CSE222 / BİL505
Data Structures and Algorithms
Homework #6 – Report

ABDULLAH ENES PATIR

1) Selection Sort

Time Analysis	Selection Sort consistently makes same number of comparisions and swaps regardless of the input array's initial order. But it is clear that it performs poorly compared to other algorithms based on the numbers.
Space Analysis	Since selection sort is an in-place and is not a recursive algorithm, its space comlexity is constant.

2) Bubble Sort

Time Analysis	Bubble sort works best when input array is sorted and it performs poorly in reversed and randomized array when i examine the comparision counter. Also swap counter is also a sign for performance because it increases when input is not sorted.
Space Analysis	Since bubble sort is an in-place and is not a recursive function, its space comlexity is constant.

3) Quick Sort

Time Analysis	Quick Sort varies its performance: it is highly efficient in the randomized array with few comparisions, yet less so in the sorted array with high number of comparisons and swaps, probably due to a poor pivot choice.
Space Analysis	Quick sort is an in-place algorithm but since it is a recursive algorithm space comlexity is logarithmic.

4) Merge Sort

Time Analysis	Merge sort performed excellent compared to other algorithms with very low number of comparisions and no swaps at all across different array orders.
Space Analysis	Since it is an in-place and recursive algorithm it has large space comlexity compared to other with a very efficent time complexity. It performs linearl time complexity.

General Comparison of the Algorithms

In sorted arrays, bubble and merge sort is more optimal compared to other two with minimal comparisions and no swaps.

In randomized arrays, quick sort performed really good with low number of comparisions and swaps. Merge sort also performed good.

In reversely sorted arrays, bubble sort and quick sort performed bad in this order with high numbers while merge sort is good.

Overall, Merge Sort is generally the most consistent and efficient, especially with its comparison counts and zero swaps but it also has the worst space complexity. Quick Sort offers high performance but varies with pivot selection. Selection and Bubble Sorts, while memory efficient, performs bad in time efficiency, particularly with in not desired order.