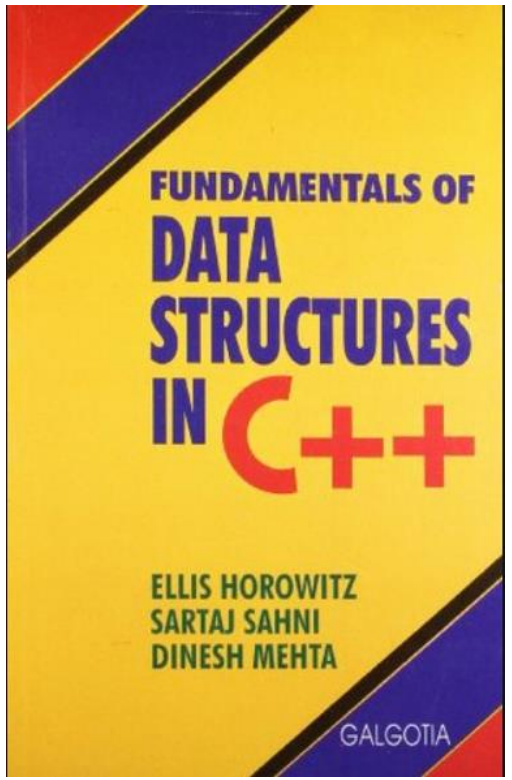
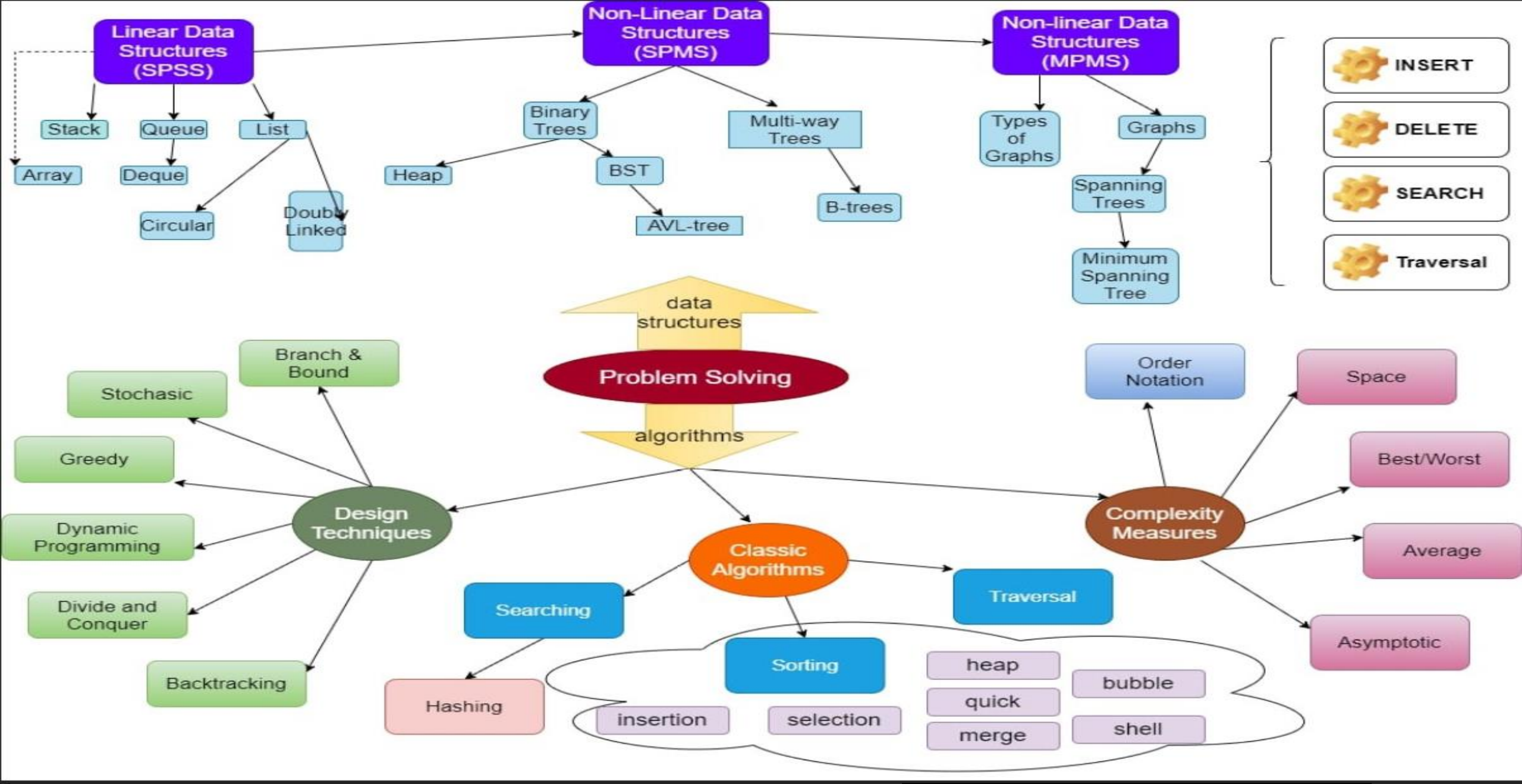
Algorithms & Data Structure

