HW-sorting-exercises

Name: Minjie Shen

CS5008 Homework 5

1. Explain what you think the worst-case, big-Oh complexity and the best-case, big-Oh complexity of bubble sort is. Why do you think that?

   In the worst case big-Oh complexity of bubble sort, the original array is in complete different opposite order of the final sorted array, which means for every element in the for loop, we will need to swap between this element and the next element until the end of the array. We have to choose every element of the array and swap till the end of the array, so together the time complexity will be N * N, which is O(N^2).

   In the best case, the big-Oh complexity of bubble sort is O(n), when we don't have to swap current element with any element after current element. But we still need to iterate till the end of the array to make sure it is sorted.

2. Explain what you think the worst-case, big-Oh complexity and the best-case, big-Oh complexity of selection sort is. Why do you think that?

   The worst case big-Oh complexity of selection sort is O(N^2). This happens when we must iterate to the last of the array to find the minimum value (or maximum value) of the sub-array and swap the current index with the minimum or maximum value we find in the sub-array. First, we need to iterate through the array to mark the start point of the swap position. Then we need to iterate through the array from the start position, in worst case we will iterate till the end of the array to find the minimum value.

   The best case big-Oh complexity of selection sort is O(N), where we only need to iterate through the array and the minimum value that we need to find is exactly at the start position. This means that the array is nearly sorted array.

3. Does selection sort require any additional storage (i.e. did you have to allocate any extra memory to perform the sort?) beyond the original array?

   The additional storage is O(1), we only need to have an integer space to store the minimum value of the sub-array for each iteration.

4. Would the big-Oh complexity of any of these algorithms change if we used a linked list instead of an array?

   The big-Oh complexity of these algorithms will not change if we used a linked list instead of an array. For an array, the search time complexity is O(1), but the add and delete time

is O(N). For a linked list, the search time complexity is O(N), but the delete and insert time is O(1).

5. Explain what you think big-Oh complexity of sorting algorithm that is built into the c libraries is. Why do you think that?

   The built in qsort() function in the std c library is a kind of quick sort. The time complexity for the qsort() is O(N log(N)). From running the experiment on the terminal, the total time spend for the c library sorting algorithm is much faster than the bubble sort and the selection sort. So I looked into the c library sorting algorithm and find out that this sort is a quick sort which is much faster than bubble sort and selection sort.