Unit 2: LINEAR DATA STRUCTURE (Array & Stack)

Prof. Ankit Koyani
CSE Department
SLTIET, Rajkot

Mahatma Gandhi Charitable Trust Managed

Shri Labhubhai Trivedi Institute of Engineering & Technology

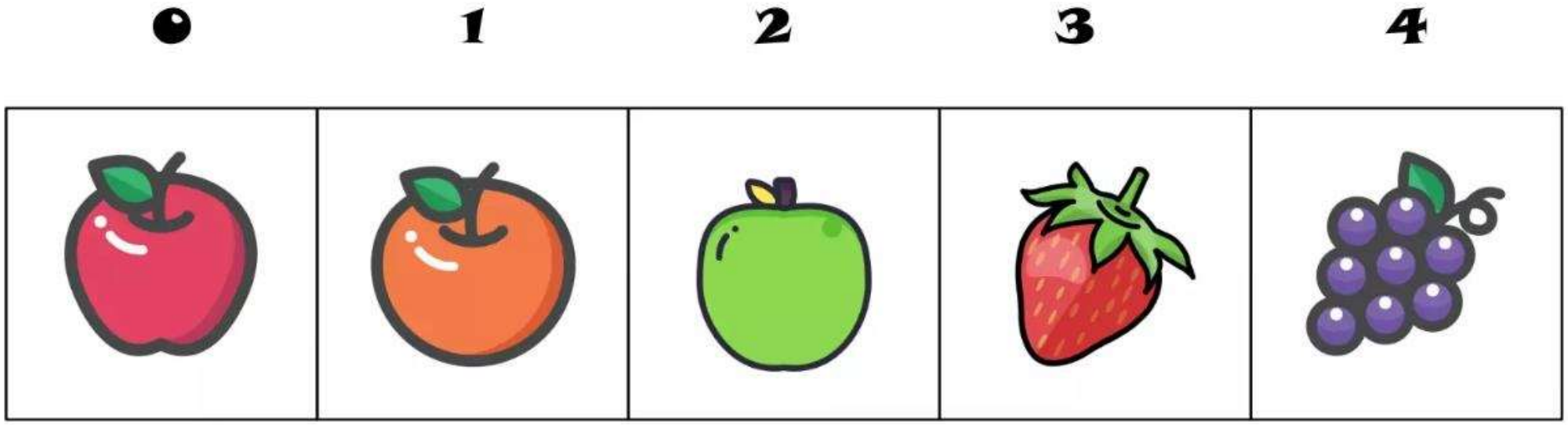
Topics to be covered:

- Introduction to Array
- Representation of Array
- Types of Array
- Sparse matrix & it’s representation
- Applications of array
- Advantages & disadvantages

Introduction to Array

- Array is collection of elements with **same data-type** having a common name.
- Array index always starts with **0 (zero).**
- Each element in the array can be accessed via its index.
- These data structures come into picture when there is a necessity to store multiple elements of similar nature together at one place.

