PNS SCHOOL OF ENGINEERING AND TECHNOLOGY

NISHAMANI VIHAR, MARSHAGHAI, KENDRAPARA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

LECTURE NOTE

Semester: 3rd Semester

Subject: ALGORITHM

Prepared By:

Mr. BISWARANJAN SWAIN

HOD DEPT. OF CSE

Unit -I

INTRODUCTION TO ALGORITHMS

Definition:

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

Example analogy: A recipe for baking a cake is an algorithm (list of steps to achieve a goal).

Flowchart:

A visual representation of a process or algorithm using standardized symbols.

Aspect	**Algorithm**	**Flowchart**
Definition	A step-by-step procedure or set of instructions to solve a problem.	A visual representation of a process or algorithm using standardized symbols.
Representation	Written in plain English, pseudocode, or a programming language.	Uses graphical symbols (e.g., ovals, rectangles, diamonds) to depict steps.
Format	Text-based (e.g., numbered steps or code-like structure).	Diagram-based with shapes connected by arrows to show flow.
Ease of Understanding	May require technical knowledge to interpret, especially in pseudocode.	Easier to understand due to visual format, even for non-technical audiences.
Example	Algorithm to find the sum of two numbers: 1. Input two numbers: A, B. 2. Calculate sum = A + B. 3. Output sum.	Flowchart for the same: - Oval (Start) → Rectangle (Input A, B) → Rectangle (Sum = A + B) → Rectangle (Output Sum) → Oval (End).
Use in Programming	Directly translatable to code; serves as the foundation for programming.	Used for planning or documentation, not directly executable.

Both are complementary tools in problem-solving and programming, with algorithms providing the logic and flowcharts offering a visual aid to understand that logic.

Key Characteristics of Algorithms:

 Finiteness

- An algorithm must terminate after a finite number of steps.
- This means it cannot run indefinitely or enter an infinite loop without producing a result.
- Example: An algorithm to calculate the sum of numbers from 1 to 10 will stop after adding the numbers, ensuring a definite end.

 Definiteness

- Each step of the algorithm must be clear, precise, and unambiguous.
- Instructions should be well-defined so that anyone (or a computer) can follow them without confusion.
- Example: Instead of saying "make the number bigger," a definite instruction would be "add 5 to the number."

 Input

- An algorithm may accept zero or more inputs, which are the data it processes to produce a result.
- Inputs are well-defined values provided before or during execution.
- Example: In an algorithm to find the average of numbers, the inputs are the numbers to be averaged.

 Output

- An algorithm must produce at least one output, which is the result of the computation or solution to the problem.
- The output must be related to the input and the problem being solved.
- Example: For an algorithm that sorts a list, the output is the sorted list.

 Effectiveness

- Each step of the algorithm must be basic enough to be executed accurately in a finite amount of time, typically by a human or computer.
- The steps should be simple and feasible, avoiding overly complex or impossible operations.
- Example: An instruction like "multiply two numbers" is effective because it can be performed precisely, whereas "guess the answer" is not.

Importance in Computing:

    - Algorithms are the foundation of programming and problem-solving in computer science.
    - Examples: Sorting data, searching the web, GPS navigation, machine learning.

Algorithm vs. Program:

    - Algorithm: A conceptual idea (e.g., steps to solve a problem).
    - Program: Implementation of an algorithm in a programming language.

Examples of Algorithms.

- Real-World Examples:
    - Following a map to reach a destination.
    - Instructions for assembling furniture.
- Computational Examples:
    - Sorting: Arranging numbers in ascending order (e.g., Bubble Sort).
    - Searching: Finding an item in a list (e.g., Binary Search for a sorted list).

Activity: Writing a Simple Algorithm (20 minutes)Objective: Apply knowledge by creating algorithms.

