SNS COLLEGE OF ENGINEERING
Kurumbapalayam (Po), Coimbatore – 641 107

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 COMPUTER SCIENCE AND ENGINEERING

19IT103 – COMPUTATIONAL THINKING AND PYTHON PROGRAMMING

❖ **A readable, dynamic, pleasant, flexible, fast and powerful language**

Recap:

- Notations (pseudocode, flow chart, programming language).
- Flowcharts are a graphical means of representing an algorithm
- Flowchart is a diagrammatic representation of sequence of logical steps of a program.
- A programming language is a formal language that specifies a set of instructions that can be used to produce various kinds of output.
- Programming languages generally consist of instructions for a computer.
- Eg : C, C++, COBAL, JAVA, Python ... Etc

1.6 Algorithmic problem solving:

- An algorithm is **a defined set of step-by-step procedures** that provides the correct answer to a particular problem.

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

- The formulation of algorithm is always been an important element of problem solving.

- We can consider algorithms to be procedural solutions to problems.

1.6 Algorithmic problem solving:

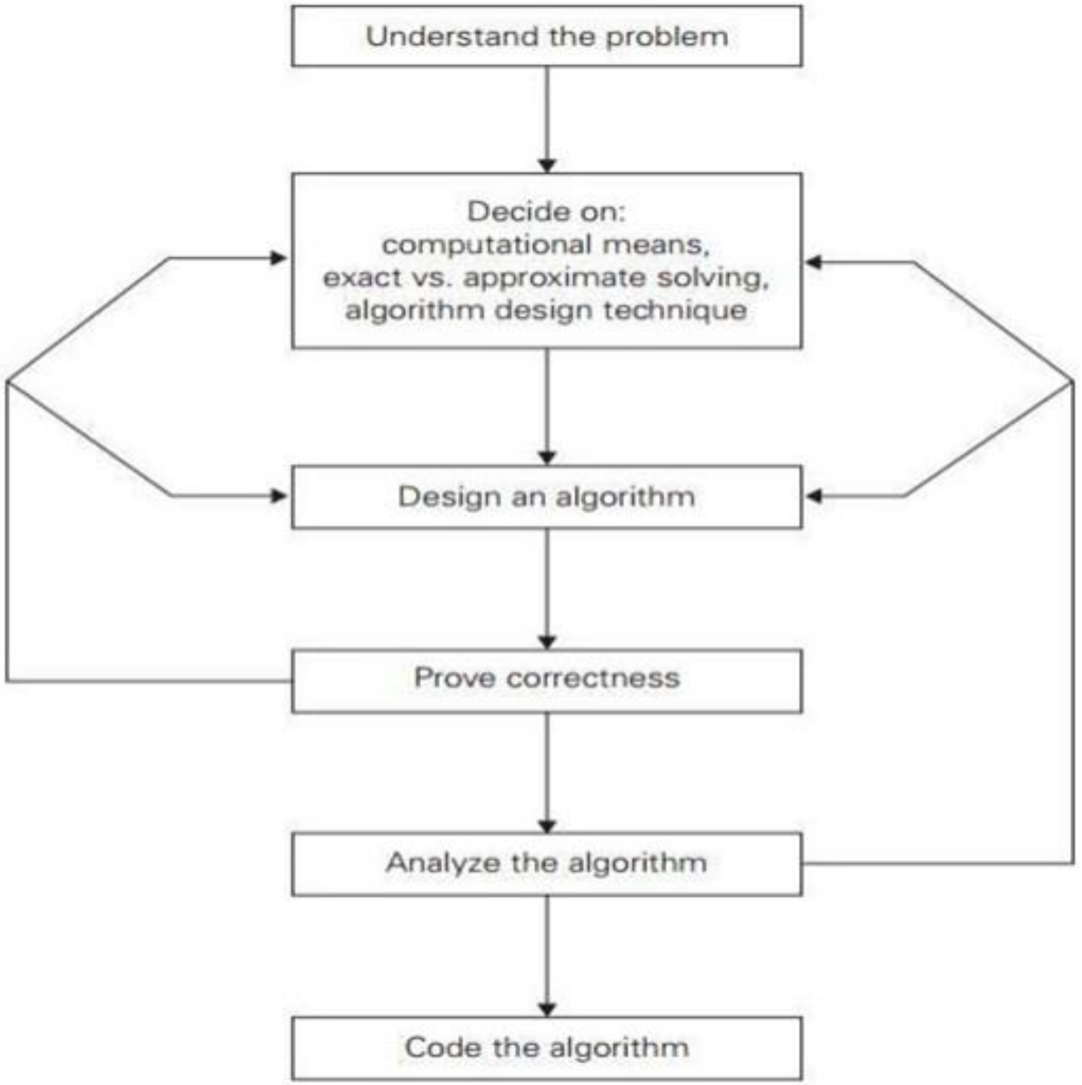

Figure 1: Algorithm design and analysis process

1.6 Algorithmic problem solving:

The fundamental steps are:

- Understanding the problem
- Ascertaining the capabilities of computational device
- Choose between exact and approximate problem solving
- Decide on appropriate data structures
- Algorithm design techniques
- Methods for specifying the algorithm
- Proving an algorithm’s correctness
- Analyzing an algorithm
- Coding an algorithm

1.6 Algorithmic problem solving:

1. **Understanding the problem :**

- The first thing we need to do before designing an algorithm is **to understand completely the problem given**.

- **Read the problem's description carefully and ask questions if you have any doubts about the problem.**

- An input to an algorithm specifies an instance of the problem the algorithm solves.

1.6 Algorithmic problem solving:

1. Understanding the problem :

- It is very important to **specify exactly the range of instances** the algorithm needs to handle.

- **Correct algorithm** is not one that works most of the time, but **one that works correctly for all legitimate inputs.**

- Do not skip on this first step of algorithmic problem-solving process; if we do, then we need to do unnecessary rework on it.

1.6 Algorithmic problem solving:

2. Ascertain the capabilities of computational device :

- Once you completely understand a problem, you need to ascertain the capabilities of the computational device the algorithm is intended for.
- If the instructions are executed one after another, one operation at a time. Algorithms designed to be executed on such machines are called ***sequential algorithm.***
- If the instructions are executed concurrently, it is called ***parallel algorithm.***

1.6 Algorithmic problem solving:

3. Choose between exact and approximate problem solving :

- Next principal decision is to Choose between solving the problem exactly or solving the problem approximately.
- Case 1: solving the problem exactly – an algorithm is called ***exact algorithm***
- Case 2: solving the problem approximately – an algorithm is called ***approximation algorithm.***
- First, some important problems cannot be solved exactly for most of their instances; example – extracting square roots solving nonlinear equations.
- Second, available algorithm for solving a problem exactly can be unacceptably slow because of the problem's intrinsic complexity

1.6 Algorithmic problem solving:

4. Decide on appropriate data structures :

- Data structure plays a vital role in designing and analysis the algorithms.
- Some of the algorithm design techniques also depend on the structuring or restructuring data specifying a problem’s instance.
- **Algorithm+ Data structure=programs.**

1.6 Algorithmic problem solving:

5. 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 atmost importance for the following reasons:
  - First, they provide guidance for designing algorithms for new problems, ex : problems for which there is no known satisfactory algorithm.
  - Second, algorithms are the cornerstone of computer science.
- Algorithm design techniques make it possible to classify algorithms according to an underlying design idea.

THANK YOU