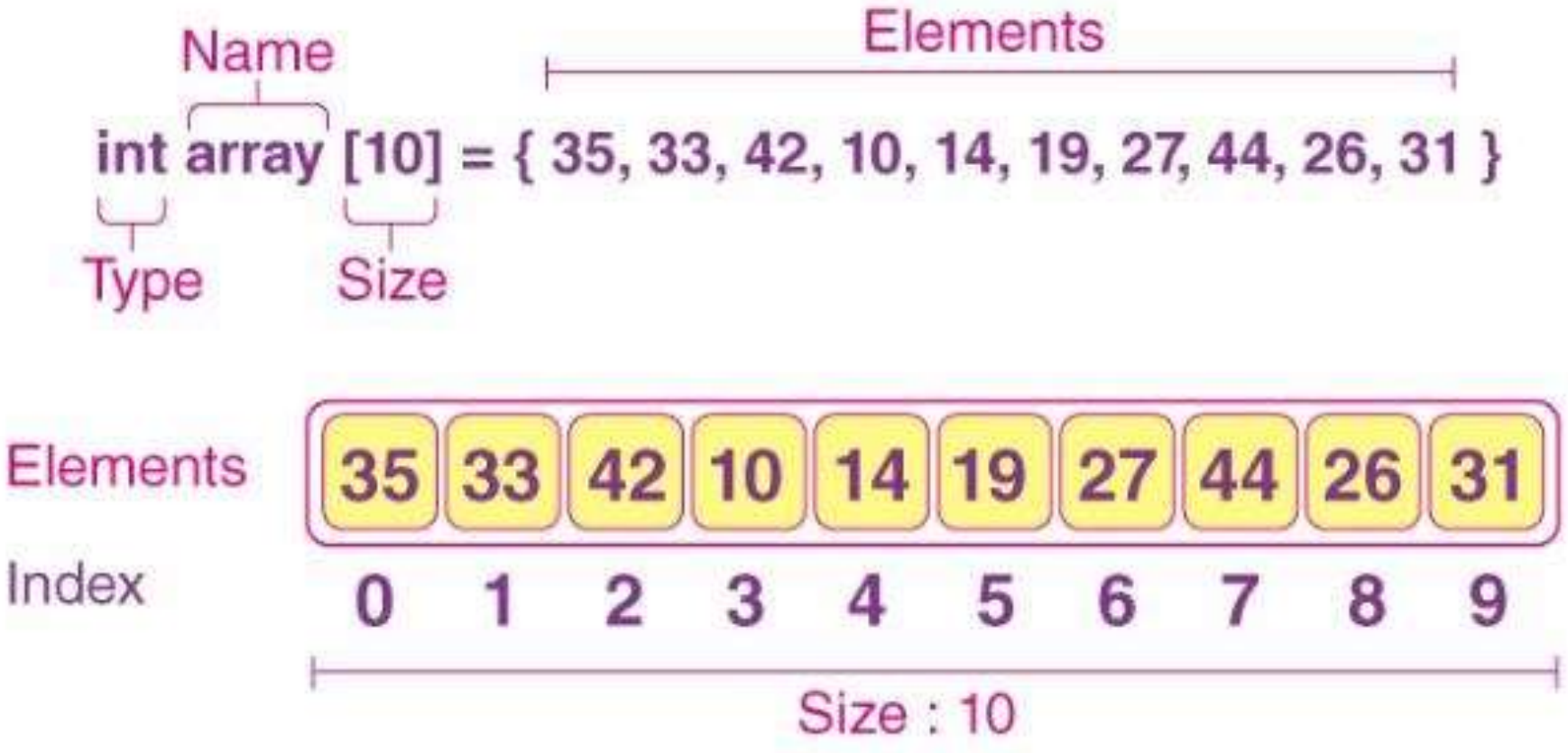
Is this array?



An Array of Fruits : fruits
fruits[4]="Grapes"
fruit = fruits[0]

fruit variable holds "Apple"

Representation of Array



- Arrays always store the same type of values.
- In the above example:
  1. int is a type of data value.
  2. Data items stored in an array are known as elements.
  3. The location or placing of each element has an index value.

Important: Array can store only the same type of data items.

Declaration Syntax of Array:

Datatype VariableName[size];

- Example 1: For integer value
- int A[10];

