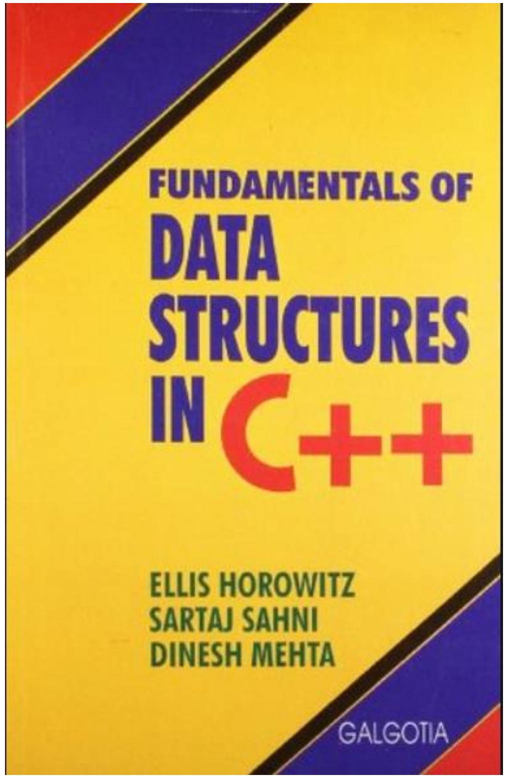
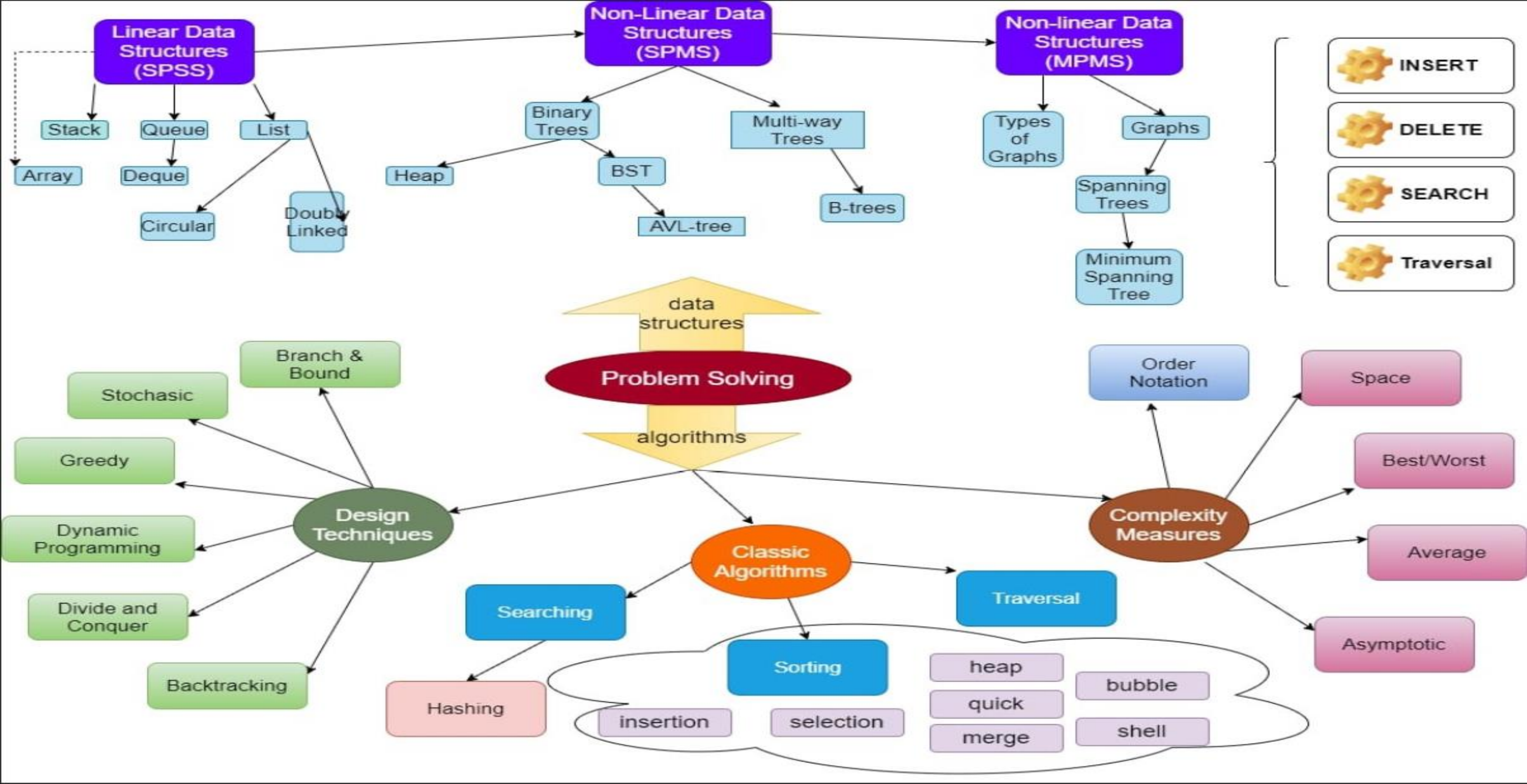
# Algorithms & Data Structure : Day 1

Kiran Waghmare

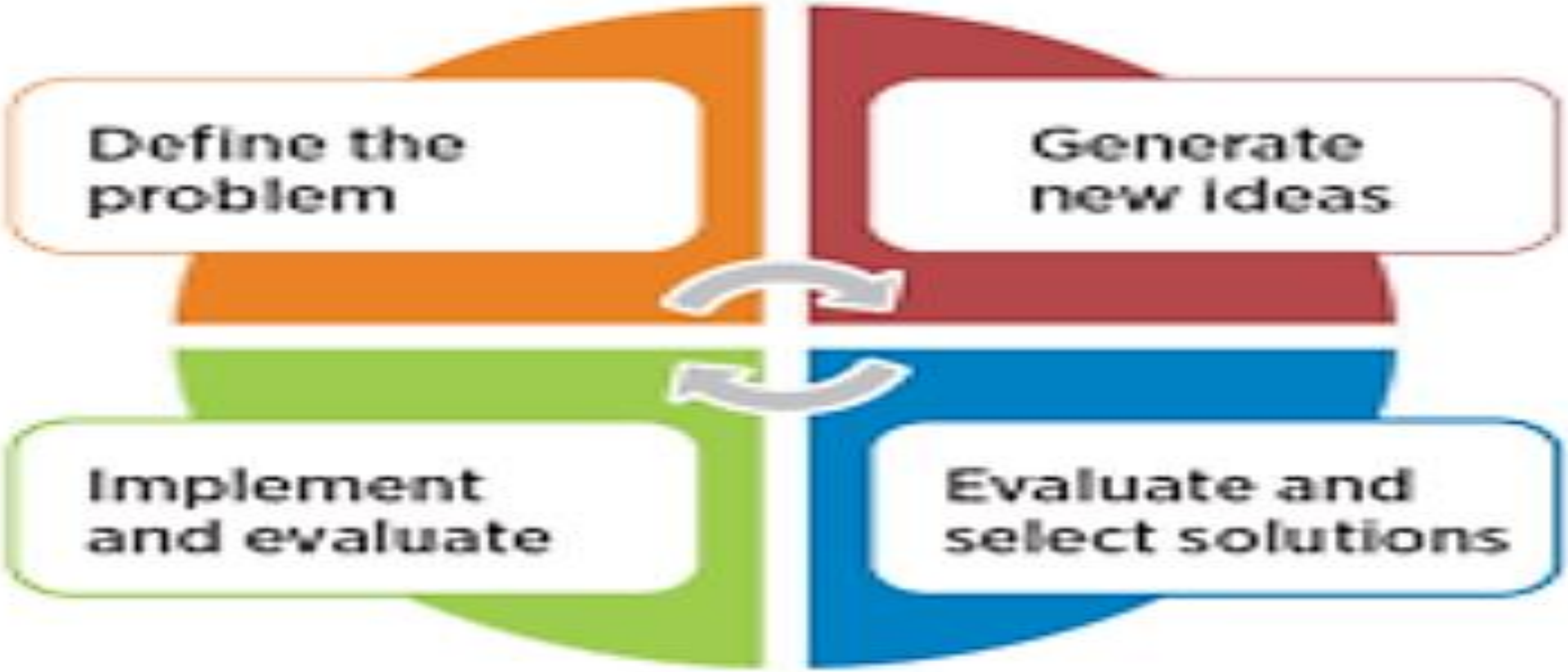
# Module 2: Algorithms and Data Structures

