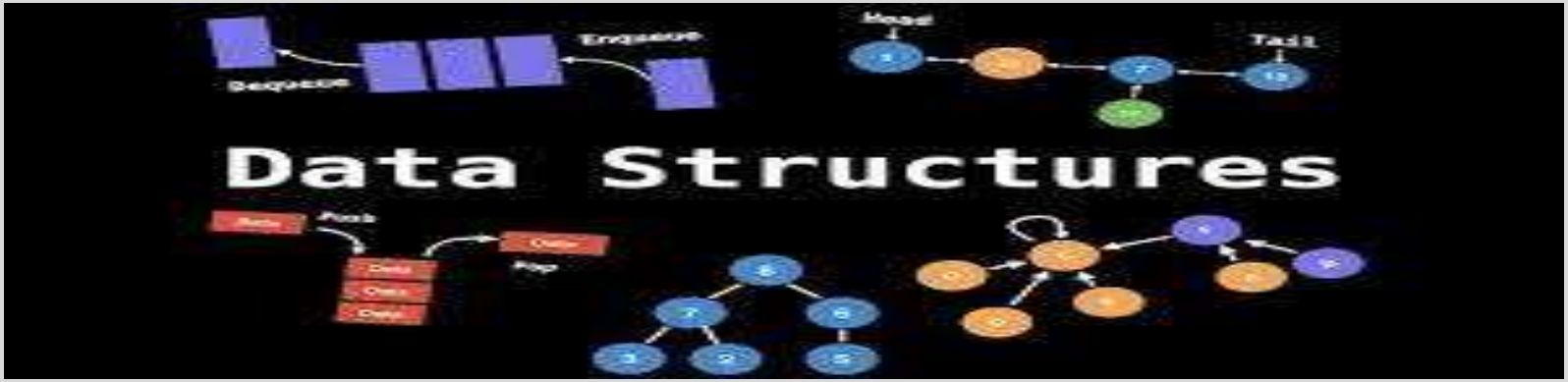


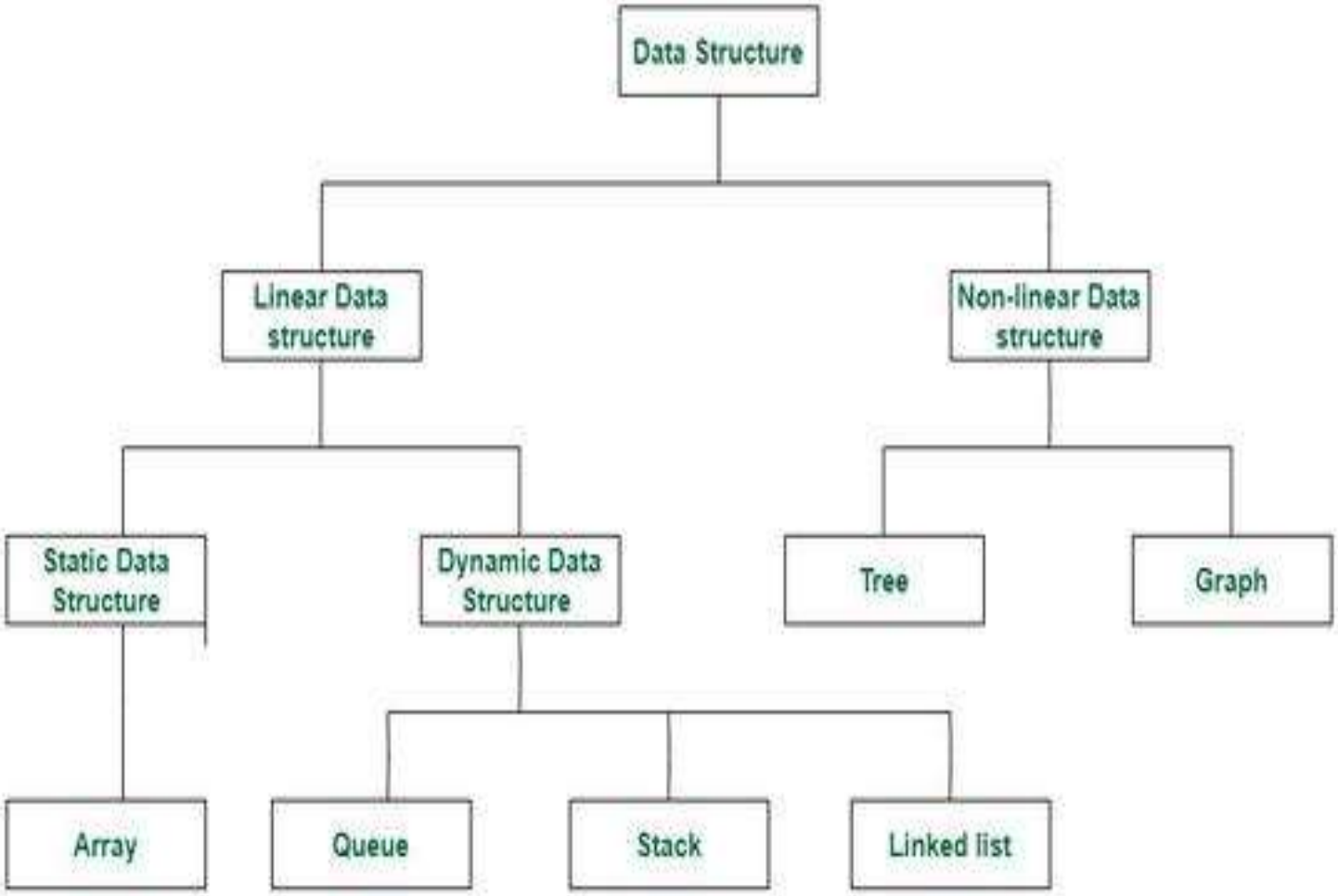
Data Structure
Stacks
Queues

Presented By:
Isha Malik Arora
Computer Science Dept.
Govt. College For Girls, Ludhiana

Definition

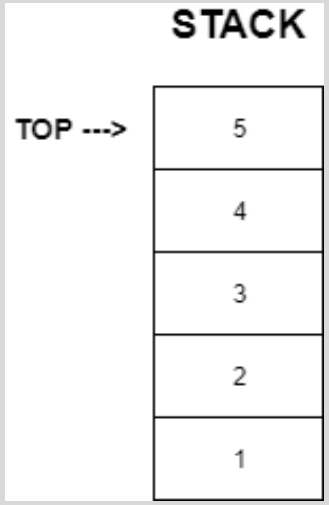
- **Data**: Collection of **raw facts & figure**.
- **Data Structure** is representation of the logical relationship existing between individual elements of data.
- **Data Structure** is a specialized format for organizing and storing data in memory that considers not only the elements stored but also their relationship to each other.

