



DATA STRUCTURES AND ALGORITHMS

Mar23 : Day 1

Kiran Waghmare
CDAC Mumbai

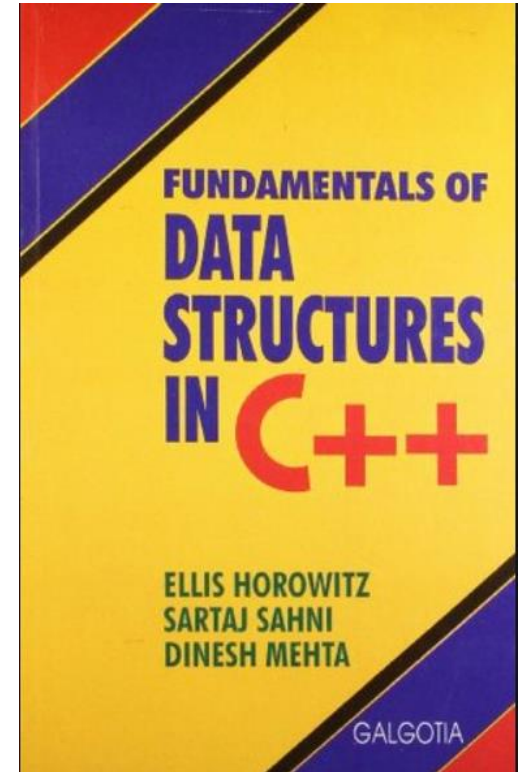
Module 2: Algorithms and Data Structures

