بِسۡمِ اللّٰہِ الرَّحۡمٰنِ الرَّحِیۡمِ

# CONTENTS

# PROBLEM SOLVING AND ALGORITHM DESIGNING

Unit 1


- Define the term problem
- Evaluate a problem in order to find out its best solution
- Design a strategy for the solution of problem
- Find feasible solutions of a problem

## 1.1 PROBLEM SOLVING

Problem solving is the process of finding solutions of difficult or complex issues. It is the process by which any kind of problem is solved.

Solving problems is the core feature of computers. A computer is not intelligent. It cannot analyze a problem and come up with a solution. A human (the programmer) must analyze the problem, develop the instructions for solving the problem, and then make the computer carry out the instructions. The major responsibility of a programmer is to provide solution of the problems by using computers. It will be easier if computer science students first understand how a human solves a problem, then understand how to translate this solution into something a computer can understand, and finally how to "write" the specific steps to get the job done. Remember, in some cases a machine will solve a problem in a completely different way than a human.

### What is Problem

A problem is a situation preventing something from being achieved. A problem can be a task, a situation or any other thing. In simple language, a problem is a question which requires an answer or a solution. In any case, a problem is considered to be a matter which is difficult to solve or settle, a doubtful case, or a complex task involving doubt and uncertainty.

## Plan the solution of Problem

Problems can be solved with the help of computers but for this programmer has to plan and strategize tasks that lead to the solution of the problem.

## Problem Solving Strategies

A very important aspect of problem-solving is developing good strategies. There are many strategies for solving a problem. A strategy is an approach (or a series of approaches) created to solve a computational problem. Strategies are designed according to the nature of the problem. Strategies are flexible and show various steps to reach the solution. A strategy in itself might produce incorrect results, but an algorithm based on such strategy will always produce correct results.

For solving any problem, it is important to know what the problem really is and what how it should actually be if there was no such problem. The question about end result helps you to find the gap and create a strategy for solution.

## Problem Solving Process

Problem solving is a step-by-step process. There are four basic step involved in finding a solution for a problem as given below:

1. Define the problem.
2. Generate alternative solutions.
3. Evaluate and select an alternative.
4. Implement and follow up on the solution.

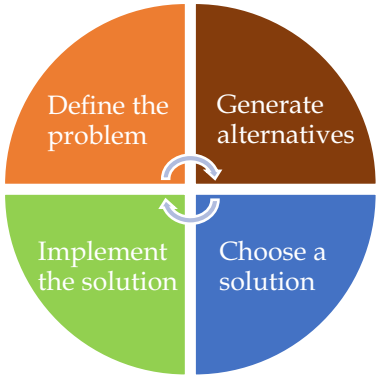



**Fig. 1.1: Problem Solving**

### 1. Define the problem

In problem solving process, the first step is defining or identifying the problem. It is the most difficult and the most important of all the steps. It involves diagnosing the situation so that the focus should be on the real problem and not on its symptoms. During this first stage of problem solving, it is important to describe the problem. A well- described problem will also help others to understand the problem.

## 2. Generate alternative solutions

For any problem, there are more solutions to it than the one that is thought of first. Postpone the selection of one solution until several problem-solving alternatives have been proposed. Considering multiple alternatives can significantly enhance the value of your ideal solution. So, it is best to develop a list of all feasible solutions that can be assessed and decide which one will be the best for the particular problem. Thinking and team problem-solving techniques are both useful tools at this stage of problem solving.

## 3. Evaluate and select an alternative

Many alternative solutions to the problem should be generated before final evaluation. A common mistake in problem solving is that alternatives are evaluated as they are proposed, so the first acceptable solution is chosen, even if it is not the best solution. If we just focus on trying to get the results we want, we can miss the chances for learning something new which is required for real improvement in the problem-solving process.

## 4. Implement and follow up on the solution

The plan for best solution also includes planning on what happens next if something goes wrong with the solution if it does not work out the way it was required. When the best solution is implemented, it is important to track and measure the results to be able to answer questions such as: Did it work? Was this a good solution? Did we learn something here in the implementation that we could apply to other potential problems?

When choosing most appropriate solution, the problem solver should consider about the possible impacts of that solution. For example, will this solution solve the current problem without creating new ones or if this solution is acceptable by everyone involved in this situation or if the solution is within the budget and achievable within a given time.