Kiran Waghmare

Module 2: Algorithms and Data Structures

- **Text Book:**
  - Fundamentals of Data Structures in C++ by Horowitz, Sahani & Mehta
- **Topics:**
  - 1.Problem Solving & Computational Thinking
  - 2.Introduction to Data Structures & Recursion
  - 3.Stacks
  - 4.Queues
  - 5.Linked List Data Structures
  - 6.Trees & Applications
  - 7.Introduction to Algorithms
  - 8.Searching and Sorting
  - 9.Hash Functions and Hash Tables
  - 10.Graph & Applications
  - 11.Algorithm Designs

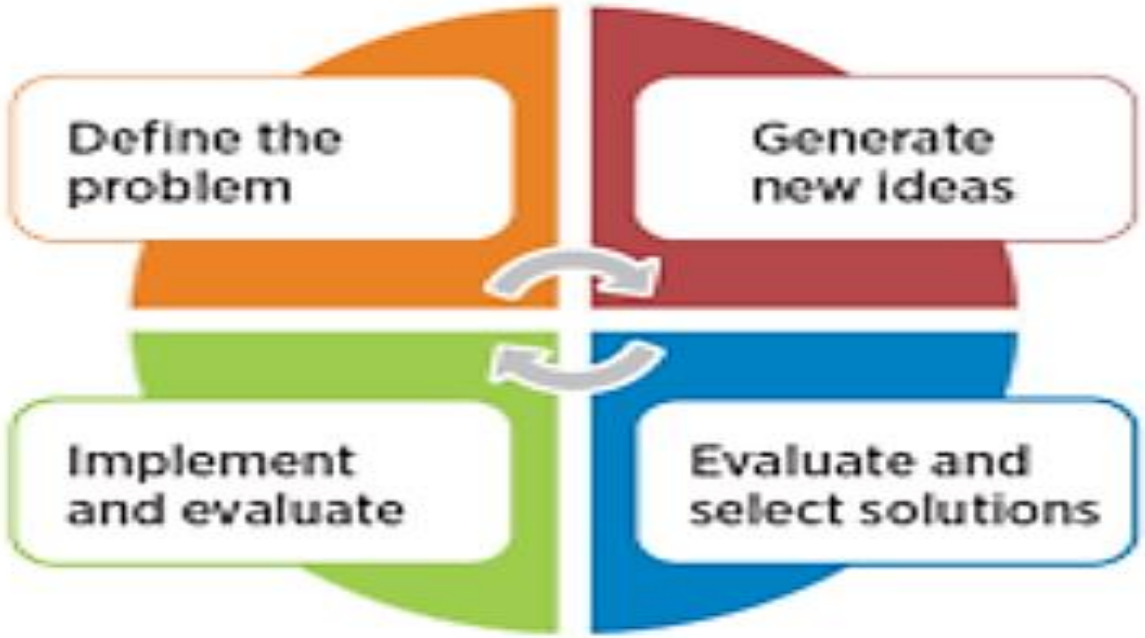



Linear Data Structures (SPSS)
Non-Linear Data Structures (SPMS)
Non-linear Data Structures (MPMS)
Stack
Queue
List
Array
Deque
Circular
Doubly Linked
Binary Trees
Multi-way Trees
Heap
BST
AVL-tree
B-trees
Types of Graphs
Graphs
Spanning Trees
Minimum Spanning Tree
INSERT
DELETE
SEARCH
Traversal
data structures
Problem Solving
algorithms
Branch & Bound
Stochasic
Greedy
Dynamic Programming
Divide and Conquer
Backtracking
Design Techniques
Classic Algorithms
Searching
Hashing
Sorting
insertion
selection
heap
quick
merge
bubble
shell
Traversal
Order Notation
Space
Best/Worst
Average
Asymptotic
Complexity Measures

What is Computational Thinking?