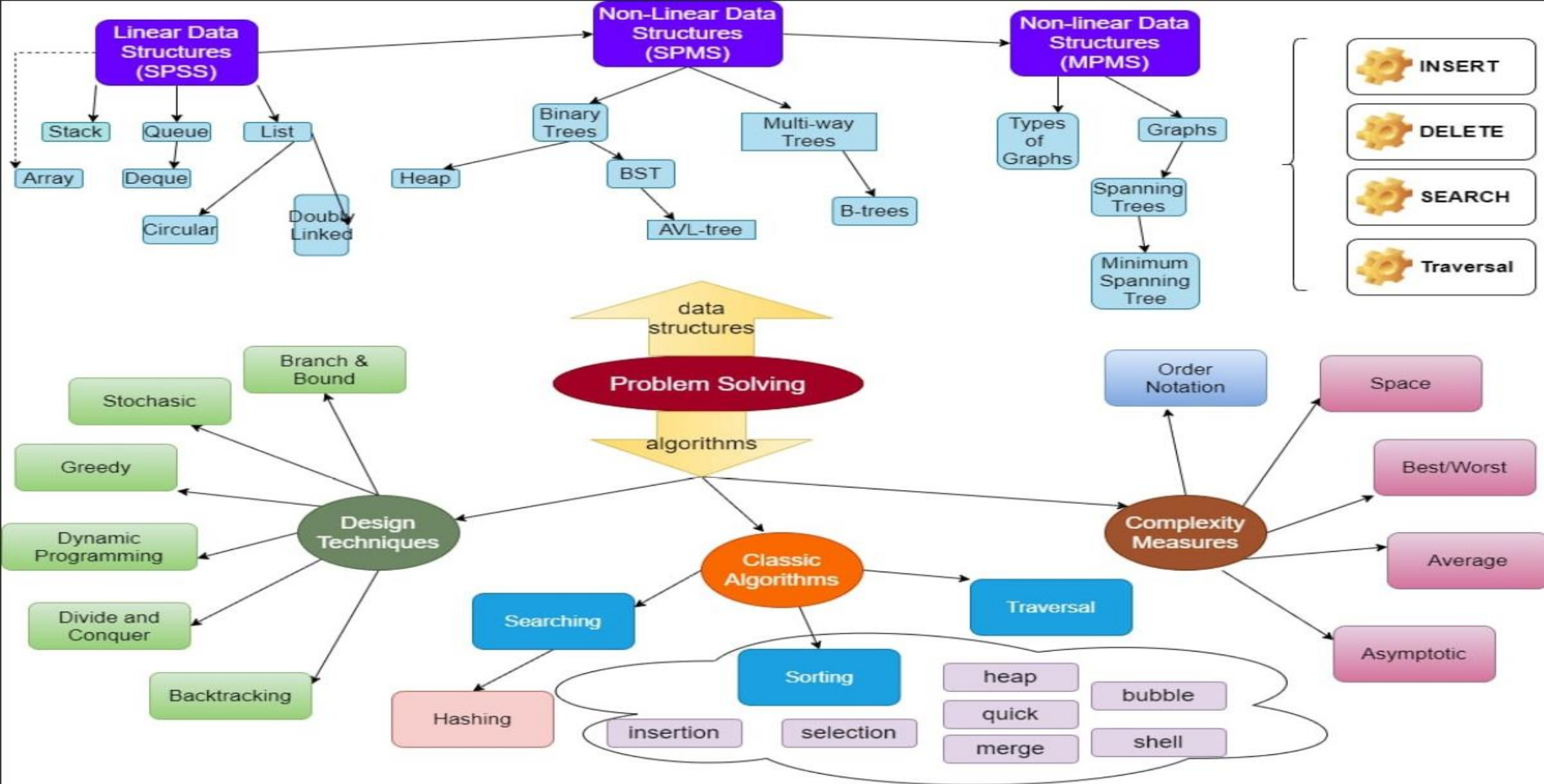
- **Text Book:**

- 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





Agenda

- **Problem Solving & Computational Thinking**

- **Algorithm & Data Structure**

 - OODesign: ADTs

- **Recursion**

 - Base condition

 - Direct & indirect recursion

 - Memory allocation

 - Pros and Cons

 - Complexity analysis

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

