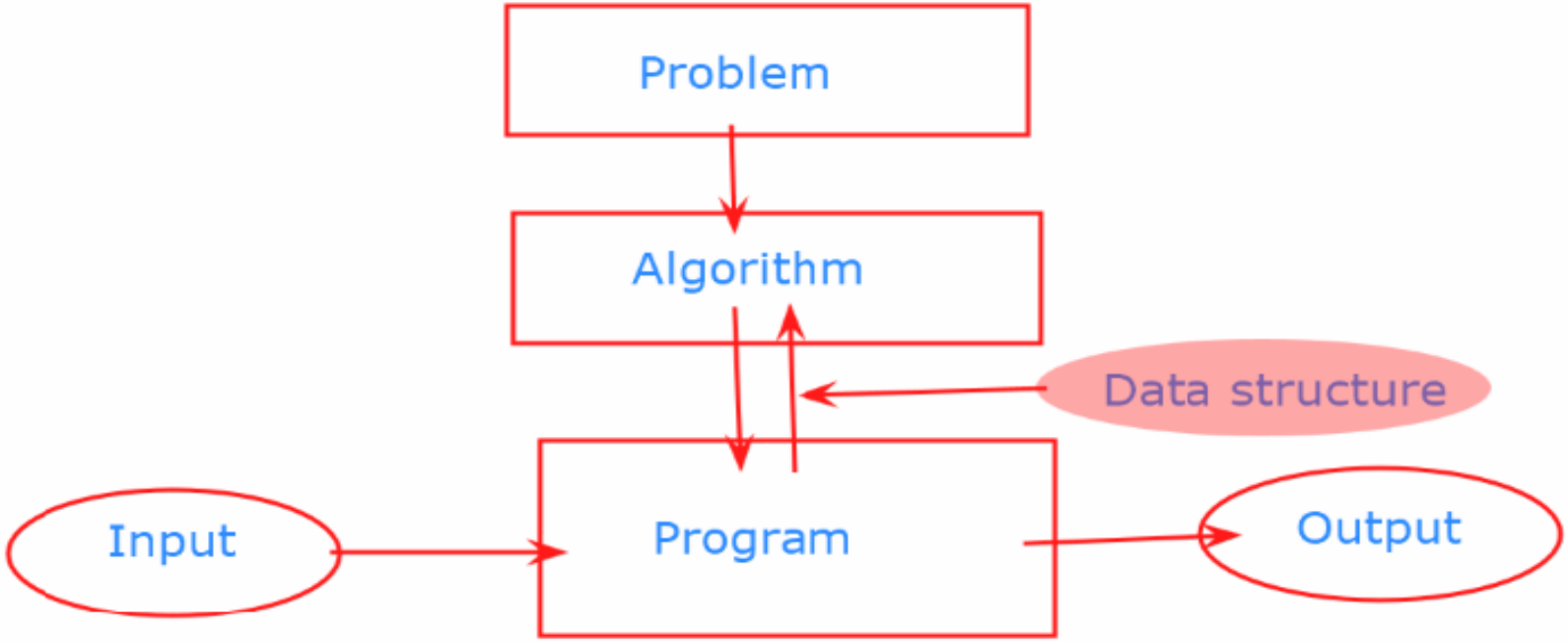
Sep22 : Day 2

Kiran Waghmare

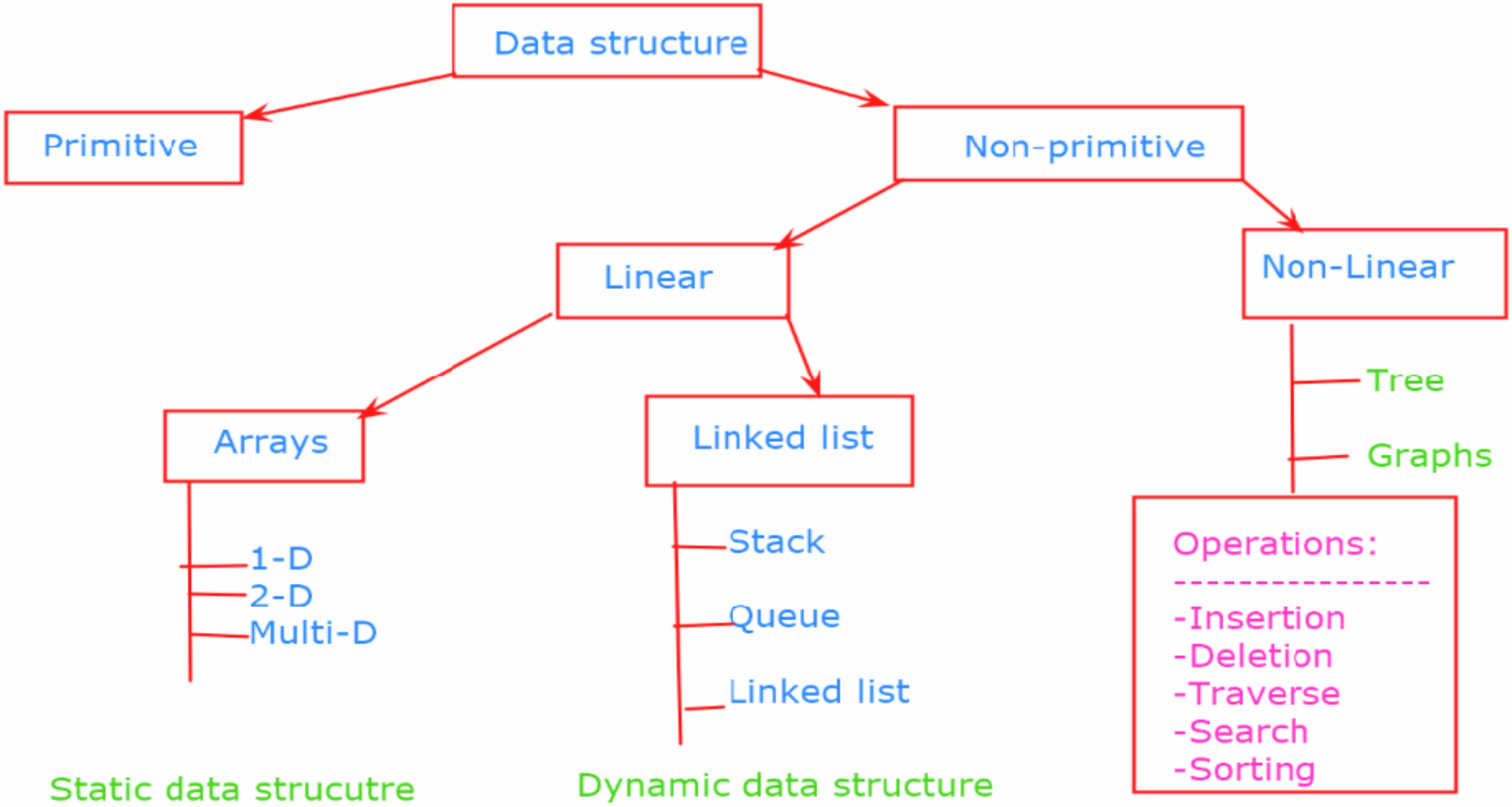
CDAC Mumbai

-An implementation of the algorithm in some programming language.

