



SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107

Accredited by NAAC-UGC with 'A' Grade

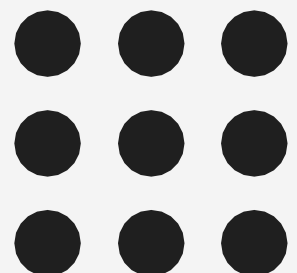
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

**Department of Artificial Intelligence and
Data Science**

**Course Name – Computational Thinking and
Python Programming**

I Year / I Semester

Unit 1-Computational thinking and problem solving





NOTATIONS OF AN ALGORITHM :

Algorithm can be expressed in many different notations, including Natural Language, Pseudo code, flowcharts and programming languages. Natural language tends to be verbose and ambiguous. Pseudocode and flowcharts are represented through structured human language.

A notation is a system of characters, expressions, graphics or symbols designs used among each others in problem solving to represent technical facts, created to facilitate the best result for a program .

Pseudocode:

Pseudocode is an informal high-level description of the operating principle of a computer program or algorithm. It uses the basic structure of a normal programming language, but is intended for human reading rather than machine reading.

It is text based detail design tool. Pseudo means false and code refers to instructions written in programming language.

Pseudocode cannot be compiled nor executed, and there are no real formatting or syntax rules. The pseudocode is written in normal English language which cannot be understood by the computer.

Example:

Pseudocode:

To find sum of two numbers

READ num1,num2

sum=num1+num2

PRINT sum

Basic rules to write pseudocode:

1. Only one statement per line. Statements represents single action is written on same line. For example to read the input, all the inputs must be read using single statement.
2. Capitalized initial keywords The keywords should be written in capital letters. Eg: READ, WRITE, IF, ELSE, ENDIF, WHILE, REPEAT, UNTIL Example: Pseudocode: Find the total and average of three subjects RAED name, department, mark1, mark2, mark3 Total=mark1+mark2+mark3 Average=Total/3 WRITE name, department,mark1, mark2, mark3
3. Indent to show hierarchy Indentation is a process of showing the boundaries of the structure.
4. End multi-line structures Each structure must be ended properly, which provides more clarity.

Example:

Pseudocode: Find greatest of two numbers

READ a, b

IF a>b then

PRINT a is greater

ELSE

PRINT b is greater











ENDIF

5. Keep statements language independent.

Pesudocode must never written or use any syntax of any programming language.

Flowchart:

A graphical representation of an algorithm. Flowcharts is a diagram made up of boxes, diamonds, and other shapes, connected by arrows. Each shape represents a step in process and arrows show the order in which they occur.

Name	Symbol	Description
Process		Process or action step
Flow line		Direction of process flow
Start/ terminator		Start or end point of process flow
Decision		Represents a decision making point
Connector		Inspection point
Inventory		Raw material storage
Inventory		Finished goods storage
Preparation		Initial setup and other preparation steps before start of process flow
Alternate process		Shows a flow which is an alternative to normal flow
Flow line(dashed)		Alternate flow direction of information flow



Rules for drawing flowchart

In drawing a proper flowchart, all necessary requirements should be listed out in logical order.

The flow chart should be clear, neat and easy to follow. There should not be any room for ambiguity in understanding the flowchart.

The usual directions of the flow of a procedure or system is from left to right or top to bottom.

Only one flow line should come out from a process symbol.

Only one flow line should enter a decision symbol, but two or three flow lines, one for each possible answer, can leave the decision symbol.

Only one flow line is used in conjunction with terminal symbol.

If flowchart becomes complex, it is better to use connector symbols to reduce the number of flow lines. Ensure that flowchart has logical start and stop.

Algorithmic problem solving is solving problem that require algorithm for the solution the formulation of an algorithm for the solution.