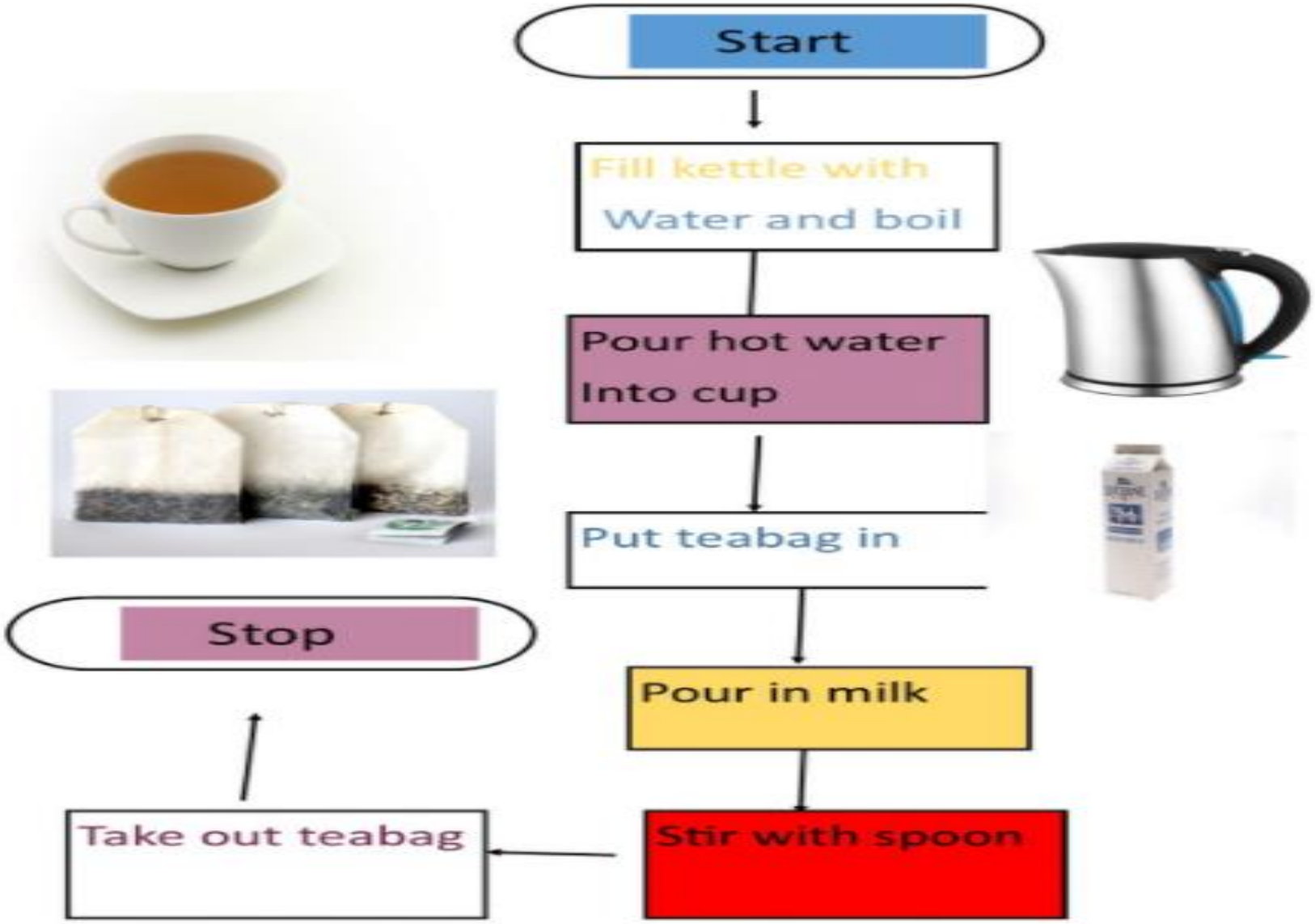
- **Computational thinking is a problem solving process that includes:**

- **Decomposition:**
  - Breaking down data, processes, or problems into smaller, manageable parts.
- **Pattern Recognition:**
  - Observing patterns, trends, and regularities in data.
- **Abstraction:**
  - Identifying the general principles that generate these patterns.
  - This involves filtering out the details we do not need in order to solve a problem.
- **Algorithm Design:**
  - Developing the step by step instructions for solving this and similar problems.



Problem Solving Chart

Write Algorithm to prepare a Tea



Definition

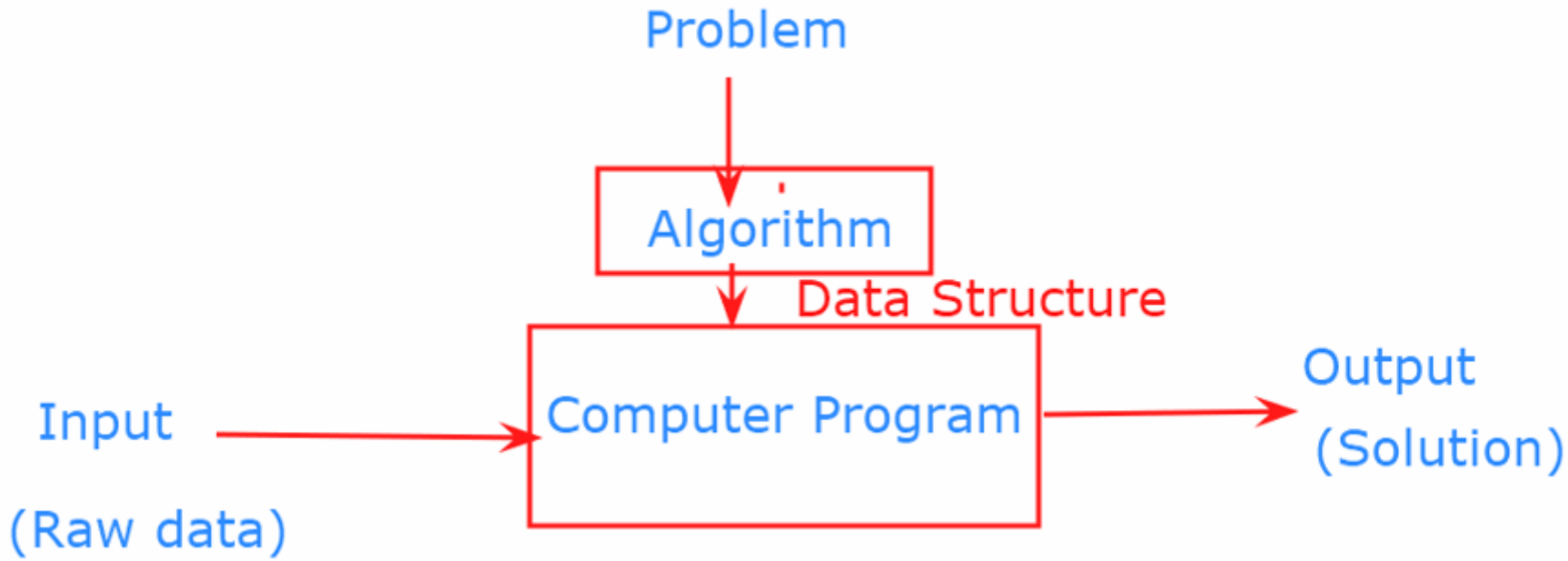
- **Data:**
  - Collection of Raw facts.
- **Algorithm:**
  - Outline, the essence of a computational procedure, step-by-step instructions.
- **Program:**
  - An implementation of an algorithm in some programming language
- **Data Structure:**
  - Organization of data needed to solve the problem.
  - The programmatic way of storing data so that data can be used efficiently

