Name: Yevgeny Omar Siregar

Student No: 2602220453

Class: L3BC

Individual Assignment

1. Divide and conquer approach breaks down a problem into multiple sub-problems recursively until it cannot be divided further. These sub-problems are solved first and the solutions are merged together to form the final solution.

   Divide: We divide the original problem into multiple sub-problems until they cannot be divided further.
   Conquer: Then these subproblems are solved separately with the help of recursion
   Combine: Once solved, all the subproblems are merged/combined together to form the final solution of the original problem.

2.

   - Merge Sort:

   Merge sort is a popular sorting algorithm that uses the divide and conquer strategy. It divides the input array into smaller subarrays, recursively sorts those subarrays, and then merges them back together in a sorted manner. It guarantees a time complexity of O(n log n) for sorting.

   - Quick Sort:

   Quick sort is another widely used sorting algorithm that follows the divide and conquer paradigm. It chooses a "pivot" element from the array and partitions the array into two subarrays, one containing elements less than the pivot and the other containing elements greater than the pivot. It then recursively sorts the subarrays. Quick sort also has an average time complexity of O(n log n) and is often faster in practice compared to merge sort due to its smaller constant factors.

   - Binary Search:

   Binary search is an efficient searching algorithm that relies on divide and conquer. Given a sorted array, it repeatedly divides the search interval in half and compares the target value with the middle element. Depending on the comparison, it narrows down the search to the left or right half of the array. This process continues until the target element is found or the search interval becomes empty. Binary search has a time complexity of O(log n).

3. A = (3, 41, 52, 26, 38, 57, 9, 49)

   Left half: (3, 41, 52, 26)

   Right half: (28, 57, 9, 49)

   Left half: (3, 41)

   Right half: (52, 26)

   Left half: (3)

   Right half: (41)

   Right half: (52)

   Left half: (26)

   Right half: (28, 57)

   Left half: (28)

   Right half: (57)

   Left half: (9, 49)

   Left half: (9)

   Right half: (49)

   Merge (3) and (41) => (3, 41)

   Merge (52) and (26) => (26, 52)

   Merge (3, 41) and (26, 52) => (3, 26, 41, 52)

   Merge (28) and (57) => (28, 57)

   Merge (9) and (49) => (9, 49)

   Merge (28, 57) and (9, 49) => (9, 28, 49, 57)

   Merge (3, 26, 41, 52) and (9, 28, 49, 57) => (3, 9, 26, 28, 41, 49, 52, 57)

4. The choice between Quick Sort and Merge Sort depends on various factors, including the specific requirements of your problem and the characteristics of the data you're sorting.

   Quick Sort is often chosen over Merge Sort when you need an efficient, in place sorting algorithm with good average case performance, and space constraints are a consideration. However, it's important to analyse your specific problem and data characteristics to make an informed decision, as the choice between these sorting algorithms can depend on the unique requirements of your application.

5. Heap Sort is typically used in scenarios where you need a stable and efficient sorting algorithm with a consistent O(n log n) time complexity, and you don't have concerns about the stability of the sort (i.e., the relative order of equal elements).

Heap Sort is a reliable and efficient sorting algorithm suitable for a wide range of scenarios, especially when you prioritize consistent performance and in-place sorting over stability. However, it's important to consider the specific requirements of your problem and your data characteristics before choosing Heap Sort or any other sorting algorithm.