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

ARİFE YURTSEVEN

1) Selection Sort

Time Analysis	Bubble Sort sorts the array by comparing consecutive elements at each step. At each iteration, it selects the smallest element and places it in the array. In the worst case it makes n*(n-1)/2 comparisons, which corresponds to O(n^2) time. This structure is not effective on large data sets.
Space Analysis	The space complexity of Selective Sort is O(1) because it does not require any additional memory for iteration, except for a few variables. Mechanics Analysis: Bubble Sort does not require any additional array or memory space. Makes changes to the array itself. Therefore, it has a constant memory usage.

2) Bubble Sort

Time Analysis	Bubble Sort is based on the process of sorting each element by comparing it with consecutive elements. At each step, one element is compared with the others and replaced if necessary. This operation performs n*(n-1)/2 comparisons, which corresponds to O(n^2) time. It is not effective on large data sets.
Space Analysis	Bubble Sort manipulates the array itself. It does not require any additional array or memory space. It only swaps elements within the array. Therefore, it has a constant memory usage. The space complexity of Bubble Sort is O(1).

3) Quick Sort

Time Analysis	Quick Sort has O(n log n) time complexity in the average case. However, it might be O(n^2) in the worst case. It divides the array into smaller substrings based on a pivot element. And it sorts each string recursively. Its efficiency mostly depends on the selected pivot element. If implemented correctly, it creates efficient performance. Therefore, it is more suitable for use in large data sets.
Space Analysis	Quick Sort does not require any additional array or memory space. However, it requires a call stack due to its recursive nature. Therefore, it uses O(log n) extra memory in the average case. In the worst case scenario it becomes O(n).

4) Merge Sort

Time Analysis	Merge Sort has O(n log n) time complexity in all cases. It constantly splits the array in half and then merges it. At each splitting step, it traverses the entire array and sorts each element. Therefore, it is effective on large data sets.
Space Analysis	Merge Sort needs a temporary array during the merge process. Therefore it uses extra memory and requires O(n) extra memory space.

General Comparison of the Algorithms

Selection Sort and Bubble Sort are not effective on large data sets because they have O(n^2) time complexity. They must visit all the staff.

Quick Sort and Merge Sort can be effective on large data sets and have O(n log n) time complexity.

Quick Sort's worst case is worse than Merge Sort's best case, but Quick Sort is faster and uses less memory on average

It has the maximum memory space due to in-place operations in quick sort and bubble sort. In other words, the space complexity is constant. No extra memory is used during pivot selection and element swapping. However, Merge Sort requires extra space for the merge operation. A temporary array is created after each division operation, resulting in additional memory usage.

When looking at sorted strings, unnecessary comparisons are made in quick sort and selective sort. It leads to inefficiency. merge sort is the most efficient. Bubble sort is also very effective on short lines, but its efficiency decreases on long lines.