Data Structure

- It is representation of the logical relationship existing between individual elements of data.
- It is a specialized format for organizing and string data in memory that considers not only the elements stored but also their relationship to each other.
- Data structure affects the design of both structural & functional aspects of a program.
- Program = Algorithm + Data Structure
- Algorithm is a step by step procedure to solve a particular function.

Algorithm

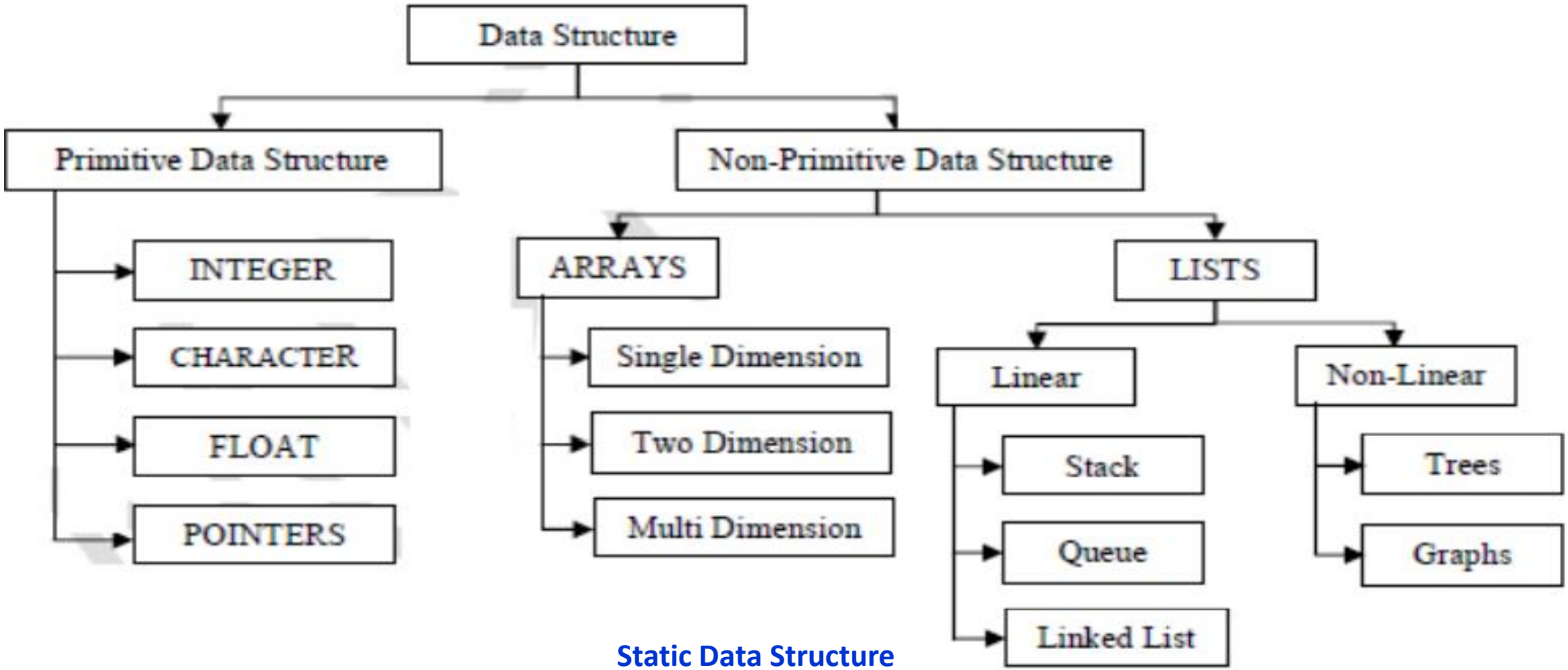
- An ***algorithm*** is a sequence of unambiguous instructions/operations for solving a problem, for obtaining a required output for any legitimate input in a finite amount of time.



