Time Complexity and Space Complexity

Introduction to Complexity Analysis

Complexity analysis is essential for understanding the efficiency of algorithms.

It allows us to predict the performance of algorithms, especially as the size of input grows.

Two critical measures are time complexity and space complexity.

Time Complexity

Time complexity measures the amount of time an algorithm takes to complete as a function of the input size (often denoted as 'n').

- **O(1)**: Constant time complexity; time taken does not depend on input size.
- **O(log n)**: Logarithmic time; reduces problem size by a constant factor each step.
- **O(n)**: Linear time; time grows directly with input size.
- **O(n log n)**: Linearithmic time; common in efficient sorting algorithms.
- **O(n^2)**: Quadratic time; typical in algorithms with nested loops.
- **O(2^n)**: Exponential time; time doubles with each increase in input.
- **O(n!)**: Factorial time; grows extremely fast and is impractical for large n.

Calculating time complexity involves analyzing loops, recursion, and conditional structures in code.

Space Complexity

Time Complexity and Space Complexity

Space complexity measures the amount of memory an algorithm uses relative to the input size.

- **Auxiliary Space**: Additional space needed aside from input data.
- **Stack Space**: Memory used by function calls (recursion increases stack space).

To calculate space complexity, consider both the input data storage and any extra variables.

Analyzing Algorithm Performance

Performance analysis often considers best, average, and worst-case scenarios.

For most applications, worst-case complexity provides a reliable measure of efficiency.

Real-World Examples

1. **Binary Search** - Time Complexity: O(log n), Space Complexity: O(1).
2. **Bubble Sort** - Time Complexity: O(n^2), Space Complexity: O(1).
3. **Merge Sort** - Time Complexity: O(n log n), Space Complexity: O(n).
4. **Recursive Fibonacci** - Time Complexity: O(2^n), Space Complexity: O(n).

Complexity Cheat Sheet

Time Complexity and Space Complexity

Algorithm	Time Complexity	Space Complexity
Binary Search	O(log n)	O(1)
Bubble Sort	O(n^2)	O(1)
Merge Sort	O(n log n)	O(n)
Quick Sort	O(n log n)	O(log n)
Fibonacci (Rec)	O(2^n)	O(n)

This table provides a quick reference for time and space complexities of popular algorithms.