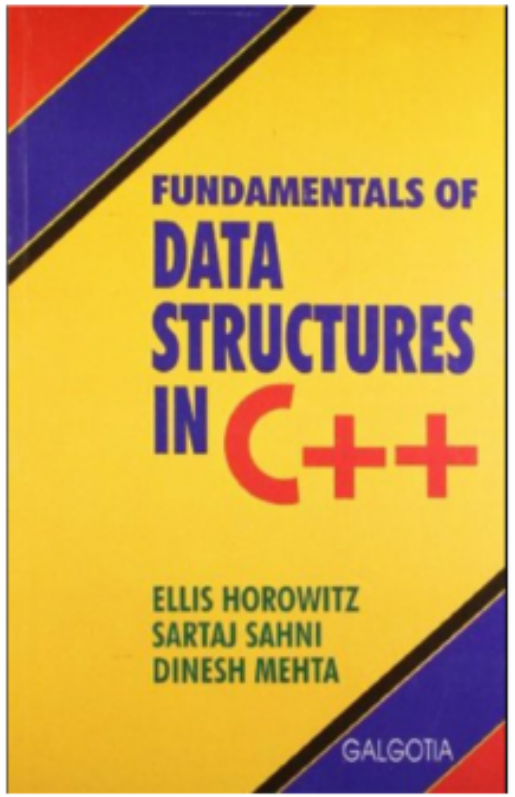
Sep22 : Day 1

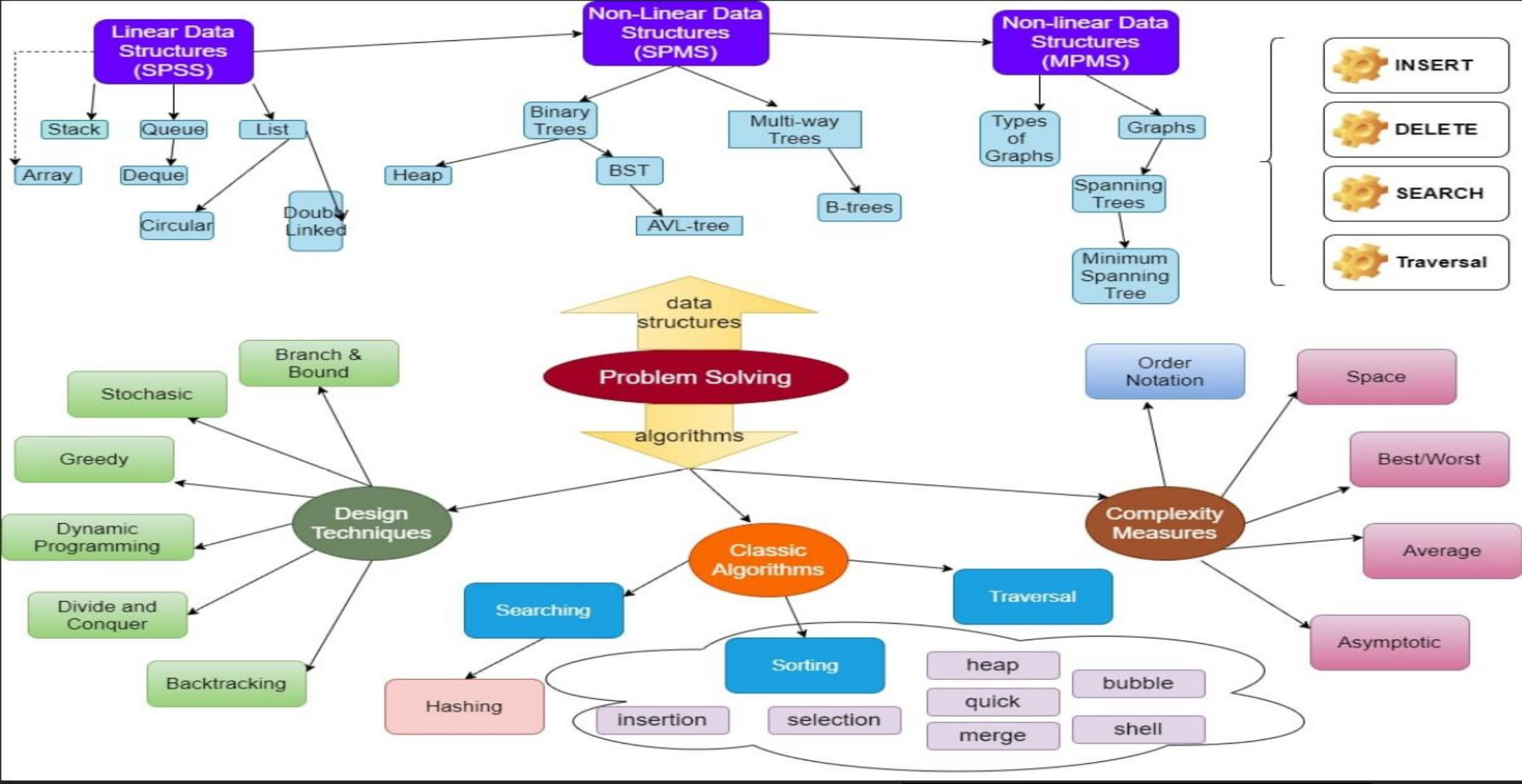
Kiran Waghmare

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




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
Design Techniques
Divide and Conquer
Backtracking
Order Notation
Space
Best/Worst
Average
Asymptotic
Complexity Measures
Classic Algorithms
Searching
Traversal
Hashing
Sorting
insertion
selection
heap
quick
merge
bubble
shell