- **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





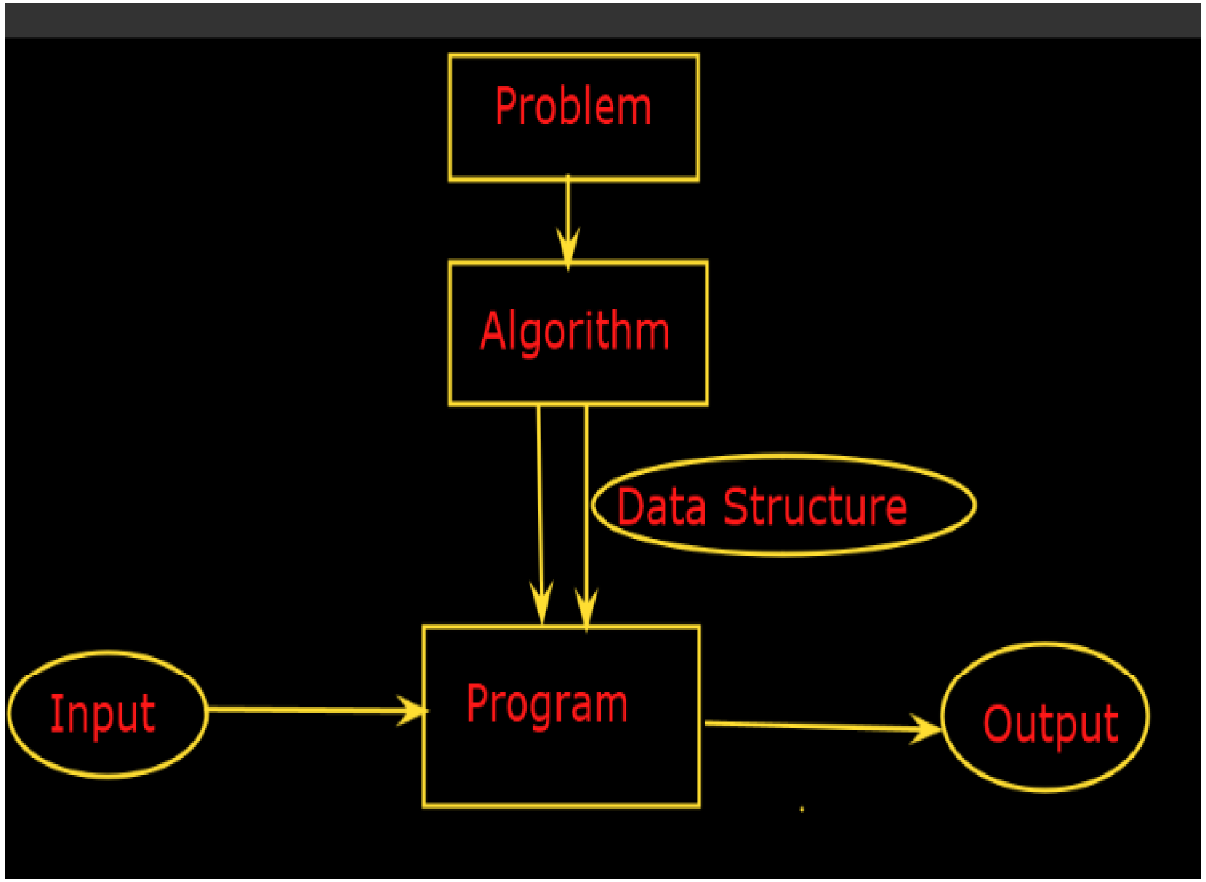
**Mind Map by Dr. M Sasikumar, CDAC Mumbai**



