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

Emir İnce

1) Selection Sort

Time Analysis	Selection Sort has a time complexity of O(n^2), where *n* is the number of elements. This complexity arises because it runs two loops: the outer loop moves through each element one by one, and the inner loop checks the rest of the array to find the smallest element to bring forward, making thorough comparisons throughout the process.
Space Analysis	The space complexity for Selection Sort is O(1). This means it is an in-place sorting method. It only needs a small amount of extra space for temporary variables necessary for swapping elements, which makes it very efficient in terms of space usage.

2) Bubble Sort

Time Analysis	Bubble Sort is known for its O(n^2) time complexity in the worst case. It compares each element to the one next to it and swaps them if needed. This swapping continues across the array until no more swaps are required, which can be quite slow for large numbers of elements.
Space Analysis	Bubble Sort's space complexity is minimal, at O(1). It manipulates the array directly, only requiring a small amount of additional space for variables that handle the swaps.

3) Quick Sort

Time Analysis	Quick Sort is typically very efficient with an average time complexity of O(n log n), although it can slow down to O(n^2) under certain conditions, like when the pivot choice is less optimal. The process works by moving smaller elements to one side of a pivot and larger elements to the other side, sorting each section recursively.
Space Analysis	The space complexity for Quick Sort is generally O(log n) because it uses recursion to sort the array, which requires storing information about the parts of the array being worked on in each recursive call.

4) Merge Sort

Time Analysis	Merge Sort always operates in O(n log n) time, making it consistently efficient. It breaks the array down into halves, sorts each half on its own, and then combines them back into a full array, using a methodical merge process that ensures overall order.
Space Analysis	Merge Sort, however, needs additional space, with a space complexity of O(n). This is to accommodate the temporary arrays used during the merging process, which can make it less space-friendly compared to other in-place sorting algorithms.

General Comparison of the Algorithms

When evaluating these algorithms, Quick Sort and Merge Sort generally perform better than Selection Sort and Bubble Sort, especially with larger datasets due to their more complex time complexity. Merge Sort delivers steady results under different conditions, while Quick Sort can be faster but might also show reduced performance in its worst-case scenario depending on how the pivot is selected.