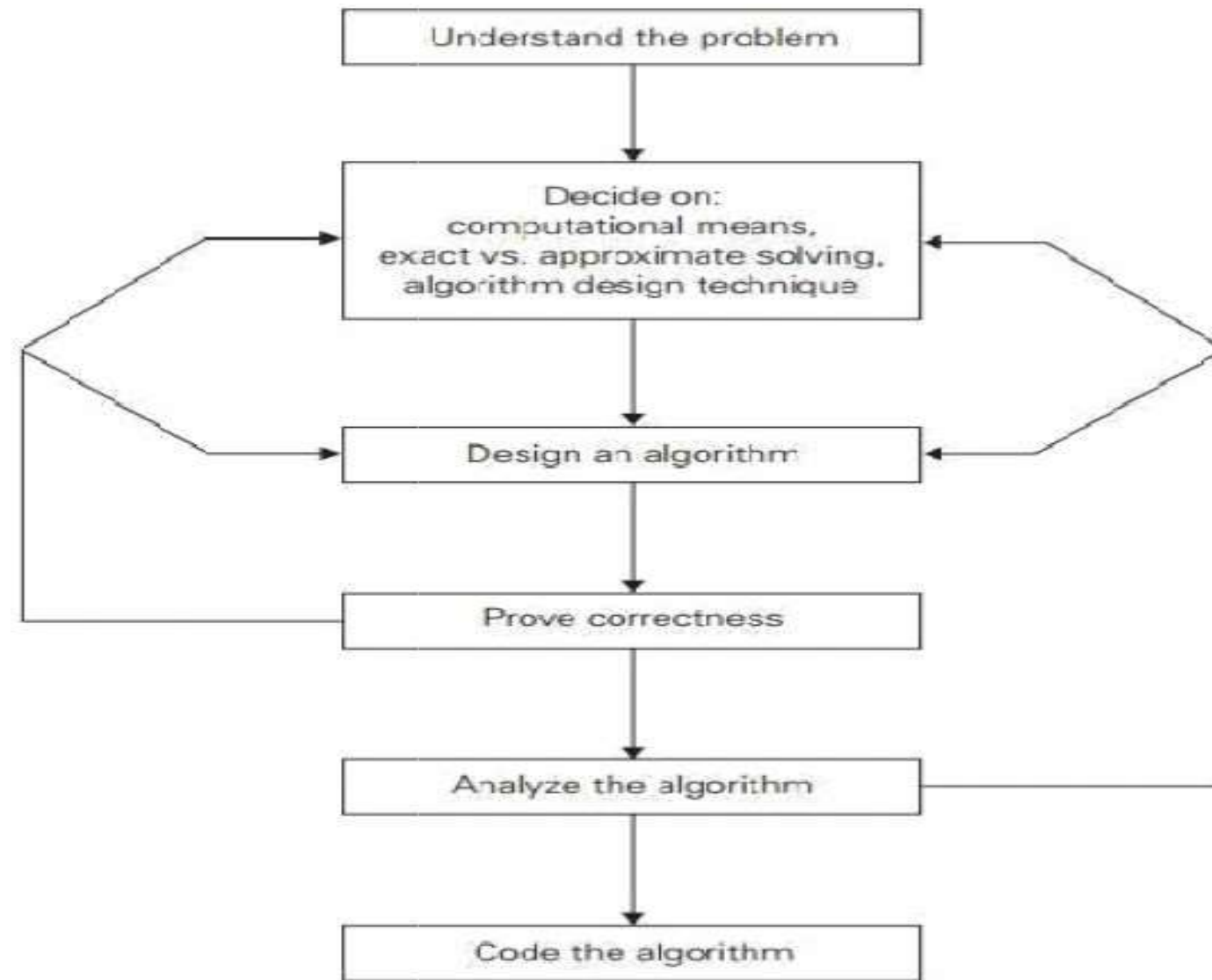


FIGURE 1.2 Algorithm design and analysis process.



Understanding the Problem

- It is the process of finding the input of the problem that the algorithm solves.
- It is very important to specify exactly the set of inputs the algorithm needs to handle.
- A correct algorithm is not one that works most of the time, but one that works Correctly for *all* legitimate inputs.

Ascertaining the Capabilities of the Computational Device

If the instructions are executed one after another, it is called sequential algorithm

Choosing between Exact and Approximate Problem Solving

The next principal decision is to choose between solving the problem exactly or solving it approximately.

Based on this, the algorithms are classified as exact *algorithm* and *approximation algorithm*.

Data structure plays a vital role in designing and analysis the algorithms.

Some of the algorithm design techniques also depend on the structuring data specifying a problem's instance

Algorithm+ Data structure=programs.

Algorithm Design Techniques

An *algorithm design technique* (or “strategy” or “paradigm”) is a general approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing.

Learning these techniques is of utmost importance for the following reasons.

First, they provide guidance for designing algorithms for new problems, Second, algorithms are the cornerstone of computer science.



Methods of Specifying an Algorithm

Pseudocode is a mixture of a natural language and programming language-like constructs. Pseudocode is usually more precise than natural language, and its usage often yields more succinct algorithm descriptions. In the earlier days of computing, the dominant vehicle for specifying algorithms was a *flowchart*, a method of expressing an algorithm by a collection of connected geometric shapes containing descriptions of the algorithm's steps.

Programming language can be fed into an electronic computer directly. Instead, it needs to be converted into a computer program written in a particular computer language. We can look at such a program as yet another way of specifying the algorithm, although it is preferable to consider it as the algorithm's implementation.

Once an algorithm has been specified, you have to prove its *correctness*. That is, you have to prove that the algorithm yields a required result for every legitimate input in a finite amount of time.

A common technique for proving correctness is to use mathematical induction because an algorithm's iterations provide a natural sequence of steps needed for such proofs.

It might be worth mentioning that although tracing the algorithm's performance for a few specific inputs can be a very worthwhile activity, it cannot prove the algorithm's correctness conclusively. But in order to show that an algorithm is incorrect, you need just one instance of its input for which the algorithm fails.



Analyzing an Algorithm

Efficiency.

Time efficiency: indicating how fast the algorithm runs,

Space efficiency: indicating how much extra memory it uses

simplicity.

An algorithm should be precisely defined and investigated with mathematical expressions.

Simpler algorithms are easier to understand and easier to program.

Simple algorithms usually contain fewer bugs.

Coding an Algorithm

Most algorithms are destined to be ultimately implemented as computer programs. Programming an algorithm presents both a peril and an opportunity.

A working program provides an additional opportunity in allowing an empirical analysis of the underlying algorithm.

Such an analysis is based on timing the program on several inputs and then analyzing the results obtained.