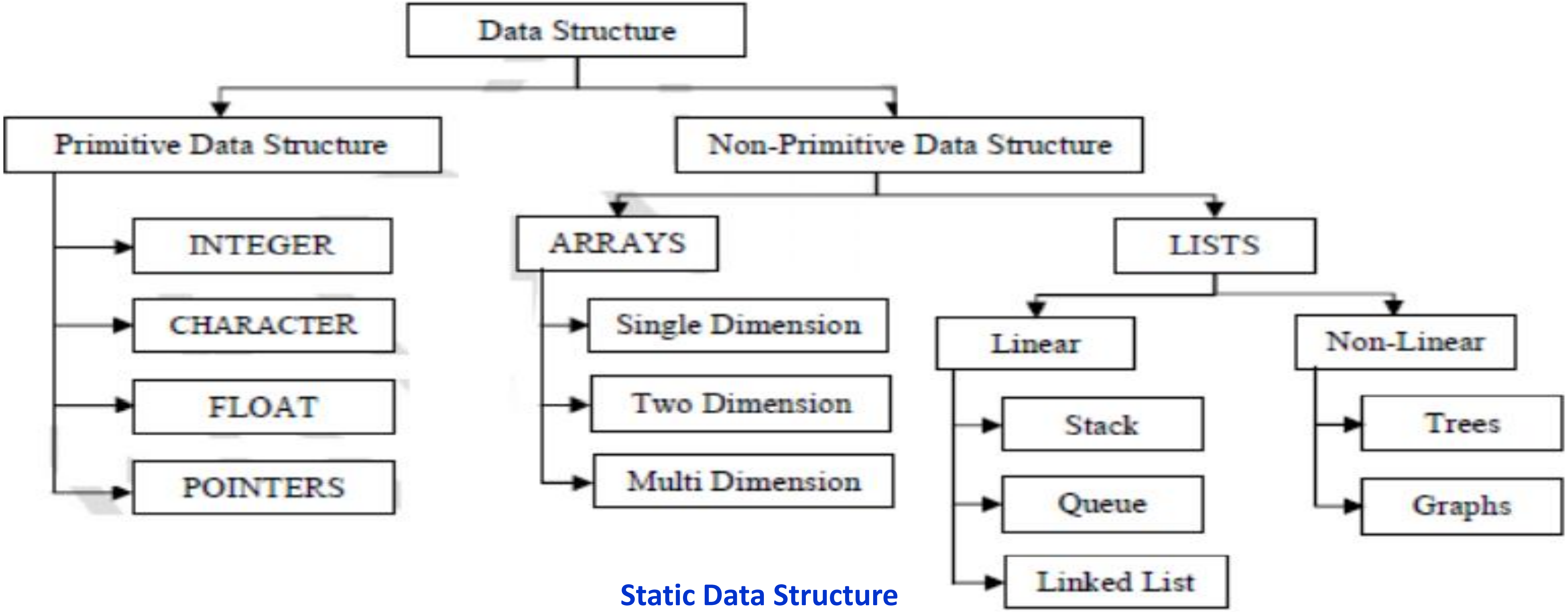
Problem Solving Chart

# 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



# Classification of Data Structure



Data Structure

Primitive Data Structure

INTEGER

CHARACTER

FLOAT

POINTERS

Non-Primitive Data Structure

ARRAYS

Single Dimension

Two Dimension

Multi Dimension

LISTS

Linear

Stack

Queue

Linked List

Non-Linear

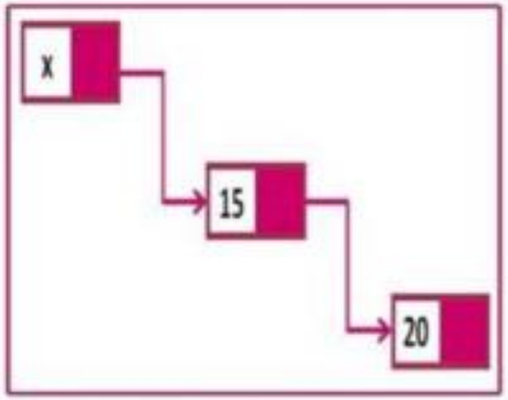
Trees

Graphs

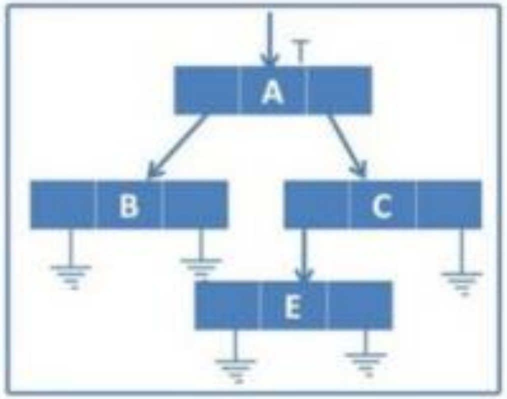
Static Data Structure

Dynamic data structure

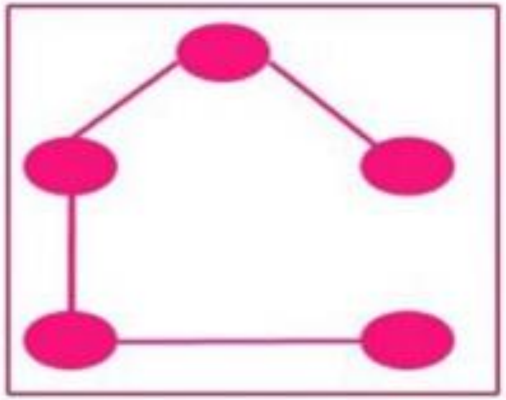
Sorting



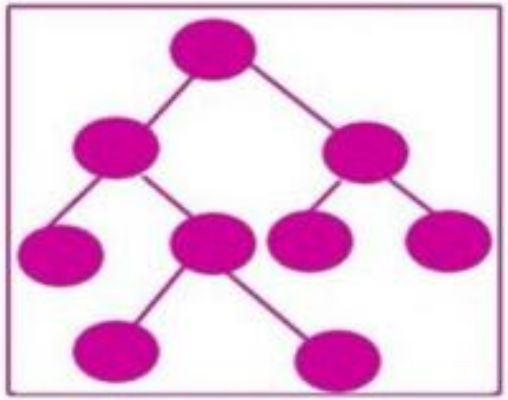
