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

**Ahmet Hakan Demir**

1. **Selection Sort**

|  |  |
| --- | --- |
| **Time Analysis** | Comparison count and swap count is always the same for a sorted array, a reversely sorted array, and a randomized array; and the time complexity is O(n^2) at best and worst case. |
| **Space Analysis** | There is no extra space usage for selection sort. So, there is no space complexity we can discuss. |

1. **Bubble Sort**

|  |  |
| --- | --- |
| **Time Analysis** | Comparison count is always same as selection sort. but the swap counter depends on array. For sorted array, there is no swap, but for the reversely sorted array, swap counter is equal to comparison count. Time complexity is O(n^2) at the worst case which is a reversely sorted array; and O(n) for the best case which is sorted array. |
| **Space Analysis** | There is no extra space usage like selection sort. |

1. **Quick Sort**

|  |  |
| --- | --- |
| **Time Analysis** | In quick sort, the time complexity is at best when the median chosen as pivot(in my code I couldn’t, I chose rightmost element), which is O(n.logn). And the worst case is in a sorted array if we choose leftmost element, because it will cause all the partitions take linear time. So, it will be O(n^2). And the comparison count depends on pivot choice and swap counter depends on initial array’s sort level. |
| **Space Analysis** | Since we create new subarrays in recursive calls, which starts from n sized and ends with 1 sized arrays, the space complexity will be O(n.log(n)). |

1. **Merge Sort**

|  |  |
| --- | --- |
| **Time Analysis** | The merge sort algorithm divides and conquers the array. Since the array divides to log(n) subarrays recursively and merged linearly; the time complexity is O(n.log(n)) for all the cases. |
| **Space Analysis** | Extra used array size is totally n sized. So, space complexity is O(n). |

**General Comparison of the Algorithms**

According to the table above, for randomly sorted arrays, it is more useful to use the quick sort or merge sort. The selection sort has O(n^2) time complexity that’s why it wouldn’t be effective and efficient for any type of the array. But for nearly sorted arrays, (for example, if you are adding to array and at each addition, you are sorting the array, that means you have a sorted array except one element, and it might be useful to use bubble sort algorithm at that point. but for the complicated and randomly given array, bubble sort algorithm will take more than quick sort and merge sort.