Classification of Data Structure

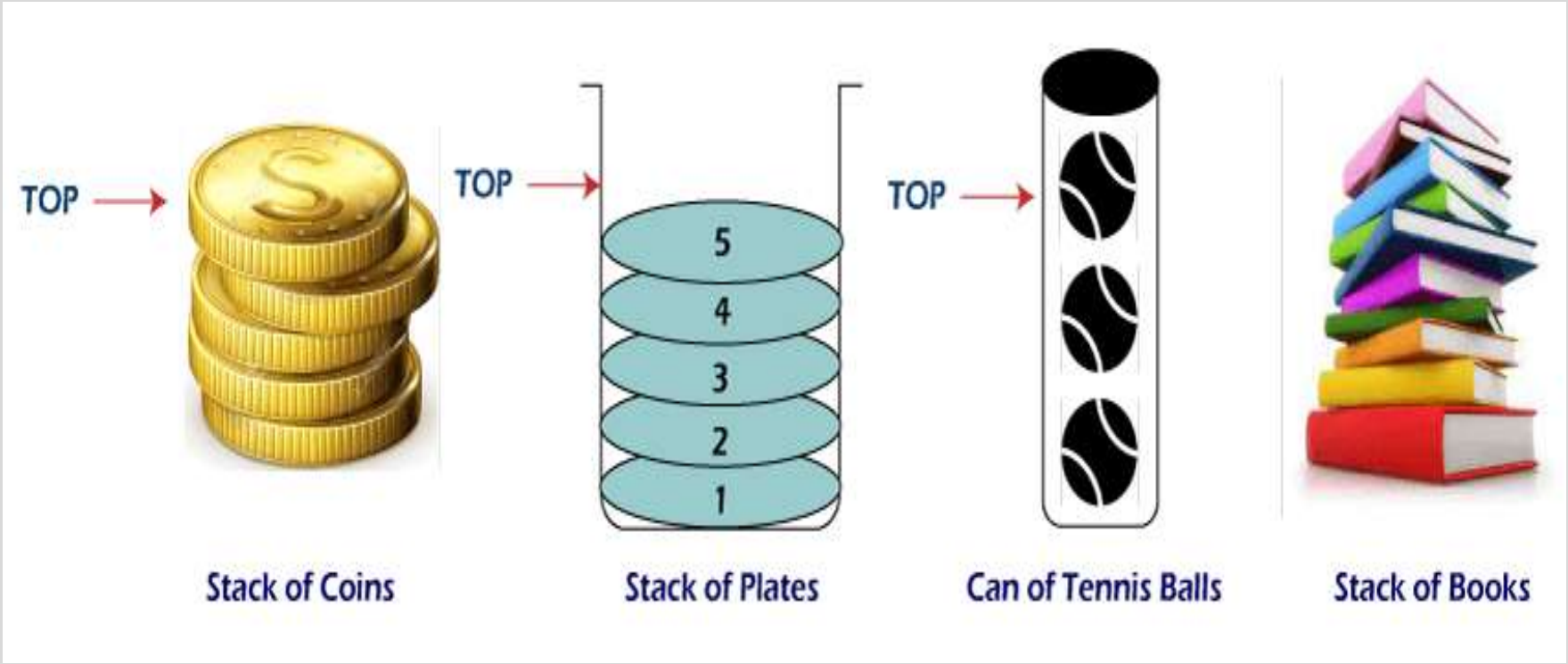


What do you mean by Stacks?

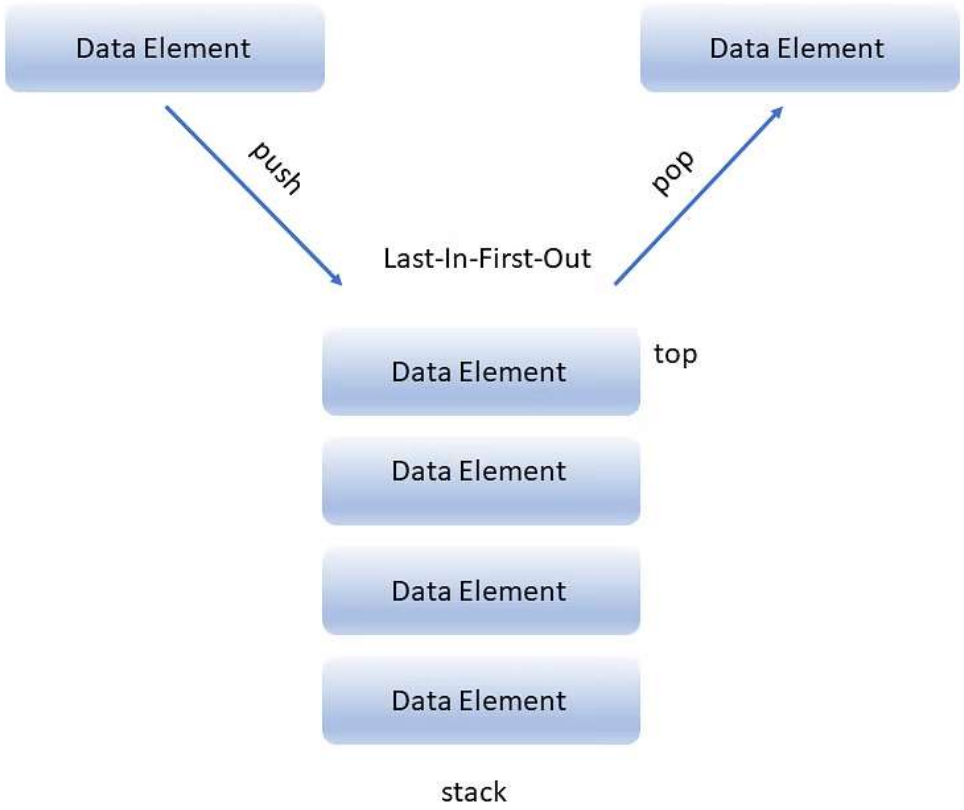
Stacks is a linear type of data structure that follows the **LIFO (Last-In-First-Out)** principle and allows insertion and deletion operations from one end of the stack data structure, that is **top**. Implementation of the stack can be done by contiguous memory which is an array, and non-contiguous memory which is a linked list. Stack plays a vital role in many applications.