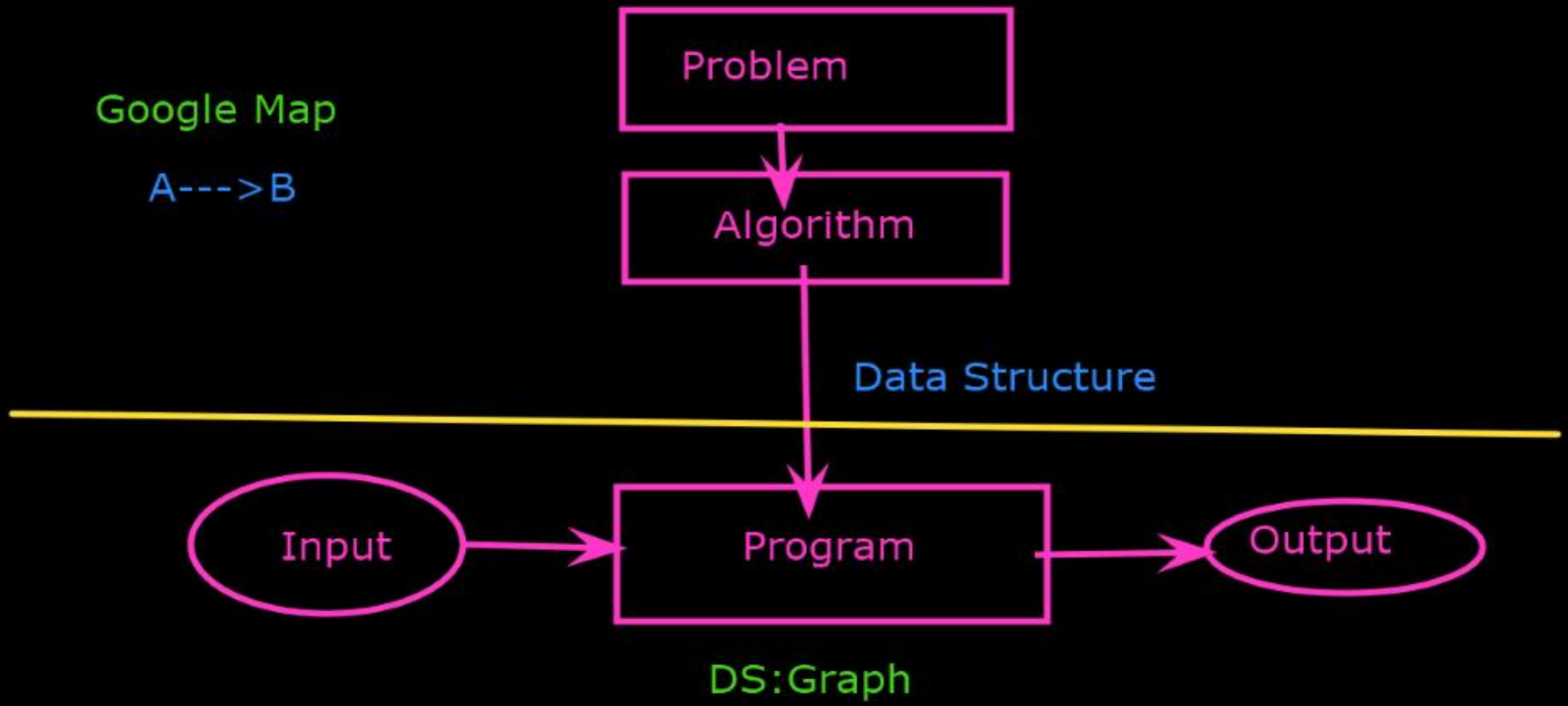
What is Computational Thinking?

- **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

Data structure:Oraganization of data needed to solve the problem.