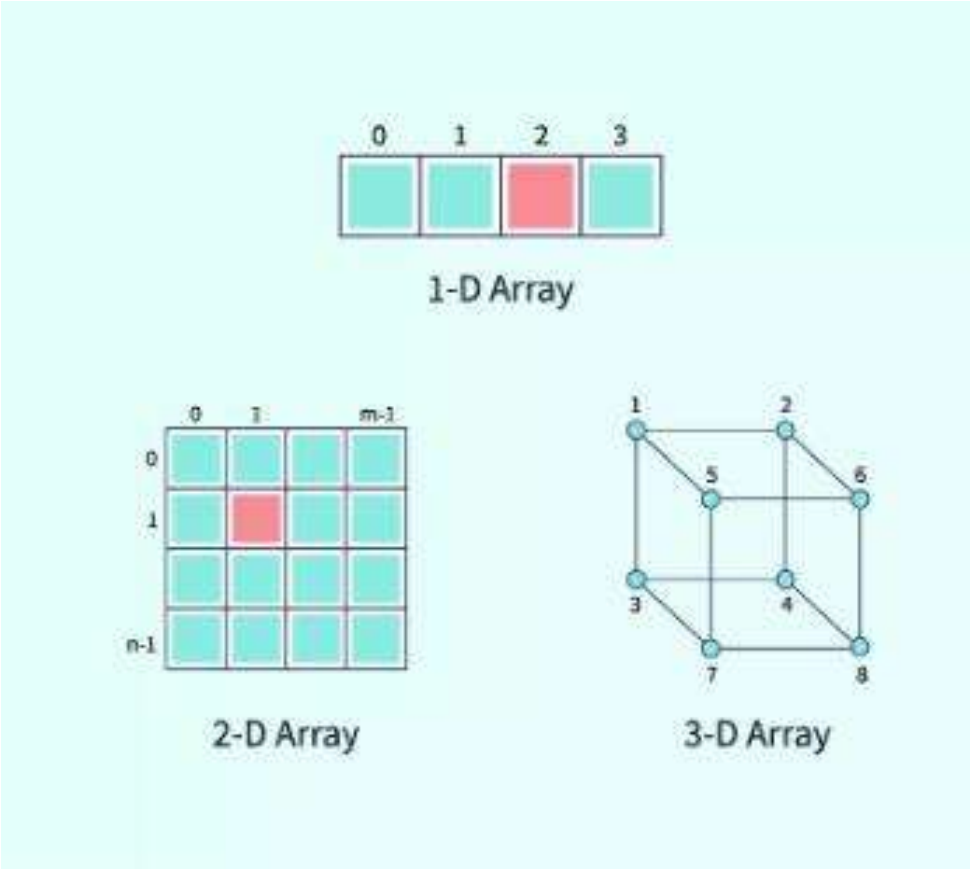
Here 10 means, this array A can have 10 integer elements.

- Example 2: For character value
- char B[10];

This array B can have 10 character elements.

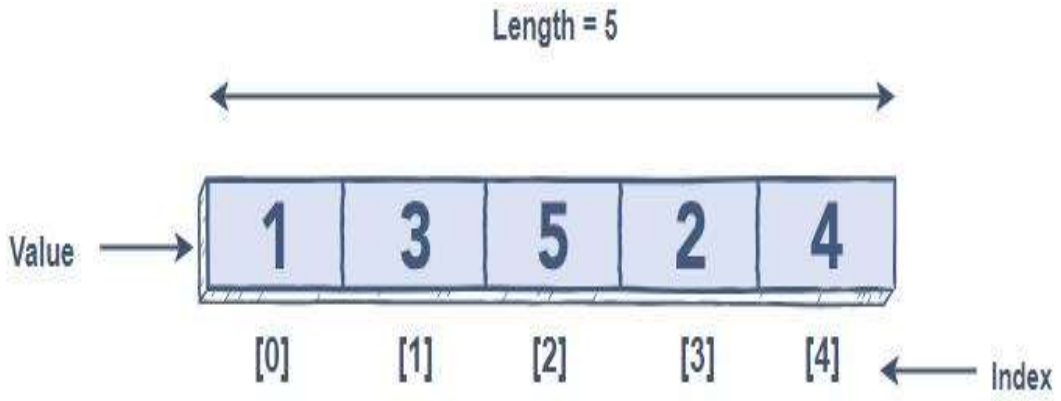
Types of Arrays:

