Name: Suraj Balaso Desai

10 Quiz Questions on Merge Sort

1] In Merge Sort, the merging of two sorted sub-arrays into a single sorted array is performed by:
A) Swapping elements of the two sub-arrays
B) Comparing elements of the two sub-arrays and merging them in order
C) Shifting elements of one sub-array to the other sub-array
D) None of the above

Answer: B) Comparing elements of the two sub-arrays and merging them in order

Explanation: Merge Sort is a divide and conquer sorting algorithm that sorts an array by dividing it into two halves, sorting each half separately, and then merging the two sorted halves. The merging process involves comparing elements of the two sub-arrays and merging them in order. During the merging process, we compare the first elements of both sub-arrays, and the smaller one is placed in the output array first. Then we compare the second element of the sub-array from which the smaller element was taken with the next element of the other sub-array, and so on. This process continues until we have merged all the elements from both sub-arrays into the output array. So, option B is the correct answer.

2] What is the time complexity of Merge Sort algorithm in the worst case?
A) O(n)
B) O(n log n)
C) O(n^2)
D) O(log n)

Answer: B) O(n log n)

Explanation: The time complexity of Merge Sort in the worst case is O(n log n), where 'n' is the number of elements in the array. This is because Merge Sort is a divide and conquer algorithm that divides the input array into two halves recursively until each sub-array has only one element. Then it merges these sub-arrays in a sorted order. The merge operation takes O(n) time, and there are log n levels of recursion, so the overall time complexity is O(n log n). Therefore, option B is the correct answer.

3] Which of the following statements is true about Merge Sort?
A) It is a stable sorting algorithm.
B) It requires less auxiliary space than Quick Sort.
C) It has a worst-case time complexity of O(n^2).
D) It is not suitable for sorting large datasets.

Answer: A) It is a stable sorting algorithm.

Explanation: A sorting algorithm is said to be stable if it maintains the relative order of equal elements in the input array. Merge Sort is a stable sorting algorithm because during the merging process, when two equal elements are encountered, the element from the left sub-array is placed in the output array first, maintaining the relative order. Therefore, option A is the correct answer.

4] What is the worst-case space complexity of Merge Sort algorithm, assuming an array of 'n' elements?

A) O(1)
B) O(n)
C) O(log n)
D) O(n log n)

Answer: B) O(n)

Explanation: The space complexity of Merge Sort depends on the implementation, specifically on how the merging is done. In a typical implementation, Merge Sort uses auxiliary space to store the two sub-arrays being merged. In the worst case, the recursion tree of Merge Sort is skewed and has a height of 'n', which means that 'n' sub-arrays are being merged simultaneously. At each level of the recursion tree, the total size of the sub-arrays being merged is 'n'. Therefore, the worst-case space complexity of Merge Sort is O(n), which means that it uses an auxiliary space proportional to the size of the input array. Option B is the correct answer.

5] Which of the following is NOT a disadvantage of Merge Sort?

A) It requires additional space proportional to the size of the input array.
B) It is not an in-place sorting algorithm.
C) It has a worst-case time complexity of O(n^2).
D) It has a higher constant factor than some other sorting algorithms.

Answer: C) It has a worst-case time complexity of O(n^2).

Explanation: Merge Sort has a worst-case time complexity of O(n log n), which is better than O(n^2) algorithms like Bubble Sort and Insertion Sort. Therefore, option C is incorrect and not a disadvantage of Merge Sort.

6] Which of the following statements is true about Merge Sort when sorting a linked list?

A) It requires additional space proportional to the size of the input list.
B) It is faster than Quick Sort when sorting a linked list.
C) It is not a stable sorting algorithm when sorting a linked list.
D) It cannot be used to sort a linked list.

Answer: A) It requires additional space proportional to the size of the input list.

Explanation: When using Merge Sort to sort a linked list, it requires additional space proportional to the size of the input list for the auxiliary array used in the merging process. Therefore, option A is the correct answer.

7] Which of the following statements is true about Merge Sort when sorting a nearly sorted array?

A) It runs in O(n log n) time complexity.
B) It has a higher time complexity than some other sorting algorithms.
C) It requires more auxiliary space than Quick Sort.
D) It performs poorly on nearly sorted arrays.

Answer: A) It runs in O(n log n) time complexity.

Explanation: Merge Sort has a worst-case time complexity of O(n log n), which is the same for all input arrays. Therefore, when sorting a nearly sorted array, it still runs in O(n log n) time complexity. Therefore, option A is the correct answer.

8]Which of the following is true about Merge Sort's stability property?
A) Merge Sort is stable when sorting primitive types but not when sorting objects.
B) Merge Sort is not stable when sorting arrays but is stable when sorting linked lists.
C) Merge Sort is always stable regardless of the type of data being sorted.
D) Merge Sort is stable for a limited range of input sizes.

Answer: C) Merge Sort is always stable regardless of the type of data being sorted.

9] Which of the following is a benefit of Merge Sort over Quick Sort?
A) It has a better worst-case time complexity.
B) It has a smaller constant factor in its time complexity.
C) It requires less auxiliary space in the worst-case.
D) It is an in-place sorting algorithm.

Answer: A) It has a better worst-case time complexity.

Explanation: Merge Sort has a worst-case time complexity of O(n log n), which is better than Quick Sort's worst-case time complexity of O(n^2). Therefore, option A is the correct answer.

10] Consider an implementation of Merge Sort that uses a recursive algorithm for the merging step. Which of the following statements is true about the worst-case space complexity of this implementation?
A) The worst-case space complexity is O(1).
B) The worst-case space complexity is O(n).
C) The worst-case space complexity is O(log n).
D) The worst-case space complexity is O(n log n).

Answer: D) The worst-case space complexity is O(n log n).

Explanation: Merge Sort's worst-case space complexity is O(n) due to the need for an auxiliary array in the merging step. However, if the merging step is implemented recursively, then the call stack space also needs to be taken into account.

When Merge Sort is applied to an array of size n, the algorithm recursively splits the array in half until the base case of a single-element array is reached. At each level of the recursion, the algorithm creates two new subarrays, and each subarray requires O(n/2) space. Therefore, the total space required for the subarrays at any level of recursion is O(n).

Since the recursion depth is O(log n) due to the halving of the array size at each level, the worst-case space complexity of the recursive Merge Sort implementation is O(n log n).