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

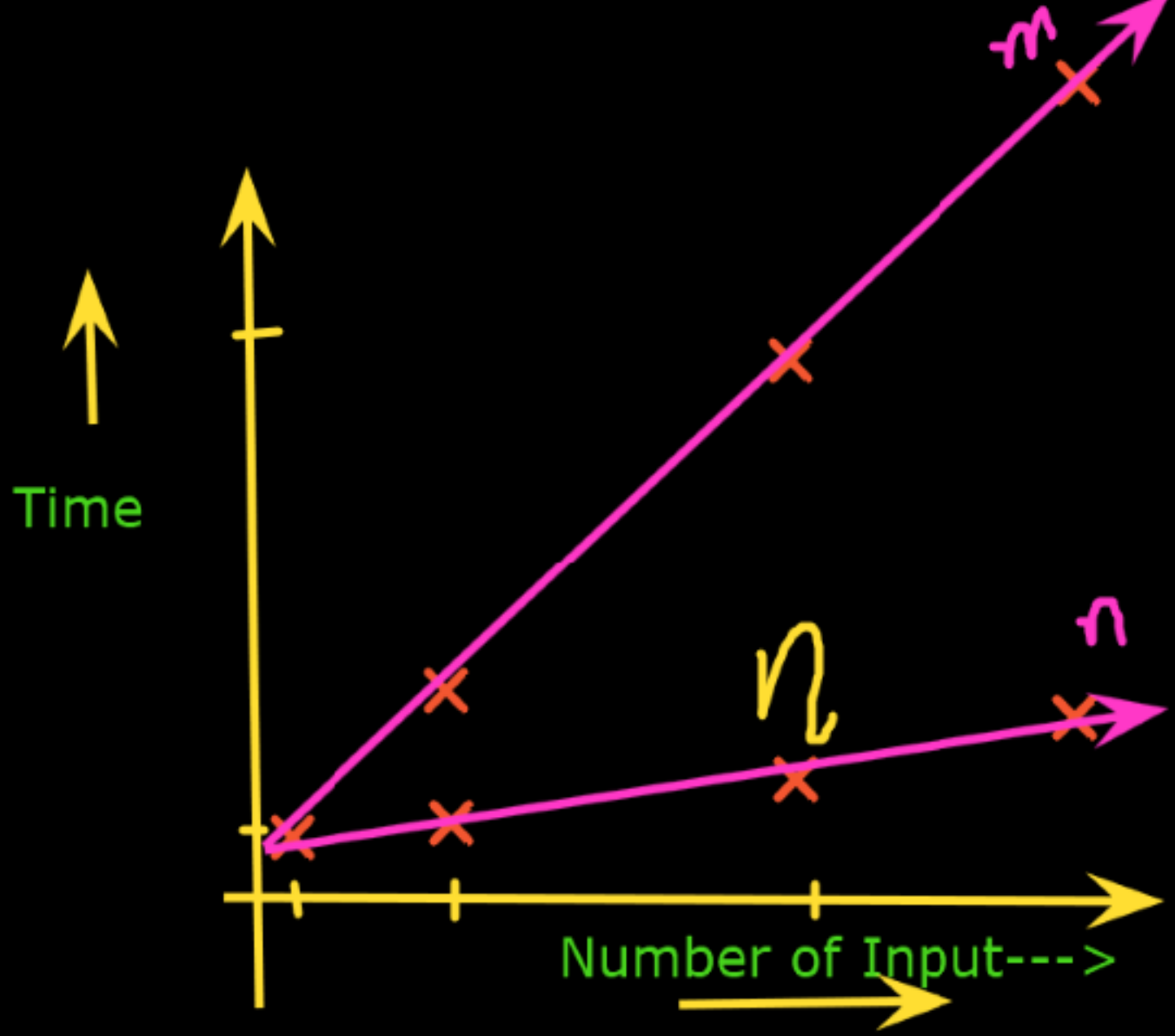
Algorithm

- 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.

Characteristics of algorithm:

-Input
-Output
-Unambiguity
-Finite
-Effective
-Language independent

-scalability of algorithm
-Performance of algorithm



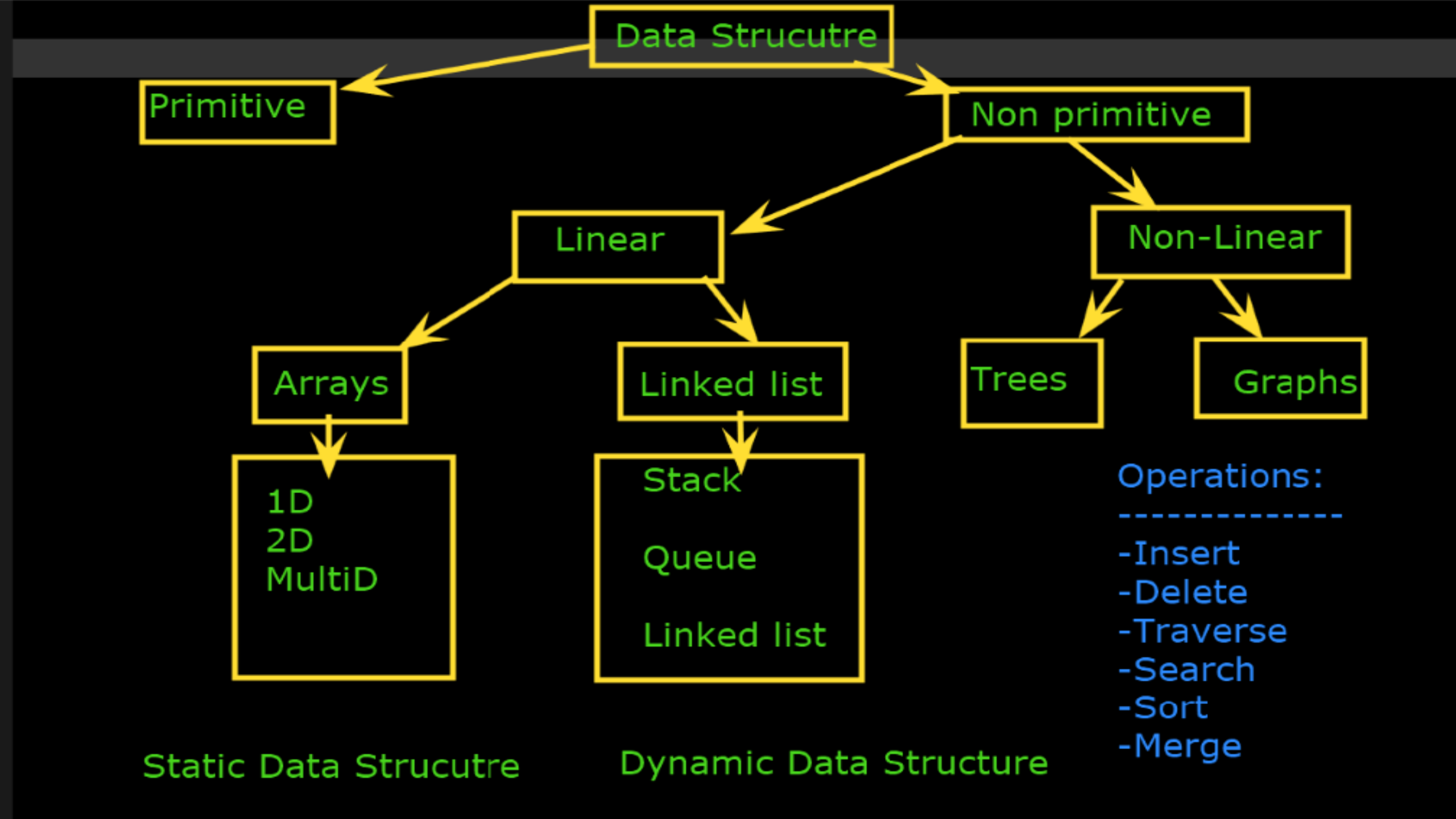
Algorithm stategies:

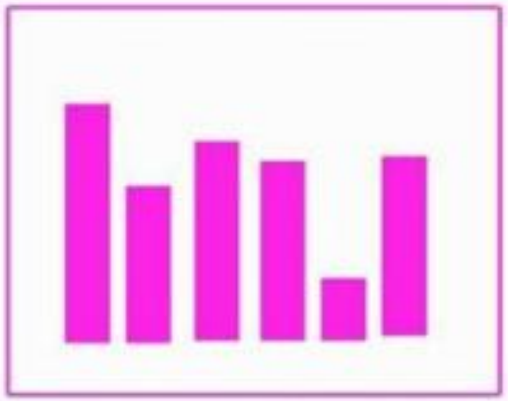
-Brute force
-

Algorithm Design Strategies

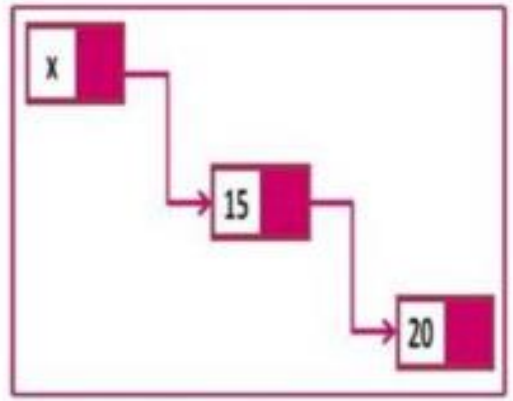
- Brute force
- Divide and conquer
- Decrease and conquer
- Transform and conquer
- Greedy approach
- Dynamic programming
- Backtracking and branch and bound
- Space and time tradeoffs

Invented or applied by many genius in CS

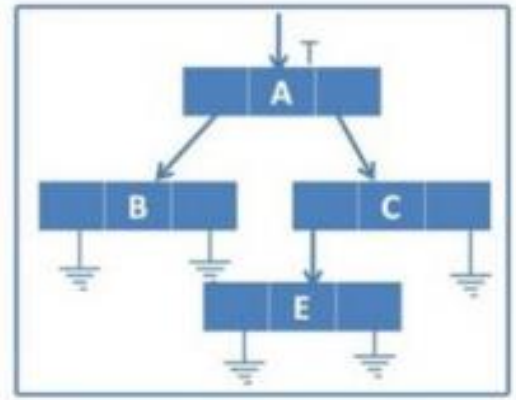

Data Strucutre
Primitive
Non primitive
Linear
Non-Linear
Arrays
Linked list
Trees
Graphs
1D
2D
MultiD
Stack
Queue
Linked list
Operations:
-Insert
-Delete
-Traverse
-Search
-Sort
-Merge
Static Data Strucutre
Dynamic Data Structure



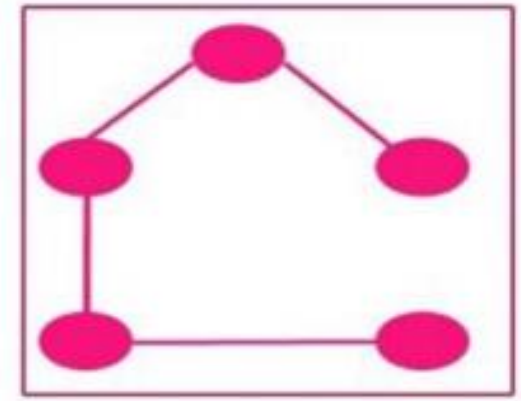
Sorting



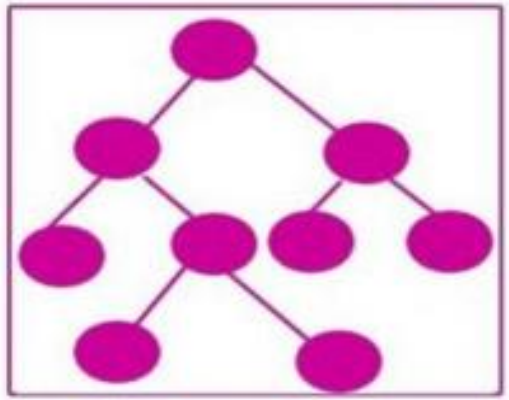
Link list



list

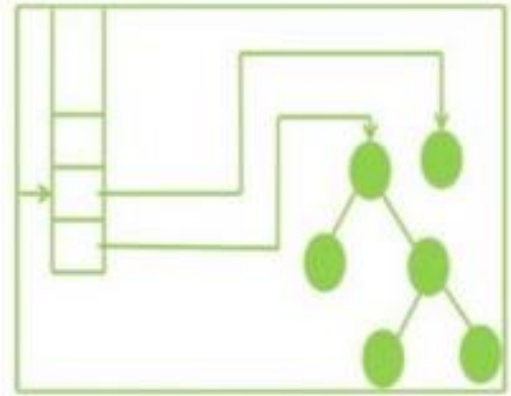


spanning tree

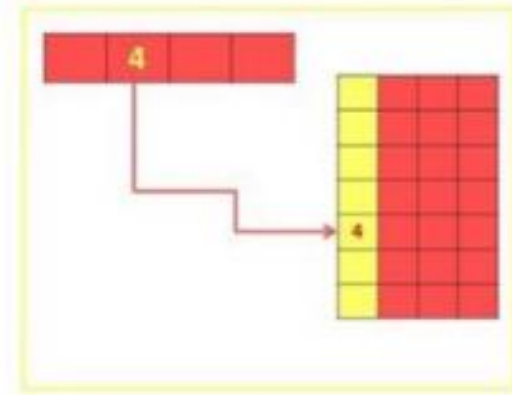


Tree

Graph



Stack



Hashing

By...navinkumardhoprephotography.com

Data Structure:

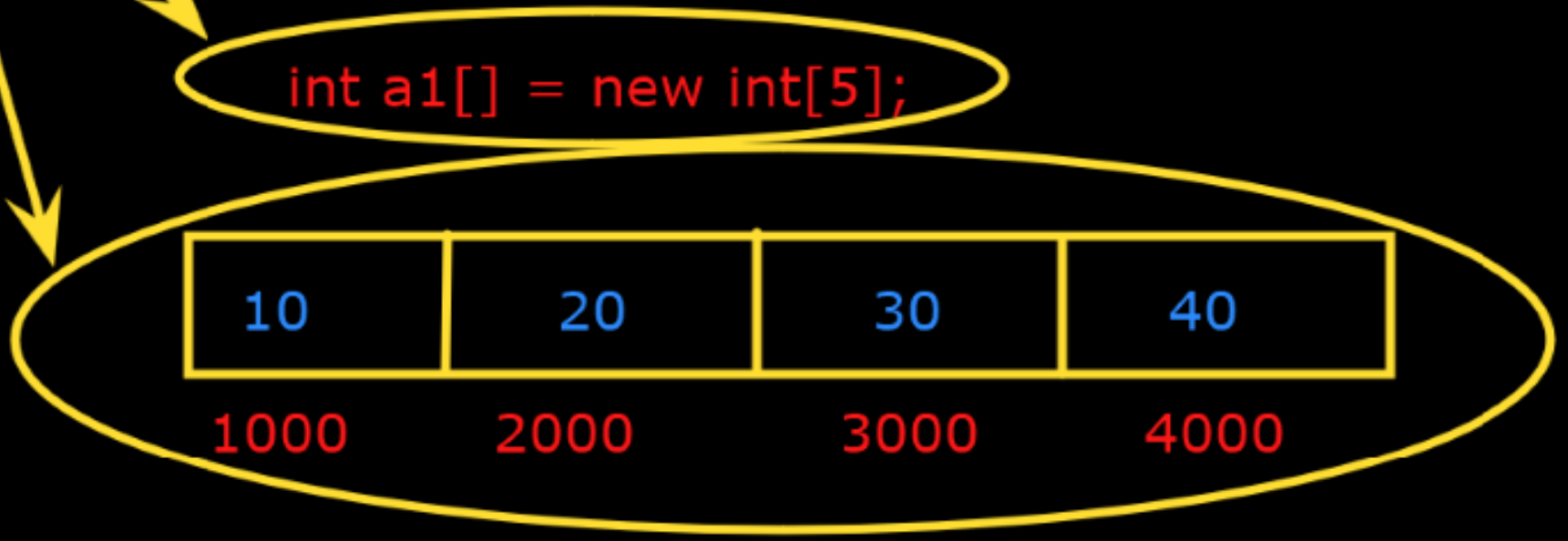
A data structure is a data oragnization, management and storage format that enables efficient access and modification.

-2 ways for Data structure

- -Logical view / Abstract view
- -Physical view / Implementation view

Abstract Data Type: (ADT)

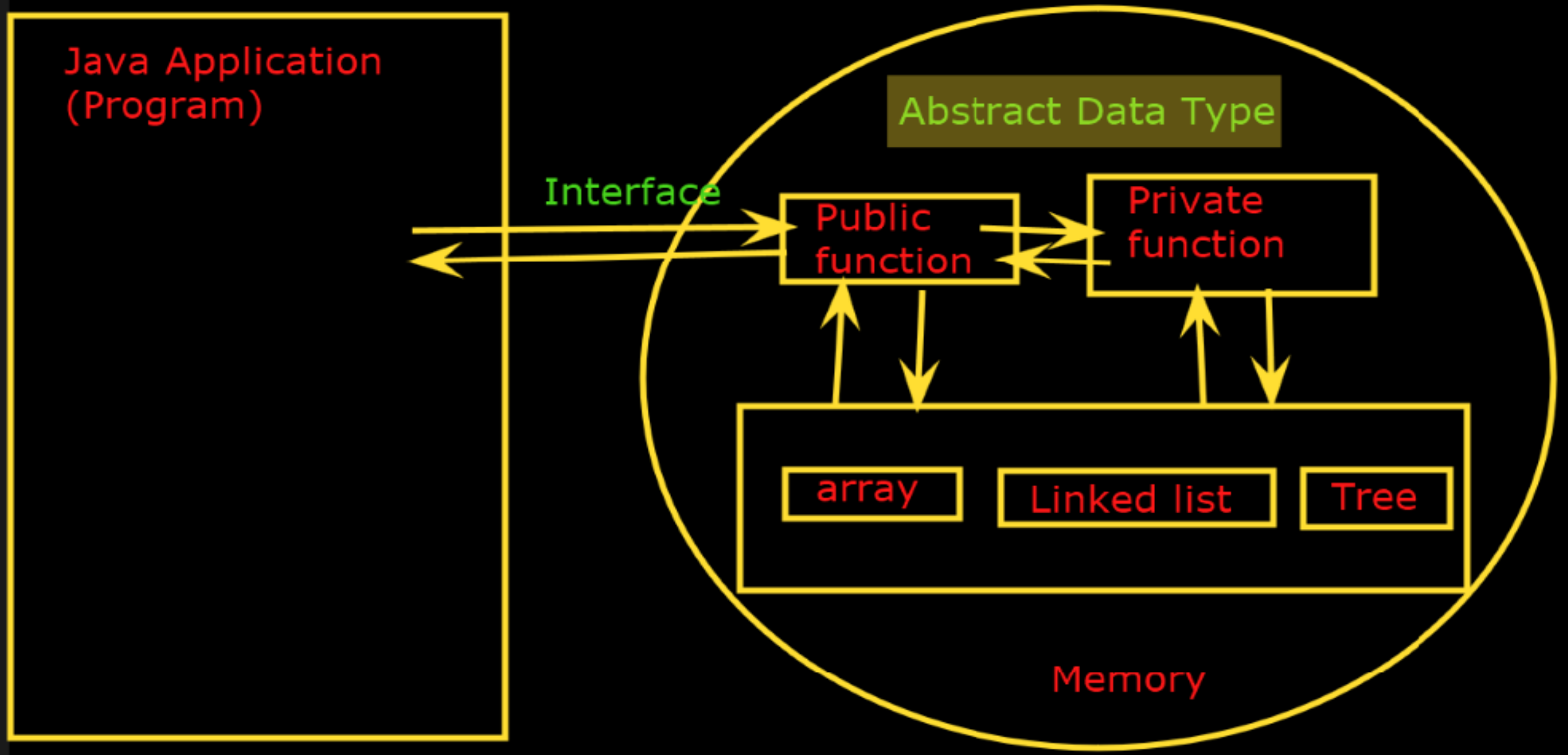
-objects whose behaviour is defined by a set of value and set of operations.

