Key Areas of Computer Science

Area	Description
Algorithms and Data Structures	Efficient methods to store, retrieve, and manipulate data
Programming	Writing code to instruct computers to perform tasks
Computer Hardware & Architecture	Understanding how computers physically operate
Operating Systems	Managing hardware and software resources
Databases	Systems to organize and access large amounts of data
Computer Networks	How data is transferred across systems (e.g., Internet)
Software Engineering	Designing, building, and maintaining reliable software
Artificial Intelligence	Creating systems that simulate human intelligence
Cybersecurity	Protecting systems from digital attacks
Human-Computer Interaction (HCI)	Designing user-friendly systems

Why Study Computer Science?

- **Problem Solving**: Build logical thinking and analytical skills
- **Creativity**: Design apps, games, and AI systems
- **Career Opportunities**: High demand across industries
- **Global Impact**: Solve real-world challenges in health, climate, finance, etc.

Foundation Skills Needed

- Logical reasoning
- Basic mathematics
- Interest in technology
- Problem-solving mindset

What is an Algorithm?

An **algorithm** is a **step-by-step set of instructions** used to solve a problem or perform a task.

Example: A recipe for making tea is an algorithm:

- Boil water
- Add tea leaves
- Brew for 3 minutes
- Pour into cup
- Add milk/sugar as needed

Representation of Algorithms

Natural Language: Descriptive steps in plain English

Pseudocode: A language-independent representation

Flowcharts: Diagrammatic representation of logic

Types of Algorithms

Type	Description	Example
Search Algorithm	Finds an element in data	Linear Search, Binary Search
Sort Algorithm	Arranges data in order	Bubble Sort, Merge Sort
Recursive Algorithm	Solves a problem by solving smaller instances of itself	Factorial, Fibonacci
Greedy Algorithm	Makes optimal choice at each step	Coin Change, Dijkstra
Dynamic Programming	Solves problems by combining solutions of subproblems	Knapsack, Fibonacci
Backtracking	Builds solution step-by-step, backtracks if needed	N-Queens, Sudoku
Divide and Conquer	Breaks problem into smaller parts	Merge Sort, Quick Sort

Why Study Algorithms?

- Improves **problem-solving** skills
- Leads to **efficient** programs (faster, less memory)
- Helps in **technical interviews** and real-world system design

Examples of Basic Algorithms

Algorithm to Add Two Numbers

Step 1:START
Step 2: Read two numbers A and B
Step 3: C=A+B
Step 4: Display C
Step 5:END

Find the Maximum of Three Numbers

Step 1: Start
Step 2: Read a, b, c
Step 3: If a > b AND a > c, then
max =a
Else if b > c, then
max = b
Else
max = c
Step 4: Print max
Step 5: Stop

Algorithm to Check Even or Odd

Step 1: Start
Step 2: Read n
Step 3: If n mod 2 = 0, then
    Print "Even"
  Else
    Print "Odd"
Step 4: Stop

Algorithm to Compute Factorial

Step 1: Start
Step 2: Read n
Step 3: fact = 1
Step 4: i = 1
Step 5: while i ≤ n
        fact =fact * i
        i = i + 1
Step 6: Print fact
Step 7: Stop

Sum of First *n* Natural Numbers

Step 1: Start
Step 2: Read n
Step 3: sum = 0
Step 4: i = 1
Step 5: while i ≤ n
sum =sum + i
i = i + 1
Step 6: Print sum
Step 7: Stop

Write an algorithm to check if a given integer is within 10 of 100 or 200.

Step 1: Start
Step 2: Read the integer n
Step 3: If $|n - 100| \leq 10$ OR $|n - 200| \leq 10$
    return True
    Else
        return False
Step 4: Stop

Check Divisibility by 3 or 7

Step 1: Start
Step 2: Read the positive integer n
Step 3: If (n mod 3 == 0) OR (n mod 7 == 0)
    return True
  Else
    return False
Step 4: Stop

Exercises

- Write an algorithm to compute the sum of the two input values. If the two values are the same, then return triple their sum.
- Write an algorithm to read two given integers and check if either of them is in the range 100..200 inclusive.
- Find the product of first n natural numbers
- Find the last digit of a given number.[input:1234 output: 4]