- There are two types of arrays:
  - One-Dimensional Arrays
  - Multi-Dimensional Arrays



One -Dimensional Arrays

- A one-dimensional array is a kind of linear array. It involves single sub-scripting.
- The [] (brackets) is used for the subscript of the array and to declare and access the elements from the array.
- Syntax: Data Type Array Name [size];
- For example: int a[5];



Multi-Dimensional Arrays

- An array involving two subscripts [] [] is known as a two-dimensional array.
- They are also known as the array of the array.
- Two-dimensional arrays are divided into rows and columns and are able to handle the data of the table.
- Syntax: Data Type Array Name[row size][column size];

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

Sparse Matrix and its representations

- A matrix is a two-dimensional data object made of m rows and n columns, therefore having total m x n values. If most of the elements of the matrix have **0 value**, then it is called a sparse matrix.
- **Why to use Sparse Matrix instead of simple matrix ?**
- **Storage:** There are lesser non-zero elements than zeros and thus lesser memory can be used to store only those elements.
- **Computing time:** Computing time can be saved by logically designing a data structure traversing only non-zero elements.

Sparse Matrix and its representations

- **Example:**

0 0 3 0 4
0 0 5 7 0
0 0 0 0 0
0 2 6 0 0

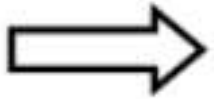
- Representing a sparse matrix by a 2D array leads to wastage of lots of memory as zeroes in the matrix are of no use in most of the cases.
- So, instead of storing zeroes with non-zero elements, we only store non-zero elements. This means storing non-zero elements with **triples- (Row, Column, value).**

Sparse Matrix and its representations

- **Using Arrays:**
- 2D array is used to represent a sparse matrix in which there are three rows named as
- **Row:** Index of row, where non-zero element is located
- **Column:** Index of column, where non-zero element is located
- **Value:** Value of the non zero element located at index – (row,column)

Sparse Matrix and its representations

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix}$$



Row	0	0	1	1	3	3
Column	2	4	2	3	1	2
Value	3	4	5	7	2	6

Application of array

- **Storing and accessing data**: Arrays are used to store and retrieve data in a specific order. For example, an array can be used to store the scores of a group of students, or the temperatures recorded by a weather station.
- **Dynamic programming**: Dynamic programming algorithms often use arrays to store intermediate results of subproblems in order to solve a larger problem.
- **Stacks and queues:** Arrays are used as the underlying data structure for implementing stacks and queues, which are commonly used in algorithms and data structures.

