

Time and Space Complexity

This document provides a clear and beginner-friendly explanation of Time and Space Complexity, which are fundamental concepts in Data Structures and Algorithms.

1. What is Time Complexity?

Time Complexity measures how the running time of an algorithm grows as the input size increases. It focuses on the number of operations performed rather than actual execution time.

Common Time Complexities:

- $O(1)$ – Constant Time
- $O(\log n)$ – Logarithmic Time
- $O(n)$ – Linear Time
- $O(n \log n)$ – Linearithmic Time
- $O(n^2)$ – Quadratic Time

2. What is Space Complexity?

Space Complexity measures how much memory an algorithm uses relative to the input size. It includes both auxiliary space and input storage.

Examples:

- $O(1)$ – Uses constant extra memory
- $O(n)$ – Uses memory proportional to input size

3. Why Time and Space Complexity Matter

Understanding complexity helps developers choose efficient algorithms, optimize performance, and write scalable code.

4. Key Takeaways

- Time complexity measures execution growth
- Space complexity measures memory usage
- Big-O notation is used to describe both