- Define term algorithm
- Discuss the importance of algorithm in problem solving

## 1.2 ALGORITHM

Algorithms are widely used throughout all areas of Information Technology (IT). A search engine algorithm, for example, takes search strings of keywords and operators as input, searches its associated database for relevant web pages, and returns results. Another common example is online advertisements where the algorithm takes age, gender, region and interests as input and then displays ads to only those people who match the criteria.

### 1.2.1 Definition

An algorithm is a set of instructions/steps/rules that are followed to solve a problem. It is a tool for solving a well-specified computational problem. In our daily lives as well, we follow various algorithms without knowing, for example:

- We follow a daily routine after waking up in the morning
- We follow a set of instructions while driving a car
- We follow recipes in cooking food or making tea

These step-by-step instructions that we follow everyday are algorithms to solve certain problems. There are two common methods to express algorithm designs; pseudocode and flowcharts.

### 1.2.2 Role of algorithm in problem solving

The advantage of using an algorithm to solve a problem or make a decision is that it produces the best possible answer every time. This is useful in solutions where accuracy is required or where similar problems need to be solved more often. In many cases, computer programs can be developed with the help of this process. Then data is entered in that program so that the algorithm can be executed to come up with the required solution.

A computer programmer has to design various algorithms to create a program. These algorithms can vary from retrieving input from a user to

computing complex formulas to reach a conclusion. This data is then rearranged into a meaningful way to present to the user. The user then makes decisions based on the presented data.

### Qualities of Good Algorithms

- Input and output should be defined precisely.
- Each step in the algorithm should be clear and unambiguous.
- Algorithms are supposed to be most effective among many different ways to solve a problem.
- An algorithm should not include computer code. Instead, the algorithm should be written in such a way that it can be used in different programming languages.


- Design algorithm to find sum, average, volume, percentage and others.

## 1.2.3 Algorithm Examples

Following are few examples of simple algorithm.

### Algorithm 1: Making a cup of tea

- **Step 1:** Start
- **Step 2:** Place the fresh water in a pot or a kettle.
- **Step 3:** Boil the water.
- **Step 4:** Put the black tea leaves in that pot.
- **Step 5:** After that add some milk into that pot.
- **Step 6:** Add some sugar.
- **Step 7:** Boil for some time.
- **Step 8: Stop**

### Algorithm 2: Sum of two numbers

- **Step 1:** Start
- **Step 2:** Declare variables num1, num2 and sum.
- **Step 3:** Read values num1 and num2.
- **Step 4:** Add num1 and num2 and assign the result to sum.
- sum = num1 + num2
- **Step 5:** Display sum
- **Step 6:** Stop

### Algorithm 3: Average of three numbers

- **Step 1:** Start
- **Step 2:** Declare variables num1, num2, num3 and avg.
- **Step 3:** Read values num1, num2 and num3.
- **Step 4:** Apply formula {Average = Sum / No. of values}
- avg = (num1 + num2 + num3) / 3
- **Step 5:** Display avg
- **Step 6:** Stop

### Algorithm 4: Volume of a box

- **Step 1:** Start
- **Step 2:** Declare variables length, width, height and volume.
- **Step 3:** Read values length, width and height.
- **Step 4:** Apply formula {Volume = length x width x height}
- volume = length x width x height
- **Step 5:** Display volume
- **Step 6:** Stop

### Algorithm 5: Percent Calculate

- **Step 1:** Start
- **Step 2:** Declare variables part, total and percentage.
- **Step 3:** Read values part and total.
- **Step 4:** Apply formula {Percentage = (part / total) × 100}
  percentage = (part / total) x 100
- **Step 5:** Display percentage
- **Step 6:** Stop


- Define the flowchart
- Identify the different symbols used in flowchart designing
- Discuss the importance of flowchart in problem solving
- Design flowchart for any problem by using various flowchart symbols
- Differentiate between algorithm and flowchart

## 1.3 FLOWCHART

The flowchart is the physical representation of problem solving process. It is used to show the sequence of steps and logic of solution for a problem, before writing the full computer program. It also helps in communicating the steps of the solution to others by using different symbols.