Data structure

- **A data structure is a data organization, management and storage format that enables efficient access and modification.**
- **It is a way in which data is stored on a computer**
- **Need of Data Structure:**
  - Each data structure allows data to be stored in specific manner.
  - Data structure allows efficient data search and retrieval.
  - Specific Data structure are decided to work for specific problems.
  - It allows to manage large amount of data such as databases and indexing services such as hash table.

Classification of Data Structure



Static Data Structure

Dynamic data structure

Advantages of Data structure

Efficiency
Abstraction
optimization
Easy of programming
Resuability

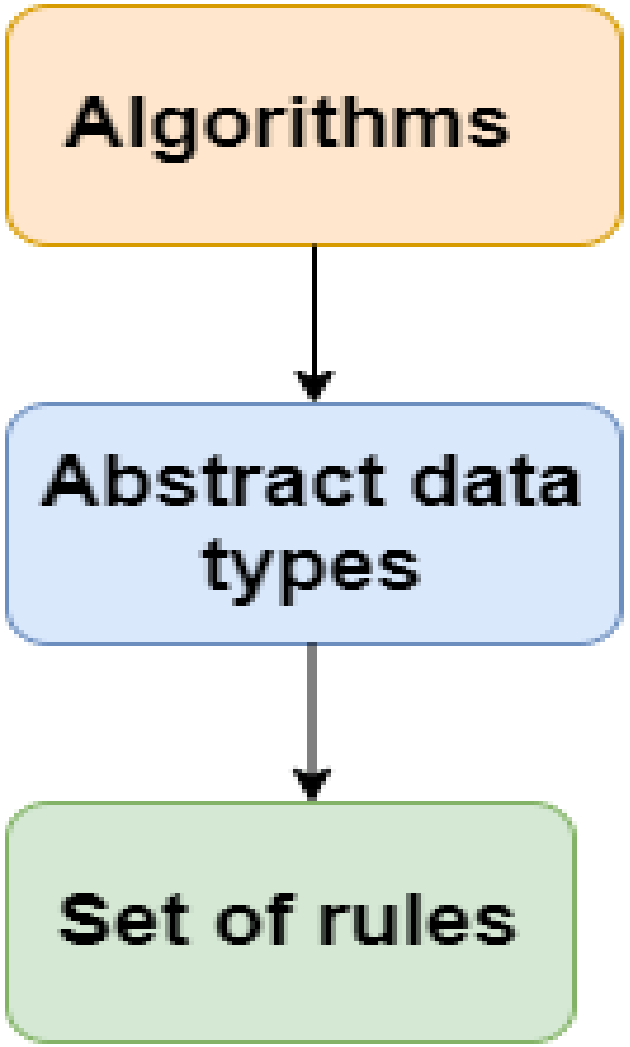
Operations on Data structures

Insert
Delete
Travese
Search
Sort
Merge

ADT : Abstract Data Type

Algorithm----->ADT------> Set of Rules

Abstract Data Type (ADT)



OnePlus 9 5G
(Winter Mist, 12G...