Example of Stacks

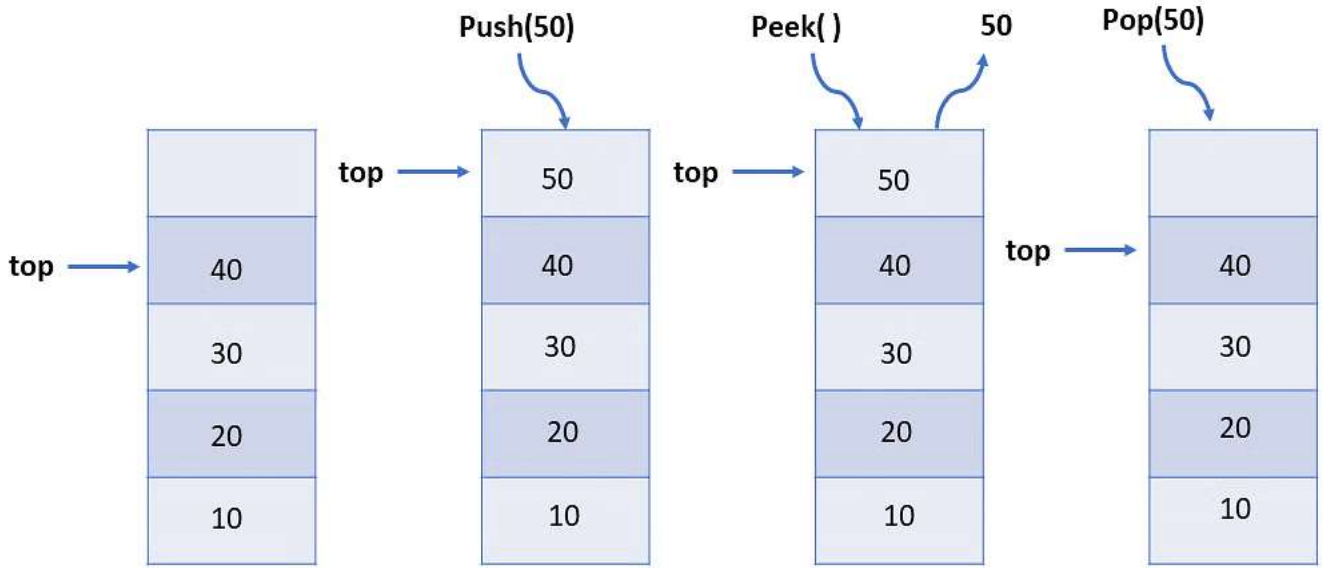


Stack Representation in Data Structures



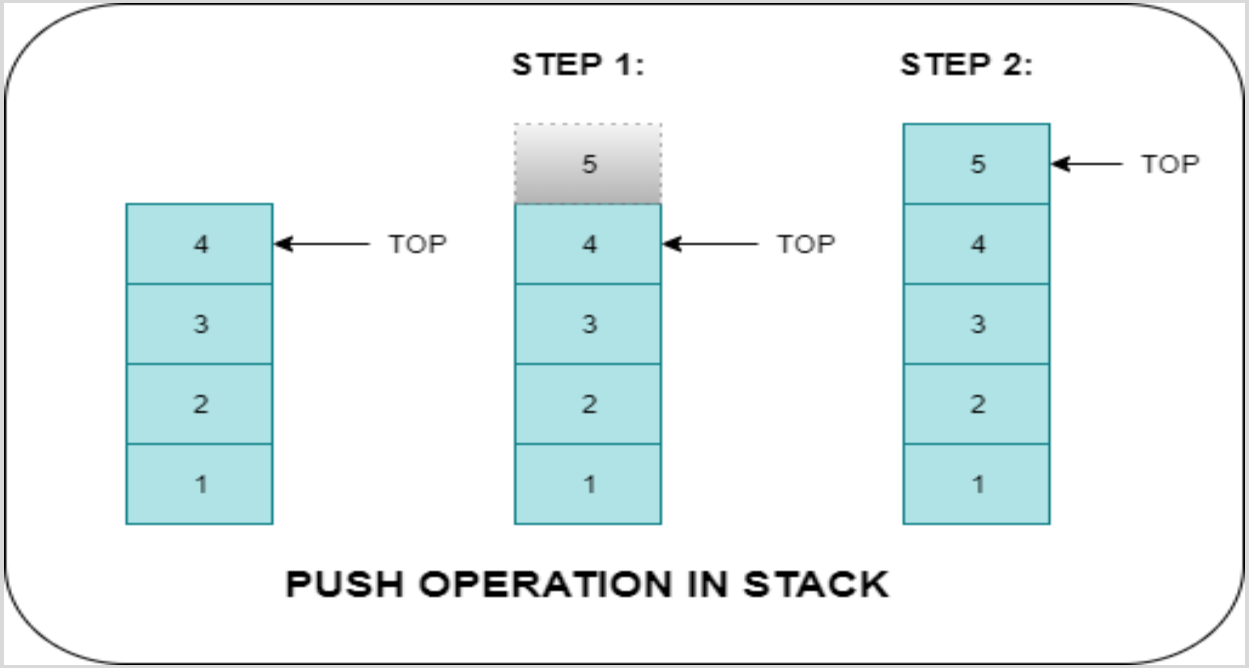
Operations On Stacks

- **push()** to insert an element into the stack
- **pop()** to remove an element from the stack
- **peek()** Returns the top element of the stack.
- **isEmpty()** returns true if the stack is empty else false.
- **size()** returns the size of the stack.


top
40
30
20
10
Push(50)
top
50
40
30
20
10
Peek( )
50
top
50
40
30
20
10
Pop(50)
top
40
30
20
10