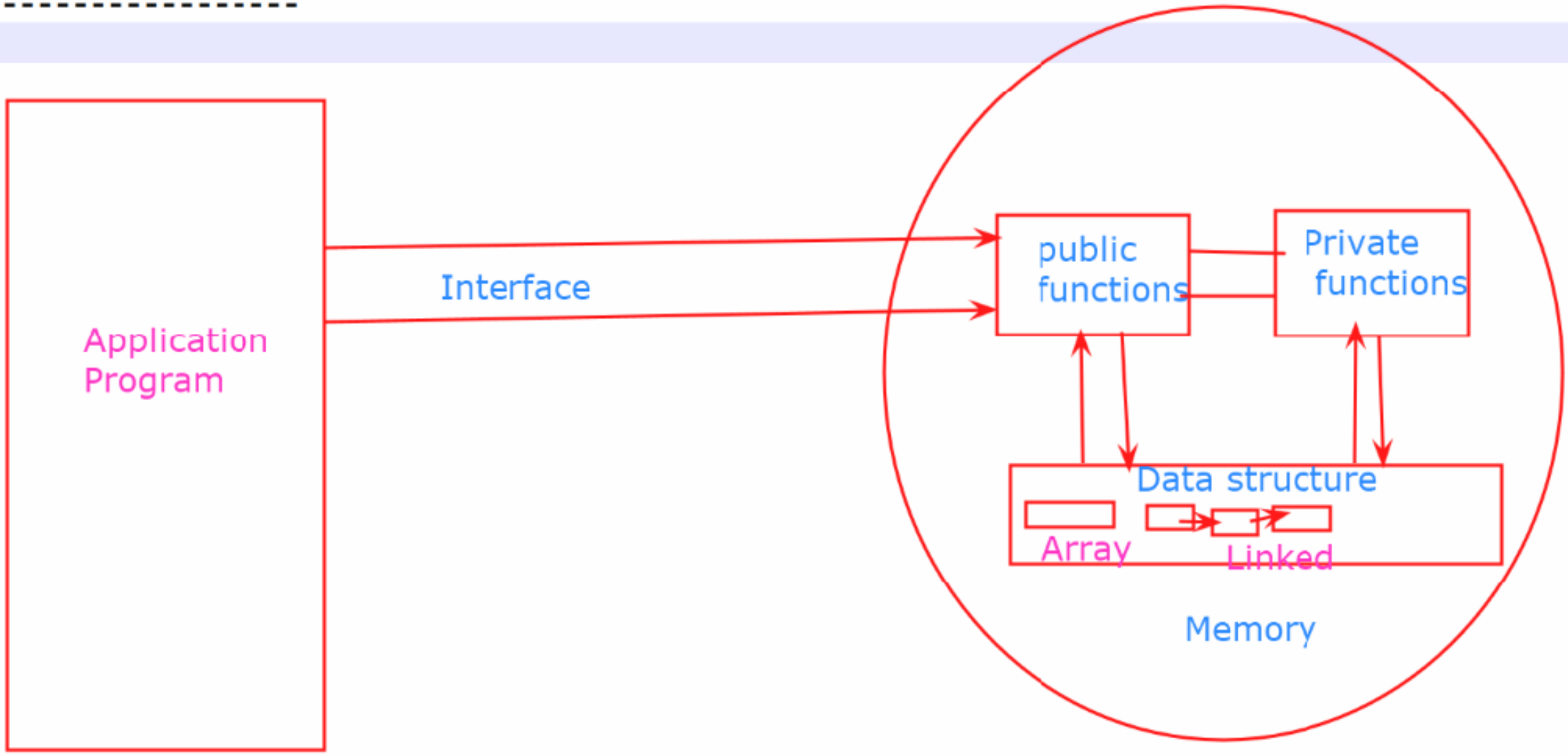
Algorithm:
-It is a sequence of unambiguous instructions/operations for solving a problem,for obtained the required output for any legitimate input in a finite amount of time.



-e.g., tree, graph, table, ...



Abstract Data type:



Abstract Data Strucutre

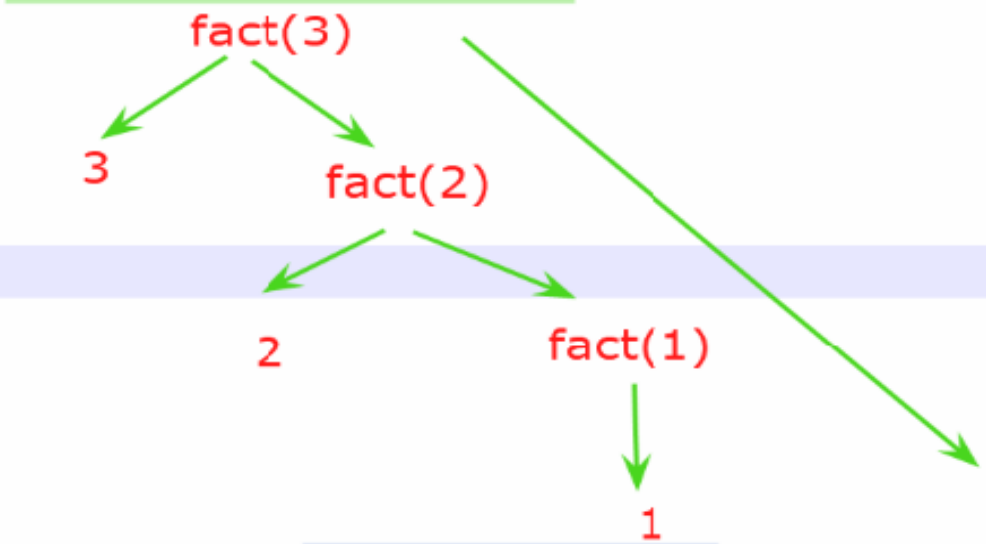
```java
class Recursion1{
    static int i=0;

    static void show()
    {
        ++i; 1 2 3 4 5 6
        if(i<=5)// base condition
        {
            System.out.println("Hello Girls !!!");
            show();// recursive call
        }
    }
    public static void main(String args[])
    {
        //System.out.println("Hello.....");
        show();
    }
}
```

