classmate
Date
Page

- A K K -

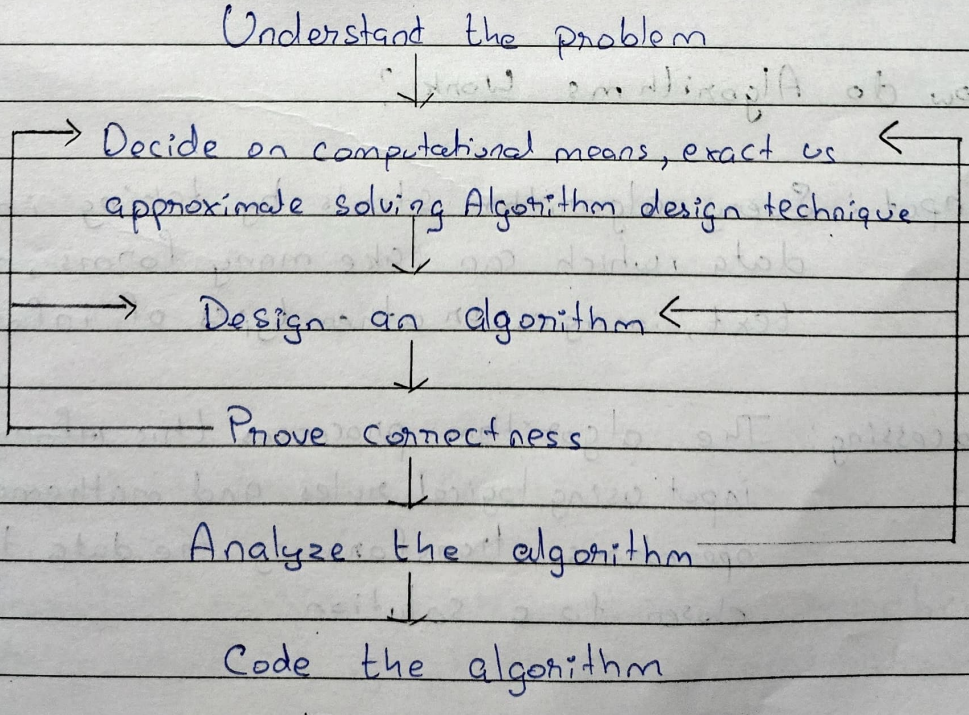
Algorithm

An algorithm is a sequence of unambiguous instructions for solving a problem i.e for obtaining a required output for any legitimate input in a finite amount of time

Characteristics of Algorithm

Input
Output
Finiteness
Definiteness
Correctness

Fundamentals of Algorithmic Problem Solving



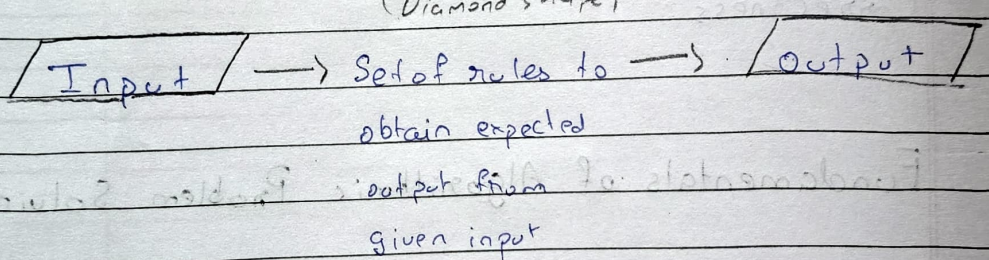
— X —

Q What is an Algorithm?

An algorithm is a step by step procedure to solve a problem or perform a task

The word algorithm means "A set of finite rules" or instructions to be followed in calculations or other problem solving operations"

"A procedure for solving a mathematical problem in a finite number of steps that frequently involves recursive operations". Therefore Algorithm refers to a sequence of finite steps to solve a particular problem



Q How do Algorithms Work?

Input: Every algorithm starts by taking input data, which can take many forms, numbers, text, images or other types of information

Processing: The algorithm processes this ~~information~~ input using logical rules and mathematical operations, transforming the data to move closer to a solution.

Output: After processing, the algorithm produces an output, an answer, a decision or some other meaningful result.

Efficiency: A major goal of any algorithm is efficiency, solving problems while using as few resources as possible

Optimization: Algorithm designers continuously work to optimize algorithms making them faster, smarter and more reliable for real world

Implementation: Finally, algorithms are created through programming languages, enabling computers to execute them and deliver the outcomes.

Fundamentals of Algorithm Problem Solving

These are basic principles used while designing algorithms

- understand the problem
- Define input and output
- Choose a strategy (loop, recursion, divide & conquer)
- Design the algorithm
- Test with examples

Important Problem Types

Important problems such as sorting, searching, string processing, graph problems, combinational problems

numerical problems are basic motivations for designing algorithm

Correctness of Algorithm

It is important for an algorithm to be correct, what does this actually mean? A correct algorithm always produces the expected output or follows the ground truth for the range of valid inputs and eventually terminates.

There are two types of correctness

(1) Partial Correctness

An algorithm is partially correct if it recieves valid input and then terminates.
We can prove the partial correctness of an algorithm through empirical analysis or tests on a few cases

(2) Total Correctness

An algorithm is totally correct if it recieves valid input, terminates and always returns the correct output. we can prove this by formal reasoning on mathematically for instance with a proof by induction.

— A K K — 06/01/2026

Efficiency of algorithm

- Analysis of an algorithm means to find out the resources required by an algorithm
- Resources may be time, space, power consumption, communication, bandwidth, computer Hardware etc
- Running time of an algorithm is measured as function of problem size
- Resource requirement increases with the size of input in some order such as linear, quadratic, logarithmic exponential etc

Q How to decide the best Algorithm ?

- We need to compare performance of all the algorithms and we can find out the best candidate of choice
- Time taken by the algorithm for execution is called **Time Complexity** of an algorithm
- The amount of space required by an algorithm for execution is called the **Space Complexity** of an algorithm
- Space requirement can't be compared directly, so important resource is computational time required by an algorithm
- The **Efficiency** of an algorithm is a measure of amount of resources consumed in solving a problem of size 'n'
- To measure the efficiency of an algorithm, requires to measure its execution time using following approaches
  (1) Emperical approach
  :- To run it and measure how much processer time is needed.

(2) Theoretical approach

:- Mathematically computing How much time is needed as a function of input size

- x -

Analysis of Algorithm

Big O cares only about how fast an algorithm grows with input size not exact time. Common examples:

O(1) → constant growth
O(n) → linear growth
O(n^2) → Quadratic growth
O(log n) → Logarithmic growth

Types of Analysis

(1) Priori Analysis

A Priori Analysis is the analysis of an algorithm's performance before its actual execution based purely on theoretical evaluation. It considers the algorithm's logic, independent of hardware, programming language or compiler, ~~key points~~:

(2) Posteriori Analysis

A posteriori Analysis is the analysis of an algorithms performance after its execution, using actual implementation and experimental results, ~~key points~~