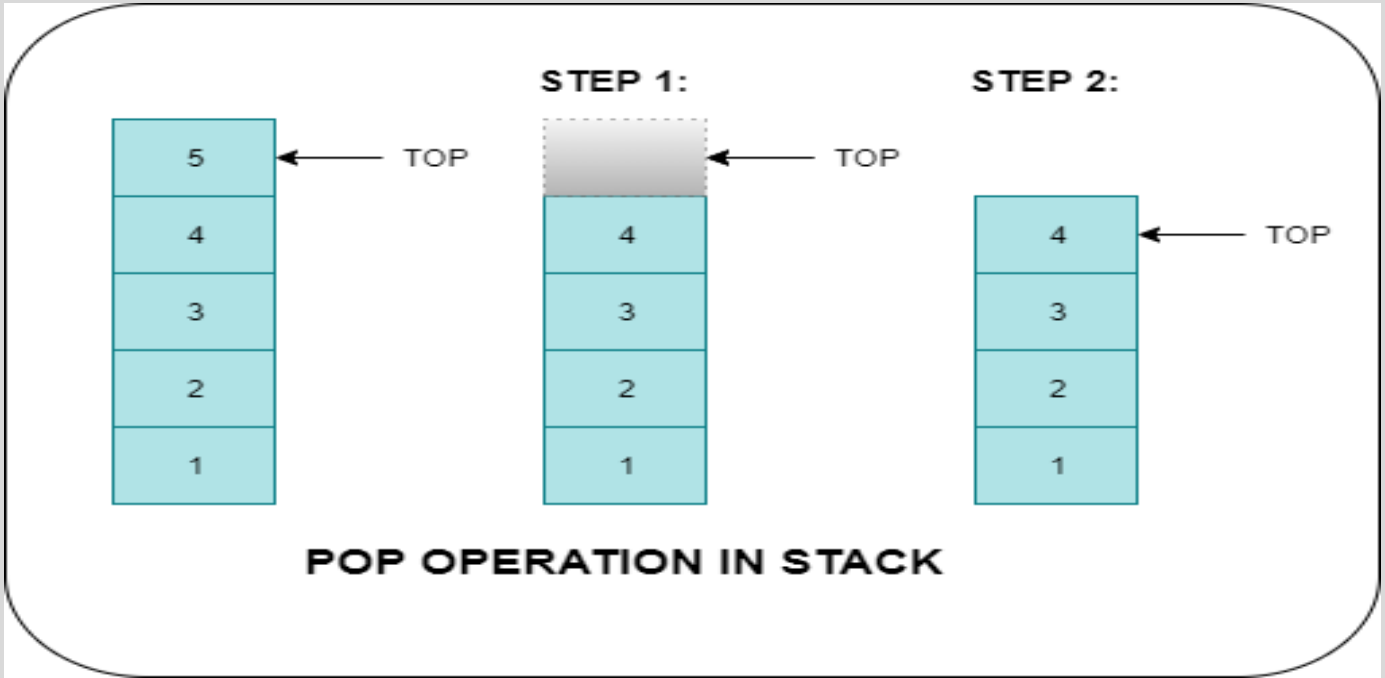
PUSH Operation

- What changes are made to the stack when a new element is pushed?
- A new element is inserted on top
- The value of top increases by 1



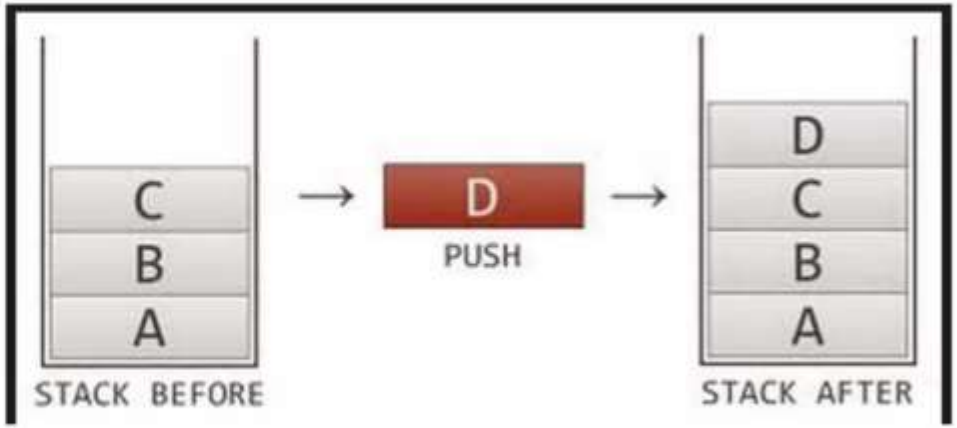
POP Operation

- What changes are made to the stack internally for a pop operation?
- The top element is removed
- The value of **top** is decreased by 1



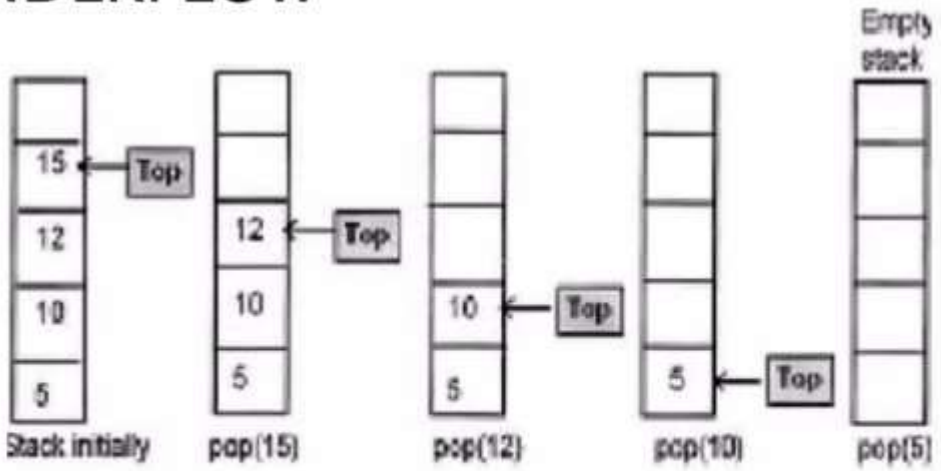
ALGORITHM OF INSERTION IN STACK: (PUSH)

1. Insertion(a,top,item,max)
2. If top=max then
   print 'STACK OVERFLOW'
   exit
   else
3. top=top+1
   end if
4. a[top]=item
5. Exit



ALGORITHM OF DELETION IN STACK: (POP)

1. Deletion(a,top,item)
2. If top=0 then
   print 'STACK UNDERFLOW'
   exit
   else
3. item=a[top]
   end if
4. top=top-1
5. Exit



Algorithm 4.3 : PEEK (S, TOP) - Given S be a one - dimensional array maintaining stack and variable TOP contain an index of the array element whose value is at the top of the stack. This algorithm returns the top element from the stack.

