Time Complexity Analysis

The time complexity of an algorithm is an essential metric to evaluate its efficiency. The table below summarizes the best-case, worst-case, and average-case time complexities of various sorting algorithms.

Algorithm	Best Case	Average Case	Worst Case
Bubble Sort	O(n)	O(n^2)	O(n^2)
Selection Sort	O(n^2)	O(n^2)	O(n^2)
Insertion Sort	O(n)	O(n^2)	O(n^2)
Merge Sort	O(n log n)	O(n log n)	O(n log n)
Quick Sort	O(n log n)	O(n log n)	O(n^2)
Heap Sort	O(n log n)	O(n log n)	O(n log n)
Radix Sort	O(nk)	O(nk)	O(nk)

Comparison of Bubble Sort with Other Algorithms

1. **Bubble Sort vs. Selection Sort**
    - Both have a worst-case time complexity of O(n^2), but Selection Sort makes fewer swaps.
    - Bubble Sort is slightly better when the list is already nearly sorted.
2. **Bubble Sort vs. Insertion Sort**
    - Insertion Sort performs much better than Bubble Sort for small or nearly sorted datasets.
    - Both have O(n^2) worst-case complexity, but Insertion Sort has O(n) best-case complexity, making it more efficient in favorable scenarios.
3. **Bubble Sort vs. Merge Sort**
    - Merge Sort consistently performs in O(n log n) time, making it significantly faster than Bubble Sort for large datasets.
    - Merge Sort requires additional space for merging, whereas Bubble Sort is in-place.
4. **Bubble Sort vs. Quick Sort**

- Quick Sort has an average-case complexity of O(n log n), outperforming Bubble Sort significantly.
- Bubble Sort is only preferred when dealing with extremely small datasets or nearly sorted lists.

5. **Bubble Sort vs. Heap Sort**
    - Heap Sort also has an O(n log n) complexity but requires additional space for the heap.
    - It is much more efficient than Bubble Sort for large data sets.
6. **Bubble Sort vs. Radix Sort**
    - Radix Sort works efficiently when sorting numbers with fixed digit sizes, while Bubble Sort remains inefficient.
    - Radix Sort can outperform comparison-based algorithms in some cases but requires extra space.