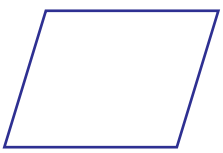
### 1.3.1 Definition

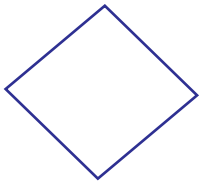
It is a general-purpose tool used to define the sequence of different types of processes or operations in information system or program. It shows processes and their flow visually using diagram. It describes graphically different steps of programs or any operation and their sequence or flow using different symbols. Information system flowcharts show flow of data from source documents to final distribution to users. Program flowcharts show the sequence of steps or instructions in a single program or subroutine. It is diagrammatic or graphical representation of algorithm and converts word of algorithm into symbols. The flowchart shows different steps with the help of

different shapes and their sequence. The processes are connected with directed lines or arrows which show the path from one procedure step to the next.

### 1.3.2 Flowchart Symbols

Flowchart is made up of different symbols to represent or show program and its flow. Some of them are as follows:

| S No. | Name | Symbol | Description |
|---|---|---|---|
| 1. | **Start/Stop** |   | It is oval shape and is used to show the start and end of a program or flowchart sequence. |
| 2. | **Arrows** |  | The arrow shape shows direction of flow from one step or box to another. |
| 3. | **Process** |  | This rectangle shape indicates any type of internal operation or process usually one step. The step is written inside the box. Only one arrow goes out of the box. |
| 4. | **Input/output** |  | This parallelogram shape shows the input or output process. It is used for any Input / Output (I/O) operation and indicates that the computer is to obtain data or output results. |

| | | | |
|---|---|---|---|
| 5. | **Decision/ Condition** |  | This diamond shape shows decision based on a condition written in the diamond. Two arrows go out of the diamond. One directs toward path if condition is true and other for false. We can also use yes/no in place of true /false. |

| | |
|---|---|
| **Teachers Note**  | Teacher should tell students about other symbols like alternate symbol for start/end, connecter symbol, etc. |

### 1.3.3 Importance of a Flowchart for Solving a Problem

Main use of a flowchart is to visualize operations and sequence of steps to perform them. It illustrates the logic to solve a problem, before writing the full computer program. It shows all steps visually or graphically which is easy to understand in one look and also helps to describe program flow to others. It also helps programmers when we modify or extend program. Following are some advantages of flowchart for solving problem

- Flowcharts help in communicating the logic of a program to all others and make it easy to understand.
- It is a useful program document that is needed for various purposes like to know about program quickly or to modify program logic.
- The flowcharts act as a guide or blueprint during the coding of program.

## A Sample Flowchart

It takes three numbers as input, calculates sum and percentage. If percentage is above 70 then prints "Well done", otherwise "Work hard".