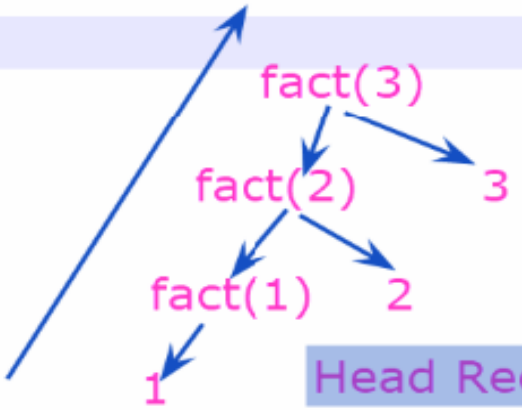
```
D:\Test>java Recursion1
Hello Girls !!!
Hello Girls !!!
Hello Girls !!!
Hello Girls !!!
Hello Girls !!!

D:\Test>
```

```java
class Recursion3{

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

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

        System.out.println(fact(5));
    }
}
```

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

Recursive tree



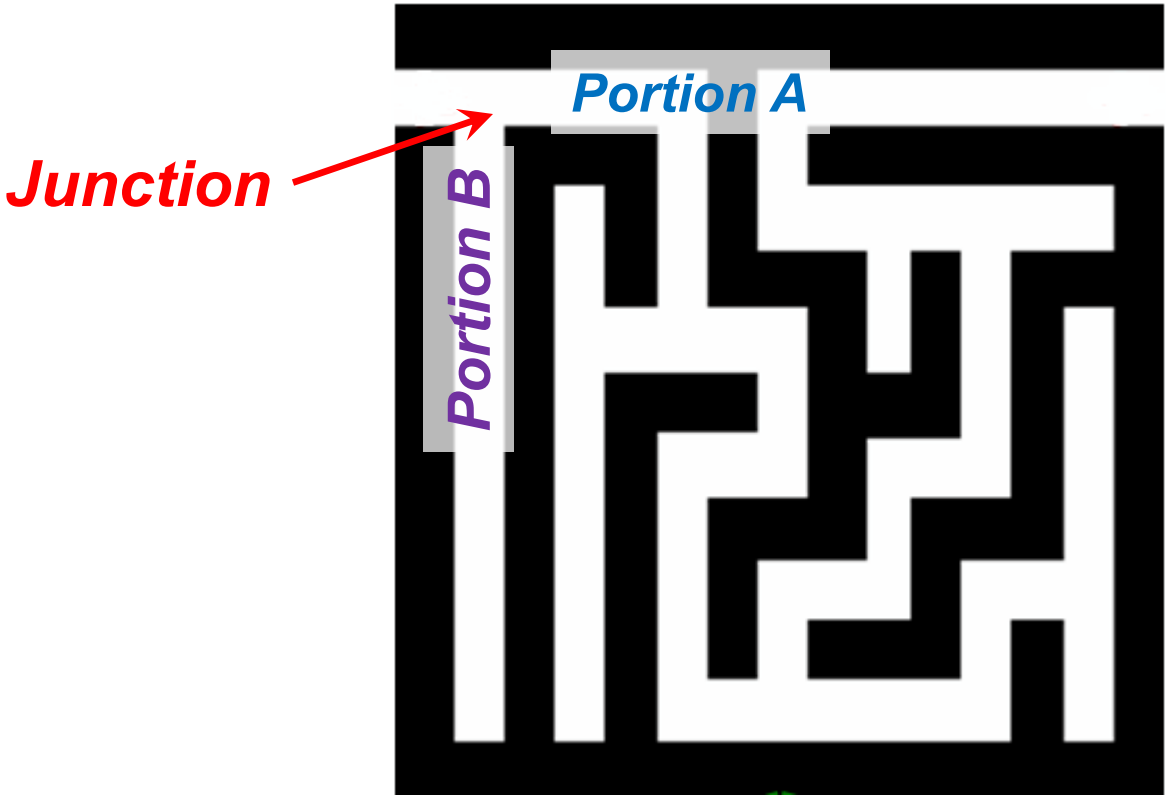


Head Recursion