Link list



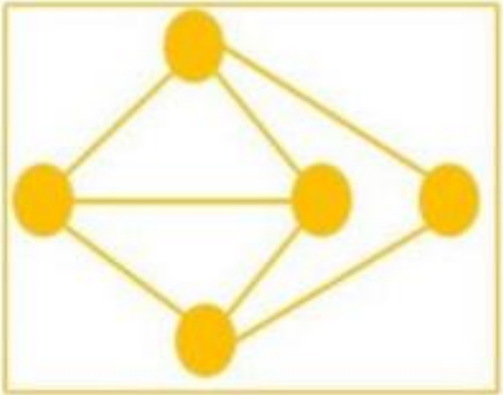
list



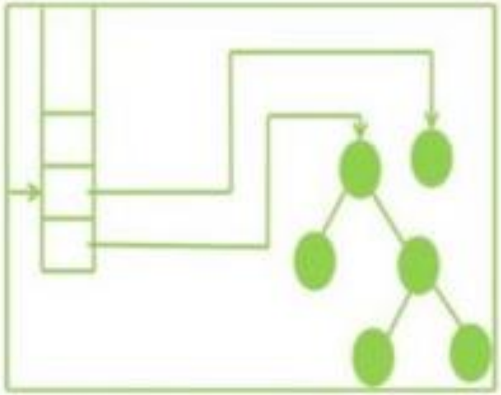
spanning tree



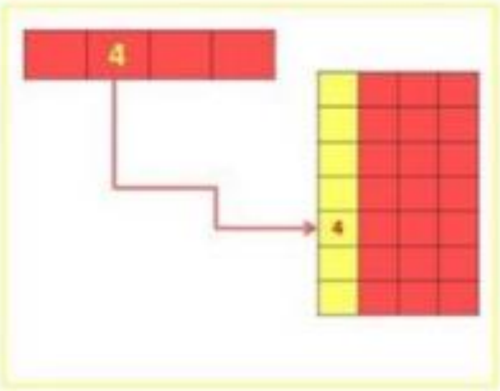
Tree



Graph



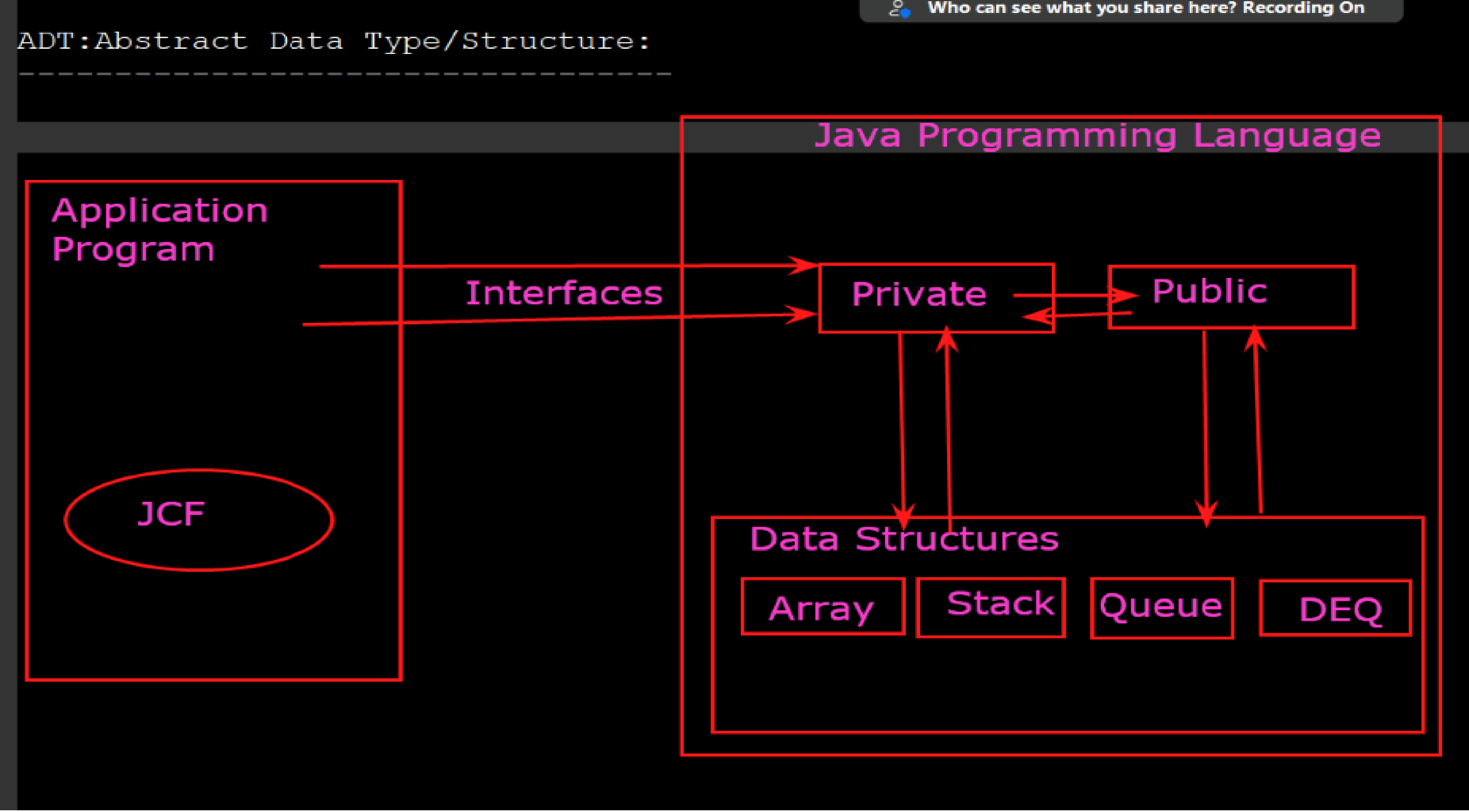
Stack



Hashing

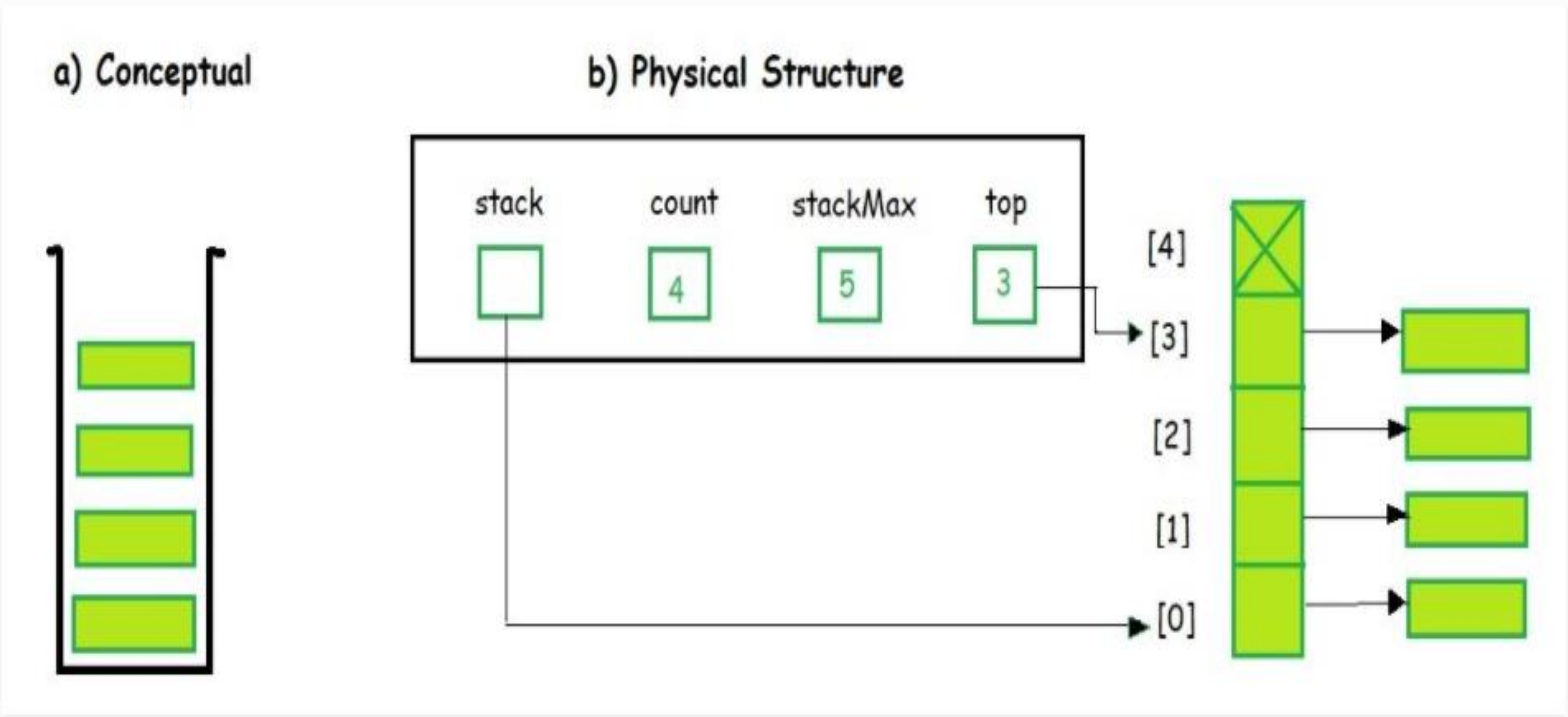
By...navinkumardhoprephotography.com

# Abstract Data Type (ADT)

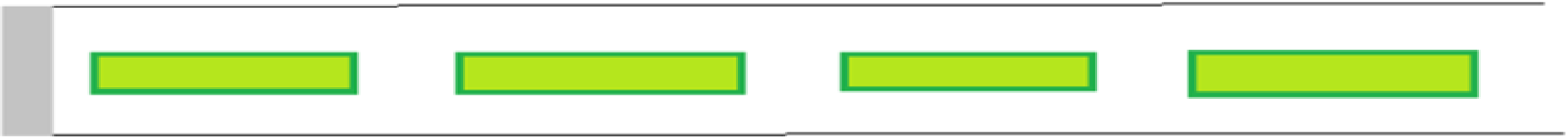


ADT:Abstract Data Type/Structure:
-----------------------------------------

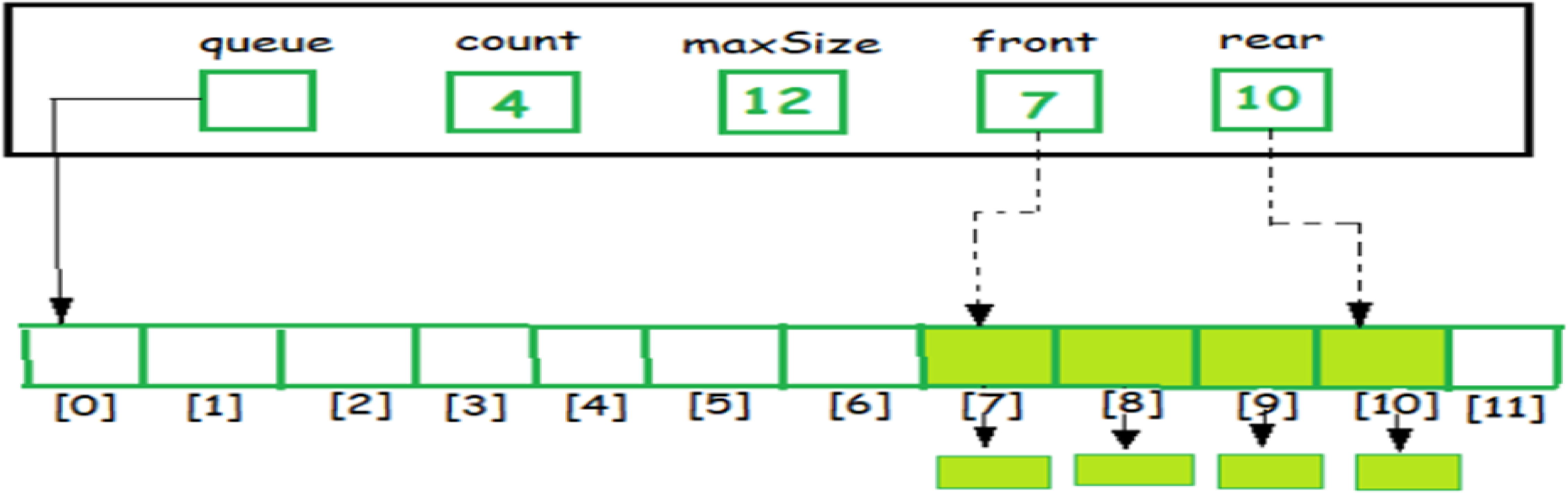
# Stack ADT


a) Conceptual
b) Physical Structure
stack
count
4
stackMax
5
top
3
[4]
[3]
[2]
[1]
[0]

