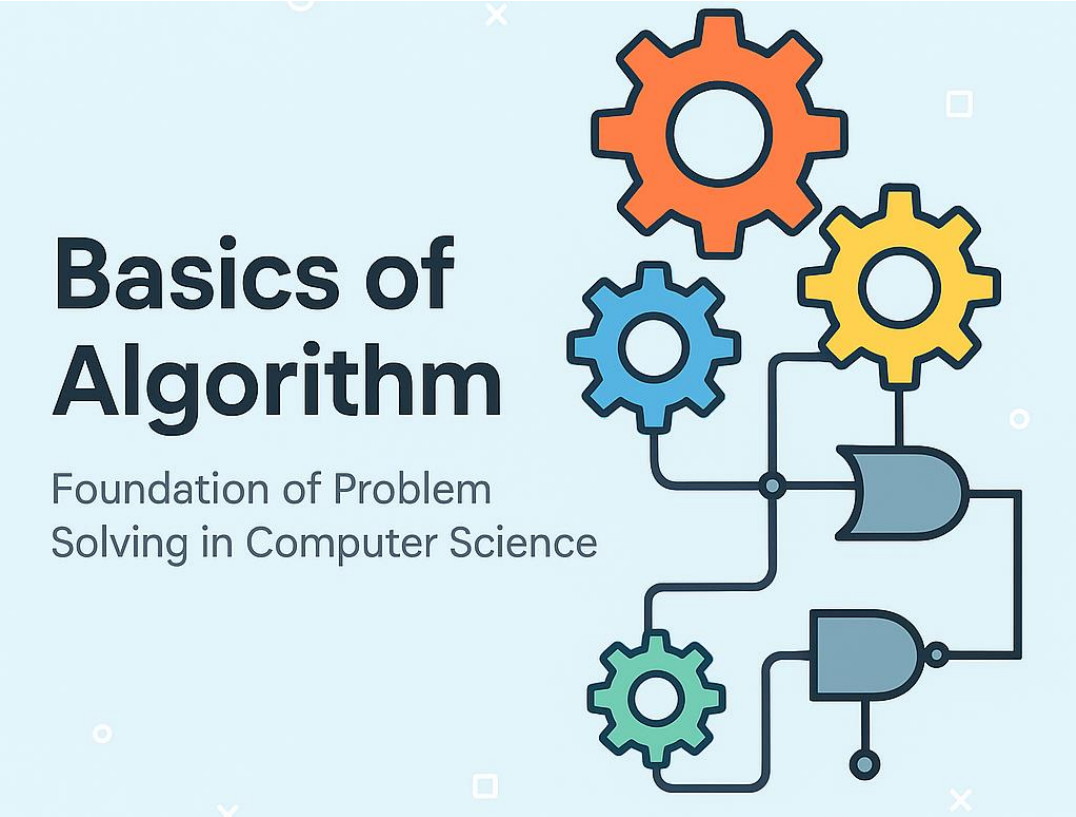
Data Structures & Algorithm

Course Code: 71203002002



What is an Algorithm?

"An algorithm is a finite, ordered sequence of unambiguous instructions that solves a problem or performs a computation."

Breakdown of definition:
Finite: It must terminate after a limited number of steps.
Ordered: Steps follow a logical sequence.
Unambiguous: Instructions are clear and precise.
Effective: It must be practically implementable.

What is an Algorithm?

