Quick Sort

1. Randomized Quicksort

Randomized Quicksort is a variation of the Quicksort algorithm that introduces randomness in the choice of the pivot element. Instead of always choosing a fixed pivot (e.g., the first or last element), a random element is selected as the pivot. This randomness helps avoid worst-case scenarios that can occur with a fixed pivot selection.

Implement and analyze the randomized Quicksort algorithm to understand the impact of random pivot selection on sorting performance. Ensure that your implementation is correct by testing it on various input arrays and comparing the results. Compare the average-case performance of randomized Quicksort with the basic Quicksort on arrays with random elements. Discuss your findings in terms of time complexity and how randomization influences the algorithm's behavior.

2. Dual Pivot Quicksort

Dual Pivot Quicksort extends the standard Quicksort algorithm by using two pivot elements during the partitioning process. It aims to improve efficiency by dividing the array into three segments rather than two.

How it Works:

- Choose two pivot elements from the array.
- Partition the array into three segments: elements less than the left pivot, elements between the pivots, and elements greater than the right pivot.
- Recursively apply the same process to the three segments.

Implement the dual pivot Quicksort algorithm and briefly explain your thoughts on the concept, such as how the steps affect the number of comparisons and swaps and why (using comments between your code lines).

3. Quicksort with Three-way Partitioning (optional)

In case you have time left to spare, try to implement the same algorithm using three way partitioning.