# Queue ADT

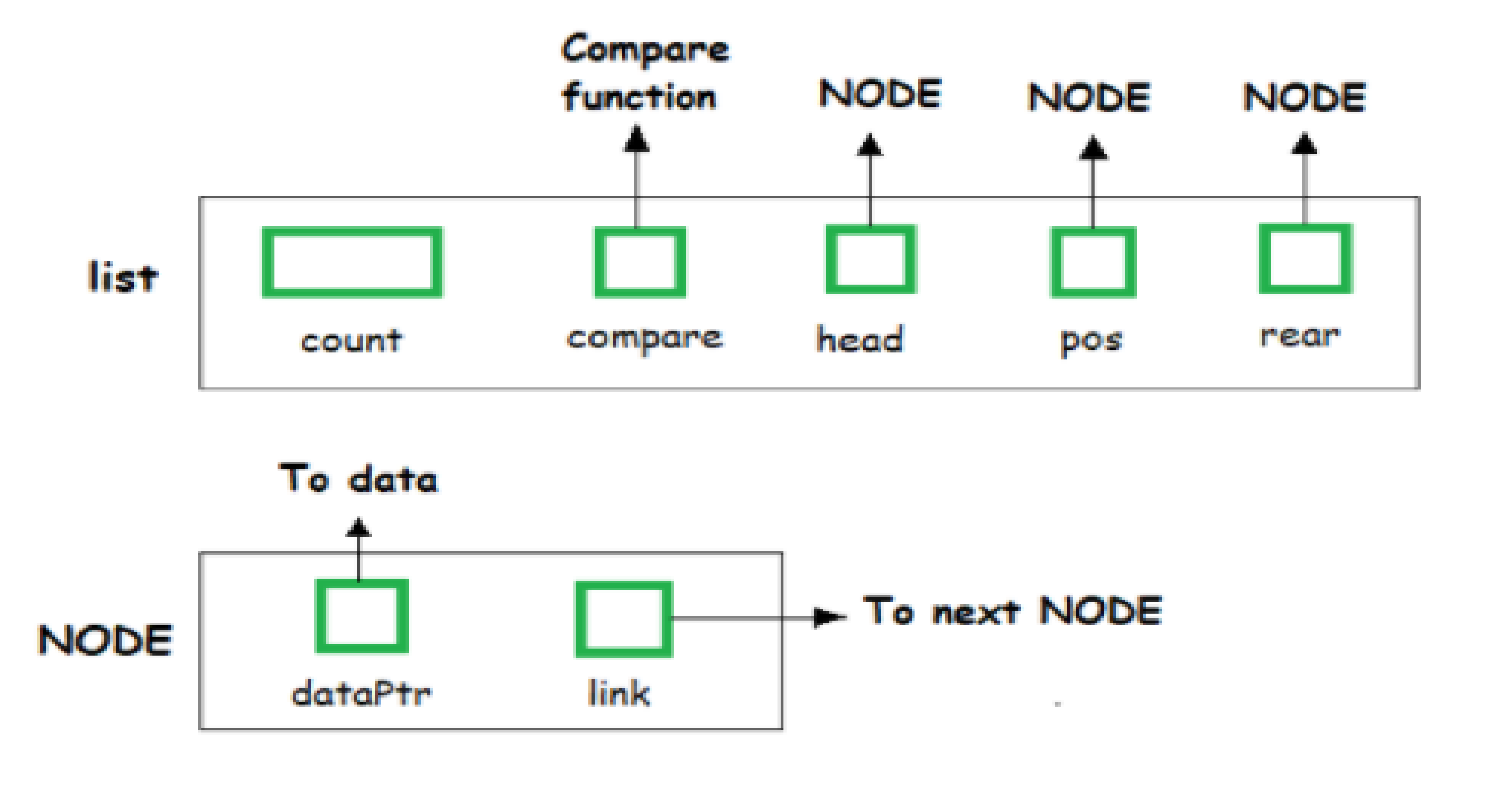


a) Conceptual



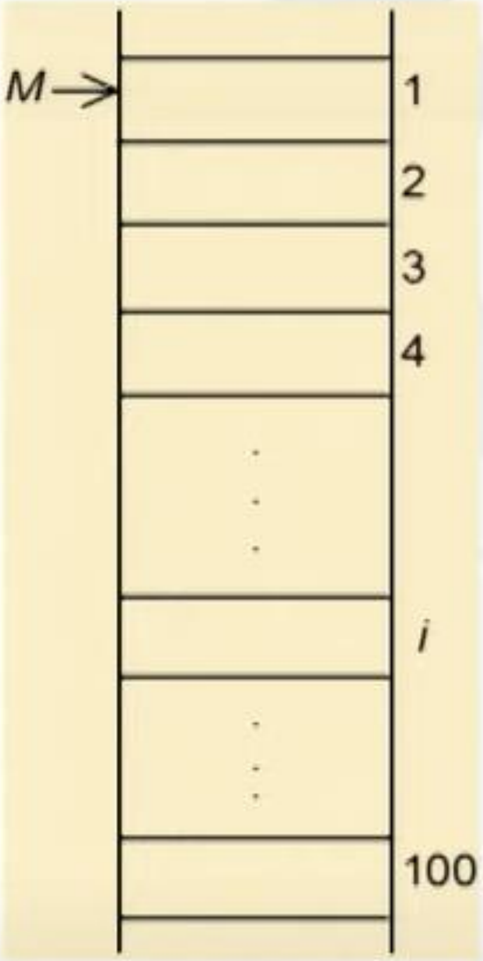
b) Physical Structures

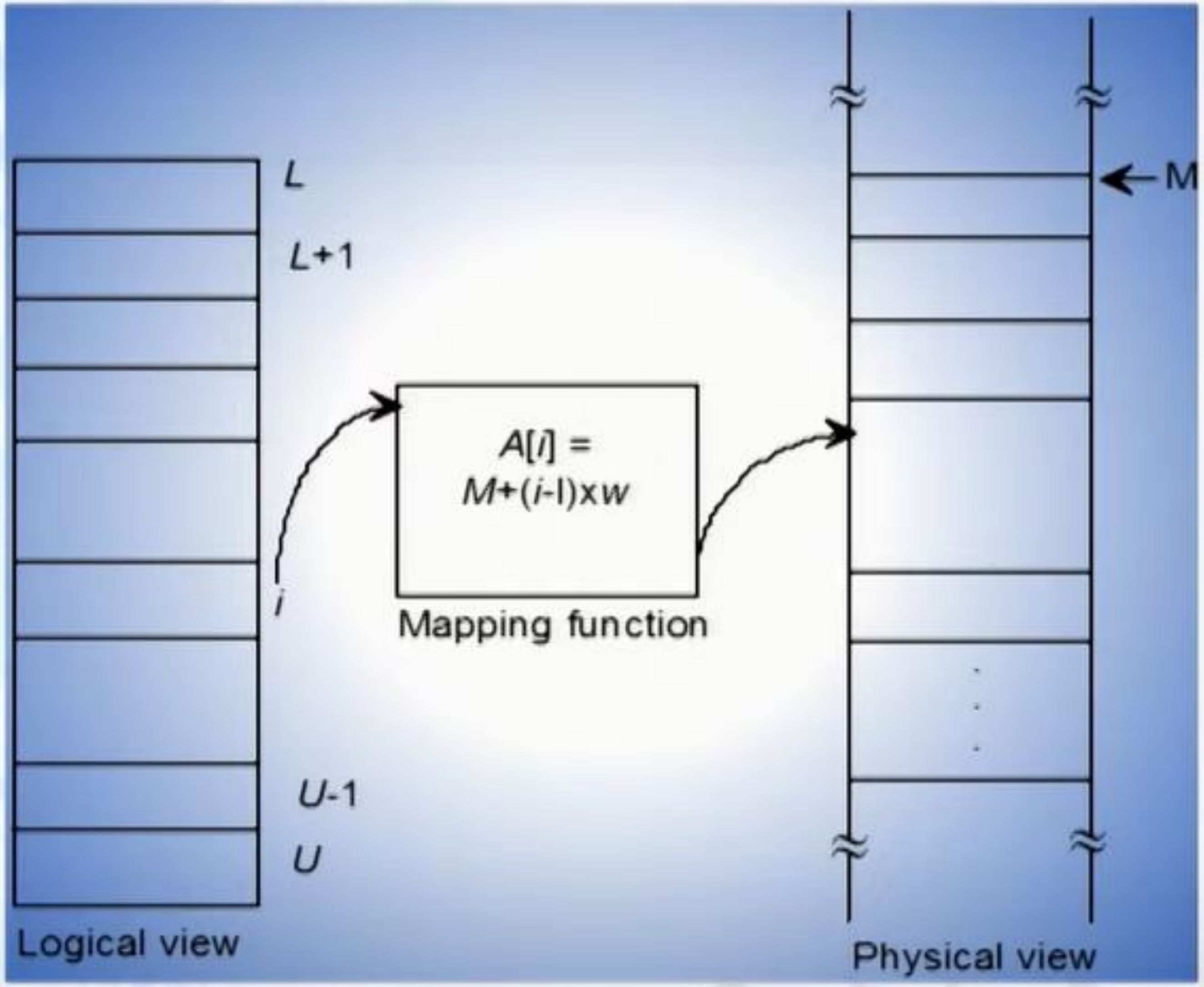
# List ADT


Compare function
NODE
NODE
NODE
list
count
compare
head
pos
rear
To data
NODE
dataPtr
link
To next NODE

# Algorithms & Data Structure
# Arrays

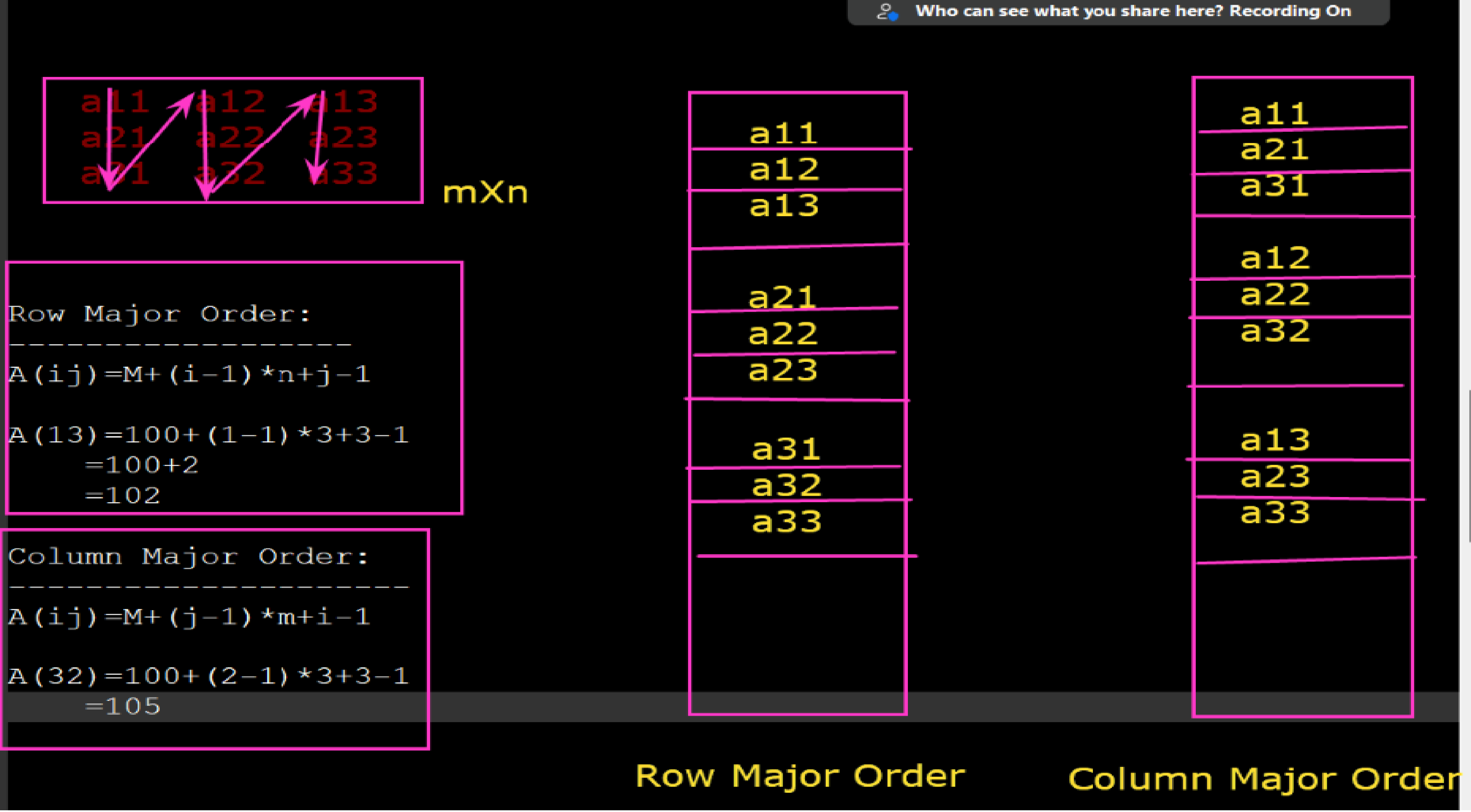
Kiran Waghmare





$$\text{Address}\,(A[i]) = M + (i - L) \times w$$

$$\text{Size}\,(A) = U - L + 1$$



Who can see what you share here? Recording On

| | | |
|---|---|---|
| a11 | a12 | a13 |
| a21 | a22 | a23 |
| a31 | a32 | a33 |

