar pattern emerges when the data is sorted. Here too, the deterministic algorithm proves to be faster than the randomized version, although the difference in performance narrows for larger input sizes. For reverse-sorted arrays, the deterministic algorithm once again outshines the randomized one, with the performance gap widening as the size of the array increases.

Theoretical vs Empirical Analysis:

- **Deterministic (Median of Medians)** achieves O(n) in the worst case, which is why its performance remains stable and slightly faster as input size grows.
- **Randomized (Quickselect)** has an expected O(n) time complexity, but the performance may degrade with poor pivot choices, especially in certain distributions like reverse-sorted arrays.

References

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). MIT Press.