1. [Check for underflow state of stack]

   If TOP = 0 then

      Write " Underflow"

      return

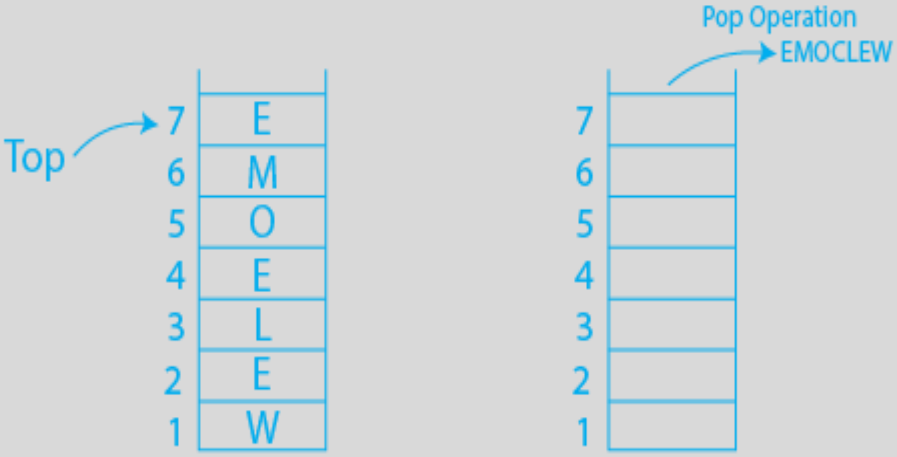
   [End of If structure]

2. ITEM ← S[TOP]

3. Return (ITEM)

Applications of Stacks

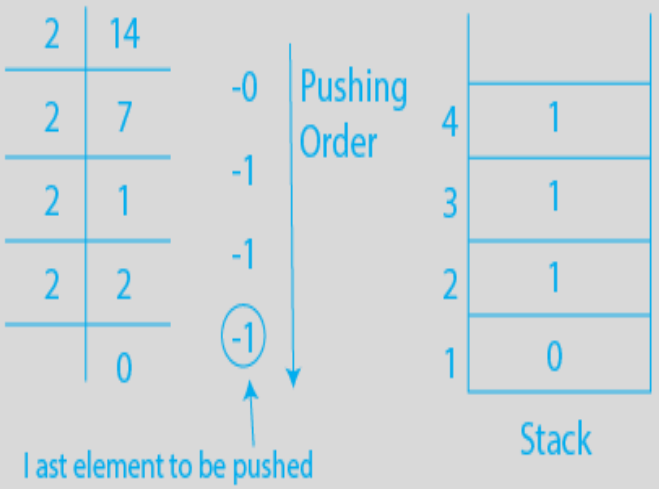
- **Reverse a String**
  A Stack can be used to reverse a string's characters. This can be done by simply popping each character off of the stack one at a time after pushing them onto the stack one at a time. The initial character of the Stack is at the bottom of the Stack, the last character of the String is at the top, and due to the Stack's last in first out property, after performing the pop operation in the Stack, the Stack returns the String in reverse order.