mXn

```
Row Major Order:
------------------
A(ij)=M+(i-1)*n+j-1

A(13)=100+(1-1)*3+3-1
    =100+2
    =102
```

```
Column Major Order:
---------------------
A(ij)=M+(j-1)*m+i-1

A(32)=100+(2-1)*3+3-1
    =105
```

Row Major Order:

a11<br>a12<br>a13<br>a21<br>a22<br>a23<br>a31<br>a32<br>a33

Column Major Order:

a11<br>a21<br>a31<br>a12<br>a22<br>a32<br>a13<br>a23<br>a33

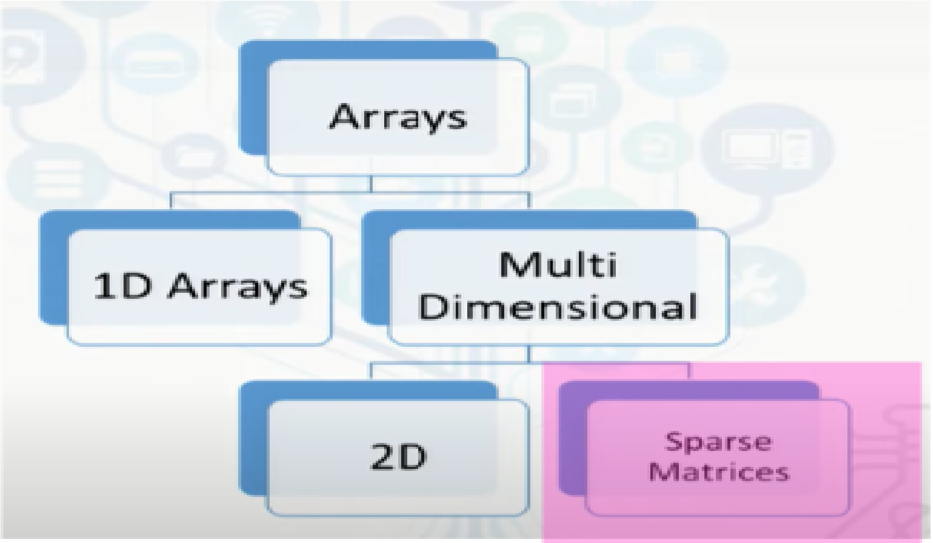
Who can see what you share here? Recording On

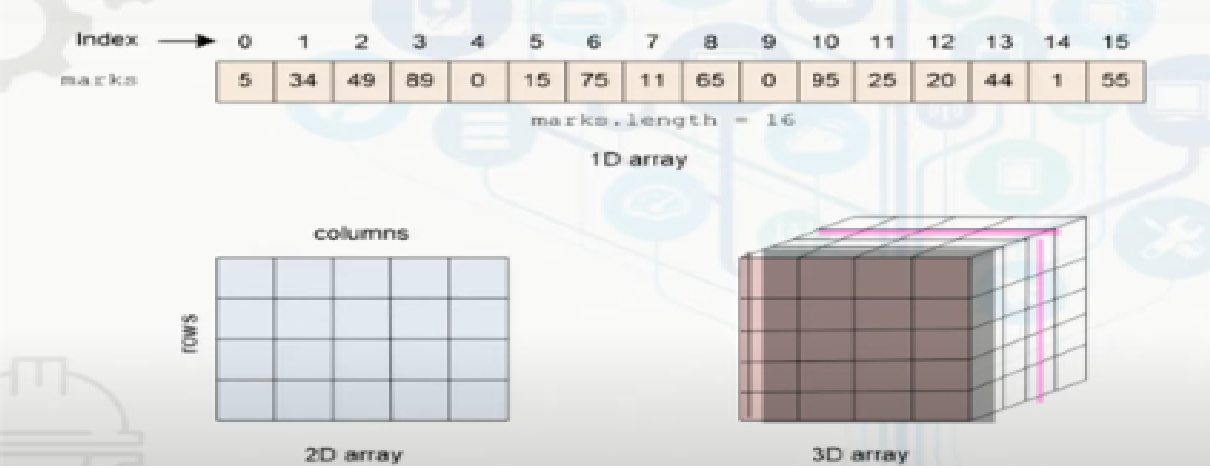
Arrays:
--------
finite
homogeneous

100

| A | R | R | A | Y | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | |

PowerPoint Slide Show - [Day1 Mar22 ADS Session 1.pptx] - PowerPoint





| Index → | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| marks | 5 | 34 | 49 | 89 | 0 | 15 | 75 | 11 | 65 | 0 | 95 | 25 | 20 | 44 | 1 | 55 |

marks.length = 16

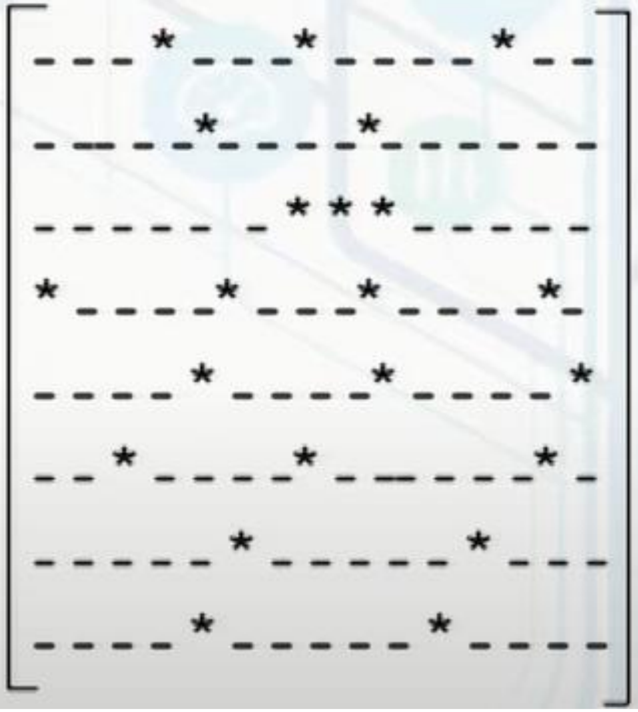
$$\begin{matrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{matrix}$$

