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

The worst-case Big-O complexity of bubble sort is O(n^2). This is because in the worst-case scenario, the largest element must be swapped n-1 times to reach its final position in the sorted list, meaning that the algorithm must perform n-1 comparisons for each of the n elements.

For the bubble sort solution provided by CS5008_ClassRepo, The best-case Big-O complexity of bubble sort is still O(n^2). But if we could try to optimize our code to add one boolean controller to check did we swap the item. If we don't swap items, we don't have to go through the inner loop. So the best case will be like O(n) under the situation that we are sorting a sorted array.

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

The worst-case and best-case Big-O complexity of the selection sort both is O(n^2). The same selection sort is to compare n(n-1)/2 times regardless of the sequence, which is the number of comparisons
And for the number of swaps: if the array is ordered then no swap is needed, if it is in reverse order then n swaps are needed
In total, the time complexity is O(N2) at best and at worst

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?

Selection sort does not require any additional storage, by swapping elements within the array. No extra memory is needed to store intermediate values during the sorting process.

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

The time complexity for the approach to swap the elements of the Linked List is O(N), The data structure used to store the elements does not change the overall algorithm's performance characteristics. The Big-O complexity of bubble sort if implemented using a linked list would still be O(n^2).

The Big-O complexity of selection sort if we used a linked list would still be O(n^2) in the worst-case scenario, where n is the number of elements in the list. This is because the algorithm requires iterating over the list to find the minimum element, and then swapping it with the current position, which is repeated for n-1 elements.

5. Explain what you think Big-O complexity of sorting algorithm that is built into the C libraries is. Why do you think that?

The sort algorithm we use in csort.c is the quicksort algorithm which has an average-case time complexity of O(n log n) and a worst-case time complexity of O(n^2). These algorithms have been optimized and widely used, making them a common choice for sorting in C libraries. Also using it in Python, and Java libraries.