Details about Time Complexity

**Best Case:**

The best case for Quick Select occurs when the chosen pivot consistently divides the array into roughly equal halves, similar to the best case of QuickSort. In the best case scenario, the pivot chosen at each step partitions the array into two nearly equal-sized subarrays. This ideal partitioning leads to efficient reduction of the problem size.

Time Complexity: O(n)

In the best case, the algorithm's time complexity approaches linear time, O(n), similar to partitioning in QuickSort when the pivot consistently divides the array evenly.

**Worst Case**:

The worst case for Quick Select happens when the chosen pivot is consistently a minimum or maximum element (i.e., the pivot doesn't effectively split the array). In this case, the algorithm can degrade to a quadratic time complexity due to inadequate partitioning.

Time Complexity: O(n^2)

The worst-case time complexity occurs when the chosen pivot is consistently either the smallest or largest element, resulting in unbalanced partitions and poor reduction in problem size.

**Average Case:**

The average case for Quick Select considers a random selection of pivots or a pivot selection strategy that, on average, leads to balanced partitions. Quick Select typically performs well on average due to randomized pivot selection, which tends to yield balanced partitions across multiple iterations.

Time Complexity: O(n)

On average, the time complexity of Quick Select is O(n). The average case performance is influenced by the random selection of the pivot, which results in effective partitioning and reduction of problem size.

**Summary:**

Best Case: O(n) when the chosen pivot consistently divides the array into roughly equal halves.

Worst Case: O(n^2) when the chosen pivot is consistently poorly chosen (e.g., always the smallest or largest element).

Average Case: O(n) when the pivot is chosen randomly or with a strategy that generally leads to balanced partitions.