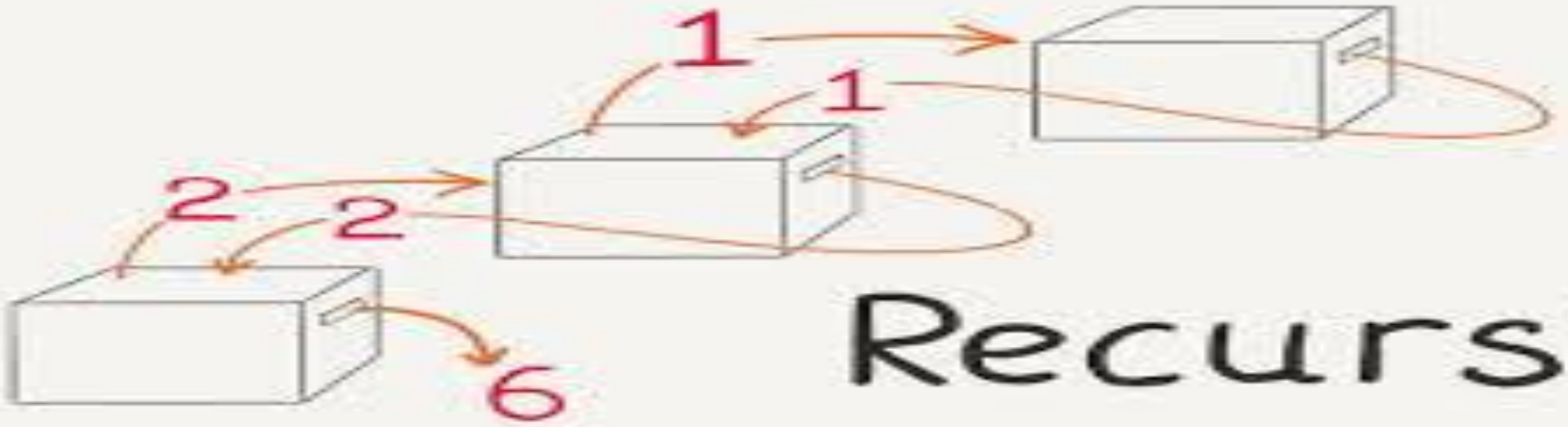


Abstract Data Type (ADT)

Abstract Data Type: (ADT)

-objects whose behaviour is defined by a set of value and set of operations.





Topics

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

Recursion:

```
void fun()
{
    -----
    -----
    fun();
    ------
    -----
}

int main()
{
    -----
    ----
    fun();
    -----
    ---
}
```

Recursion

function : recursive function

copy of a function

fun():3

fun():2

fun():1

main()

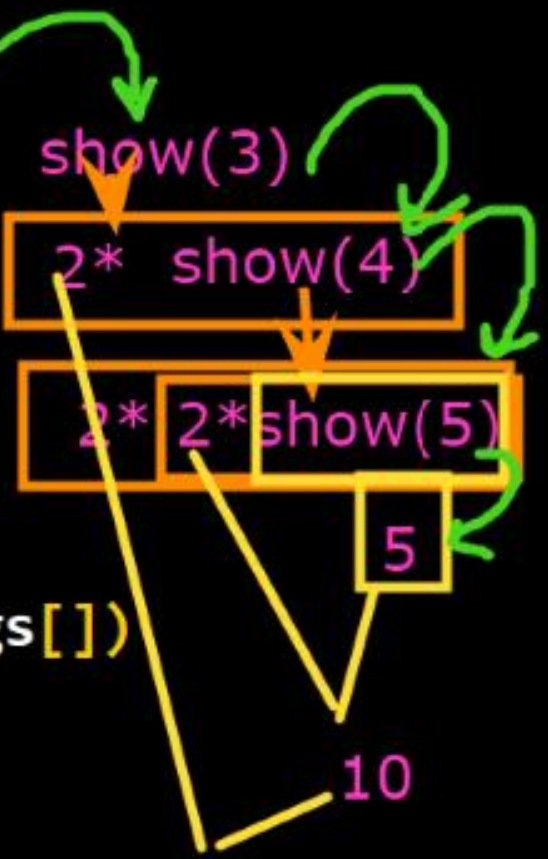
Outline of a Recursive Function

```
if (answer is known)
    provide the answer & exit
else
    call same function with
    a smaller version
    of the same problem
```

