Written Exercise for Lab04 Sorting

Name: Rong Huang
Date:02/18/2023

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?**

Bubble sort compares each pair of adjacent items and swaps them if they are in the wrong order. In the worst case, every possible comparison and swap needs to be made. In the best case, if the array is already sorted, only comparisons are made, and no swaps are needed, making it linear.

Worst-case Big-O complexity of bubble sort: O(n^2). This happens when the array is in reverse order, and we need to compare and swap every single element with every other element to get it in the right place.

Best-case Big-O complexity of bubble sort: O(n). This happens when the array is already sorted, and we just need to make one pass through the array to check that everything is in order, with no swaps needed.

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?**

In selection sort, we are always looking for the minimum (or maximum) element in the unsorted part of the array to place it in the sorted part. This process doesn't change regardless of the array's initial order.

Worst-case Big-O complexity of selection sort: O(n^2). This is because, for each element in the array, you search through the rest of the array to find the minimum element and place it in the correct position.

Best-case Big-O complexity of selection sort: O(n^2). Unlike bubble sort, even if the array is sorted, you still need to look for the minimum element for each position in the array, making the number of comparisons the same.

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?**

No, selection sort does not require any additional storage beyond the original array. It is an in-place sorting algorithm, meaning it only needs a small, constant amount of additional memory space for its operations (like storing the index of the minimum element).

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

The Big-O complexity for the overall algorithm remains the same (O(n^2) for both bubble sort and selection sort) because the number of comparisons doesn't change. However, the operations themselves (like swapping elements in bubble sort or finding the minimum in selection sort) could become more expensive in terms of time due to the nature of linked lists (e.g., needing to traverse the list to access 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 C standard library function qsort is typically implemented using a version of quicksort, which has a worst-case complexity of O(n^2) and an average-case complexity of O(n log n).

Quicksort is chosen for its efficient average-case performance and relatively simple implementation. Its average-case complexity is O(n log n) because it divides the array into partitions and then sorts each partition recursively. The worst case is rare and usually mitigated by using techniques like choosing good pivot values or switching to a different sorting algorithm for small arrays.