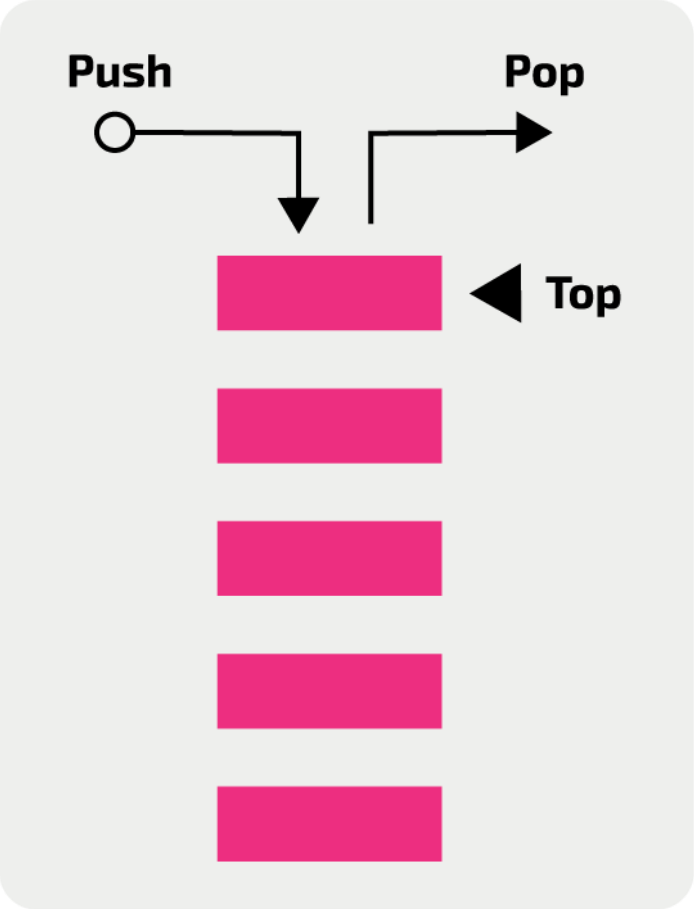
Stack and its operations

Stack

- A Stack is a linear data structure that follows the **LIFO (Last-In-First-Out)** principle. Stack has one end, whereas the Queue has two ends (**front and rear**).
- It contains only one pointer **top pointer** pointing to the topmost element of the stack.
- Whenever an element is added in the stack, it is added on the top of the stack, and the element can be deleted only from the stack.
- In other words, a ***stack can be defined as a container in which insertion and deletion can be done from the one end known as the top of the stack.***

Stack

- It is called as stack because it behaves like a real-world stack, piles of books, etc.
- A Stack is an abstract data type with a pre-defined capacity, which means that it can store the elements of a limited size.
- It is a data structure that follows some order to insert and delete the elements, and that order can be LIFO or FILO.



Standard Stack Operations

- **push():** When we insert an element in a stack then the operation is known as a push. If the stack is full then the overflow condition occurs.
- **pop():** When we delete an element from the stack, the operation is known as a pop. If the stack is empty means that no element exists in the stack, this state is known as an underflow state.
- **isEmpty():** It determines whether the stack is empty or not.
- **isFull():** It determines whether the stack is full or not.'

