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

BERKAY EMRE KESKİN

1) Selection Sort

Time Analysis	We have 8 elements in the array. It's comparison counter is 28 for each array types (sorted, reversly sorted & randomized). It's swap counter is 7 for each array types also. This makes that this algorithm has O(n^2) time complexity at each case. Even if the array is sorted it swaps the elements. This is an unnecessary work.
Space Analysis	In this algorithm, space complexity is O(1) because we dont use any partial memory rather than the array and temporary elements to swap the array elements.

2) Bubble Sort

Time Analysis	It's comparison counter is same as the selection sort for each case (28). But if the array is sorted, bubble sort does not swap any elements only iterates through all the array. This is the best case for this algorithm and it's time complexity is O(n). But if the array is reversly sorted, it swaps 28 times. This is the worst case scenario for the bubble sort which is O(n^2). If the array is in the randomized order, it's swap counter changes due to how much it is close to a sorted array. But in the worst case of it it would be O(n^2).
Space Analysis	Like the selection sort, we don't use any extra memory rather than temporary elements which holds O(1) space complexity.

3) Quick Sort

Time Analysis	In quick sort, the worst case scenario is the array is in sorted way. If we choose the pivot poorly, our time complexity will be O(n^2). In my algorithm, i chose the pivot as my last element and as seen in the outputs, it compares 28 times and swaps 35 times. This is because for each partitioning, the pivot would be the larges element in the array. In the best case scenario, time complexity would be O(n * log(n)).
Space Analysis	Space complexity would be different than the selection and bubble sort because we use recursion in this algorithm. In the average case it would be O(log(n)). But if the partioning is not balanced it could be O(n) in the worst case.

4) Merge Sort

Time Analysis	Merge sort is the best sort algorithm among these 4 sort algorithms according to the counters. It doesn't use swap function, instead it merges two arrays. Even if the array is sorted or not, time complexity of the merge sort algorithm is O(n * log(n)) for each case. It divides the array until the divided arrays has 1 elements and after that it merges them together recursively.
Space Analysis	For the merge sort algorithm, we use arrays to divide the array into two halves and we fill them with the initial array. So we have an O(n) space complexity.

General Comparison of the Algorithms

In general, we can compare these algorithms according to their time and space complexities according to their behaviour on the different sorted type arrays.

Selection sort is the worst sort algorithm among these algorithms because it's time complexity at it's best and worst cases is O(n^2). It iterates through the array to find the minimum element and swaps it with the indexed element. Bubble sort gives better time complexity if the array is nearly or fully sorted.It gives amortized O(n) time complexity if it is sorted but it's worst case is if the array is reversly sorted. It compares each duel element for all the elements. Quick sort is better than these two in average case time complexity with O(n * log(n)) but it gives O(n^2) if the array is sorted. This is due to poor selection of pivot. If we choose pivot as our first or last element, for each partition the pivot would be the greatest or lowest element. Merge sort is the best sorting algorithm among these algorithms. It gives O(n * log(n)) time complexity for average. It divides the arrays until they have one element and it merges from single elements to bigger arrays recursively. It holds O(n) space but it gives really great time complexity.

If we compare them for the sorted array, Quick Sort and Selection Sort gives the worst efficiency. Their time complexity is O(n^2) for the sorted array. For Bubble Sort algorithm, sorted array is it's best case scenario and it gives a O(n) time complexity. Merge Sort algorithm gives the same time complexity for the all types of sorted array and it is O(n * log(n)).

For the reversly sorted array, Selection Sort and Bubble Sort gives the worst efficiency. Their time complexity is O(n^2). Quick sort gives an average of O(n * log(n)) and Merge Sort gives O(n * log(n)) also.

At last, if we compare them for the randomized arrays, Bubble Sort and Selection Sort gives an average case of O(n^2). Quick Sort gives O(n * log(n)) and Merge Sort gives O(n * log(n)) also.