Introduction to Algorithms

What is an Algorithm?

An **algorithm** is a step-by-step procedure to solve a problem or perform a task. It consists of **finite**, **well-defined**, and **ordered** instructions that take input, process it, and produce an output.

Example (Real-life Algorithm): Making tea

1. Boil water.
2. Add tea leaves.
3. Let it steep.
4. Add sugar/milk if needed.
5. Serve in a cup.

Characteristics of a Good Algorithm

1. **Correctness** – It should solve the problem correctly.
2. **Efficiency** – It should run in a reasonable amount of time.
3. **Clarity** – Easy to understand and implement.
4. **Finiteness** – It must have an end.
5. **Generality** – It should work for multiple inputs.

Algorithms are presented using **pseudocode, Flow Chart** or implemented in a programming language.

Types of Algorithms

1. Sequential Algorithm (Step-by-step Execution)

An algorithm that follows a fixed sequence of steps.

Example: Finding the Sum of Two Numbers

1. take two numbers as input.
2. Add them together.
3. Return the sum

```
Algorithm Sum(a, b)
  return a + b
End Algorithm
```

```
int sum(int a, int b)
{
return a + b;
}
```

2. Iterative Algorithm (Using Loops)

An algorithm that **repeats** a process multiple times using loops.

Example: Finding Factorial of a Number (Iterative)

1. Set the result to 1.
2. Start from 1 and multiply the result by each number up to the given number.
3. When finished, return the result.

```
Algorithm Factorial(n)
    result ← 1
    FOR i ← 1 TO n
            DO result ← result * i
    END FOR
    return result
End Algorithm
```

```
int Factorial(int n) {
  int result = 1;
  for (int i = 1; i <= n; i++)
  {  result *= i }
  return result;
}
```

----

3. Recursive Algorithm (Function Calls Itself)

An algorithm that solves a problem by **calling itself** with a smaller input.

1. If the number is 0 or 1, return 1.
2. Otherwise, multiply the number by the factorial of (number - 1).
3. Keep calling itself until reaching 1.

```
Algorithm Factorial(n)
         IF n = 0 OR n = 1 THEN return 1
         ELSE return n * Factorial(n - 1)
 End Algorithm
```

```
int Factorial(int n)
{
 if (n == 0 || n == 1) return 1; // Base case
return n * Factorial(n - 1); // Recursive call   }
```

Algorithms Examples:

1. Find the Largest Number in a List

Problem: Given a list of numbers, find the largest one.
Pseudocode:

```
Algorithm FindLargestNumber(list)
  Set largest to first element in list
  For each element in list:
    If element is greater than largest:
      Update largest to element
  End For
  Return largest
End Algorithm
```

2. Count the Number of Vowels in a String

Problem: Given a string, count how many vowels (a, e, i, o, u) it contains.
Pseudocode:

```
Algorithm CountVowels(string)
  Set count to 0
  For each letter in string:
    If letter is 'a', 'e', 'i', 'o', or 'u':
      Increase count by 1
  End For
  Return count
End Algorithm
```

3. Reverse a String

Problem: Reverse the order of characters in a given string.
Pseudocode:

```
Algorithm ReverseString(string)
  Set result to empty string
  For each letter from end of string to start:
    Add letter to result
  End For
 Return result
End Algorithm
```