Unit-1: Introduction to Algorithm

Algorithm Definition and its Properties:

Algorithm:

An algorithm is a finite set of well-defined instructions that takes some set of values as *input* and produces some set of values as *output* to solve a problem. An algorithm is thus a sequence of computational steps that transform the input into the output.

Or

A step-by-step problem-solving procedure, especially an established, recursive computational procedure for solving a problem in a finite number of steps.

Key properties of Algorithms include:

Basic Criteria of an Algorithm:

Input: Algorithms take one or more inputs, which are the initial data or values upon which the algorithm operates to produce an output.

Output: Algorithms generate an output based on the provided input and the sequence of steps prescribed by the algorithm. The output represents the solution or result of the problem the algorithm addresses.

Deterministic: Each step in the algorithm must be clear and deterministic, meaning that given the same input, the algorithm will always produce the same output and follow the same set of steps.

Finiteness: An algorithm must have a finite number of steps. It should eventually terminate after executing a finite number of operations, producing the desired result.

Effectiveness: Algorithms must be effective, meaning they must be capable of being carried out in practice using a reasonable amount of time and resources. The steps of the algorithm should be feasible and within the limits of computational capabilities.

Distinct Area of an Algorithm:

1. Design of an Algorithm: This part is used to design a new algorithm for the specific given problem. The different algorithms may be designed for a single problem, but during the design of an algorithm, we must have to follow the basic criteria.

2. Validate of an Algorithm: Validate an algorithm means we have to check that the designed algorithm produces the correct output or not for all legal sets of inputs.

3. Analysis of an Algorithm: In the analysis part we calculate that if for a single problem, there is more than one algorithm designed then we have to analysis that which algorithm takes minimum time and minimum space to produce the output.

4. Testing: It is the process to make a program or algorithm error-free.

Requirement to study algorithm:

Studying algorithms involves understanding the principles and techniques for solving computational problems efficiently and effectively.

Basic Programming Knowledge: Understand the fundamentals of a programming language, data structures, and basic algorithms.

Mathematical Foundation: Familiarize yourself with basic mathematical concepts like discrete mathematics, linear algebra, and calculus. These are important for analyzing and understanding algorithm complexity.

Data Structures: Learn about fundamental data structures such as arrays, linked lists, stacks, queues, trees, graphs, and hash tables. Understand their properties, operations, and when to use them.

Algorithm Design Techniques: Study different algorithm design paradigms such as Divide and Conquer, Dynamic Programming and Greedy Algorithms

Algorithm Analysis: Learn how to analyze algorithms in terms of time and space complexity understand best-case, worst-case, and average-case scenarios.

Algorithm vs. Program

Algorithm:

It is a finite set of well-defined instructions or rules that, when followed, leads to a specific outcome or solution. Algorithms can be expressed in natural language, pseudocode, flowcharts, or any other formal representation.

Program:

A program is a set of instructions written in a specific programming language that, when executed by a computer, performs a particular task or achieves a specific goal. A program is a concrete implementation of an algorithm using the syntax and features of a programming language.

characteristics of programs:

Written in a Programming Language: Programs are written in a programming language such as C++, Python, Java, etc.

Executable: Programs are meant to be executed by a computer to produce the desired output based on the given inputs.

Implements Logic: A program translates the logical steps defined in an algorithm into executable code.

Algorithm Design Techniques:

Algorithm design techniques are fundamental approaches used to create efficient and effective algorithms for solving various computational problems

Divide and Conquer:

- Divide the problem into smaller subproblems of the same type.
- Solve each subproblem independently.
- Combine the solutions of the subproblems to obtain the solution for the original problem.
- Examples: Merge sort, quicksort, Strassen's matrix multiplication.

Greedy Algorithms:

- Make a series of choices, each leading to the locally optimal solution at that step, hoping to find a global optimum.
- At each step, choose the best option without considering the overall situation.
- Examples: Dijkstra's algorithm, Kruskal's algorithm for minimum spanning tree.

Dynamic Programming:

- Break down a complex problem into simpler, overlapping subproblems.
- Solve each subproblem only once and store the solutions in a table to avoid redundant work.
- Combine the solutions of subproblems to solve the original problem.
- Examples: Fibonacci sequence calculation, knapsack problem, longest common subsequence.

Backtracking:

- Build solutions incrementally by making a series of choices.
- If a choice doesn't lead to a valid solution, backtrack and make a different choice.
- Used for optimization, enumeration, and constraint satisfaction problems.
- Examples: N-Queens problem, Sudoku, graph coloring.

Branch and Bound:

- Divide the problem into smaller subproblems and solve them recursively.
- Prune parts of the search space that cannot lead to a better solution than the current best solution.
- Examples: Traveling Salesman Problem (TSP), 0/1 knapsack problem.

Randomized Algorithms:

- Use a random or probabilistic component to make decisions.
- The randomness helps achieve a desired solution with high probability.
- Examples: QuickSort with random pivot selection, randomized primality testing.