# Search:

**Question 1:**

Q: What is linear search, and how does it work?

A: Linear search is a simple search algorithm that iterates through each element in a list or array until the desired element is found or the end of the list is reached. It compares each element sequentially until a match is found.

# Question 2:

Q: Describe the time complexity of linear search.

A: The time complexity of linear search is (O(n)), where (n) is the number of elements in the list. In the worst case, the algorithm may need to examine each element.

# Question 3:

Q: What is binary search, and when is it applicable?

A: Binary search is a divide-and-conquer search algorithm used on sorted arrays. It repeatedly divides the search interval in half until the target element is found. It is applicable only to sorted lists.

# Question 4:

Q: Explain the time complexity of binary search.

A: The time complexity of binary search is (O(log n)), where (n) is the number of elements in the sorted array. The algorithm reduces the search space by half at each step.

# Sort:

**Question 5:**

Q: What is bubble sort, and how does it work?

A: Bubble sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The pass-throughs are repeated until the entire list is sorted.

# Question 6:

Q: State the time complexity of bubble sort.

A: The time complexity of bubble sort is (O(n2)) in the worst and average cases. It is not the most efficient sorting algorithm for large datasets.

# Question 7:

Q: What is quicksort, and why is it considered efficient?

A: Quicksort is a divide-and-conquer sorting algorithm that partitions the array into smaller subarrays, recursively sorts them, and then combines them. It is efficient due to its average- case time complexity of (O(n log n)) and low overhead.

# Question 8:

Q: Explain the pivot selection in the quicksort algorithm.

A: In quicksort, a pivot element is chosen from the array. The array is then partitioned into two subarrays: elements less than the pivot and elements greater than the pivot. The pivot is then placed in its final sorted position.

# Question 9:

Q: Define insertion sort.

A: Insertion sort is a simple sorting algorithm where elements are iteratively taken from an unsorted part of the array and inserted into their correct positions in the sorted part.

# Question 10:

Q: State the time complexity of insertion sort.

A: The time complexity of insertion sort is (O(n2)) in the worst case. However, it performs well for small datasets and partially sorted arrays, making it adaptive in certain scenarios.