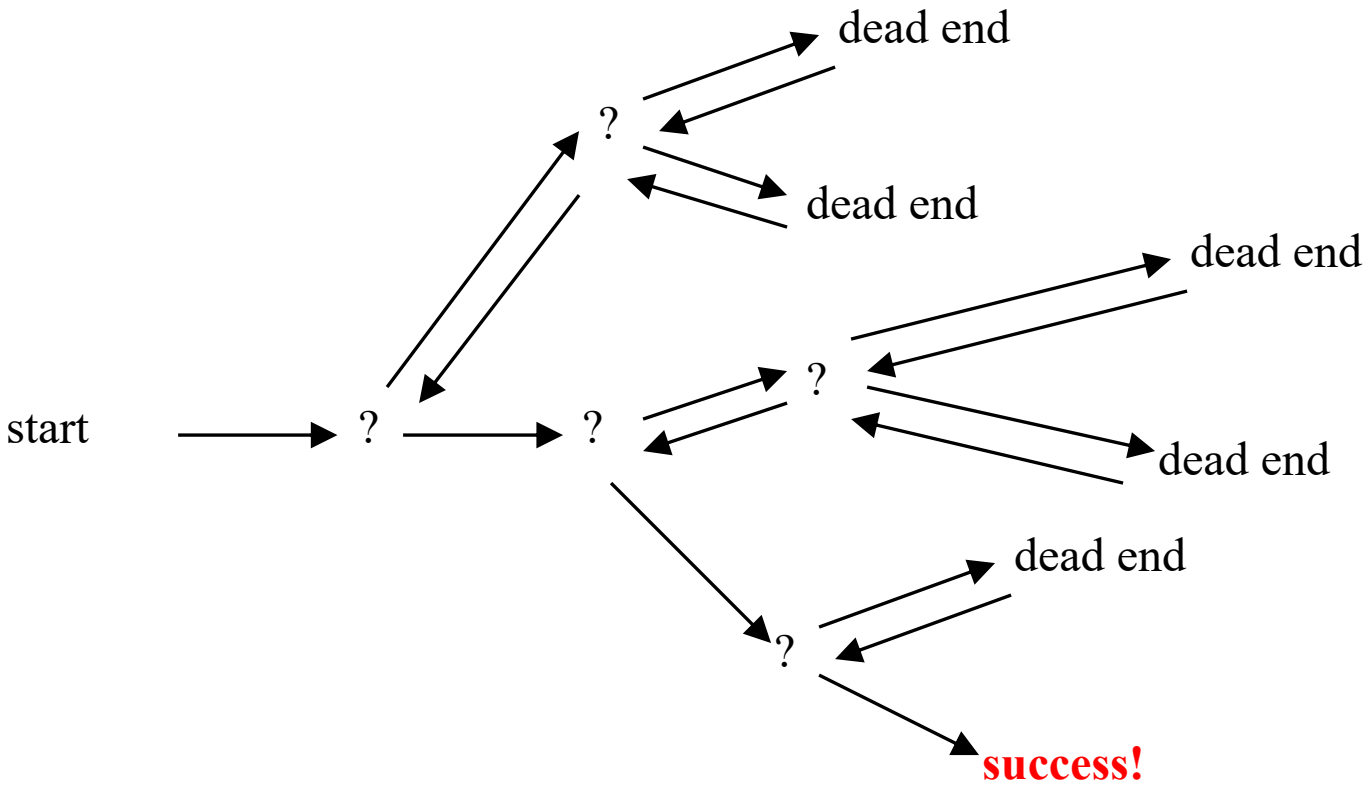
Tail Recursion

Backtracking: Idea

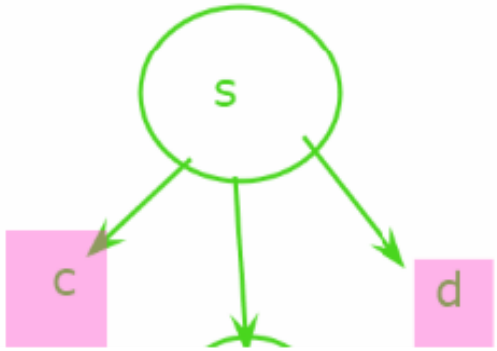
- **Backtracking is a technique used to solve problems with a large search space, by systematically trying and eliminating possibilities.**
- **A standard example of backtracking would be going through a maze.**
  - At some point, you might have two options of which direction to go:

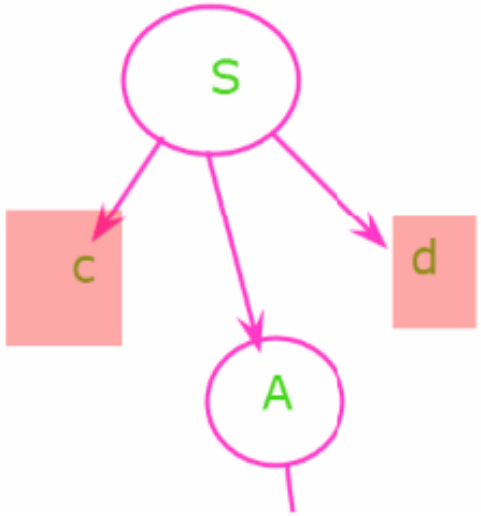


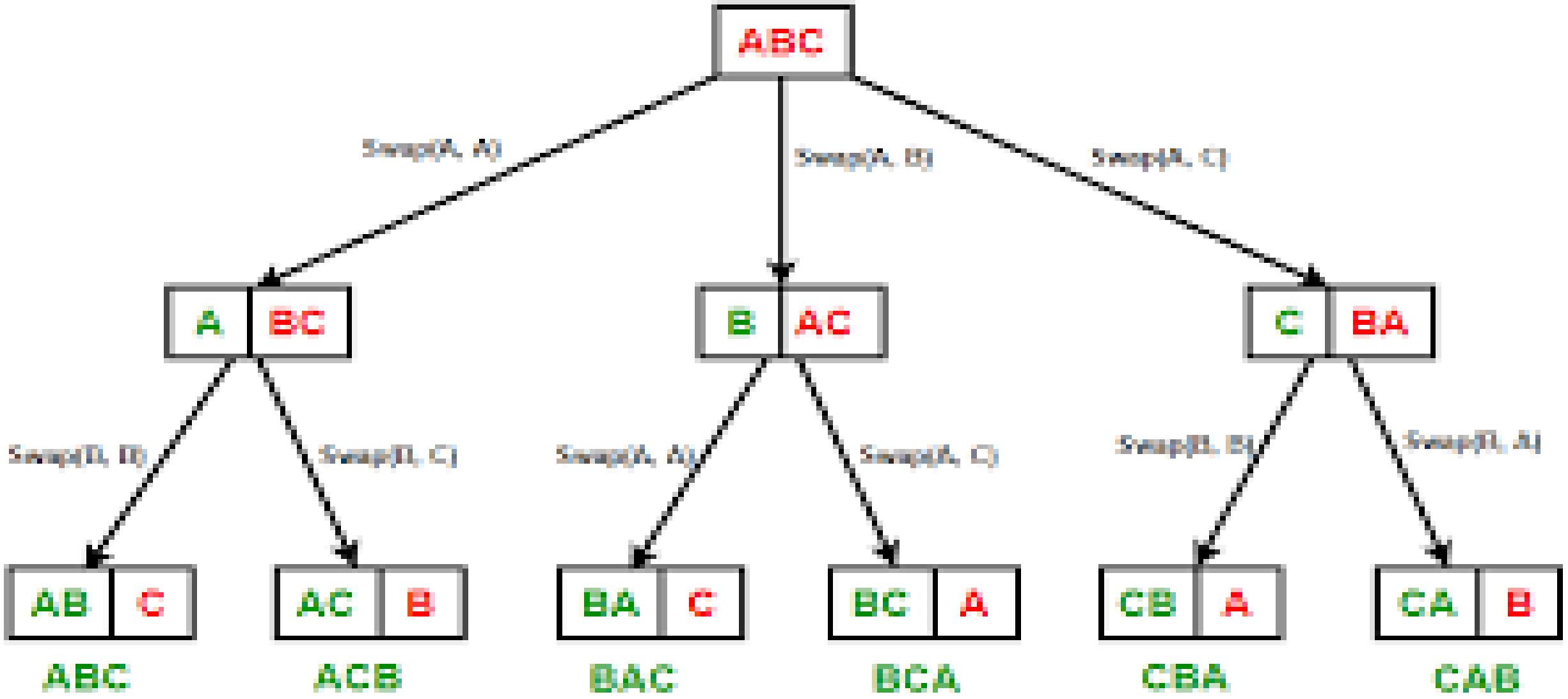
Backtracking (animation)