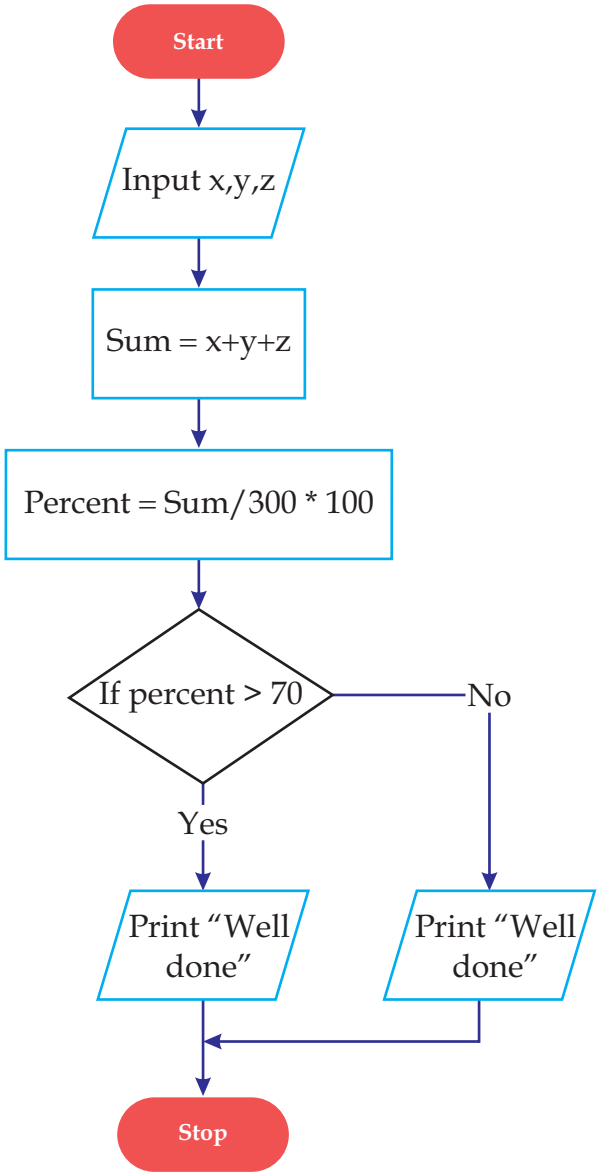



Fig. 1.2: Sample Flowchart

### 1.3.4 Difference between Algorithm and Flowchart

| S.No. | Algorithm | Flowchart |
|---|---|---|
| 1 | Algorithm is step by step solution of a problem | Flowchart is a diagram of different shapes which shows flow of data through processing system. |
| 2 | In algorithm text is used. | In flowchart, symbols or shapes are used |
| 3 | Algorithm is easy to debug. | Flowchart is difficult to debug |
| 4 | Algorithm is difficult to write and understand. | Flowchart is easy to construct and understand. |
| 5 | Algorithm does not follow any rules. | Flowchart follows rules for its construction. |
| 6 | Algorithm is the pseudo code of the program. | Flowchart is just graphical or visual representation of program logic. |


- Define Linear data types
  - Stack, Queue, Array
- Define Non-Linear data types
  - Tree & Graph

## 1.4 DATA STRUCTURE

A **data structure** is a particular way of organizing data in a computer to use it effectively. For example, array data structure is used to store a list of items having the same data-type. Data structure may be linear or non-linear.

### 1.4.1 Linear Data structures

In Linear Data Structure data elements are arranged in sequential order and each of the elements is connected to its previous and next element. This structure helps to convert a linear data structure in a single level and in single run. They are easy to implement as computer memory is also in a sequential form. Linear data structures are not efficient in memory utilization. Examples of linear data structures are Stack, Queue, Array etc.

### (a) Stack

Stack is a linear data structure which follows a particular order to perform different operations. Items may be added or removed only at the top of stack. The order may be LIFO (Last In First Out) or FILO (First In Last Out). The data which is placed first is removed in last and which is placed last is removed first. We cannot remove data from the bottom or middle. Examples of a stack are plates that are stacked or put over one another in the canteen. The plate which is at the top most is the first one to be removed and the plate which is placed at the bottom most position remains in the stack until last plate is removed.

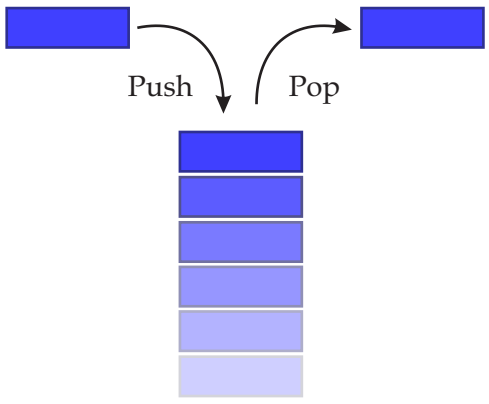



**Fig. 1.3: Stack (Push- Pop)**

The term *push* is used to insert a new element into the stack and *pop* is used to remove an element from the stack. Insertion and removal can be done at one end called top. Stack is in overflow state when it is completely full and is in underflow state if it is completely empty. In overflow state we cannot add an item and in underflow state we cannot remove an item from it.

### (b) Queue

A Queue is a linear data structure which follows a particular order in which operations are performed in FIFO (First In First Out) method, which means that element inserted first will be removed first. Example of a queue is any queue of students in school or people in cinema where one who comes first gets the ticket first. Deletions take place at one end called *front or head* and insertions take place only at the other end called *rear or tail*. Once a

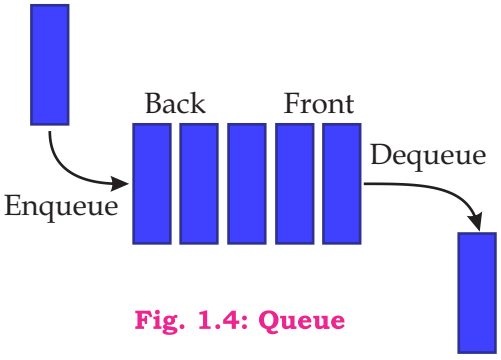



**Fig. 1.4: Queue**

new element is inserted into the Queue, it cannot be removed until all the elements inserted before it in the queue are removed. The process to add an element into queue is called Enqueue and the process to remove an element from queue is called Dequeue.