- Task: Write an algorithm for a simple task (e.g. “How to calculate the average of three numbers”).
    - Example:
      Problem: Find the largest of three numbers (A, B, C).
      Algorithm:
        - Input three numbers: A, B, C.
        - If A > B and A > C, then A is the largest.
        - Else if B > A and B > C, then B is the largest.
        - Else, C is the largest.
        - Output the largest number.
- Quick Recap:
    - What is an algorithm? (Step-by-step solution to a problem.)
    - Why are algorithms important? (They enable efficient problem-solving in computing.)
- Homework/Extension:
    - Research one famous algorithm (e.g., Dijkstra’s algorithm for shortest path) and write a short paragraph about its use.
- Reference: “Introduction to Algorithms” by Cormen et al. (simplified excerpts for advanced students).

Pseudocode:

1. Add Two Numbers

Problem: Add two numbers and display the result.

```
START
```

```
  READ A
  READ B
  SET SUM = A + B
  DISPLAY SUM
END
```

2. Check Even or Odd

Problem: Check if a number is even or odd.

```
START
  READ NUM
  IF NUM MOD 2 == 0 THEN
    DISPLAY "Even"
  ELSE
    DISPLAY "Odd"
  ENDIF
END
```

3. Find the Largest of Three Numbers

```
START
  READ A, B, C
  IF A > B AND A > C THEN
    DISPLAY "A is the largest"
  ELSE IF B > C THEN
    DISPLAY "B is the largest"
  ELSE
    DISPLAY "C is the largest"
  ENDIF
END
```

4. Calculate Factorial of a Number

```
START
  READ N
  SET FACT = 1
  FOR I = 1 TO N
    FACT = FACT * I
  ENDFOR
  DISPLAY FACT
END
```

5. Find Sum of Elements in an Array

```
START
  SET SUM = 0
  FOR I = 1 TO N
    READ A[I]
    SUM = SUM + A[I]
  ENDFOR
  DISPLAY SUM
END
```

6. Pseudocode for Linear Search

```
START
  READ N, ARRAY[1 to N], TARGET
  SET FOUND = FALSE
  FOR I = 1 TO N
    IF ARRAY[I] == TARGET THEN
      SET FOUND = TRUE
      DISPLAY "Element found at position", I
      BREAK
    ENDIF
  ENDFOR
  IF FOUND == FALSE THEN
    DISPLAY "Element not found"
  ENDIF
END
```

Codes in C for the above Examples:

1. Add Two Numbers

```c
#include <stdio.h>
int main() {
    int a, b, sum;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
    sum = a + b;
    printf("Sum = %d\n", sum);
    return 0;
}
```

2. Check Even or Odd

```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    if (num % 2 == 0)
        printf("Even\n");
    else
        printf("Odd\n");
    return 0;
}
```

3. Find the Largest of Three Numbers

```
#include <stdio.h>
int main() {
    int a, b, c;
    printf("Enter three numbers: ");
    scanf("%d %d %d", &a, &b, &c);

    if (a > b && a > c)
        printf("A is the largest\n");
    else if (b > c)
        printf("B is the largest\n");
    else
        printf("C is the largest\n");

    return 0;
}
```

4. Calculate Factorial of a Number

```
#include <stdio.h>
int main() {
    int n, i;
    long long fact = 1;
    printf("Enter a number: ");
    scanf("%d", &n);

    for (i = 1; i <= n; i++) {
        fact *= i;
    }

    printf("Factorial = %lld\n", fact);
    return 0;
}
```

5. Sum of Array Elements

```
#include <stdio.h>
int main() {
    int n, i, sum = 0;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    int arr[n];

    printf("Enter %d numbers:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
        sum += arr[i];
    }

    printf("Sum = %d\n", sum);
    return 0;
}
```

6. Linear Search in an Array

```
#include <stdio.h>
int main() {
    int n, i, target, found = 0;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter element to search: ");
    scanf("%d", &target);

    for (i = 0; i < n; i++) {
        if (arr[i] == target) {
            printf("Element found at position %d\n", i + 1);
            found = 1;
            break;
        }
    }

    if (!found)
        printf("Element not found\n");

    return 0;
}
```

Reference:

- Online algorithm visualizers (e.g., VisuAlgo, Sorting.at) for future lessons.