base case

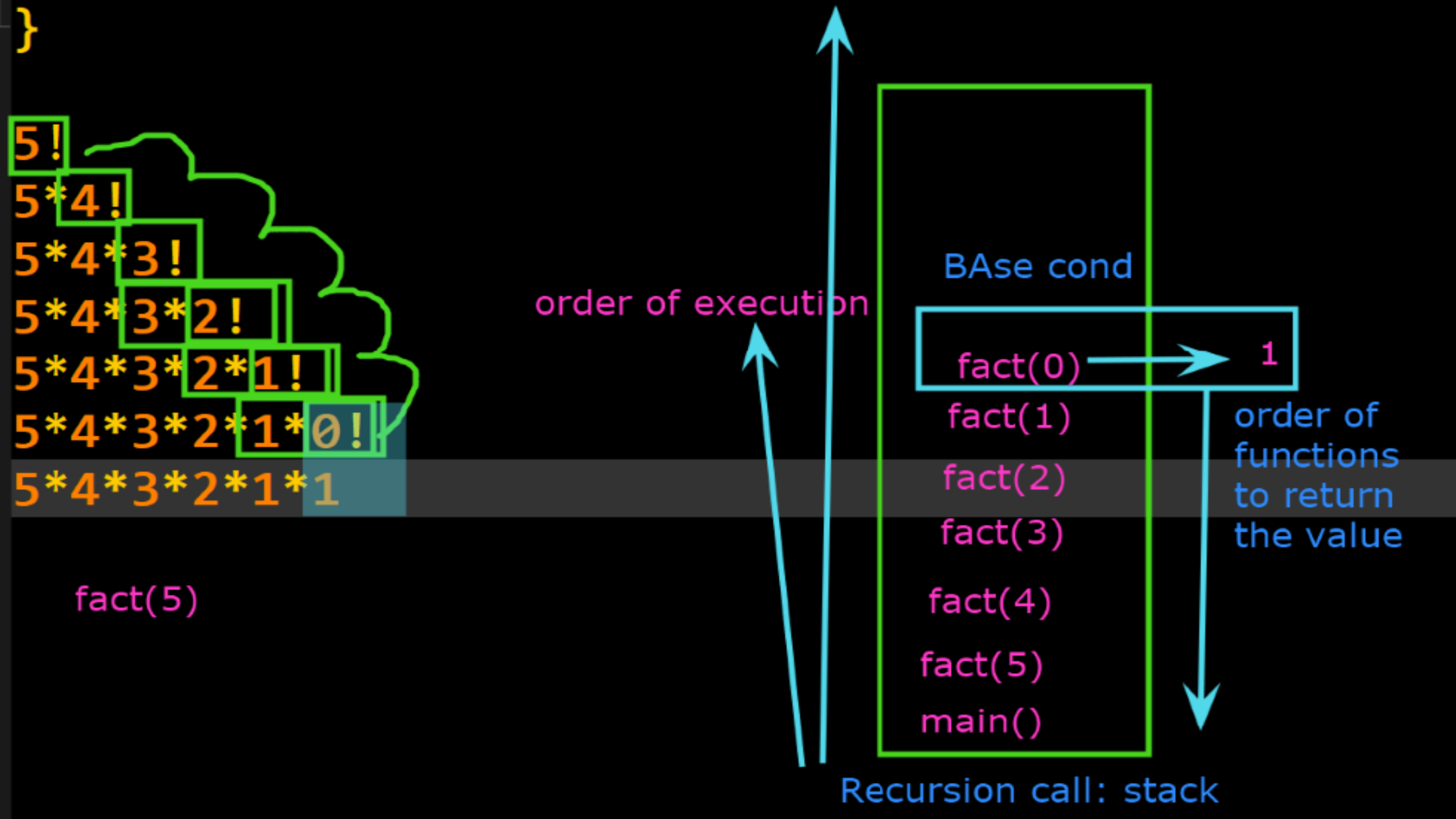
recursive case

```java
class Recursion2
{

    static int show(int n)
    {
        if(n==5)//base condition
            return n;   5
        else
            return 2*show(n+1);

    }

public static void main(String args[])
{
    System.out.println( show(3));
}
}
```

