Explain different sorting algorithms (Bubble Sort, Insertion Sort, Quick Sort, Merge Sort).

* Bubble Sort is an elementary sorting algorithm that goes back and forth through the list, comparing adjacent elements and if necessary swapping them. This process continues until the list is completely ordered. In both the worst and average cases, the time complexity is O(n^2), but in the best case when the array is already sorted it can achieve O(n) too. Bubble Sort is an in-place sorting algorithm with a space complexity of O(1), which means that it does not need extra storage beyond the input array to execute.
* Insertion Sort creates the sorted array one sequence of characters at a time by copying an element from the input data and then placing it in the last position within the fully sorted section of the list. The insertion sort algorithm has time complexity of O(n^2) in the worst and average cases but it is better with a time complexity of O(n) in the best case when the array is already sorted. The Insertion Sort approach in the list literature may refer to an in-place sorting algorithm with a space complexity of O(1).
* Quick Sort is one of the most efficient sorting algorithms and a favorite example of the divide and conquer problem-solving strategy. It has the following sub-steps that include choosing an element known as pivot, breaking up the list into sub-lists with elements greater than or equal to the pivot located in one portion and those less than the pivot in another portion, sorting each sub-list, and then combining the results. The average and best-case time complexity for Quick Sort are O(n log n), but worst-case time complexity can be as high as O(n^2) when the pivot is not chosen properly. Quick Sort, on the other hand, has a space complexity of O(log n) resulting from the recursion stack, which makes it memory efficient unlike others.
* Merge Sort is the process in which the original array is divided into two halves recursively both of which should be in the ordered manner. Then the merge will be performed to obtain a fully sorted sequence of numbers. Regardless of how the elements of the list were initially, Merge sort still delivers a time complexity of O(n log n). The extra storage required by Merge Sort comes from the merge, thus the space complexity of O(n). The extra space is used to create temporary arrays for the merge process.

Compare the performance (time complexity) of Bubble Sort and Quick Sort.

Bubble Sort has a time complexity of O(n^2), which means that it is not a good option for large datasets because its complexity is quadratic and the number of comparisons and swaps required grows with the size of the dataset. Its space complexity is O(1), as it sorts the data in place without additional memory usage. As a result, Bubble Sort is generally not as fast and efficient as other sorting methods for very large arrays or lists.

Quick Sort is much more efficient with an average time complexity of O(n log n), making it faster and better suited for large datasets. Still, n^2 remains as the worst-case time complexity which is likely to happen when poor pivot choices are made. Quick Sort has a space complexity of O(log n) due to the recursion stack, which is significantly lower than Bubble Sort. The main advantage of Quick Sort is that it is typically the fastest and most efficient sorting algorithm, particularly when we need to deal with huge data sets.

Discuss why Quick Sort is generally preferred over Bubble Sort.

Quick Sort is generally faster due to its average time complexity of O(n log n) compared to Bubble Sort’s O(n^2). It performs better with larger datasets and is more scalable. It can be optimized further with different pivot strategies to avoid worst-case scenarios.