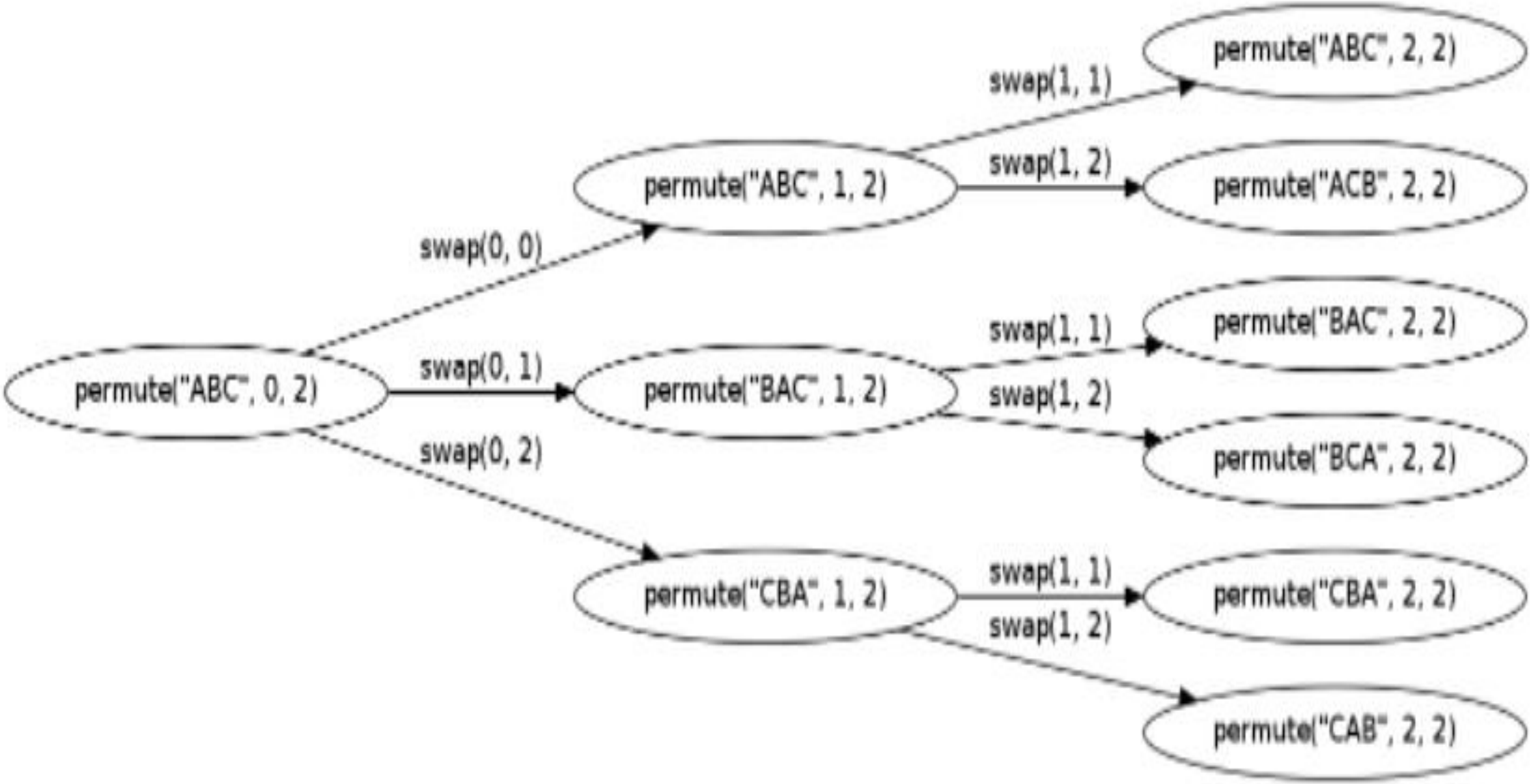
```
March 2023: Algorithms and Data Structures
-----------------------------------------
Date : 26-04-2023
Topics:
    -Recursion
    -Brute Force algorithm
    -Backtracking
    -Arrays

    S->cAd
    A->ab/a
```