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

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.

Algorithm Design Strategies

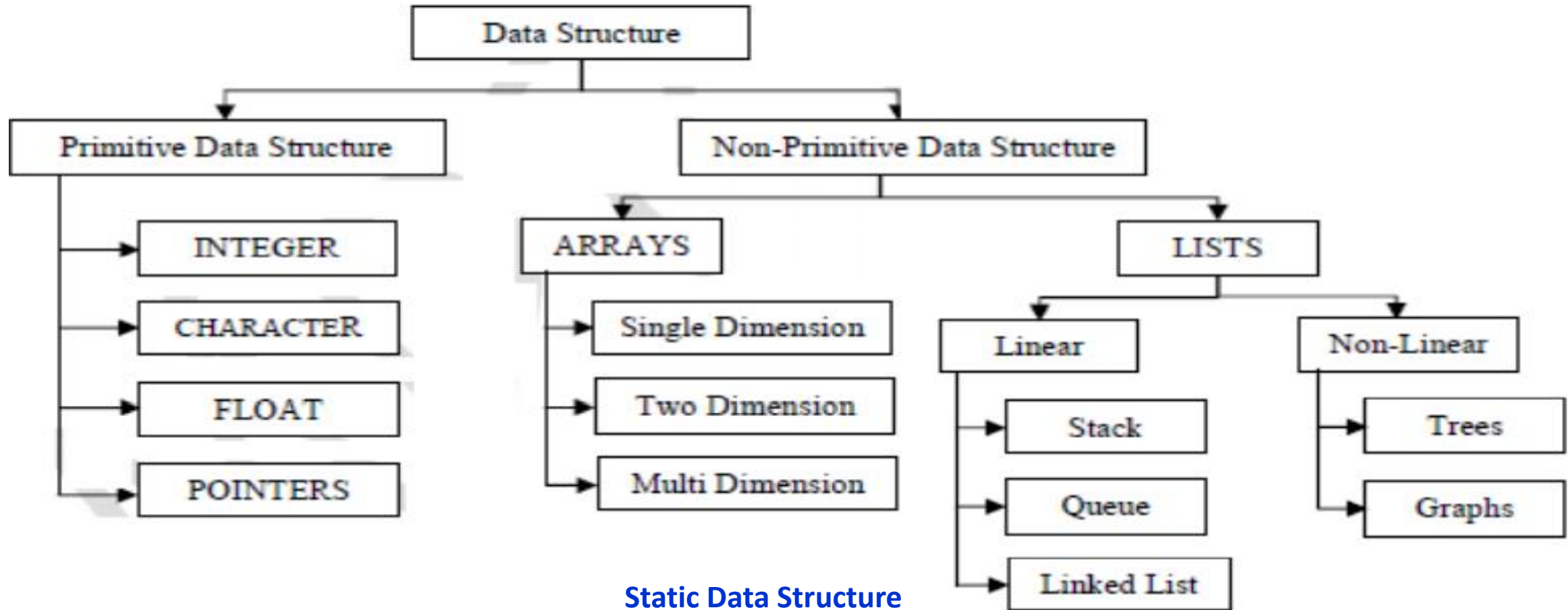
- Brute force
- Divide and conquer
- Decrease and conquer
- Transform and conquer
- Greedy approach
- Dynamic programming
- Backtracking and branch and bound
- Space and time tradeoffs

