1. My choice algorithm is a merge sort because if satisfies the time complexity of O(NlogN). This is because it sorts it uses recursion which gives a time complexity of logN. But the recursion is nested in a while loop. Thus giving a O(NlogN).

2. The best case time complexity of this algorithm should be if the list is already sorted. However, even if its already sorted, the algorithm still needs to split them apart and attempt to sort it. Thus time complexity is still O(NlogN).

3. The worst case time complexity would be O(NlogN) because it would have to loop through the every element and call the function every time.