The worst-case, best-case, and average-case time complexity of Quick Sort are:

* Worst-case complexity: The worst-case time complexity of Quick Sort occurs when the input array is already sorted or nearly sorted, and the pivot element is always chosen as the largest or smallest element. In this case, the partitioning step will always divide the array into two sub-arrays of size n-1 and 1, respectively, resulting in a worst-case time complexity of O(n^2). However, this worst-case scenario is rare and can be avoided by choosing the pivot element randomly or by using a median-of-three pivot selection method. With these improvements, the worst-case time complexity of Quick Sort is O(n log n).
* Best-case complexity: The best-case time complexity of Quick Sort occurs when the pivot element divides the input array into two sub-arrays of roughly equal size. In this case, the recursion tree will have a balanced structure, and each level will process n elements, resulting in a best-case time complexity of O(n log n).
* Average-case complexity: The average-case time complexity of Quick Sort is O(n log n). This can be proven using mathematical analysis or by using probabilistic arguments.

In terms of space complexity, Quick Sort has a worst-case space complexity of O(n) due to the recursion stack. However, the average-case space complexity is O(log n) due to the balanced recursion tree.