The worst-case, best-case, and average-case time complexity of Merge Sort can be analyzed as follows:

* Worst-case complexity: The worst-case complexity of Merge Sort occurs when the input array is in reverse order. In this case, the algorithm must perform the maximum number of comparisons and merges. The worst-case time complexity of Merge Sort is O(n log n), where n is the size of the input array.
* Best-case complexity: The best-case complexity of Merge Sort occurs when the input array is already sorted. In this case, the algorithm can detect that the sub-arrays are already sorted and skip the merge step. The best-case time complexity of Merge Sort is also O(n log n).
* Average-case complexity: The average-case complexity of Merge Sort is also O(n log n). This can be proven using mathematical induction or by analyzing the recursive tree of sub-problems that Merge Sort creates.

In terms of space complexity, Merge Sort has a worst-case space complexity of O(n) because it needs to create temporary arrays to hold the sorted sub-arrays during the merging phase. However, this can be improved to O(log n) by using an iterative version of the algorithm or by implementing an in-place version of the merge step.