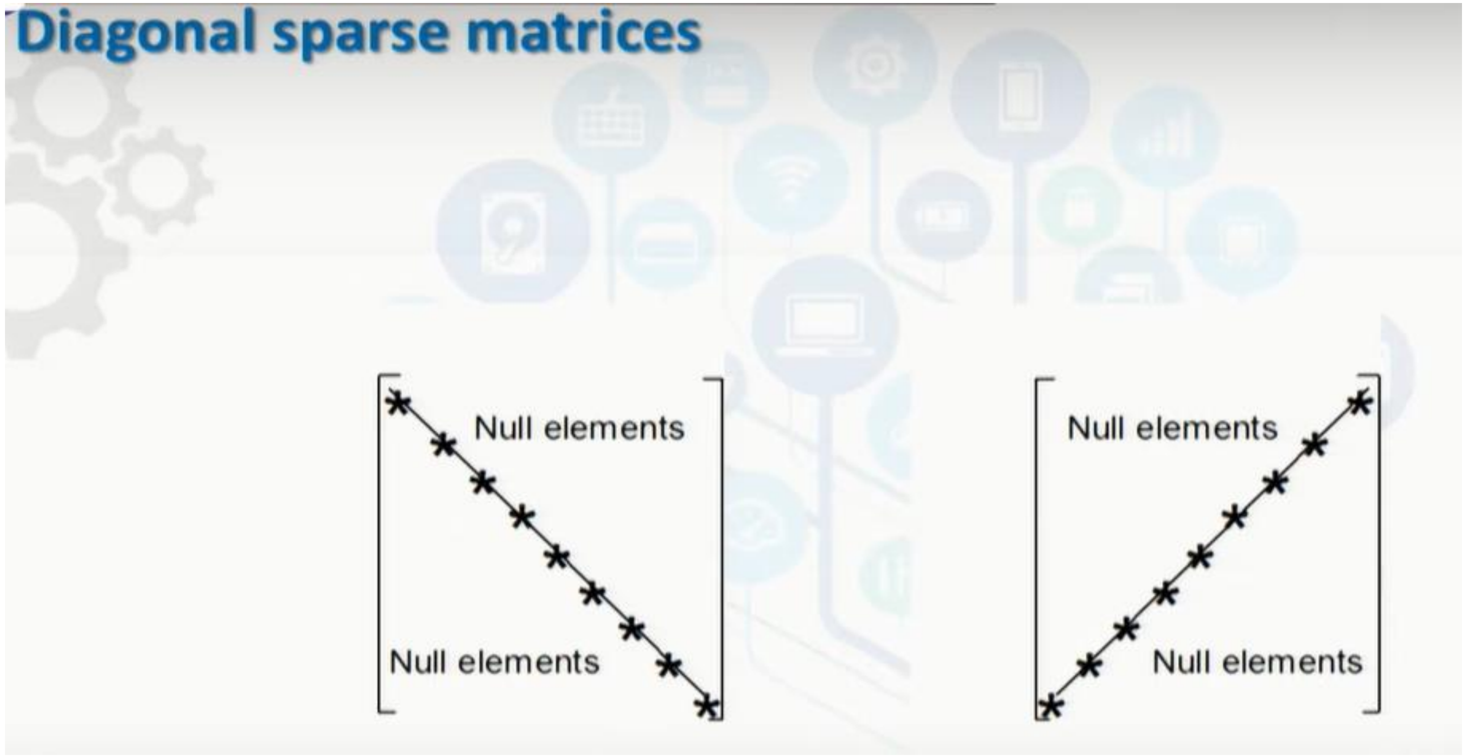
A(2) =100+(2-0)*2 =104

=4-0 + 1=5

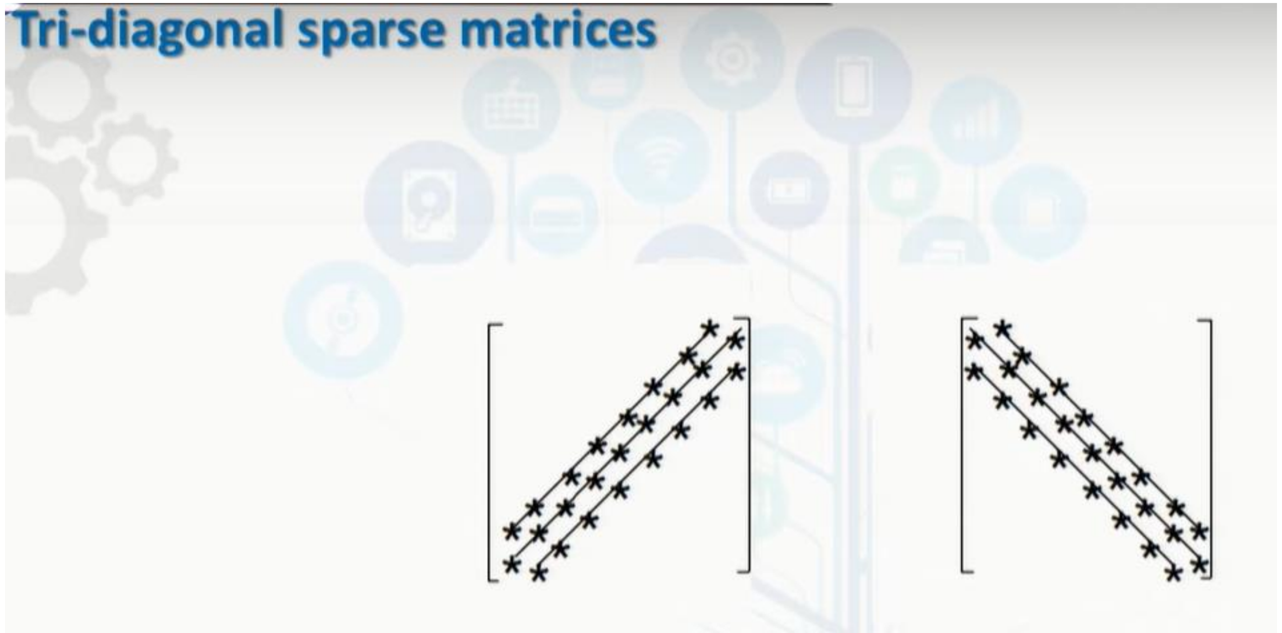
# Sparse matrix

A *sparse* matrix is a two-dimensional array having the value of majority elements as null

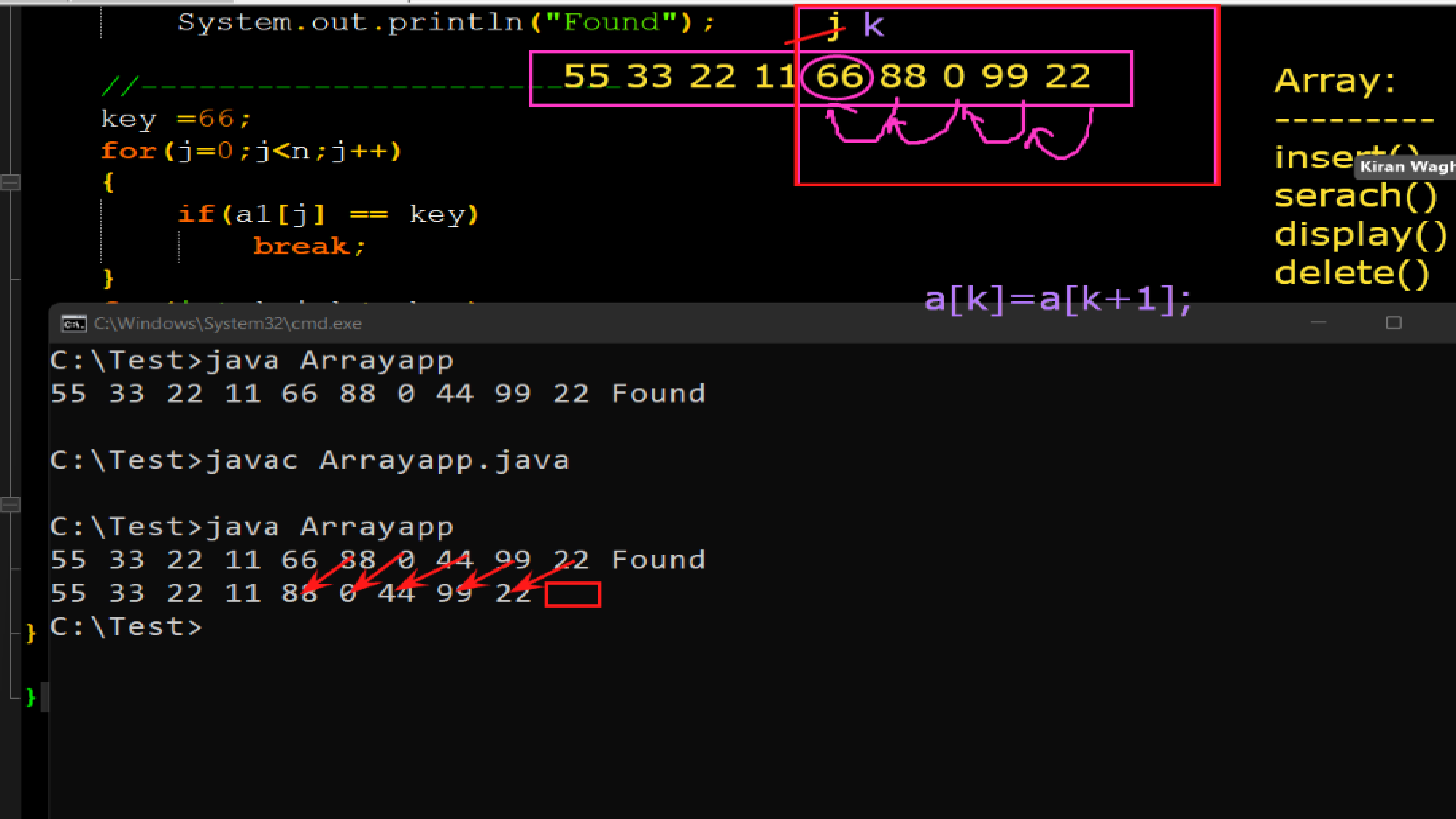




## Diagonal sparse matrices



## Tri-diagonal sparse matrices



```
        System.out.println("Found");

    //------------------------
    key =66;
    for(j=0;j<n;j++)
    {
        if(a1[j] == key)
            break;
    }
```