Standard Stack Operations

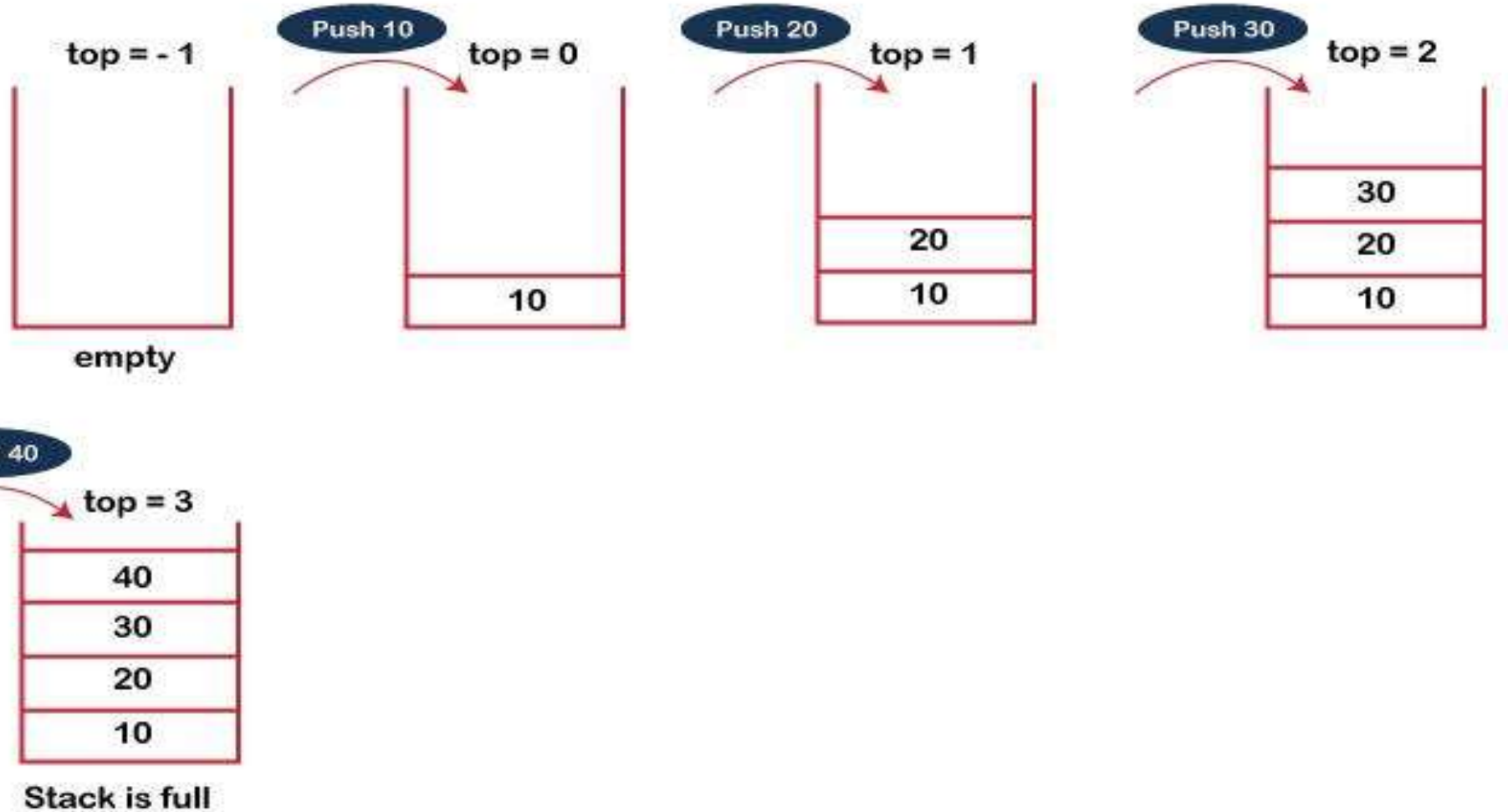
- **peek():** It returns the element at the given position.
- **count():** It returns the total number of elements available in a stack.
- **change():** It changes the element at the given position.
- **display():** It prints all the elements available in the stack.

- https://yongdanielliang.github.io/animation/web/Stack.html

PUSH and POP operations

- **The steps involved in the PUSH operation is given below:**
- Before inserting an element in a stack, we check whether the stack is full.
- If we try to insert the element in a stack, and the stack is full, then the ***overflow*** condition occurs.
- When we initialize a stack, we set the value of top as -1 to check that the stack is empty.
- When the new element is pushed in a stack, first, the value of the top gets incremented, i.e., **top=top+1,** and the element will be placed at the new position of the **top**.
- The elements will be inserted until we reach the ***max*** size of the stack

PUSH and POP operations



Algorithm: PUSH (S, TOP, X)

1.[Check for stack overflow]

If TOP ≥ N Then

write ('STACK OVERFLOW')

Return

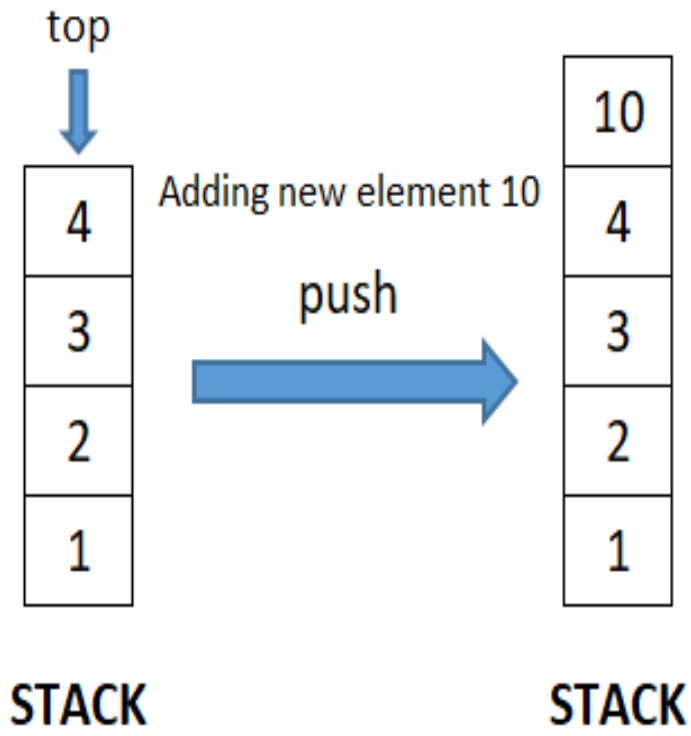
2. [Increment TOP]