Invented or applied
by many genius in
CS

Analysis of Algorithms

- An algorithm is said to be efficient and fast, if it **takes less time to execute and consumes less memory space.**
- The performance of an algorithm is measured on the basis of following properties :
 - 1.Time Complexity
 - 2.Space Complexity

Classification of Data Structure

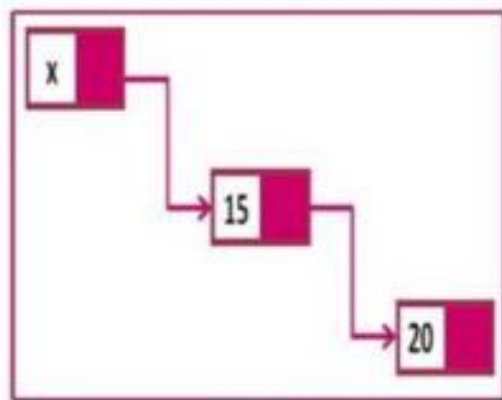


Static Data Structure

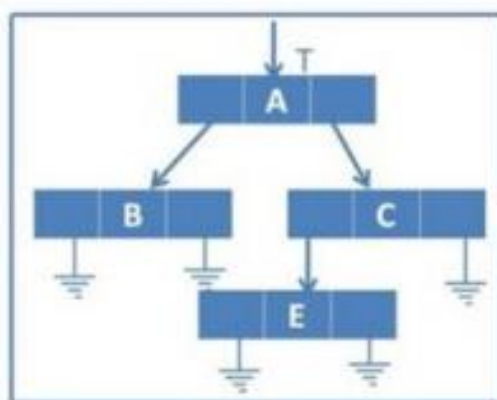