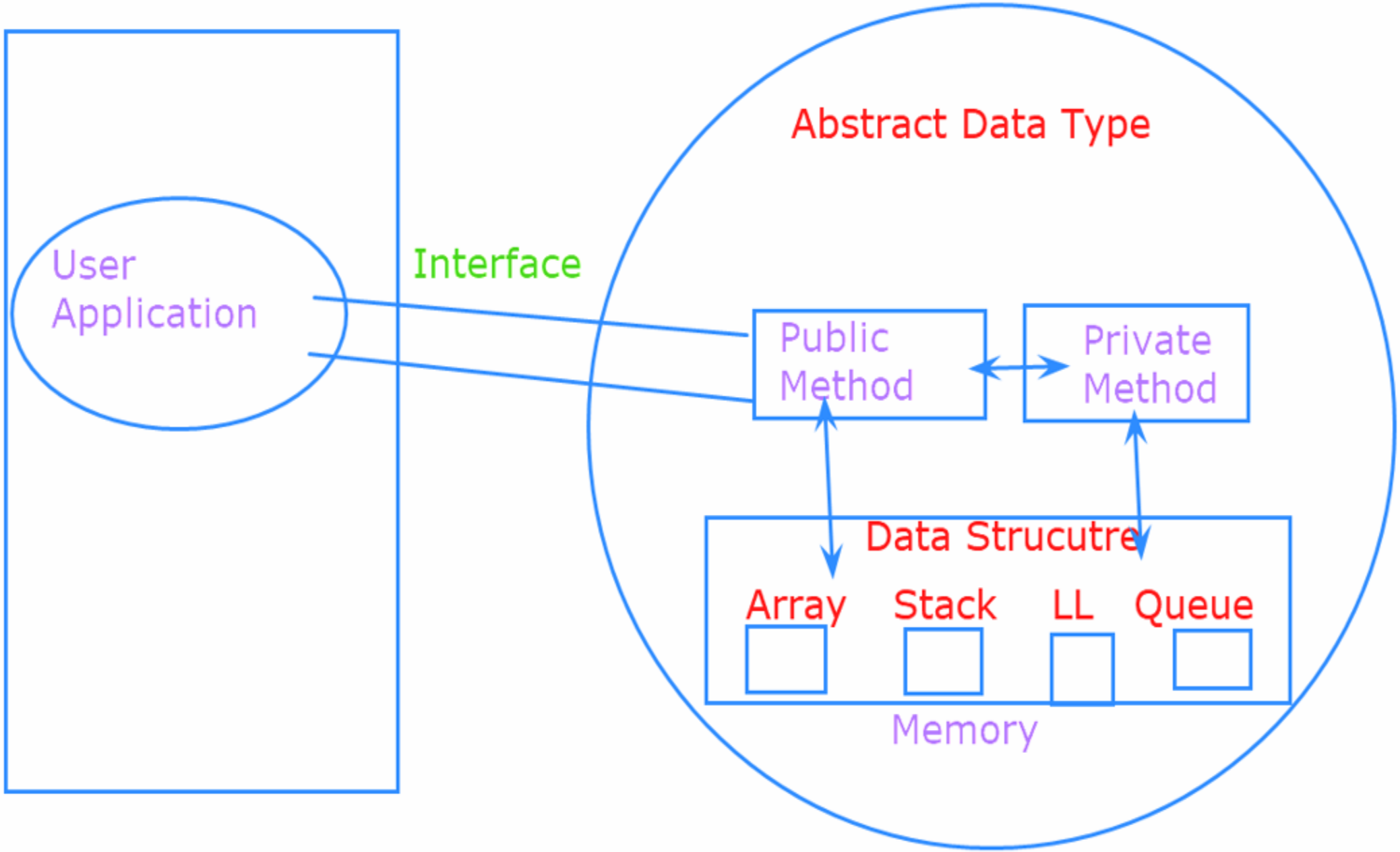
₹54,999

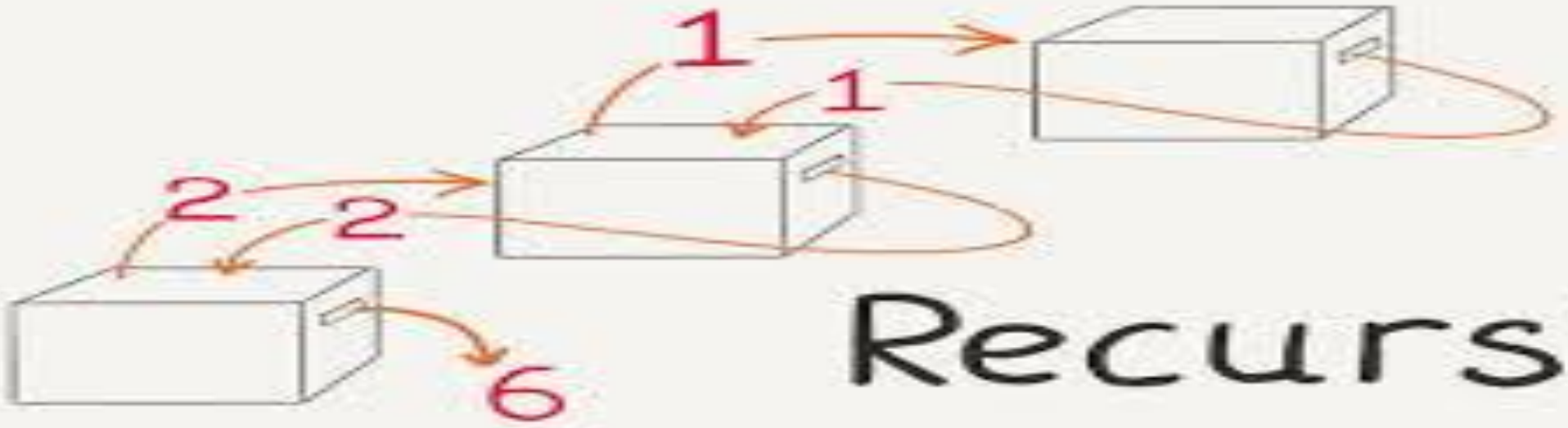
Amazon.in

Free shipping

Logical view

Implementation

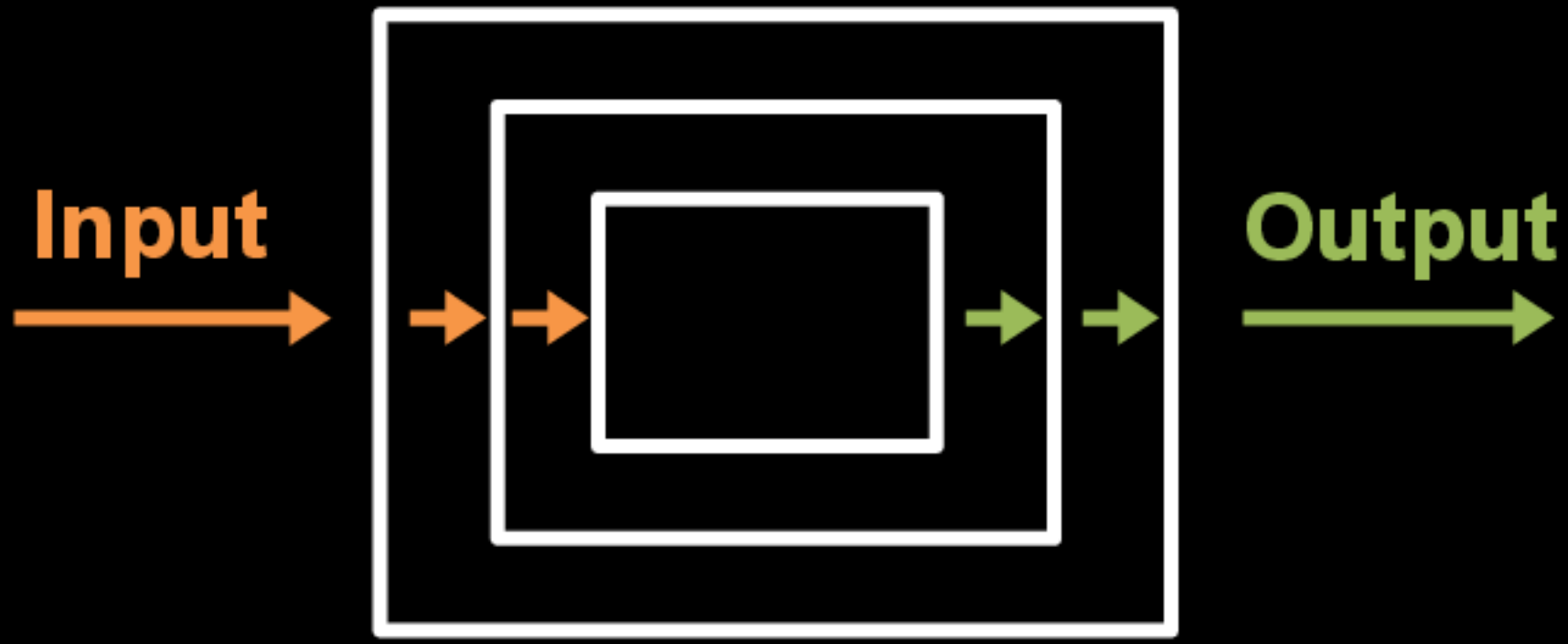

Abstract Data Type
User Application
Interface
Public Method
Private Method
Data Strucutre
Array
Stack
LL
Queue
Memory



Topics

1. Recursive definitions and Processes
2. Writing Recursive Programs
3. Efficiency in Recursion
4. Towers of Hanoi problem.

Recursion



Recursion

- Any function which calls itself directly or indirectly is called Recursion and the corresponding function is called as recursive function.
- A recursive method solves a problem by calling a copy of itself to work on a smaller problem.
- It is important to ensure that the recursion terminates.
- Each time the function call itself with a slightly simple version of the original problem.
- Using recursion, certain problems can be solved quite easily.
- E.g: Tower of Hanoi (TOH), Tree traversals, DFS of Graph etc.,

Day 1: 30/10/2021

Who can see what you share here? Recording On

Topics:

-Problem Solving & Computational Thinking

-Algorithm & Data Structure

-Recursion

```
//recursive function
static void show()
{
    show();//recursive call
}
```