Array:
---------
insert()
serach()
display()
delete()

a[k]=a[k+1];

C:\Windows\System32\cmd.exe

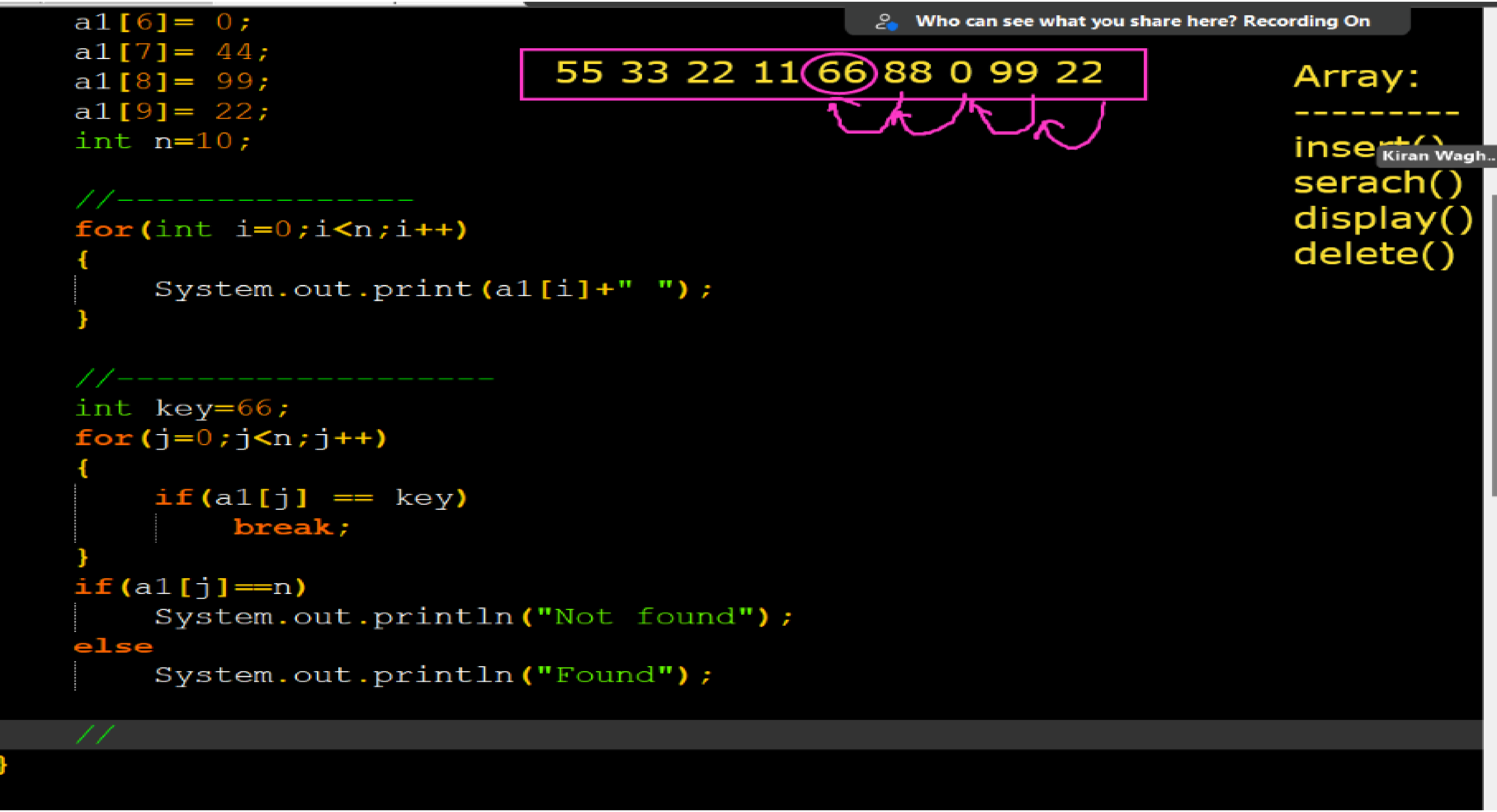
```
C:\Test>java Arrayapp
55 33 22 11 66 88 0 44 99 22 Found

C:\Test>javac Arrayapp.java

C:\Test>java Arrayapp
55 33 22 11 66 88 0 44 99 22 Found
55 33 22 11 88 0 44 99 22
C:\Test>
```

```
}

}
```



55 33 22 11 66 88 0 99 22

Array:
---------
insert()
serach()
display()
delete()

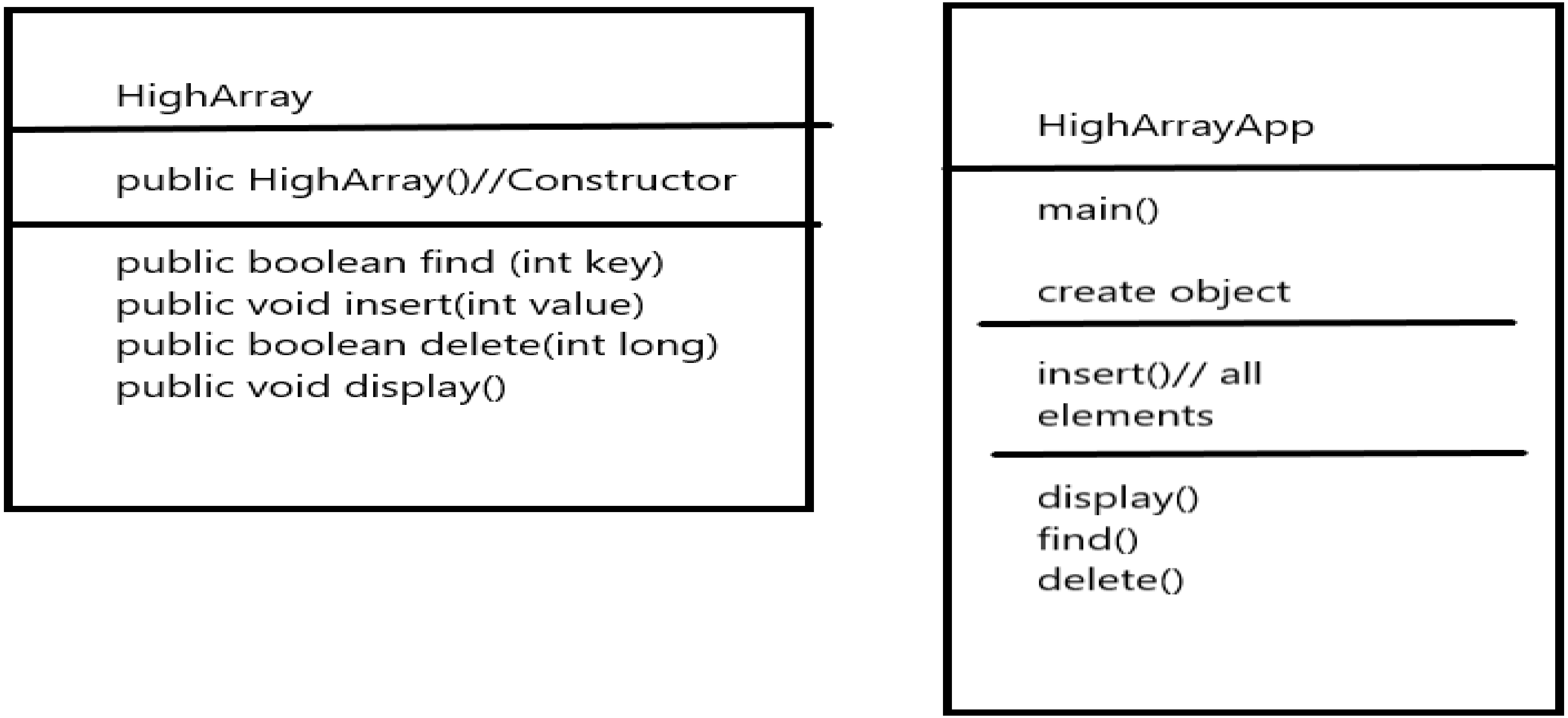
```
    a1[6]= 0;
    a1[7]= 44;
    a1[8]= 99;
    a1[9]= 22;
    int n=10;

    //---------------
    for(int i=0;i<n;i++)
    {
        System.out.print(a1[i]+" ");
    }

    //-------------------
    int key=66;
    for(j=0;j<n;j++)
    {
        if(a1[j] == key)
            break;
    }
    if(a1[j]==n)
        System.out.println("Not found");
    else
        System.out.println("Found");

    //
}
```

# Program 2



HighArray

public HighArray()//Constructor

public boolean find (int key)
public void insert(int value)
public boolean delete(int long)
public void display()

HighArrayApp

main()

create object

insert()// all
elements

display()
find()
delete()