Dynamic data structure



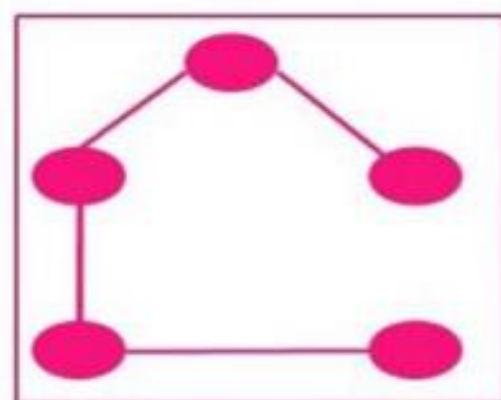
Sorting



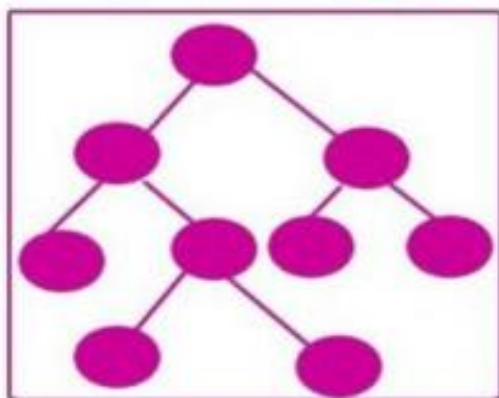
Link list



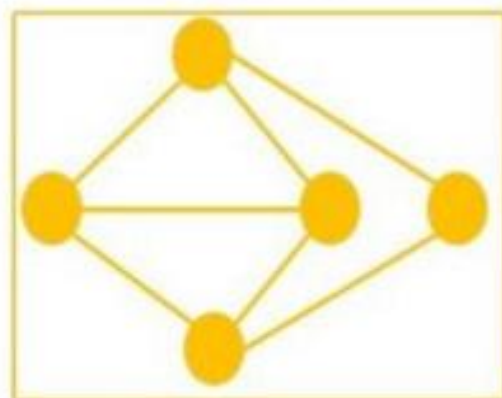
list



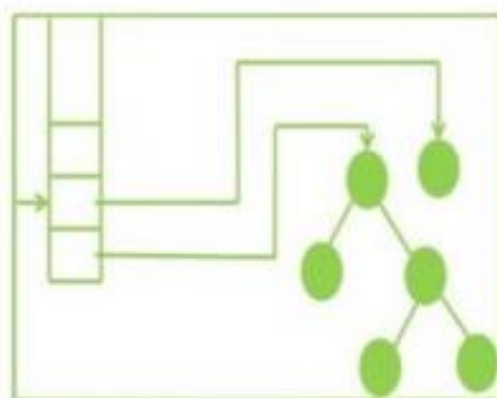
spanning tree



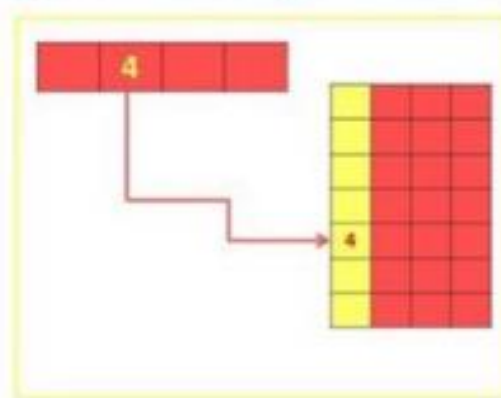
Tree



Graph

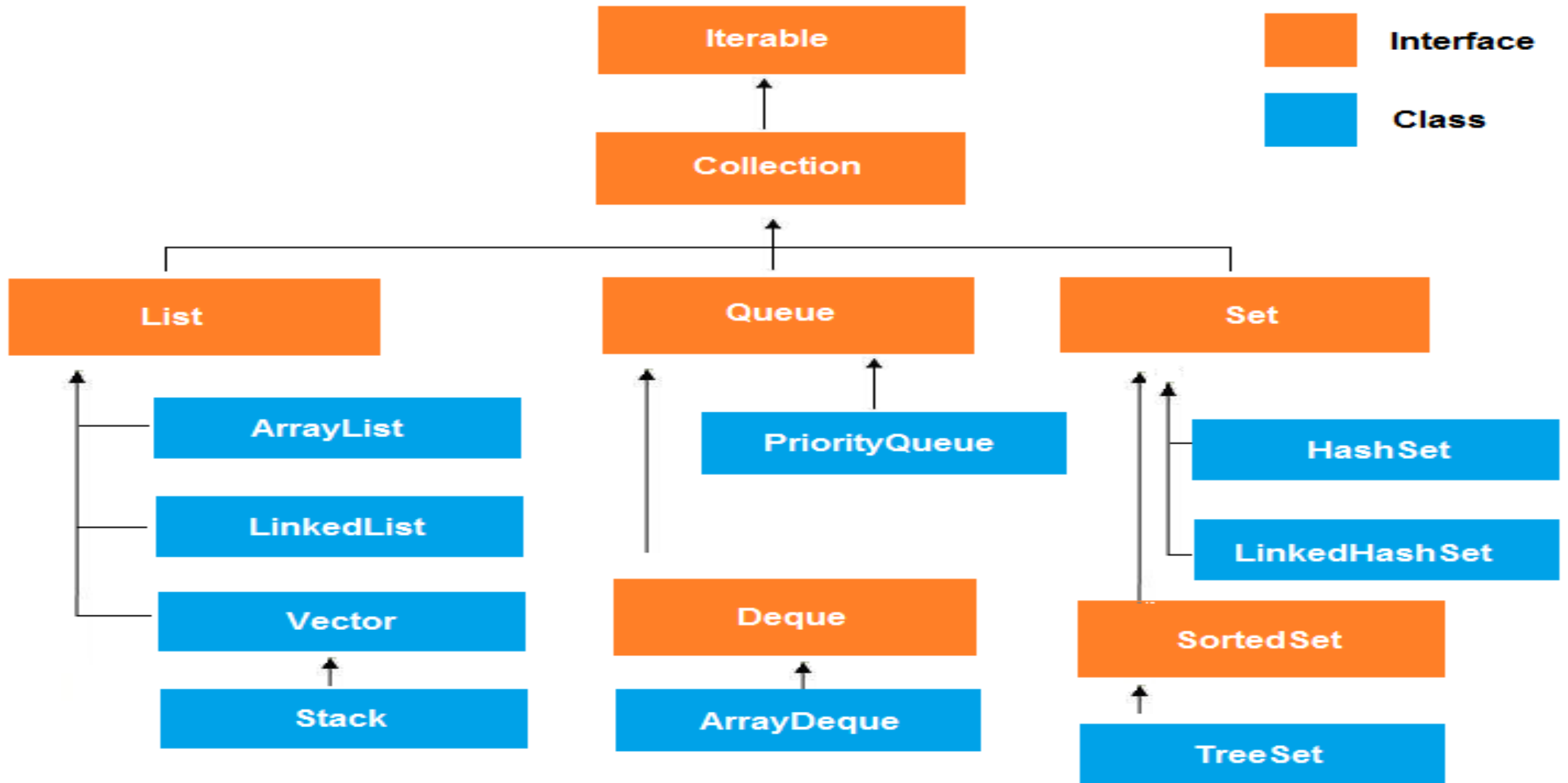


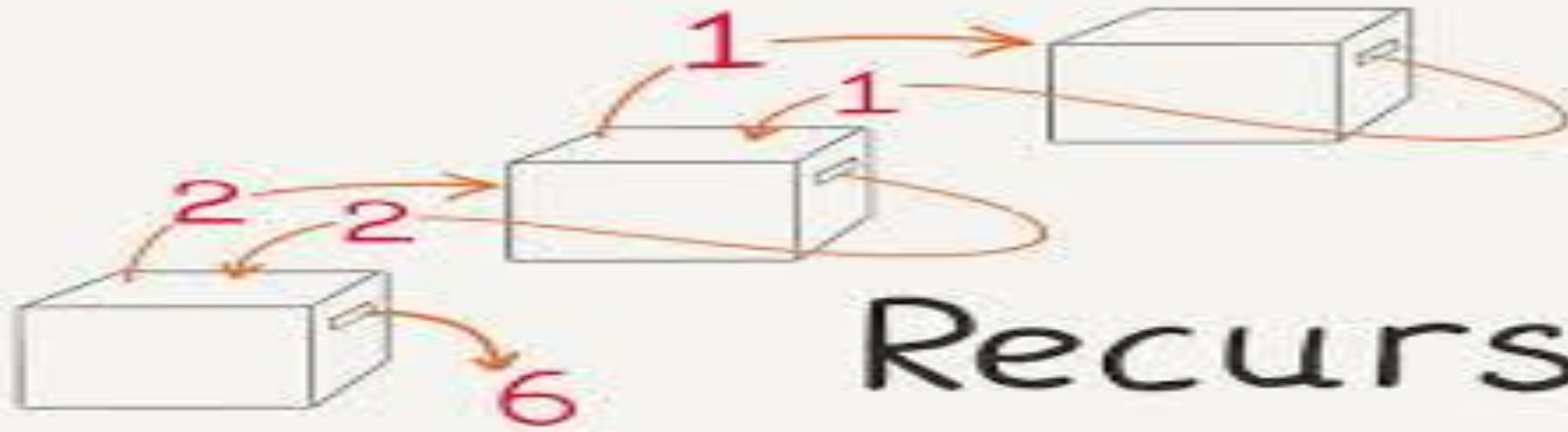
Stack



Hashing

Abstract Data Type (ADT)