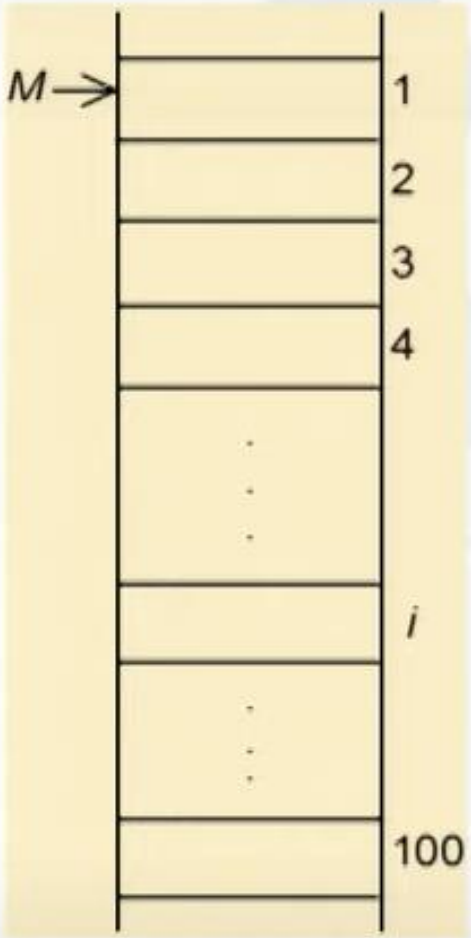


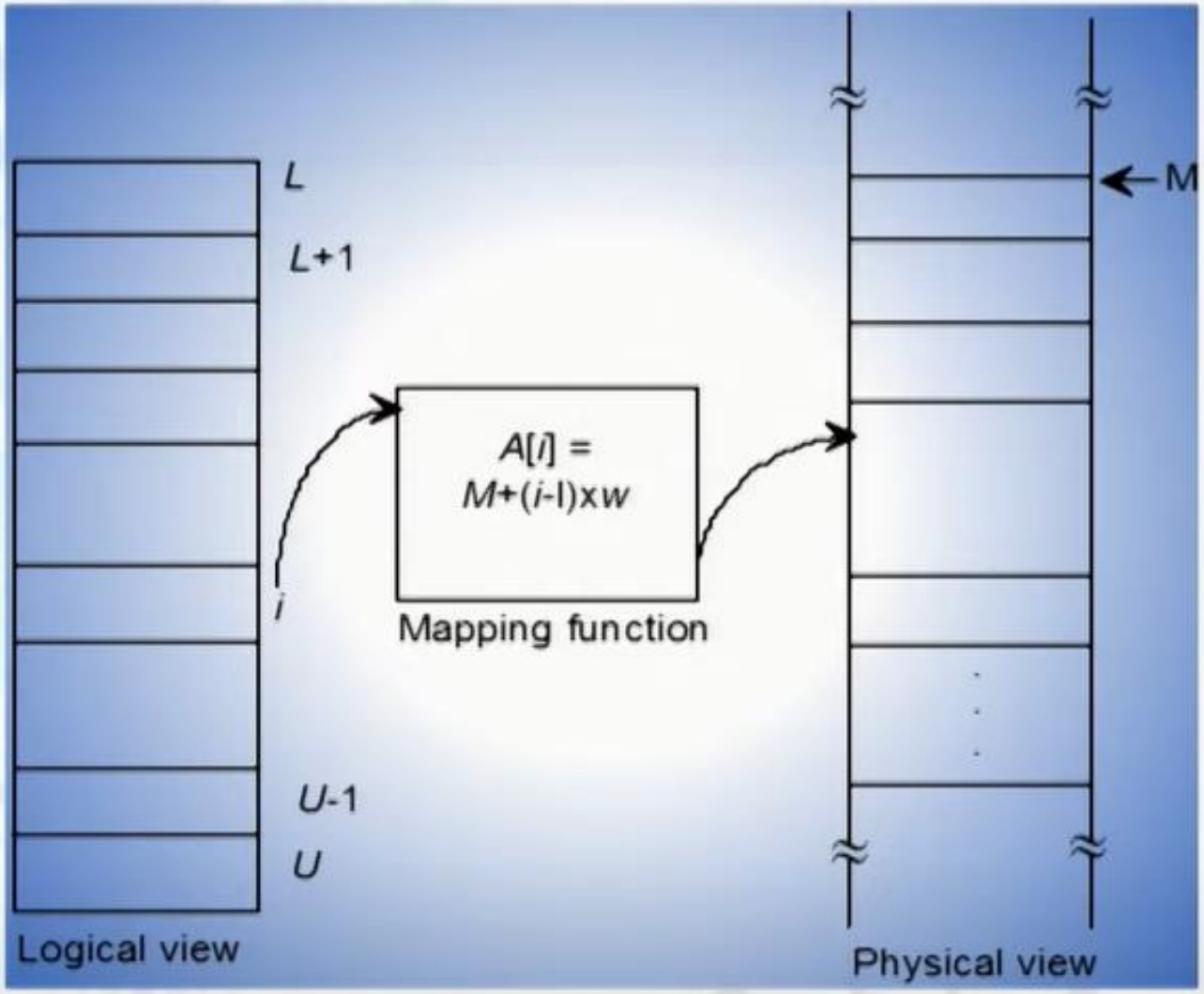


Recursion Tree for string "ABC"


permute("ABC", 0, 2)
swap(0, 0)
swap(0, 1)
swap(0, 2)
permute("ABC", 1, 2)
permute("BAC", 1, 2)
permute("CBA", 1, 2)
swap(1, 1)
swap(1, 2)
swap(1, 1)
swap(1, 2)
swap(1, 1)
swap(1, 2)
permute("ABC", 2, 2)
permute("ACB", 2, 2)
permute("BAC", 2, 2)
permute("BCA", 2, 2)
permute("CBA", 2, 2)
permute("CAB", 2, 2)

