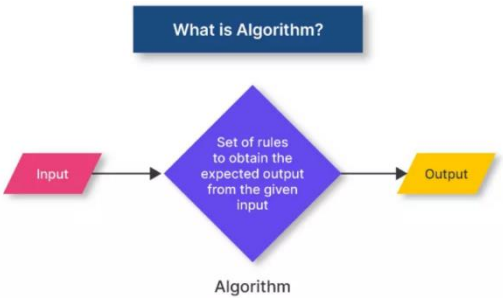
DS Algorithm:

What is an algorithm?



- An algorithm is a step-by-step, well-defined procedure or set of rules designed to perform a specific task or solve a problem.
- it contains the finite set of instructions which are being carried in a specific order to perform the specific task.
- It is not the complete program or code;
- it is just a solution (logic) of a problem, which can be represented either as an informal description using a Flowchart or Pseudocode.

Characteristics of an Algorithm

The following are the characteristics of an algorithm:

- **Input:** An algorithm has some input values. We can pass 0 or some input value to an algorithm.
- **Output:** We will get 1 or more output at the end of an algorithm.
- **Unambiguity:** An algorithm should be unambiguous which means that the instructions in an algorithm should be clear and simple.
- **Finiteness:** An algorithm should have finiteness. Here, finiteness means that the algorithm should contain a limited number of instructions, i.e., the instructions should be countable.
- **Effectiveness:** An algorithm should be effective as each instruction in an algorithm affects the overall process.
- **Language independent:** An algorithm must be language-independent so that the instructions in an algorithm can be implemented in any of the languages with the same output.

Why do we need Algorithms?

- We need **algorithms** because they provide a structured and efficient way to solve problems and perform tasks.

Here are some reasons why algorithms are essential:

1. Problem-Solving

Algorithms break complex problems into manageable steps, offering a clear pathway to find a solution.

2. Efficiency

s ensure tasks are completed in the most optimal way, saving time, memory, and resources.

3. Consistency

Algorithms guarantee predictable and consistent results if the input remains the same.

Let's understand the algorithm through a real-world example. Suppose we want to make a lemon juice, so following are the steps required to make a lemon juice:

Step 1: First, we will cut the lemon into half.

Step 2: Squeeze the lemon as much you can and take out its juice in a container.

Step 3: Add two tablespoon sugar in it.

Step 4: Stir the container until the sugar gets dissolved.

Step 5: When sugar gets dissolved, add some water and ice in it.

Step 6: Store the juice in a fridge for 5 to minutes.

Step 7: Now, it's ready to drink.

Factors of an Algorithm:

1. **Modularity**: Break the problem into smaller, manageable parts.
2. **Correctness**: Produces desired output for given inputs.
3. **Maintainability**: Simple structure for easy updates.
4. **Functionality**: Logical steps solve the problem effectively.
5. **Robustness**: Clearly defines and handles the problem.
6. **User-Friendly**: Easy to explain and implement.
7. **Simplicity**: Easy to understand and follow.
8. **Extensibility**: Adaptable for reuse or integration by others.

Importance of Algorithms

Theoretical importance:

- **Helps us understand the problem, and the approach required to solve it.**

Practical importance:

- **Provides us with a approach that helps us solve the problem practically.**

Approaches of Algorithm

1. **Brute Force Algorithm**:
    - **Optimizing**: Searches all solutions and selects the best one.
    - **Sacrificing**: Stops as soon as the best solution is found.
2. **Divide and Conquer**:
   Breaks the problem into smaller parts, solves them, and combines the results.
3. **Greedy Algorithm**:
   Makes optimal choices at each step, aiming for the best solution; fast but not always accurate.
4. **Dynamic Programming**:
    - Breaks problems into subproblems.
    - Stores intermediate results to avoid recomputation (memoization).
    - Combines subproblem results for efficiency.
5. **Branch and Bound**:
   Splits feasible solutions into subsets and evaluates them to find the best solution (used for integer programming).
6. **Randomized Algorithm**:
   Introduces randomness in inputs to produce varied, efficient, and simple solutions compared to deterministic algorithms.
7. **Backtracking**:
   Recursively tries solutions and removes those that violate constraints.

Major categories of algorithm:

- **Sort:** Algorithm developed for sorting the items in a certain order.
- **Search:** Algorithm developed for searching the items inside a data structure.
- **Delete:** Algorithm developed for deleting the existing element from the data structure.
- **Insert:** Algorithm developed for inserting an item inside a data structure.
- **Update:** Algorithm developed for updating the existing element inside a data structure.

Algorithm Analysis

The algorithm can be analyzed in two levels:

- Prior Analysis: Here, prior analysis is the theoretical analysis of an algorithm which is done before implementing the algorithm. Various factors can be considered before implementing the algorithm. It has no effect on the implementation part.
- Posterior Analysis: Here, posterior analysis is a practical analysis of an algorithm. The practical analysis is achieved by implementing the algorithm using any programming language. This analysis basically evaluates that how much running time and space taken by the algorithm.