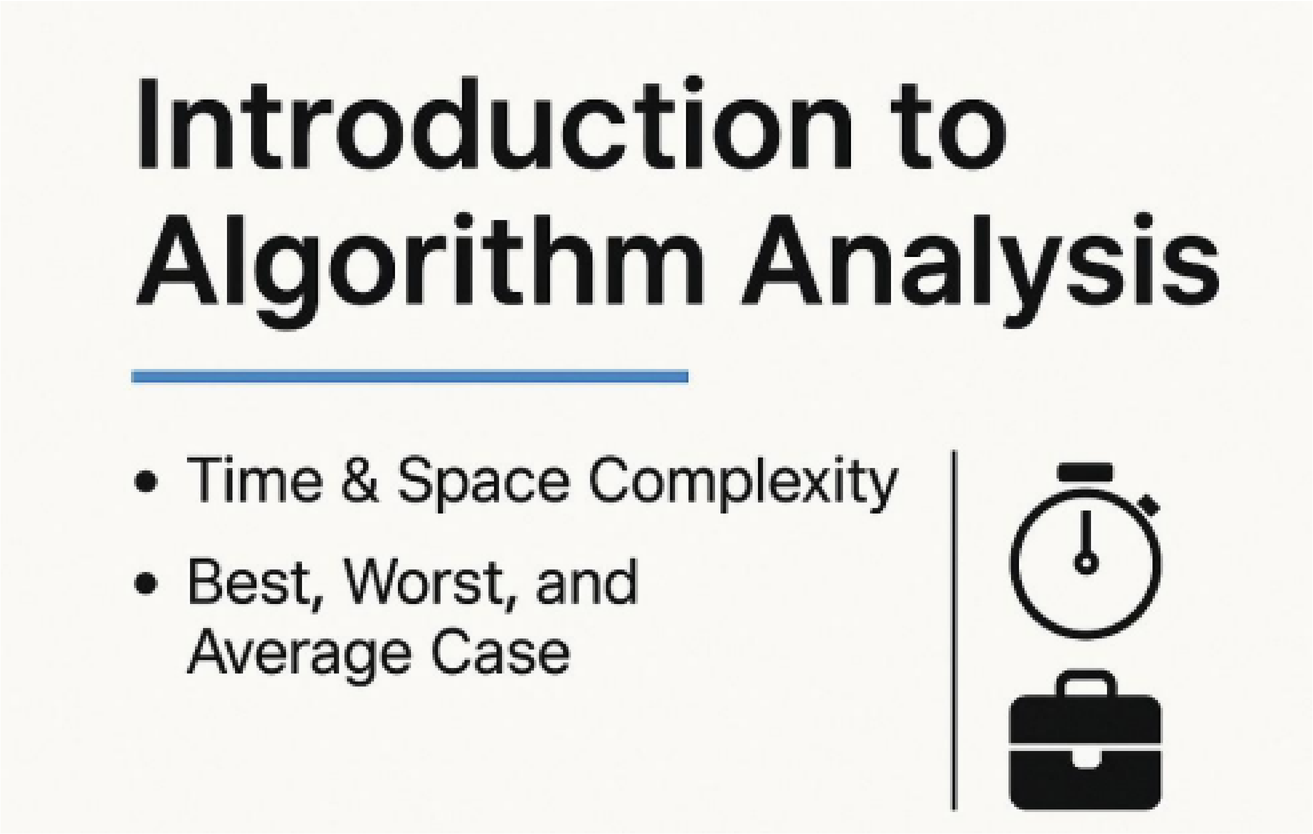
KARNAVATI UNIVERSITY

Course Code: 71203002002

Data Structures & Algorithm

Unitedworld Institute of Technology


Introduction to Algorithm Analysis
• Time & Space Complexity
• Best, Worst, and Average Case

Analysis

Time & Space Complexity | Best, Worst, and Average Case

What is Algorithm Analysis?

Algorithm analysis is the process of determining the amount of **computational resources** (such as **time** and **space**) that an algorithm uses to solve a problem, typically as a function of the input size.

Simple Definition:

"Algorithm analysis is the study of how efficient an algorithm is in terms of time and memory usage as the input size increases."

Purpose of Algorithm Analysis

•To **predict performance** without running the code •To **compare** different algorithms for solving the same problem •To identify **bottlenecks** and inefficiencies
•To guide in choosing the **best algorithm** for a problem

Analytical method to evaluate performance of an

algorithm

- Two main metrics:
  Time Complexity
  Space Complexity
- Helps compare algorithms independent of hardware

What is Time Complexity?

Time complexity refers to the **amount of time** taken by an algorithm to run as a **function of the length of the input**.

Definition:
Time complexity is a theoretical measure that evaluates how the runtime of an algorithm increases as the size of input n increases.

Why Time Complexity Matters?

- Algorithms can be written in many ways to solve the same problem.
- Code execution time depends on:
    1. **Instruction sequence**
    2. **Programming language & syntax**
    3. **Machine (CPU, OS, hardware)**
- Time complexity allows **machine-independent** comparison of algorithms. •

Helps optimize for performance, scalability, and efficiency.

How It Works?