TOP ← TOP + 1

3. [Insert Element]

S[TOP] ← X

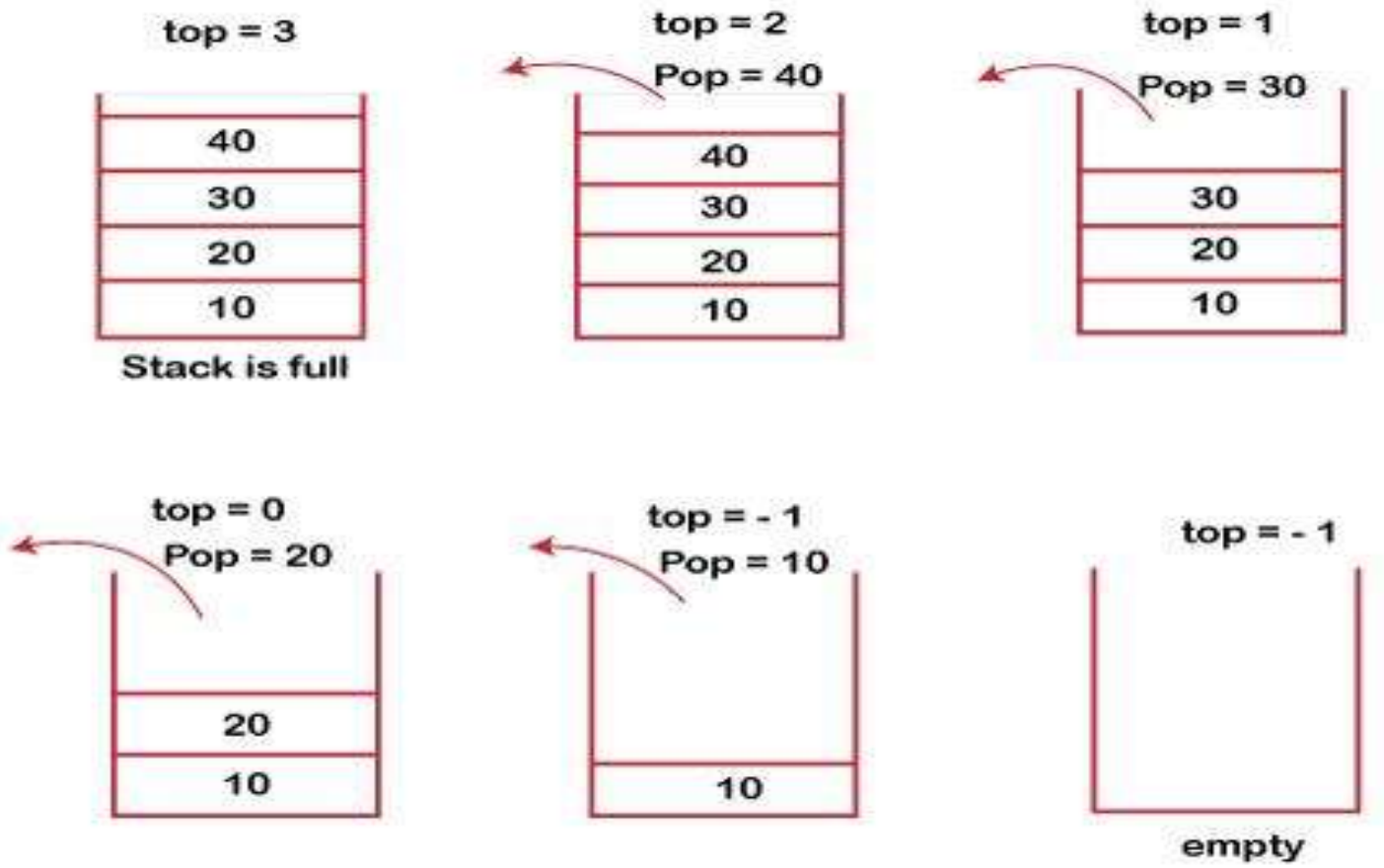
4. [Finished]



