SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAMMING FOR PROBLEM SOLVING

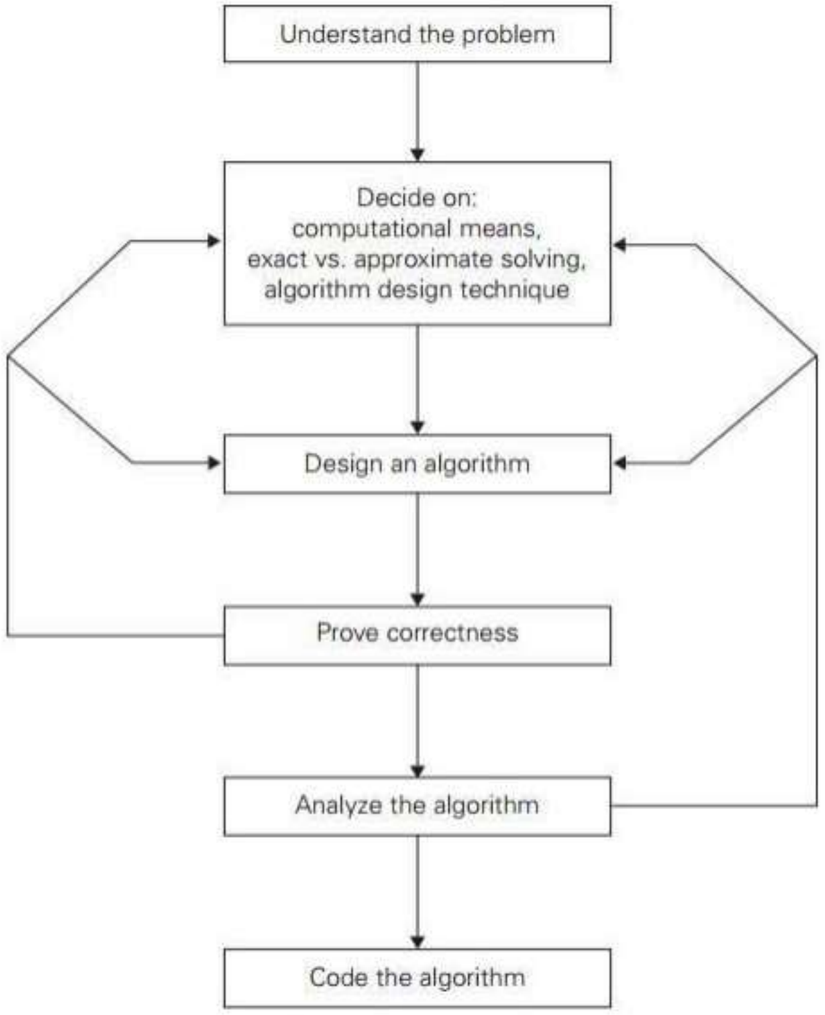
I YEAR - I SEM

UNIT 1 – Introduction to Problem Solving Techniques

TOPIC 5 – Algorithmic Problem Solving

ALGORITHMIC PROBLEM SOLVING

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



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**
  - If the instructions are executed concurrently, it is called **parallel 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.**
- Deciding a data structure:
  - 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:**
  - 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 success in 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:**
  - 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.

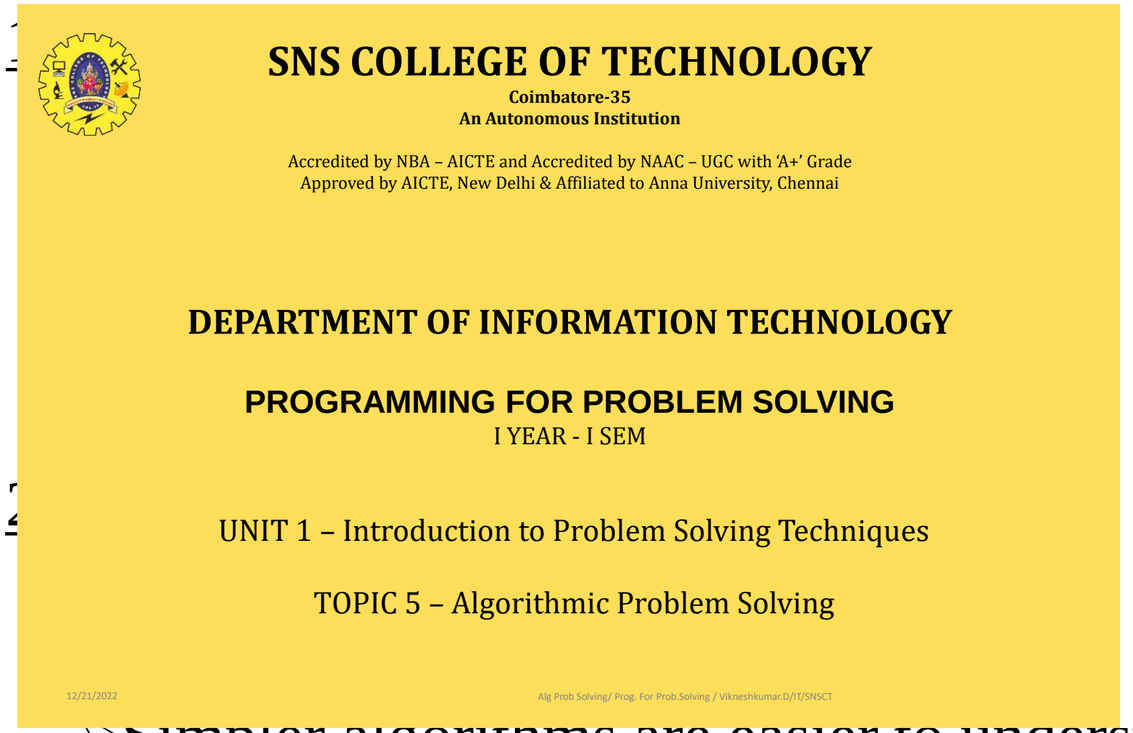
PROVING AN ALGORITHM'S CORRECTNESS

- 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.

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 (Pattern and Observations) of the underlying algorithm.
- Such an analysis is based on timing the program on several inputs and then analyzing the results obtained.

ANALYZING AN ALGORITHM



investigated with

- Simpler algorithms are easier to understand and easier to program.
- Simple algorithms usually contain fewer bugs.