Lab07 - Sorting

Prasun Surana
CS5008, Summer 1

1. The worst case time complexity for bubble sort would be O(n^2), which would be for a reverse sorted array (for instance, if we are trying to sort by ascending order, the worst case would be if the array is in descending order). This is because for an unsorted/reverse ordered array, the bubble sort algorithm has to go through each element in the array and shift the largest value to the end of the array. It goes through N-1 iterations in the first pass, N-2 iterations in the second pass, and so on, N-1 times. So the worst case is O(n^2).

The best case time complexity for bubble sort would be O(n^2) for our implementation. The algorithm checks all N elements, and even if there is no swap, it continues and completes N iterations. However, if we improved our bubble sort algorithm by adding a flag variable, we could terminate the loop if there was no swapping, i.e. if the array became sorted. In the best case, i.e. an already-sorted array, the algorithm still checks all N elements in the array, but since there is no swap, we break out of the loop. Therefore, with this improved implementation, the best-case time complexity of bubble sort could be O(n).

2. The worst case time complexity for selection sort would be O(n^2). For the first pass of selection sort, we have to find the smallest value in the array, and swap it with the first position. For the next pass, we have to find the smallest value in the unsorted portion of the array, and put it into the second position, etc, until we get to the end of the array. Since we are iterating through the array to find the minimum value, and then doing that for each value in the array, the worst case time complexity would be O(n^2).

The best case time complexity will still be O(n^2). Even if the array is sorted, and the algorithm performs no swaps, in the next iteration, it still has to check the remained of the array for the minimum. Hence, its time complexity does not change.

3. We did not have to allocate any additional memory for the selection sort implementation.

4. The time complexity for a linked list vs. an array when using these sorting algorithms would not change. This is because the time complexity for searching through an array and a linked list are both O(n). With a linked list for bubble sort, the algorithm still has to start from the beginning and iterate through all the nodes and do the swaps, and do this until the array is sorted, so the worst case is still O(n^2), and if it were already sorted, it would iterate through each element, and once it finds that no swaps were made, we could break out of the loop, thus having a time complexity of O(n).

If we were running selection sort on the linked list, it would be the same case as an array, where we would have to sequentially check each element, find and store the minimum, and then perform the swap, and then do this for the unsorted array until the entire array was in the correct order. Hence, it would not change for selection sort either, and would still have a best and worst case time complexity of O(n^2).

5. The time complexity for the qsort function seems to be nlog(n). From the runtimes, we can see that the runtimes are multiplied by a factor of roughly 10 every time the input size is multiplied by 10.