# Lecture 1.2

Data Structures with Python: Analysis Terms

# Agenda

- Introduction to Algorithm Analysis
- Time and Space Complexity
- Asymptotic Notations Overview
- Big O Notation
- Best, Average, and Worst Cases

# Why Analyze Algorithms?

- Performance measurement independent of hardware and language.
- Predict resource consumption (time & memory).
- Compare different solutions to the same problem.

# Time Complexity

- Definition: Amount of time an algorithm takes to run as a function of the input size (n).
- Not actual time in seconds, but the number of basic operations.
- Crucial for scaling applications to handle large datasets.

# Space Complexity

- Definition: Total amount of memory space required by the algorithm during execution.
- Includes both fixed part (code, constants) and variable part (dynamic variables, stack).
- Auxiliary Space: Extra space or temporary space used.

# Asymptotic Notations

- Mathematical tools to represent time and space complexities.
- Describes the limiting behavior of a function when the argument tends towards a particular value or infinity.
- Key notations: Big-O, Omega (), and Theta ().

# Big 'O' Notation (O)

- Represents the upper bound of the running time.
- Guarantees that the algorithm will not take more than a specific time.
- Focuses on the worst-case scenario.
- Used most commonly in industry.

# Big 'O' Examples

- O(1) - Constant Time (e.g., accessing an array element).
- O(n) - Linear Time (e.g., looping through an array).
- O(n^2) - Quadratic Time (e.g., nested loops).
- O(log n) - Logarithmic Time (e.g., binary search).

# Other Notations: Omega and Theta

- Omega () Notation: Represents the lower bound. The minimum time required.
- Theta () Notation: Represents both upper and lower bounds. The exact asymptotic behavior.
- Useful for tight theoretical analysis.

# Complexity Cases Overview

- Algorithm performance varies based on the specific input.
- We categorize performance into three cases:
- 1. Best Case
- 2. Average Case
- 3. Worst Case

# Best Case Time Complexity

- The minimum time required by an algorithm for a given input size.
- Occurs when the input is optimally arranged for the algorithm.
- Example: Finding the first element in a linear search.

# Average Case Time Complexity

- The expected time required over all possible inputs of a given size.
- Calculated using probabilities of all input permutations.
- Often close to the worst case in many standard algorithms.

# Worst Case Time Complexity

- The maximum time required by an algorithm for a given input size.
- Occurs when the input is arranged in the most disadvantageous way.
- Provides a guaranteed upper bound on performance.

# Summary

- Time and space complexities measure algorithm efficiency.
- Asymptotic notations abstract away hardware-specific details.
- Big O notation focuses on the worst-case upper bound.
- Best, Average, and Worst cases describe performance based on input variation.

# Preview of Next Lecture

- Python Specific Data Structures
- In-depth look at List, Tuple, Set, and Dictionary
- Practical examples and use cases in Python