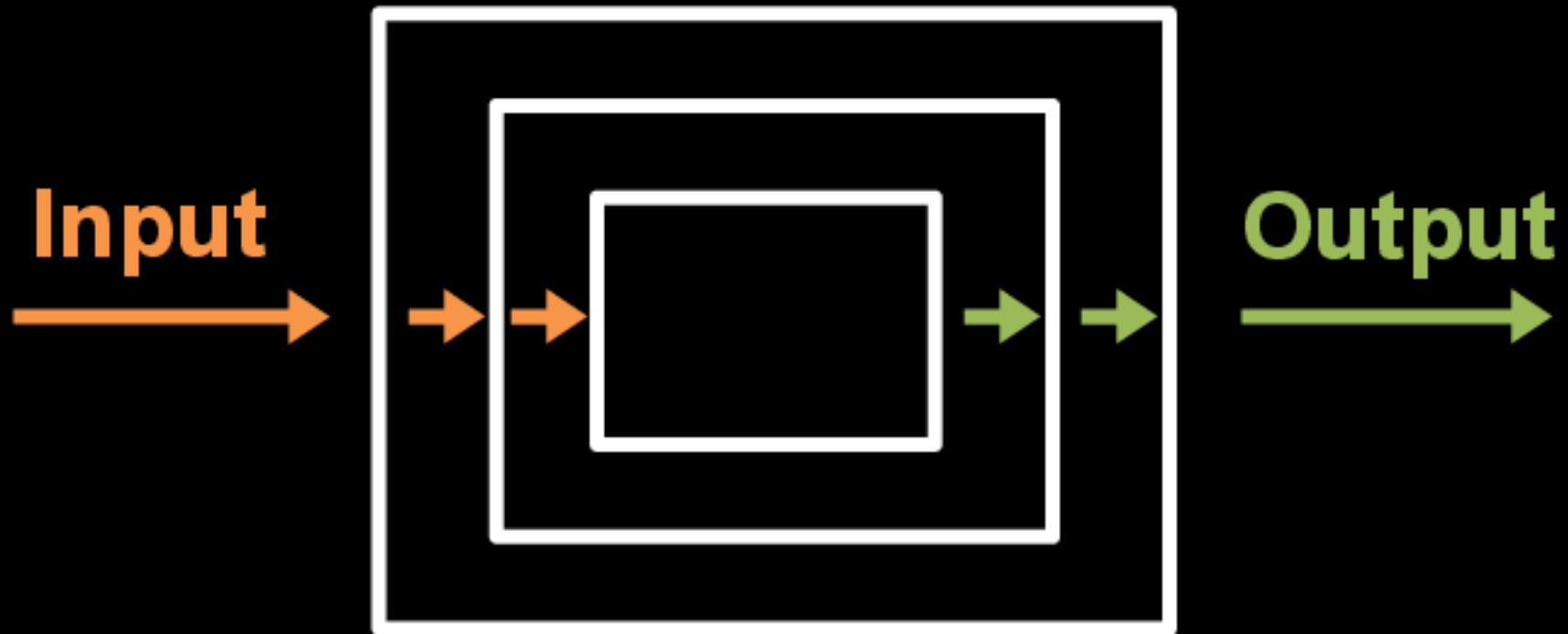


Recursion

Topics

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

Recursion



How does Recursion works?

```
void recurse()
{
    ... ..
    recurse();
    ... ..
}

int main()
{
    ... ..
    recurse();
    ... ..
}
```

The diagram illustrates the flow of recursive calls. A line from the `recurse();` statement in the `main()` function extends to the right and then turns upwards to point at the `recurse()` function definition. Another line from the `recurse();` statement inside the `recurse()` function extends to the right and then turns upwards to point at the same `recurse()` function definition. The text "recursive call" is placed between these two lines, indicating the nature of the self-referencing calls.

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.,

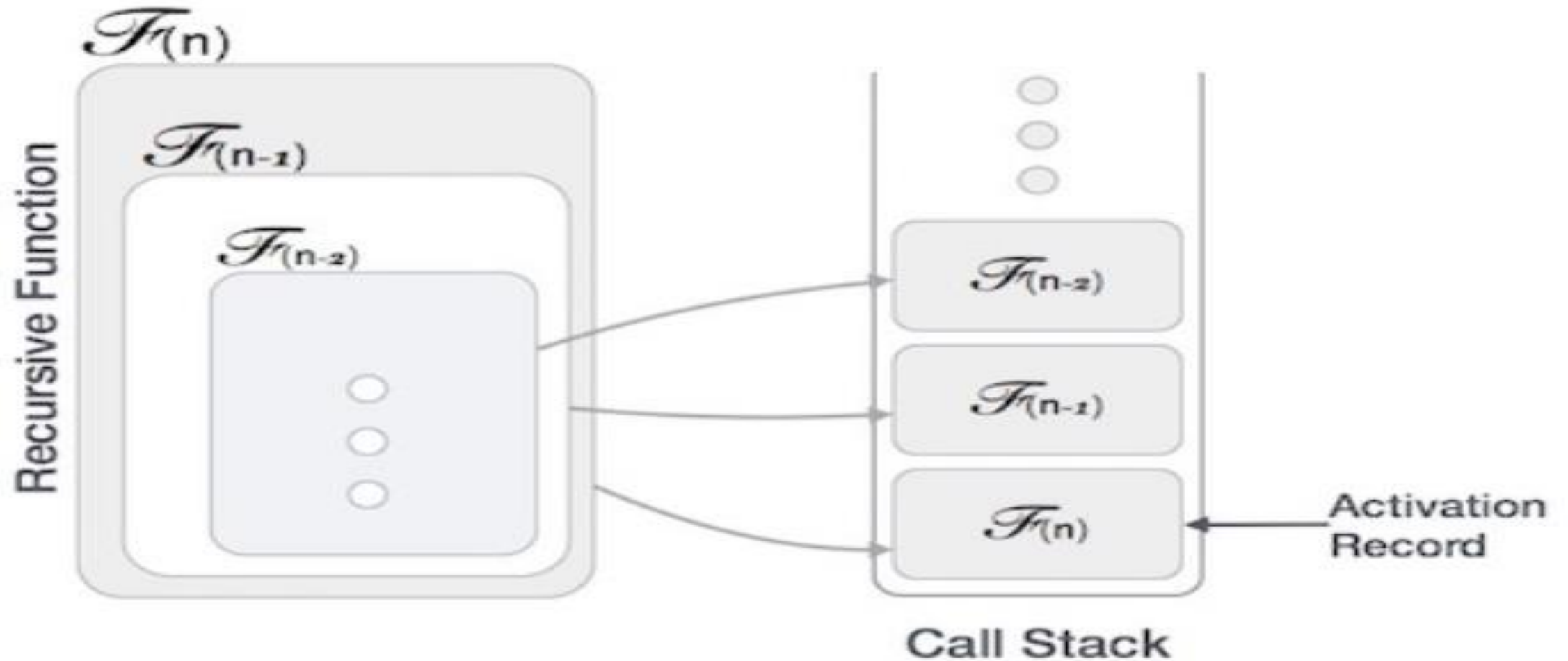
What is base condition in recursion?