- Each line or block in an algorithm takes time to execute.
- If a line runs once → **Constant Time**
- If it runs inside a loop → **Depends on loop iterations**
- Nested loops → Multiply the complexity

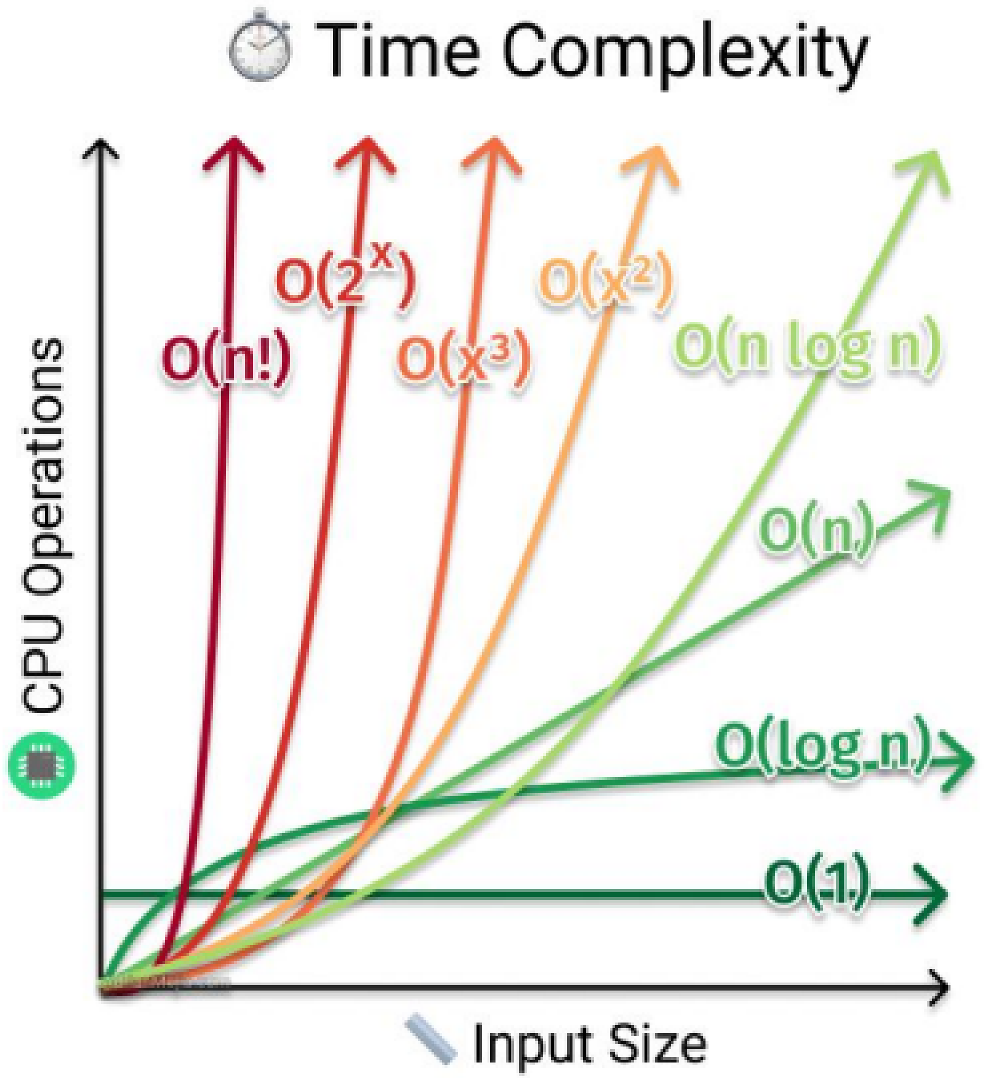
***Note:** Time complexity doesn't measure actual time in seconds, but **how the number of operations grows with input size**.*

KARNAVATI UNIVERSITY Unitedworld Institute of Technology

Common Time Complexity Notations (Big-O)

Notation	Name	Example Use
O(1)	Constant Time	Accessing array element
O(n)	Linear Time	Traversing a list
O(log n)	Logarithmic Time	Binary Search

O(n log n)	Quasilinear Time	Merge Sort, Heap Sort
$O(n^2)$	Quadratic Time	Bubble Sort, nested loops
$O(n^3)$	Cubic Time	3-level nested loops
$O(2^n)$, O(n!)	Exponential, Factorial	Recursion-heavy or brute force algorithms


Time Complexity
O(n!)
O(2^x)
O(x³)
O(x²)
O(n log n)
O(n)
O(log n)
O(1)
CPU Operations
Input Size

Real-World Analogy

•Reading the first page of a book → $O(1)$
•Reading every page of a book → $O(n)$
•Searching a word using an index → $O(\log n)$ •Reading each word on each page → $O(n^2)$

KARNAVATI UNIVERSITY Unitedworld Institute of Technology

What is Space Complexity?

Definition:
Space complexity is the amount of memory required by an algorithm to **complete its execution**, as a function of the **input size** n.

It tells us how efficiently an algorithm uses **memory resources** while solving a problem.

Why is Space Complexity Important?

• Determines whether an algorithm can run on memory-limited systems.
• Critical for large datasets (e.g., big data, embedded systems). • Helps reduce resource consumption and optimize performance.

Unitedworld Institute of Technology

Components of Space Complexity

- **Fixed Part** (independent of input):

  Code, constants, simple variables, compiler overhead.

- **Variable Part** (dependent on input):

1. Input data structures (arrays, maps, lists, etc.)
2. Call stack (for recursion)
3. Auxiliary structures (temporary arrays, hash tables, etc.)

Unitedworld Institute of Technology

Best, Worst, and Average Case

Definitions:

- **Best Case:** Minimum effort (e.g., searching first item)
- **Worst Case:** Maximum effort (e.g., item not found)
- **Average Case:** Expected effort over all inputs

Importance of Algorithm Analysis

- Choose the right algorithm for the right situation
- Avoid inefficient solutions
- Predict and control system performance

Unitedworld Institute of Technology

Suggested Demo / YouTube

Demo Idea:

• Compare Linear Search vs Binary Search in Python •
Plot time taken for increasing array sizes

YouTube Link:
Big-O Notation Explained – CS50 Harvard