### (c) Array

Array is a **linear data structure**, which holds a list of finite data elements of same data type. Each element of array is referenced by a set of index of consecutive numbers. The elements of array are stored in successive memory locations. Most of the data structures make use of arrays to implement their algorithms. Two terms are necessary to understand array.

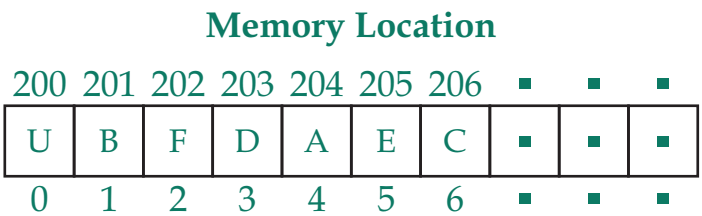



**Fig. 1.5: Array (Memory Locations)**

- **Element:** Each item stored in an array is called an element.
- **Index:** Each location of an element in an array has a numerical value called index, which is used to identify the element. Set of indexes in an array are consecutive numbers.

Operations like Traversal, Search, Insertion, Deletion and Sorting can be performed on arrays.

| **Teachers Note**  | Teacher should tell students about operations performed on an array like traversal, sorting, etc. |
|---|---|

### 1.4.2 Non-Linear Data Structures

The elements of a non-linear data structure are not connected in a sequence. Each element can have multiple paths to connect to other elements. They support multi-level storage and often cannot be traversed in single run. Such data structures are difficult to implement but are more efficient in utilizing computer memory. Examples of non-linear data structures are Tree, Graphs etc.

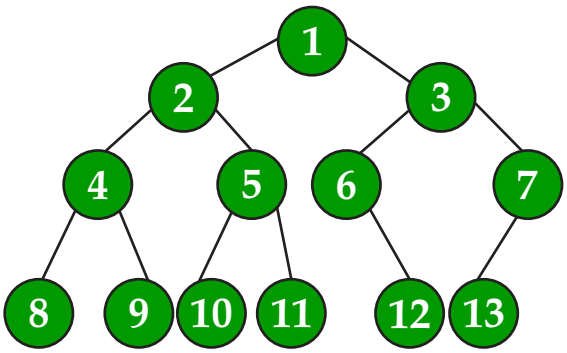



**Fig. 1.6: Tree Data Structure**

#### (a) Tree

This non-linear data structure is used to represent data containing a hierarchical relationship between elements. Tree represents its elements as the nodes connected to each other by edges. In each tree collection, we have one root node, which is the very first node in our tree. If a node is connected to another node element, it is called a parent node and the connected node is called its child node. There is also a binary tree or binary search tree. A binary tree is a special data structure used to store data in which each node can have a maximum of two children. Each node element may or may not have child nodes.

#### (b) Graph

A graph is a **non-linear data structure** consisting of data elements (finite set) called nodes/vertices and edges that are lines that connect any two nodes in that *graph*. Each element or node can contain information like roll number, name of student, marks, etc. In graph each node can have any number of edges, there is no any node

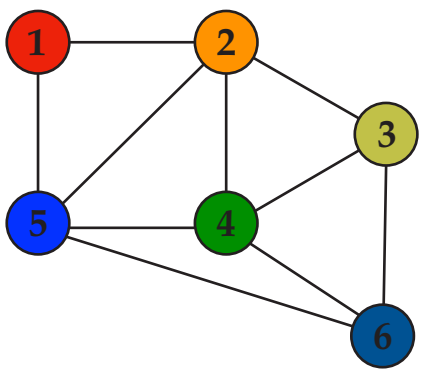



**Fig. 1.7: Graph Data Structure**

called root or child. A cycle can also be formed. In the given figure, circles represent nodes or vertices, while lines represent edges.

Graphs are used to solve network problems. Examples of networks include telephone networks, social networks like Facebook, etc. For example, a single mobile phone is represented as a node (vertex) whereas its connection with other phones can be shown as an edge between nodes.

### Types of graphs:

There are two types of graphs.

1. **Undirected Graph:**

In an undirected graph, nodes are connected by edges that are all bidirectional. For example if an edge connects node 1 and 2, we can traverse from node 1 to node 2, and from node 2 to 1.

2. **Directed Graph:**

In a directed graph, nodes are connected by directed edges – they only go in one direction. For example, if an edge connects node 1 and 2, but the arrow head points towards 2, we can only traverse from node 1 to node 2 – not in the opposite direction.