Example: "Make a cup of tea" is an algorithm:
Step1: Boil water
Step2: Add tea leaves
Step3: Pour water
Step4: Add milk/sugar
Step5: Serve

Importance of Algorithms

Content:

1. Essential for solving problems efficiently
2. Central to computer science, AI, data processing, networking
3. Key to writing optimized code
4. Help improve **speed**, **memory usage**, and **scalability**

Example Use-Cases:

1. Google Search uses ranking and indexing algorithms
2. Navigation apps use path-finding algorithms
3. E-commerce uses recommendation algorithms

Real-Life Algorithms

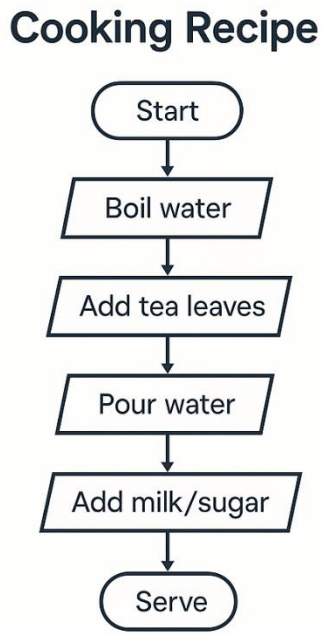
Examples:

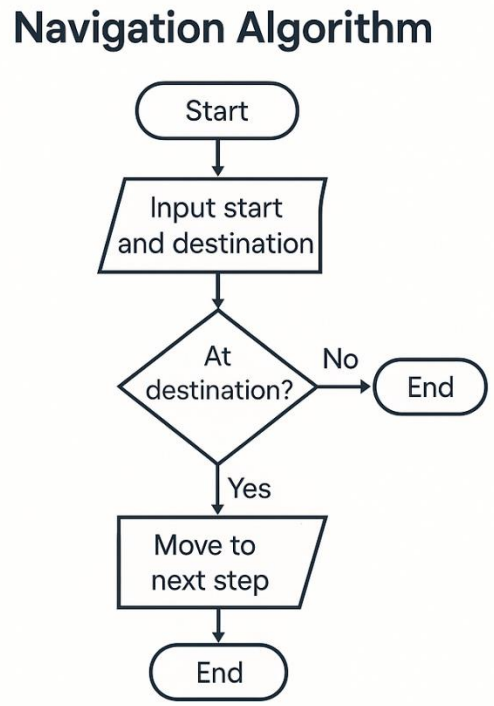
Cooking recipe (sequence of instructions)

Traffic light system (time-based conditional logic)

Boarding an airplane (priority queue system)

Google Maps (shortest path algorithm: Dijkstra's or A*)





Properties of a Valid Algorithm

Use a table or checklist format:

Property	Description
Input	Accepts zero or more inputs
Output	Produces at least one output
Finiteness	Terminates after a finite number of steps
Definiteness	Each step is unambiguous and precise
Effectiveness	Steps can be performed using basic operations

These properties ensure the process is computable, understandable, and solvable. If a procedure lacks one of these, it’s not a valid algorithm.

Algorithm Representation - Pseudocode

Pseudocode Example:

```
Algorithm FindMax(A[1..n]):
1. max ← A[1]
2. for i = 2 to n do
3.     if A[i] > max then
4.         max ← A[i]
5. return max
```

Advantages of Pseudocode:

- No language-specific syntax
- Focuses on logic, not implementation
- Easy to translate into any programming language

Pseudocode is like writing the algorithm in structured English. It lets us focus on the logic before thinking about how to implement it in code.

Algorithm Representation - Flowchart

Start → Input → Initialize max → Loop (i = 2 to n) → Compare A[i] to max → Update max → End

Symbols Used:

Rectangle = Process

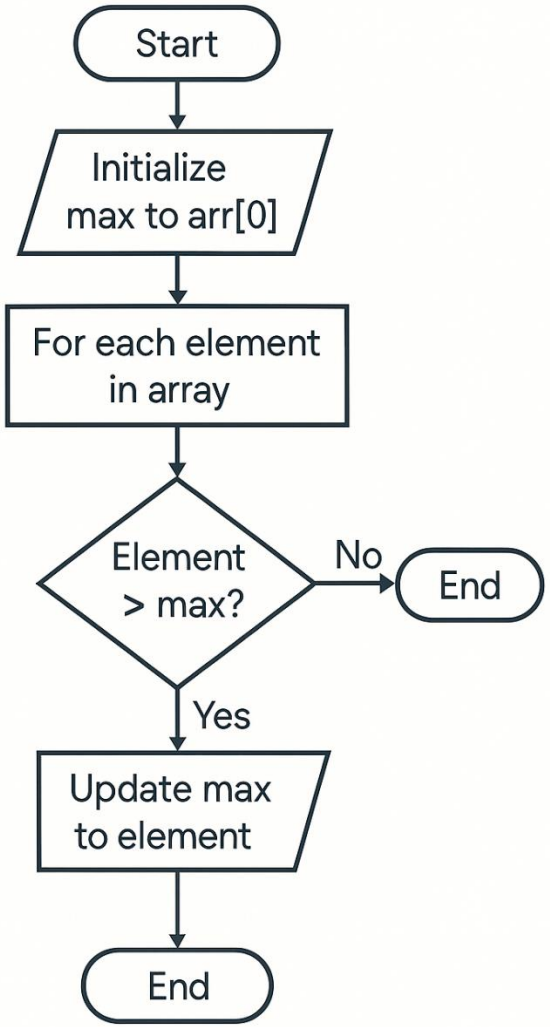
Parallelogram = Input / Output

Diamond = Decision

Arrow = Flow

Flowcharts give a visual structure to logic. They're helpful for planning and debugging, especially when working in teams.

Find Maximum in an Array



Algorithm vs Program

Aspect	Algorithm	Program
Nature	Logical steps	Actual implementation
Syntax rules	No strict rules	Follows language syntax
Execution	Abstract idea	Executes on a machine
Language	Language-independent	Language-specific (e.g., Python, C++)
Output	Describes what to achieve	Actually produces output

An algorithm is the blueprint. A program is the building. The same algorithm can be implemented in multiple programming languages.

Types of Algorithms

By Technique:

Divide and Conquer – e.g., Merge Sort

Greedy – e.g., Huffman Encoding

Dynamic Programming – e.g., Longest Common Subsequence

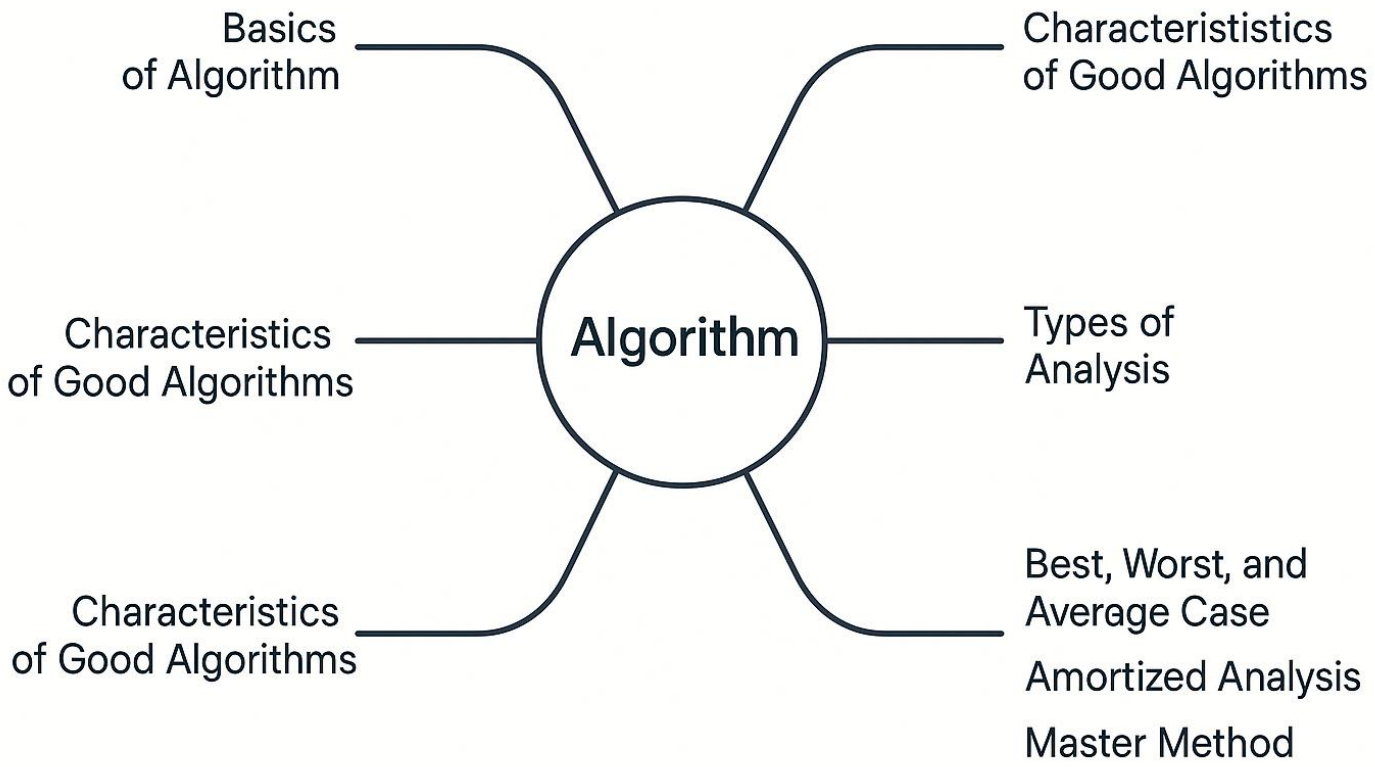
Backtracking – e.g., N-Queens problem

Brute Force – e.g., Checking all permutations

By Purpose:

Sorting (Quick Sort)
Searching (Binary Search)
Graph (Dijkstra)
Cryptography (RSA)
Compression (Huffman)

Classifying algorithms helps us choose the right strategy. Each design paradigm has its strengths depending on the problem type.

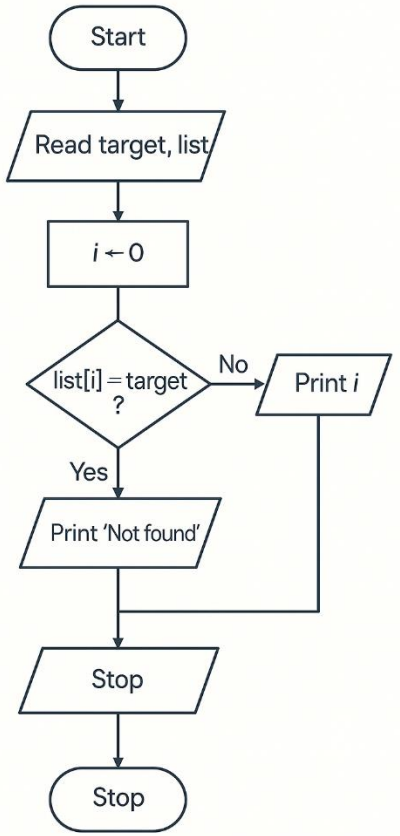

Basics of Algorithm
Characteristics of Good Algorithms
Characteristics of Good Algorithms
Algorithm
Characterististics of Good Algorithms
Types of Analysis
Best, Worst, and Average Case
Amortized Analysis
Master Method

Example: Linear Search Algorithm

Pseudocode:

```
Input: List A of n elements, target value x
1. for i from 0 to n-1
2.     if A[i] == x
3.         return i
4. return -1
```

Flowchart:



Time Complexity: O(n)

Summary & Recap

Bullet Points:

1. Algorithms solve problems in a structured, logical way.
2. Must have input, output, definiteness, finiteness, effectiveness.
3. Represented using pseudocode or flowcharts.
4. Algorithms are not the same as programs.
5. Many categories and real-world applications.