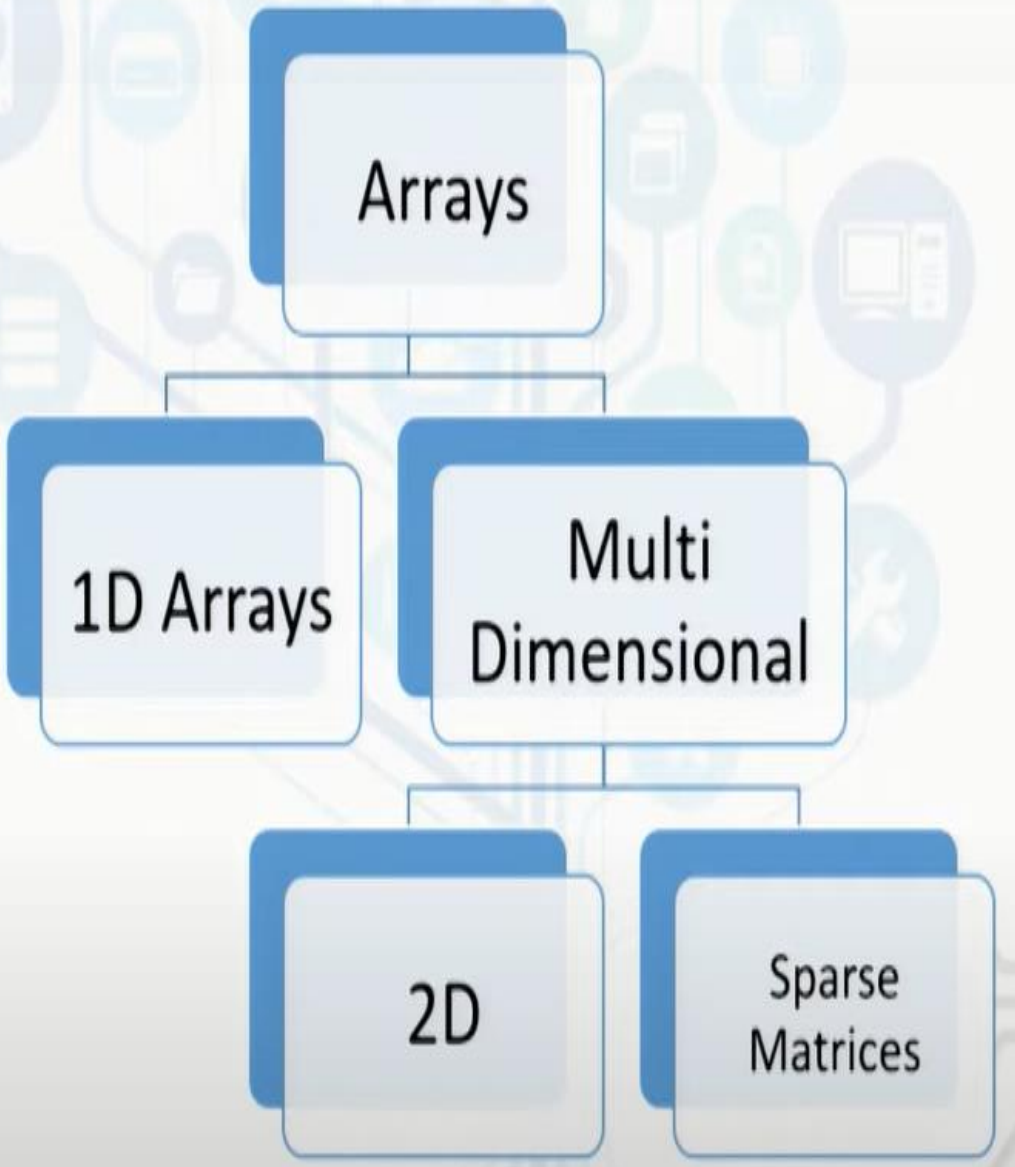
Concept of array



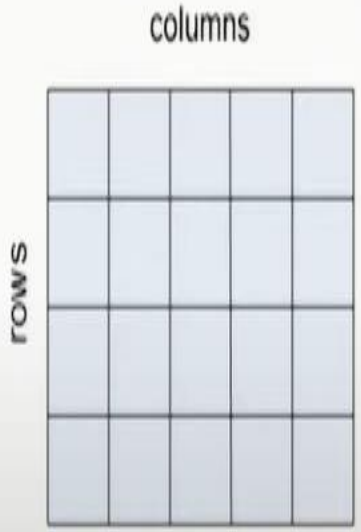


Address (A[i]) = $M + (i - L) \times w$

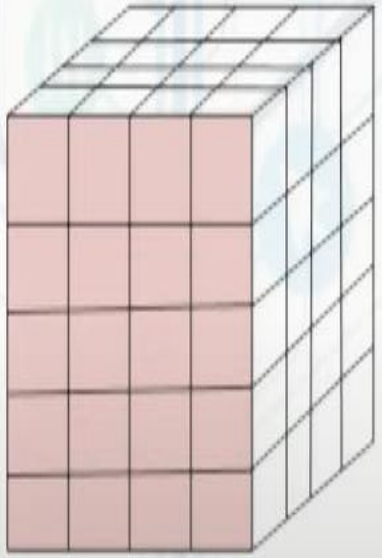
Size (A) = $U - L + 1$



1D array



2D array



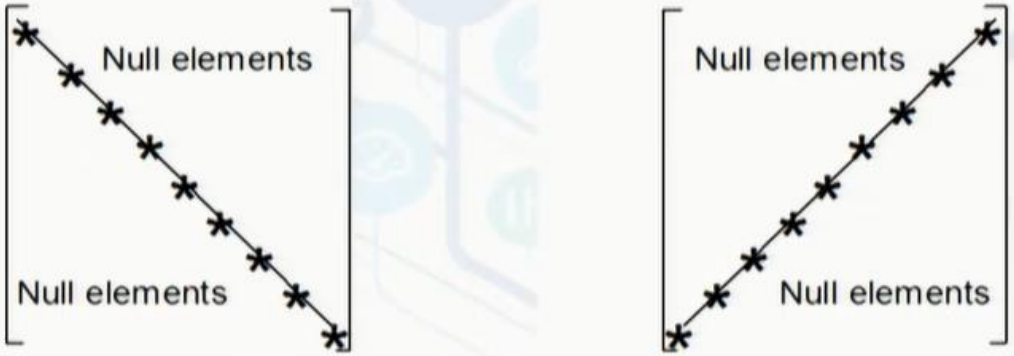
3D array

Sparse matrix

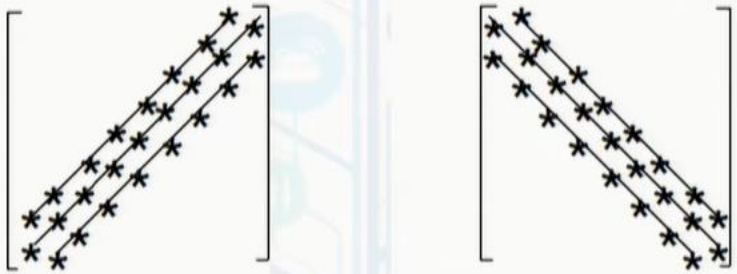
A *sparse* matrix is a two-dimensional array having the value of majority elements as null

```
[ - - - * - - - * - - - - * - - ]
[ - - - - * - - - * - - - - - - ]
[ - - - - - - * * * - - - - - ]
[ * - - - - * - - - * - - - - * ]
[ - - - - * - - - - * - - - - * ]
[ - - * - - - - * - - - - - * - ]
[ - - - - - * - - - - - * - - - ]
[ - - - - * - - - - - * - - - - ]
```

Diagonal sparse matrices



Tri-diagonal sparse matrices



```
}

public void delete(int value)
{
    int j;
    for(j=0;j<n;j++)
    {
        if(a1[j] == value)
            break;
```

```
55 99 74 4 0
D:\Test>javac Array1.java
Array1.java:57: error: missing return statement
```

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On

```
D:\Test>javac Array1.java

D:\Test>java Array1
44 55 66 99 34 74 24 4 89 0 Found
44 55 66 99 34 24 4 89 0
```

```java
arr.insert(2,66);
arr.insert(3,99);
arr.insert(4,34);
arr.insert(5,74);
arr.insert(6,24);
arr.insert(7,4);
arr.insert(8,89);
arr.insert(9,0);

arr.display();
int key =34;
if(arr.find(key))
    System.out.println("Found");
else
```