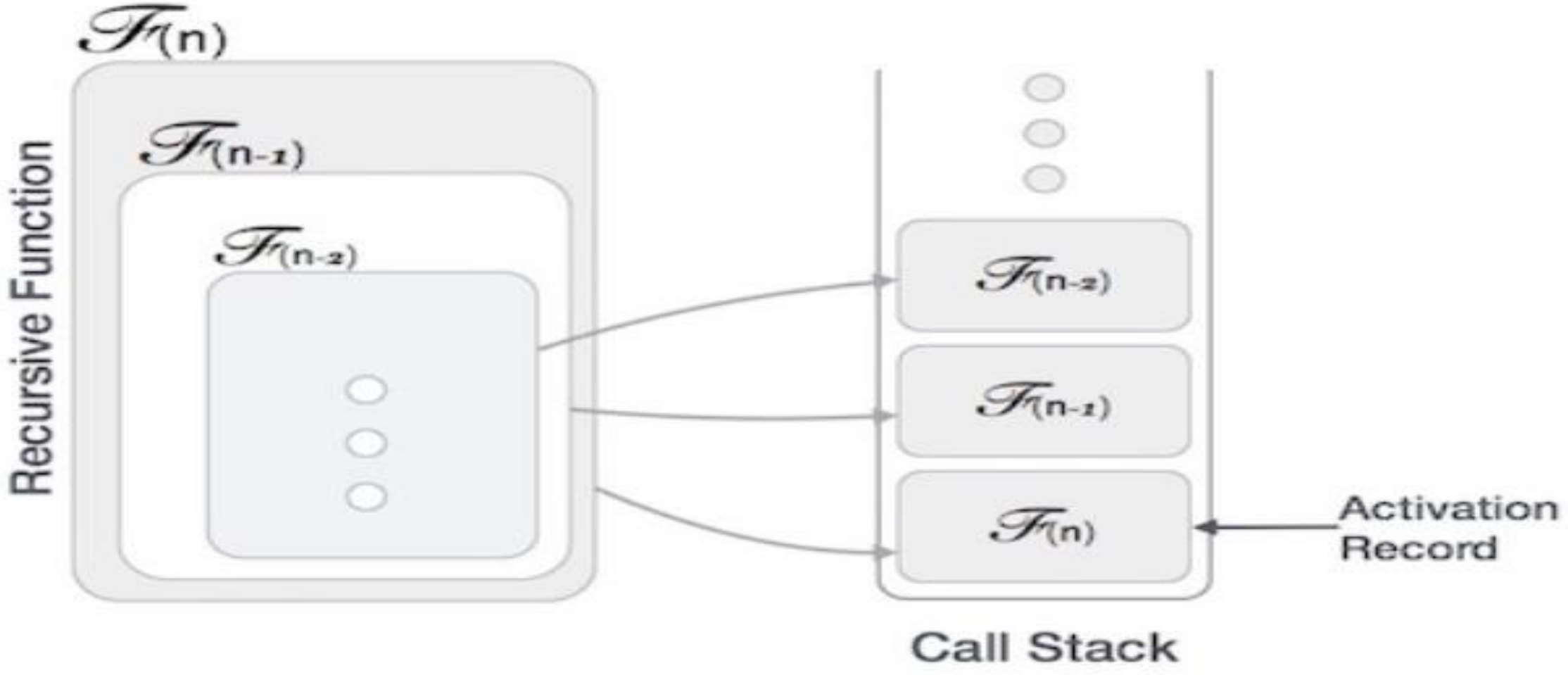


Command Prompt

```
C:\Test>java Recursion1
Happy diwali !!!
Happy diwali !!!
Happy diwali !!!
Happy diwali !!!
Happy diwali !!!

C:\Test>javac Recursion2.jav

C:\Test>java Recursion2

C:\Test>javac Recursion2.jav

C:\Test>java Recursion2
20

C:\Test>
```


}
5!
5*4!
5*4*3!
5*4*3*2!
5*4*3*2*1!
5*4*3*2*1*0!
5*4*3*2*1*1
fact(5)
order of execution
BAse cond
fact(0)
1
fact(1)
fact(2)
fact(3)
fact(4)
fact(5)
main()
order of functions to return the value
Recursion call: stack

How Data Structure Recursive function is implemented?

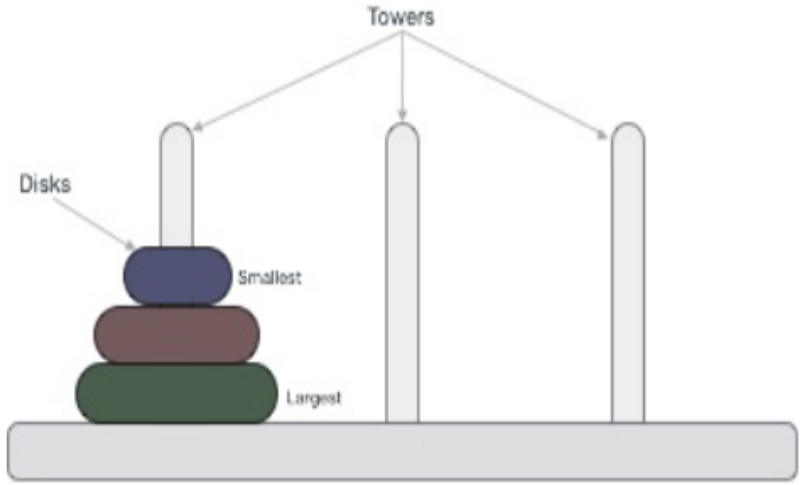


What is Tower of Hanoi?

- 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.



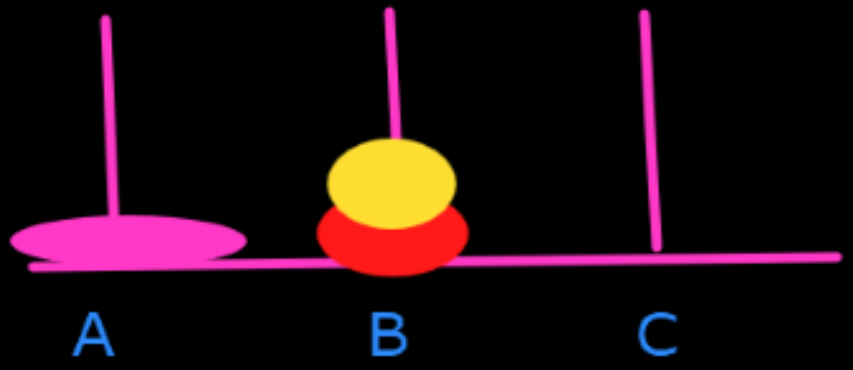
What are the rules to be followed by Tower of Hanoi?

- **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.

```
    {
        if(n==1)
            System.out.println(s+" to "+d);
        else
        {
            toh(n-1,s,d,inter);
            System.out.println(s+" to "+d);
            toh(n-1,inter,s,d);


        }
    }
    public static void main(String args[])
    {
        int n=7;
        toh(n,'A','B','C');
    }
}
```



A-C
A-B
C-B
A-C
B-A
B-C
A-C