- In the recursive program, the solution to the base case is provided and the solution of the bigger problem is expressed in terms of smaller problems.

```
int fact(int n)
{
    if (n <= 1) // base case
        return 1;
    else
        return n*fact(n-1);
}
```

- In the above example, **base case for $n \leq 1$** is defined and larger value of number can be solved by converting to smaller one till base case is reached.

How Data Structure Recursive function is implemented?

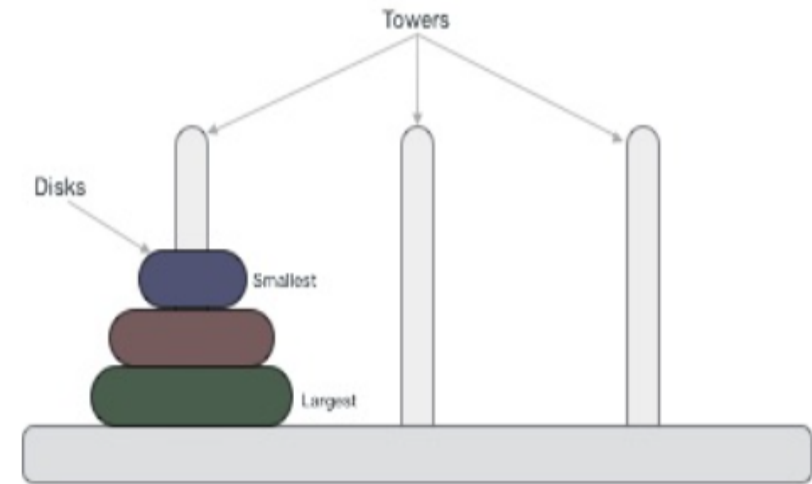


Why Algorithms?

- Fibonacci numbers
 - Compute first N Fibonacci numbers using iteration.
 - ... using recursion.
- Write the code.
- Try for N=5, 10, 20, 50, 100
- What do you see? Why does this happen?

What is Tower of Hanoi?

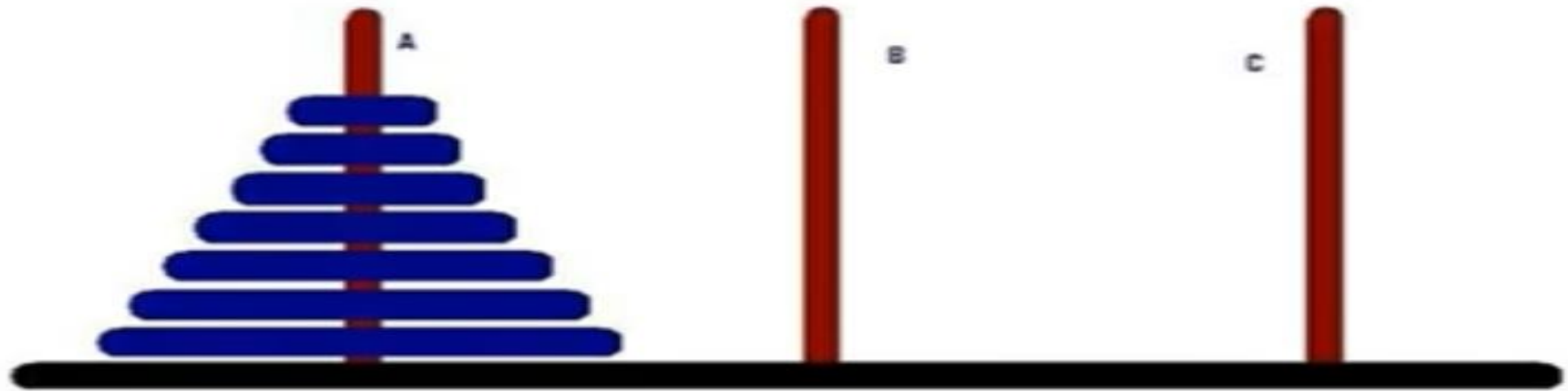
- A mathematical puzzle consisting of three towers and more than one ring is known as Tower of Hanoi.
- Tower of Hanoi
- The rings are of different sizes and are stacked in ascending order, i.e., the smaller one sits over the larger one. In some of the puzzles, the number of rings may increase, but the count of the tower remains the same.