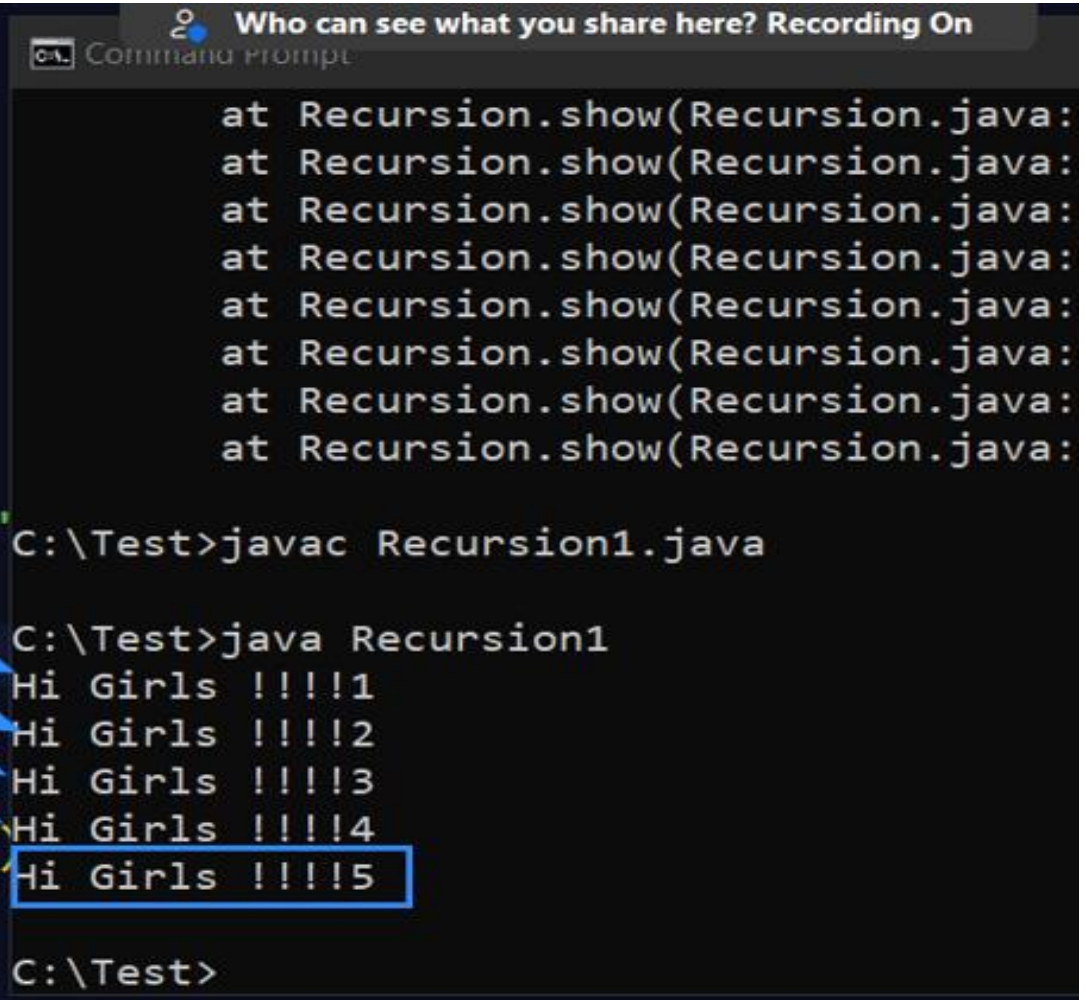
directly or indirectly call itself

RECURSION

```
p.s.v.main()
{
show()
}
```

```java
class Recursion1
{
    static int i=0;
    static void show()
    {
        ++i;
        if(i<=5)
        {
        System.out.println("Hi Girls !!!!"
        show();
        }
    }

    public static void main(String args[])
    {
        show();
    }
}
```

Base condition



```
class Recursion2
{

    static int show(int n)  2, 3 , 4
    {
        if(n==4)  2=4,  3=4,  4=4
            return n;
        else
            return 2*show(n+1);

    }                   2+show(n+1)

    public static void main(String args[])
    {
        System.out.println(show(2));
    }
}
```

Who can see what you share here? Recording On

show(2)

2*show(2+1)

2* [2*show(3+1)]

2 * 2* [4]

16

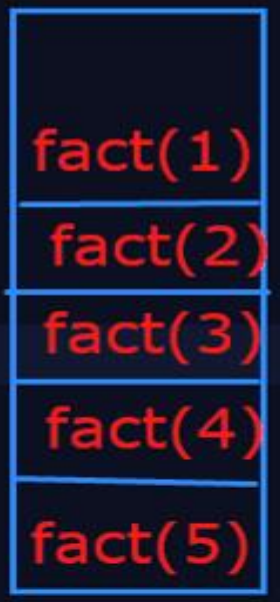
Who can see what you share here? Recording On

```
class Recursion3
{
    static int fact(int n)
    {
        if(n<=1)//base condition
            return 1;
        else
            return n*fact(n-1);
    }

    public static void main(String args[])
    {
        System.out.println(fact(5));
    }
}
```

5, 4, 3, 2, 1

5! = 5*4!
= 5*4*3!
= 5*4*3*2!
= 5*4*3*2*1!
= 5*4*3*2*1

Base condition



stack

fact(5) = 5*fact(4)
= 5*4*fact(3)
= 5*4*3*fact(2)
= 5*4*3*2*fact(1)
= 5*4*3*2*1

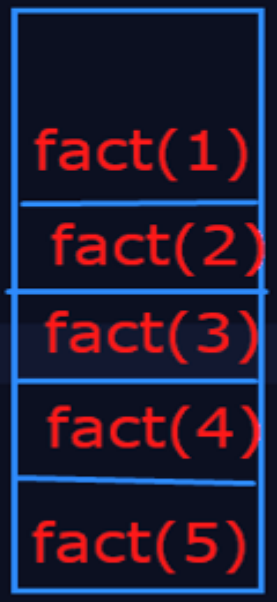
```
class Recursion3
{
    static int fact(int n)
    {
        if(n<=1)//base condition
            return 1;
        else
            return n*fact(n-1);
    }

    public static void main(String args[])
    {
        System.out.println(fact(5));
    }
}
```

5, 4, 3, 2, 1

fact(5) = 5*fact(4)
= 5*4*fact(3)
= 5*4*3*fact(2)
= 5*4*3*2*fact(1)
= 5*4*3*2*1

5! = 5*4!
= 5*4*3!
= 5*4*3*2!
= 5*4*3*2*1!
= 5*4*3*2*1

Base condition



Fact(5)
5*fact(4)
4*fact(3)
3*fact(2)
2*fact(1)
1