PUSH and POP operation

- **The steps involved in the POP operation is given below:**
- Before deleting the element from the stack, we check whether the stack is empty.
- If we try to delete the element from the empty stack, then the ***underflow*** condition occurs.
- If the stack is not empty, we first access the element which is pointed by the ***top***
- Once the pop operation is performed, the top is decremented by 1, i.e., **top=top-1**.

PUSH and POP operations



Algorithm: POP (S, TOP)

1. [Check for stack underflow

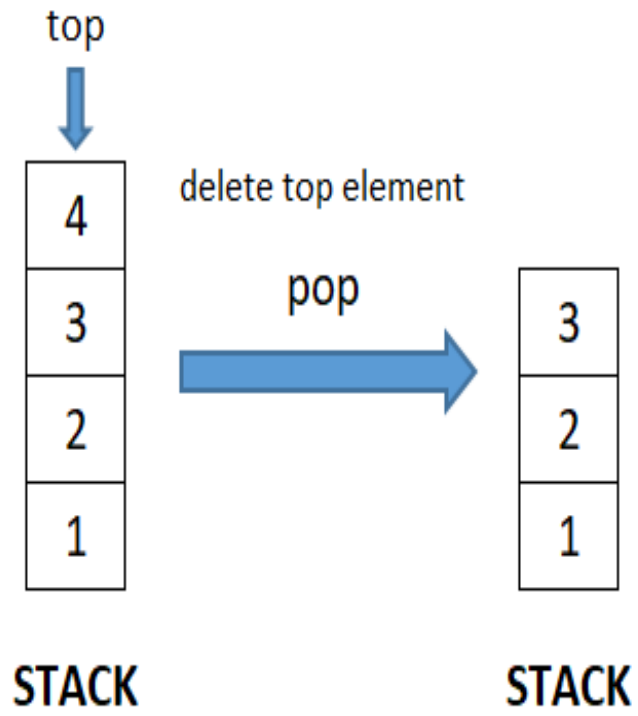
If TOP = 0 Then

write ('STACK UNDERFLOW')

Return (0)

2. [Decrement TOP]

TOP ← TOP - 1



Application of Stack

- **Recursion:** The recursion means that the function is calling itself again. To maintain the previous states, the compiler creates a system stack in which all the previous records of the function are maintained.
- **DFS(Depth First Search):** This search is implemented on a Graph, and Graph uses the stack data structure.
- **Memory management:** The stack manages the memory. The memory is assigned in the contiguous memory blocks. The memory is known as stack memory as all the variables are assigned in a function call stack memory.

Application of Stack

- **Expression conversion:** Stack can also be used for expression conversion. This is one of the most important applications of stack.
- The list of the expression conversion is given below:

1. Infix to prefix
2. Infix to postfix
3. Prefix to infix
4. Prefix to postfix
5. Postfix to infix


ANY
Questions?

CREATING CREATORS - SLTIET