- Problem solving is the process of finding solutions of difficult or complex issues.
- A problem is a situation preventing something from being achieved.
- There are four basic steps involved in finding a solution; define the problem, generate alternative solutions, evaluate and select an alternative and implement and follow up on the solution.
- It is important to understand the problem and set a starting point of solution.
- There are various strategies that can be used to formulate an algorithm for solving the problem.

- Using the strategy, various solutions to a given problem are planned and the most feasible solution is identified.
- Algorithm is a technical term for a set of instructions for solving a problem or sub-problem.
- Algorithms enable breaking down of problems and conceptualize solutions step-by-step.
- Algorithms are defined as generic steps of instructions so they can be written in any programming language.
- A flowchart writes the sequence of steps and logic of solving a problem, using graphical symbols.
- Flowcharts help in communicating the logic of a program to all others and make it easy for understanding.
- A **data structure** is a particular way of organizing data in a computer to use it effectively.
- In *Linear data structure* data elements are arranged in sequential order and each of the elements is connected to its previous and next element.
- Stack is a linear data structure in which items may be added or removed only at one end i.e. at the top of stack.
- Queue is a linear data structure in which that element inserted first will be removed first.
- Array is a **linear data structure**, which holds a list of finite data elements of same data type. Each element of array is referenced by a set of index of consecutive numbers.
- The elements of a non-linear data structure are not connected in a sequence. Each element can have multiple paths to connect to other elements.
- Tree non-linear data structure is used to represent data containing a hierarchical relationship between elements.
- A graph is a **non-linear data structure** consisting (finite set) of data elements called nodes/vertices and edges that are lines that connect any two nodes in that *graph.*


## A. ENCIRCLE THE CORRECT ANSWER:

1. To "bake a cake" is an example of:
   **a.** problem **b.** strategy
   **c.** algorithm **d.** solution
2. To find a feasible solution to a problem, the first step is to:
   **a.** establish starting point **b.** find available solutions
   **c.** create a strategy **d.** identify and analyze the problem
3. Step by step solution of a problem in simple language is called:
   **a.** Problem Solving **b.** Algorithm
   **c.** Flowchart **d.** Data Structure
4. _____ shows the logic of program graphically.
   **a.** Data Structure **b.** Graph
   **c.** Algorithm **d.** Flowchart
5. _____ Symbol is used for input/output in flowchart.
   **a.** Triangle **b.** Square
   **c.** Parallelogram **d.** Rectangle
6. Elements of _____ data structure are not connected sequentially.
   **a.** Array **b.** Graph
   **c.** Queue **d.** Stack
7. _____ stores data in hierarchal manner.
   **a.** Stack **b.** Queue
   **c.** Array **d.** Tree
8. When that data is Pushed in stack, it means that data is:
   **a.** inserted **b.** deleted
   **c.** sorted **d.** edited
9. In binary tree, each child can have maximum:
   **a.** one node **b.** two nodes
   **c.** three nodes **d.** four nodes
10. Traversing an array means accessing:
    **a.** first element **b.** last element
    **c.** any specific element **d.** each and every element of the array

## B. RESPOND THE FOLLOWING:

1. Describe the steps involved in problem solving.
2. What are the advantages of developing algorithms?
3. List any three advantages of designing flowcharts.
4. What is the difference between tree and graph data structure?
5. What is the difference between queue and stack data structure?
6. What is the need of index in an array?
7. With the help of a sketch define: Push, Pop, Overflow and Enqueue, Dequeue.


1. Design an algorithm to find the greater number by taking two numbers as input.
2. Design an algorithm to find area of a triangle.
3. Sort the following steps of the algorithm in correct order for baking a cake:
   - Step: Gather the ingredients
   - Step: End
   - Step: Grease a pan
   - Step: Preheat the oven
   - Step: Put the pan in the oven
   - Step: Start
   - Step: Pour the batter into the pan
   - Step: When the timer goes off, take the pan out of the oven
   - Step: Set a timer
   - Step: Mix together the ingredients to make the batter
4. Draw a flow chart to calculate gross salary by adding 20% house rent and 30% medical allowances in basic salary.
5. Draw a flow charts for all the algorithms given in this unit.
6. Draw the following structure
   a. Tree with six nodes.
   b. Graph with five nodes.
7. Search about **Algorithmic Thinking** and in groups, discuss this concept.