1. Explain what you think the worst-case, big-Oh complexity and the best-case, big-Oh complexity of merge sort is. Why do you think that?

   In the worst-case, the time complexity of merge sort is O(nlogn) because we will need to divide elements logN times and each time we will need to compare all elements in the subarray so the time will be O(n). In the best-case, the time complexity is also O(nlogn) because we will need to divide the list, compare and merge even the list is already sorted.

2. Merge sort as we have implemented in there is a recursive algorithm as this is the easiest way to think about it. It is also possible to implement merge sort iteratively. Explain what you think the worst-case, big-Oh complexity and the best-case, big-Oh complexity is for this iterative merge sort. Why do you think that?

   The time complexity of this iterative merge sort for worst case and best case is also O(nlogn). Just like the recursive mergo sort, this iterative version will divide the array nlogn times because the first for loop will run logN times no matter the arrary is sorted or not. The second for loop will compare all the elements in the subarray so the time complexity will be O(nlogn).