A L G O R I T H M S

F O R A B S O L U T E B E G I N N E R S

PART 1

A R U N A R U N I S T O

In programming, an **algorithm** is a set of well-defined instructions or a step-by-step procedure used to solve a specific problem or accomplish a particular task. It is a sequence of precise instructions designed to solve a problem efficiently and accurately.

An algorithm provides a logical and systematic approach to problem-solving by breaking down a complex task into simpler, more manageable steps. It describes the exact order in which these steps should be executed to achieve the desired outcome.

Algorithms are fundamental to computer programming and play a crucial role in software development. They are used to solve a wide range of problems, such as sorting and searching data, graph traversal, pathfinding, encryption, and many more. Good algorithm design is essential for creating efficient and reliable software solutions.

A way of designing algorithm is called **algorithmic strategy**.

An algorithm should have the following characteristics:

- **Input**: Zero or more quantities to be supplied.
- **Output**: At least one quantity is produced.
- **Finiteness**: Algorithms must terminate after a finite number of steps.
- **Definiteness**: All operations should be well-defined. For example, operations involving division by zero or taking a square root for a negative number are unacceptable.
- **Effectiveness**: Every instruction must be carried out effectively.
- **Correctness**: The algorithms should be error-free.
- **Simplicity**: Easy to implement.
- **Unambiguous**: The algorithm should be clear and unambiguous. Each of its steps and their inputs/outputs should be clear and must lead to only one meaning.
- **Feasibility**: This should be feasible with the available resources.
- **Portable**: An algorithm should be generic, independent of any programming language, or an operating system able to handle all ranges of inputs.
- **Independent**: An algorithm should have step-by-step directions, which should be independent of any programming code.

There are various types of algorithms, each designed to solve specific types of problems efficiently. Here are some common types of algorithms:

1. **Searching Algorithms**: Searching algorithms are used to find the presence or location of a particular element within a collection of data. Common searching algorithms include Linear Search, Binary Search, and Hashing-based search algorithms.
2. **Sorting Algorithms**: Sorting algorithms arrange a collection of data elements in a specific order, such as ascending or descending. Examples include Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, and Heap Sort.
3. **Graph Algorithms**: Graph algorithms operate on graphs, which are structures composed of nodes (vertices) and edges. They are used to solve problems related to graph traversal, shortest paths, connectivity, and more.

Examples include Breadth-First Search (BFS), Depth-First Search (DFS), Dijkstra's Algorithm, Bellman-Ford Algorithm, and Kruskal's Algorithm.

4. **Dynamic Algorithms**: Dynamic programming algorithms break down complex problems into simpler overlapping subproblems and solve them in a bottom-up manner. They use memoization or tabulation techniques to efficiently store and reuse computed results. Examples include the Fibonacci sequence, the Knapsack problem, and the Longest Common Subsequence problem.

5. **Greedy Algorithms**: Greedy algorithms make locally optimal choices at each step with the hope of finding a globally optimum solution. They do not backtrack or reconsider previous decisions. Examples include the Greedy Knapsack algorithm, Prim's Algorithm, and Kruskal's Algorithm for minimum spanning trees.

6. **Divide and Conquer Algorithms**: Divide and conquer algorithms divide a problem into smaller subproblems, solve each subproblem independently, and then combine the results to obtain the final solution. Examples include Merge Sort, Quick Sort, and Strassen's Algorithm for matrix multiplication.
7. **Backtracking Algorithms**: Backtracking algorithms explore all possible solutions by incrementally building candidates and undoing or "backtracking" when a solution is found to be invalid. They are useful for solving problems like the N-Queens problem and Sudoku puzzles.
8. **Genetic Algorithms**: Genetic algorithms are inspired by the process of natural selection and evolution. They involve generating a population of candidate solutions, applying genetic operations like mutation and crossover, and iteratively improving the population to find the optimal solution.

9. **Machine Learning Algorithms**: Machine learning algorithms are used in artificial intelligence and data analysis to automatically learn patterns and make predictions or decisions based on training data. Examples include decision trees, support vector machines, neural networks, and clustering algorithms like K-means.

These are just a few examples of the many types of algorithms used in computer science and programming.

In the next part, we will learn each algorithm method by simple coding