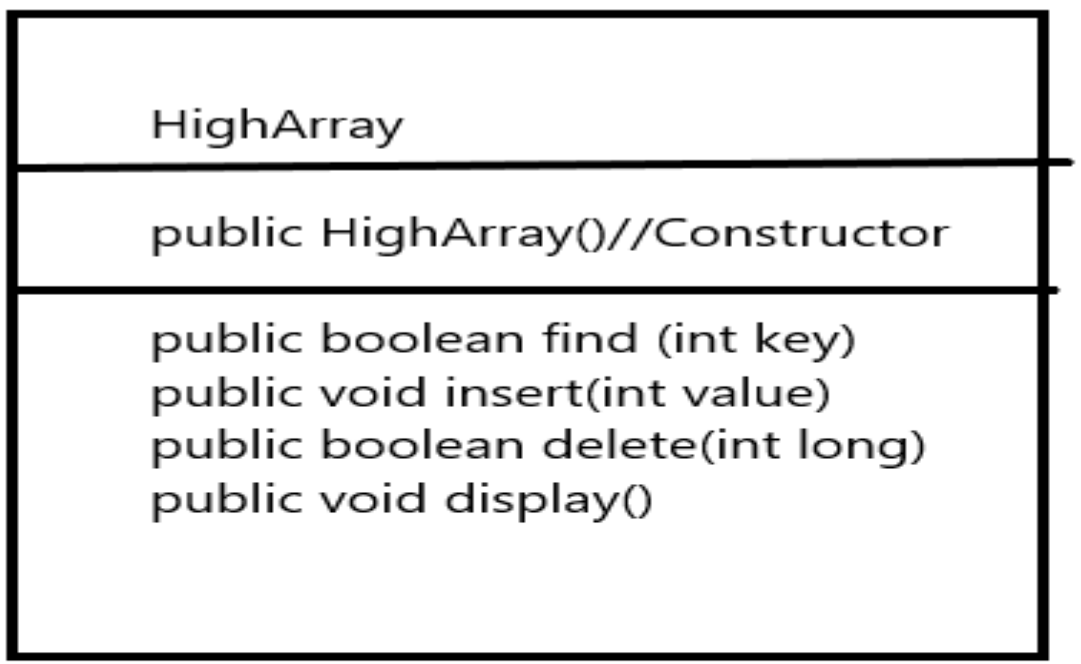
```
Array1.java:57: error: missing return statement
        }
        ^
```

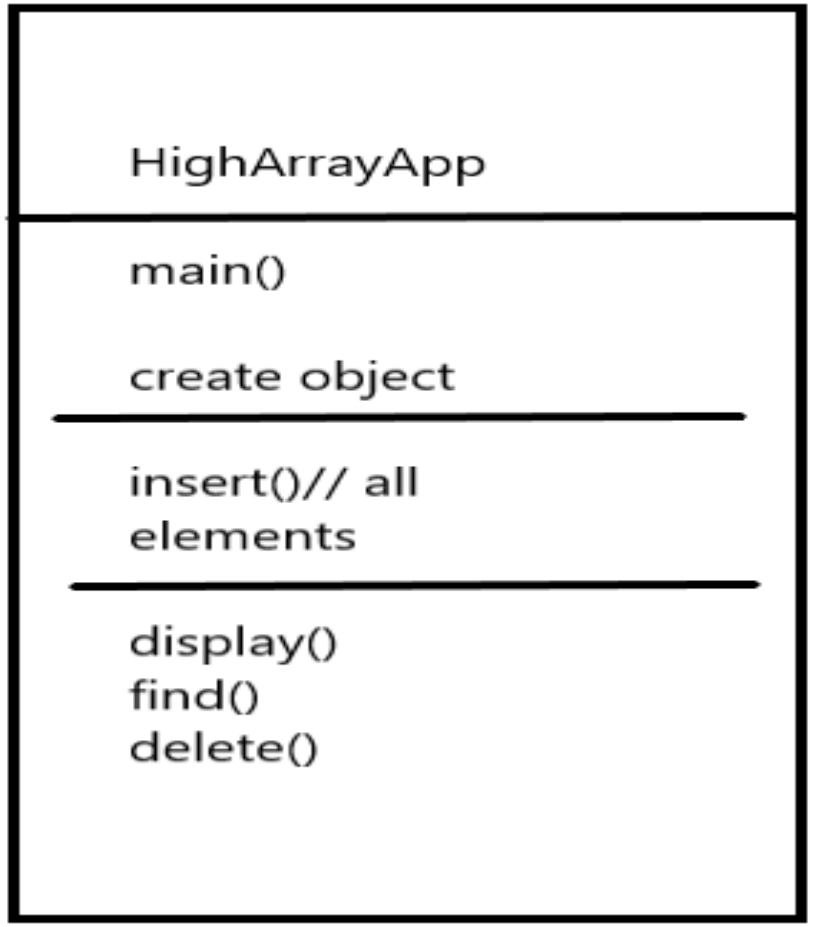
Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

Who can see what you share here? Recording On

```
D:\Test>java Array1
44 55 66 99 34 74 24 4 89 0 Found
44 55 66 99 34 24 4 89 0
D:\Test>javac Array1.java

D:\Test>java Array1
44 55 66 99 34 74 24 4 89 0 Found
44 55 66 99 34 24 4 89 0 44 55 66 99 34 24 4 89
D:\Test>javac Array1.java

D:\Test>java Array1
44 55 66 99 34 74 24 4 89 0 Found
44 55 66 99 34 24 4 89 0
```

Problem statement





Problem statement: Find duplicates in an array

- Given an array a1[] of size N which contains elements from 0 to N-1, you need to find all the elements occurring more than once in the given array.

- **Example 1:**
  - Input:
    - N = 4
    - a[] = {0,3,1,2}
  - Output: -1
  - Explanation: N=4 and all elements from 0 to (N-1 = 3) are present in the given array. Therefore output is -1.

- **Example 2:**
  - Input:
    - N = 5
    - a[] = {2,3,1,2,3}
  - Output: 2 3
  - Explanation: 2 and 3 occur more than once in the given array.

Problem statement: Removing punctuations from a given string

- **Given a string, remove the punctuation from the string if the given character is a punctuation character, as classified by the current C locale. The default C locale classifies these characters as punctuation:**
  - ! " # $ % & ' ( ) * + , - . / : ; ? @ [\ ] ^ _ ` { | } ~

- **Example 1:**
  - Input : %welcome' to @cdacmumbai?<s
  - Output : welcome to cdacmumbai

- **Example 2:**
  - Input : Hello!!!, he said ---and went**.
  - Output : Hello he said and went

Problem statement: Program to find the initials of a name.

- Given a string name, we have to find the initials of the name
- Examples 1:
  - Input : Kabhi Haa Kabhi Naa
  - Output : K H K N
    - We take the first letter of all
    - words and print in capital letter.
- Example 2:
  - Input : Mahatma Gandhi
  - Output : M G
- Example 3:
  - Input : Shah Rukh Khan
  - Output : S R K

  - Example 4: your own name

Thanks