Tower of Hanoi

Tower oh Hanoi

- The tower of Hanoi is mathematical puzzle.



- The objective of the puzzle is to move the entire stack to another rod.

What are the rules to be followed by Tower of Hanoi?

- **The Tower of Hanoi puzzle is solved by moving all the disks to another tower by not violating the sequence of the arrangements.**

The rules to be followed by the Tower of Hanoi are -

1. Only one disk can be moved among the towers at any given time.
2. Only the "top" disk can be removed.
3. No large disk can sit over a small disk.

Algorithm 1: Recursive algorithm for solving Towers of Hanoi

```
1 function recursiveHanoi( $n$ ,  $s$ ,  $a$ ,  $d$ )
2   if  $n == 1$  then
3      $\text{print}(s + \text{'' to ''} + d);$ 
4     return;
5   end
6   recursiveHanoi( $n - 1$ ,  $s$ ,  $d$ ,  $a$ );
7    $\text{print}(s + \text{'' to ''} + d);$ 
8   recursiveHanoi( $n - 1$ ,  $a$ ,  $s$ ,  $d$ );
9 end
```

Home Work

- Implement Tower of Hanoi Program
- No of Disk=3
- No of Disk=5
- No of Disk= n

Assignment 1

1. Print a series of numbers with recursive Java methods
2. Sum a series of numbers with Java recursion
3. Calculate a factorial in Java with recursion
4. Print the Fibonacci series with Java and recursion
5. A recursive Java palindrome checker

Problem 1

Recursive program to find the Sum of the series $1 - 1/2 + 1/3 - 1/4 \dots 1/N$

Given a positive integer N, the task is to find the sum of the series $1 - (1/2) + (1/3) - (1/4) + \dots (1/N)$ using recursion.

Examples:

Input: N = 3

Output: 0.8333333333333333

Explanation:

$$1 - (1/2) + (1/3) = 0.8333333333333333$$

Input: N = 4

Output: 0.5833333333333333

Explanation:

$$1 - (1/2) + (1/3) - (1/4) = 0.5833333333333333$$

Problem 2

Recursive Program to print multiplication table of a number

Given a number N, the task is to print its multiplication table using recursion.

Examples

Input: N = 5

Output:

5 * 1 = 5

5 * 2 = 10

5 * 3 = 15

5 * 4 = 20

5 * 5 = 25

5 * 6 = 30

5 * 7 = 35

5 * 8 = 40

5 * 9 = 45

5 * 10 = 50

Input: N = 8

Output:

8 * 1 = 8

8 * 2 = 16

8 * 3 = 24

8 * 4 = 32

8 * 5 = 40

8 * 6 = 48

8 * 7 = 56

8 * 8 = 64

8 * 9 = 72

8 * 10 = 80

Day 1 : Questions

- 1. WHAT IS AN ALGORITHM?**
- 2. WHY WE NEED TO DO ALGORITHM ANALYSIS?**
- 3. WHAT ARE THE CRITERIA OF ALGORITHM ANALYSIS?**
- 4. WHAT ARE ASYMPTOTIC NOTATIONS?**
- 5. BRIEFLY EXPLAIN THE APPROACHES TO DEVELOP ALGORITHMS.**
- 6. GIVE SOME EXAMPLES GREEDY ALGORITHMS.**
- 7. WHAT ARE SOME EXAMPLES OF DIVIDE AND CONQUER ALGORITHMS?**
- 8. WHICH PROBLEMS CAN BE SOLVED USING RECURSION?**
- 9. HOW DOES RECURSION WORK IN JAVA?**
- 10. WHAT IS TOWER OF HANOI?**
- 11. WHY IS RECURSION USED?**
- 12. WHAT ARE THE ADVANTAGES AND DISADVANTAGES OF RECURSION?**
- 13. DIFFERENTIATE BETWEEN RECURSION AND ITERATION.**
- 14. WHAT IS HEAD AND TAIL RECURSION?**
- 15. DISCUSS APPLICATIONS OF RECURSION.**