**Merge-Sort Algorithm**

Here are some multiple-choice questions related to the merge sort algorithm:

**Easy:**

1. What is the key advantage of using the merge sort algorithm?  
   a) It has a best-case time complexity of O(n log n).  
   b) It is an in-place sorting algorithm.  
   c) It is suitable only for small datasets.  
   d) It has an average time complexity of O(n^2).
2. Which step of the merge sort algorithm involves dividing the array into smaller subarrays?  
   a) Divide  
   b) Conquer  
   c) Merge  
   d) Combine
3. Merge sort is considered stable because:  
   a) It sorts elements in an unstable manner.  
   b) It preserves the relative order of equal elements.  
   c) It is faster than other sorting algorithms.  
   d) It has a worst-case time complexity of O(n log n).

**Medium:**

1. In merge sort, what is the purpose of the "conquer" step?  
   a) Dividing the array into subarrays  
   b) Sorting the subarrays individually  
   c) Merging the subarrays together  
   d) Combining sorted subarrays into a final result
2. Which of the following statements is true about merge sort?  
   a) It uses recursion only for the merging step.  
   b) It guarantees the best-case time complexity of O(n).  
   c) It can be parallelized to achieve linear time complexity.  
   d) It can sort linked lists more efficiently than arrays.
3. What is the time complexity of merging two sorted subarrays of lengths m and n?  
   a) O(m \* n)  
   b) O(m + n)  
   c) O(log m + log n)  
   d) O(max(m, n))

**Hard:**

1. Merge sort is not an in-place sorting algorithm because:  
   a) It uses divide and conquer strategy.  
   b) It is inherently unstable.  
   c) It requires auxiliary memory for the merging step.  
   d) It has a worst-case time complexity of O(n log n).
2. Which data structure can be used to optimize the merging step of the merge sort algorithm to achieve O(n) time complexity?  
   a) Priority queue  
   b) Binary search tree  
   c) Hash table  
   d) Heap
3. Merge sort's stability and predictable time complexity make it an ideal choice for:  
   a) Sorting small datasets.  
   b) Real-time systems.  
   c) Parallel processing.  
   d) Sorting large datasets efficiently.