44 Lecture - CS301

Important Subjective

1. **What is Selection Sort? Explain with an example.**
   Answer: Selection Sort is an algorithm that sorts an array by repeatedly finding the minimum element from the unsorted part of the array and putting it at the beginning. This process is continued until the whole array is sorted. For example, consider the following array: [64, 25, 12, 22, 11]. The steps to sort this array using selection sort are:

   Find the minimum element in the unsorted array, which is 11.
   Swap the minimum element with the first element of the unsorted array, which results in [11, 25, 12, 22, 64].
   Repeat the above two steps for the remaining unsorted array, resulting in [11, 12, 22, 25, 64].

2. **What is the time complexity of Selection Sort? Explain how you arrived at this answer.**
   Answer: The time complexity of Selection Sort is O(n^2), where n is the number of elements in the array. This is because for each element in the array, we need to find the minimum element in the remaining unsorted part of the array, which takes O(n) time. Since we repeat this process n times, the overall time complexity becomes O(n^2).

3. **Can Selection Sort be used to sort a linked list? If yes, explain how. If no, explain why not.**
   Answer: Yes, Selection Sort can be used to sort a linked list. In a linked list, we can find the minimum element in the remaining unsorted part of the list by traversing the list and keeping track of the minimum element. Once we find the minimum element, we can remove it from its current position and insert it at the beginning of the sorted part of the list. We repeat this process until the whole list is sorted.

4. **What is the best-case time complexity of Selection Sort? Explain when this scenario occurs.**
   Answer: The best-case time complexity of Selection Sort is O(n^2). This scenario occurs when the array is already sorted or nearly sorted, as the algorithm still needs to check each element in the unsorted part of the array to ensure that it is in the correct position.

5. **Compare and contrast Selection Sort and Bubble Sort.**
   Answer: Selection Sort and Bubble Sort are both simple sorting algorithms with a time complexity of O(n^2). However, Selection Sort is more efficient than Bubble Sort as it makes fewer comparisons. In Selection Sort, we find the minimum element in the remaining unsorted part of the array and swap it with the first element of the unsorted part. In Bubble Sort, we repeatedly compare adjacent elements and swap them if they are in the wrong order. This means that Bubble Sort makes more comparisons than Selection Sort, making it less efficient.

6. **How does the number of elements in the array affect the performance of Selection Sort?**
   Answer: The time complexity of Selection Sort is O(n^2), where n is the number of elements in the array. This means that as the number of elements in the array increases, the time taken to sort the array increases quadratically. Therefore, Selection Sort is not efficient for large arrays.

7. **Can Selection Sort be used to sort an array in descending order? If yes, explain how. If no, explain why not**.
   Answer: Yes, Selection Sort can be used to sort an array in descending order. Instead of finding the minimum element in the unsorted part of the array, we find the maximum element and swap it with the first element of the unsorted part. We repeat this process until the whole array is sorted in descending order.

8. **Explain the concept of in-place sorting in Selection Sort.**
   Answer: In-place sorting refers to the property of sorting algorithms that