**Time Complexity Analysis:**

**Quicksort:**

Average Case: O(n log n)

Worst Case: O(n^2) (unlikely with a good pivot strategy)

Best Case: O(n log n)

**Comparison with Merge Sort:**

**Quicksort:**

**Advantages:**

In-place sorting, requiring only a constant amount of additional memory.

Often faster in practice for average cases, especially on cache-aware architectures.

More cache-friendly due to its in-place nature.

**Disadvantages:**

Worst-case time complexity of O(n^2) can occur with poor pivot choices, leading to performance issues.

Unstable sorting (equal elements may not retain their original order).

**Merge Sort:**

**Advantages:**

Stable sorting, ensuring that equal elements retain their relative order.

Guaranteed O(n log n) time complexity in all cases.

Well-suited for linked lists.

**Disadvantages:**

Requires additional memory proportional to the size of the input array.

Slower in practice for small arrays due to the overhead of merging.

**Strengths of Quicksort:**

1. Efficiency: Quicksort is generally faster than many other sorting algorithms, especially for large datasets.

2. In-place Sorting: Quicksort is an in-place sorting algorithm, making it memory-efficient.

3. Adaptive: It performs well in practice even when the input is partially sorted.

**Weaknesses of Quicksort:**

1. Worst-Case Performance: The worst-case time complexity of O(n^2) can occur with poor pivot choices, making it less predictable in certain situations.

2. Unstable Sorting: Quicksort is unstable, meaning that equal elements may not maintain their original order.

**Performance Comparison:**

Quicksort's performance is generally competitive with other O(n log n) algorithms, and it often outperforms them in practice for average cases. However, it can be sensitive to the initial order of elements and the choice of the pivot.

**Conclusion:**

In summary, Quicksort is an effective sorting algorithm that has several noteworthy advantages, such as in-place sorting and efficiency. Its shortcomings—such as worst-case performance and instability—must be taken into account based on the particular needs of the application. The decision between them comes down to stability, memory limitations, and the type of data, in comparison to other sorting algorithms like Merge Sort. Because of its general good performance, Quicksort is still a popular choice for sorting, particularly when average-case efficiency is important.