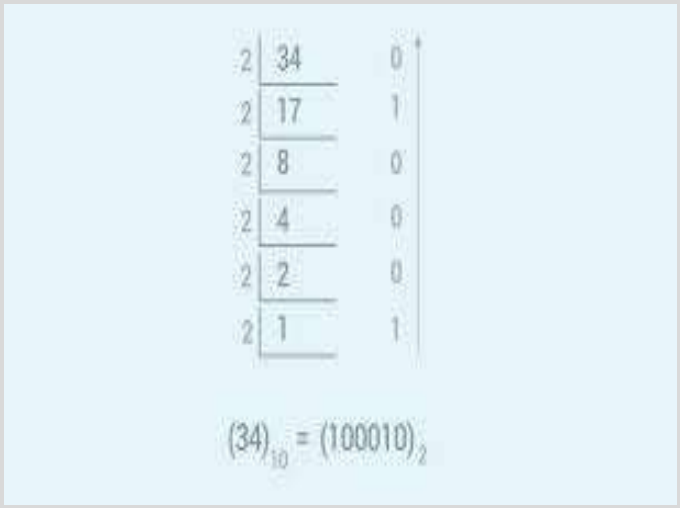


Converting decimal no.to binary

A number can be transformed from decimal to binary, octal, or hexadecimal using a stack. Any decimal number can be converted to a binary number by continually dividing it by two and pushing the residue of each division onto the stack until the result is 0. The binary counterpart ofthe provided decimal number is then obtained by popping the entire stack.

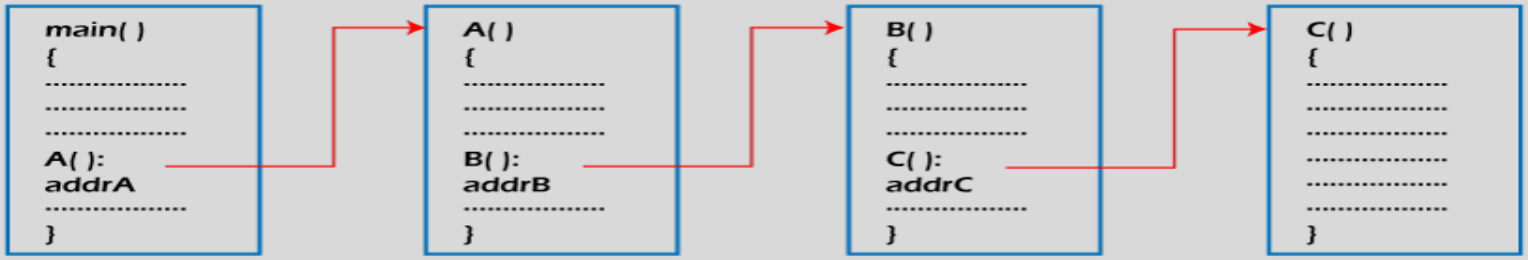
Example: Converting 14 number Decimal to Binary:



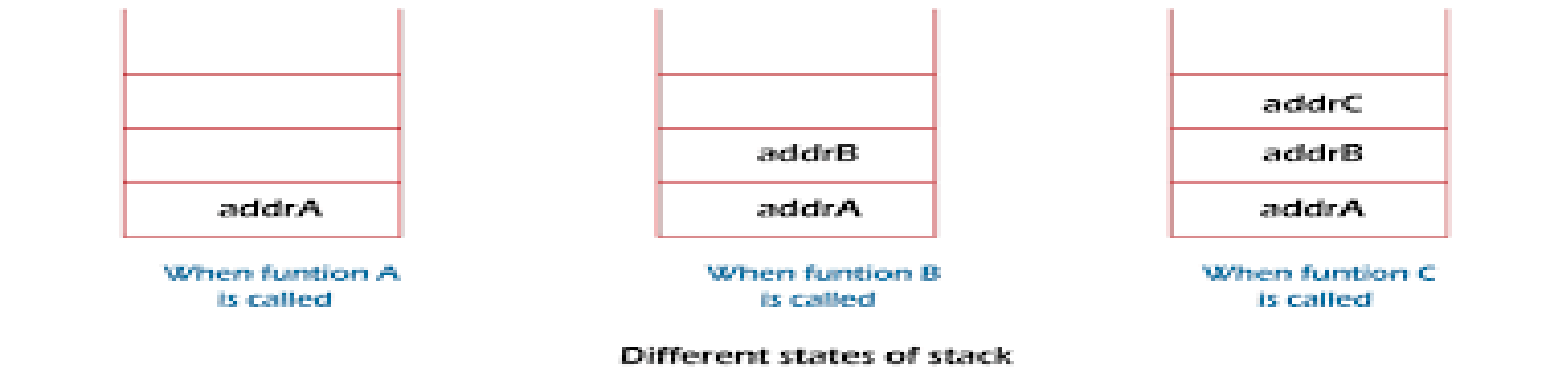


Processing Function Calls:

In programs that call multiple functions in quick succession, the stack is crucial. Assume we have a program with three A, B, and C functions. Function A calls Function B, and Function B calls Function C.



Function call



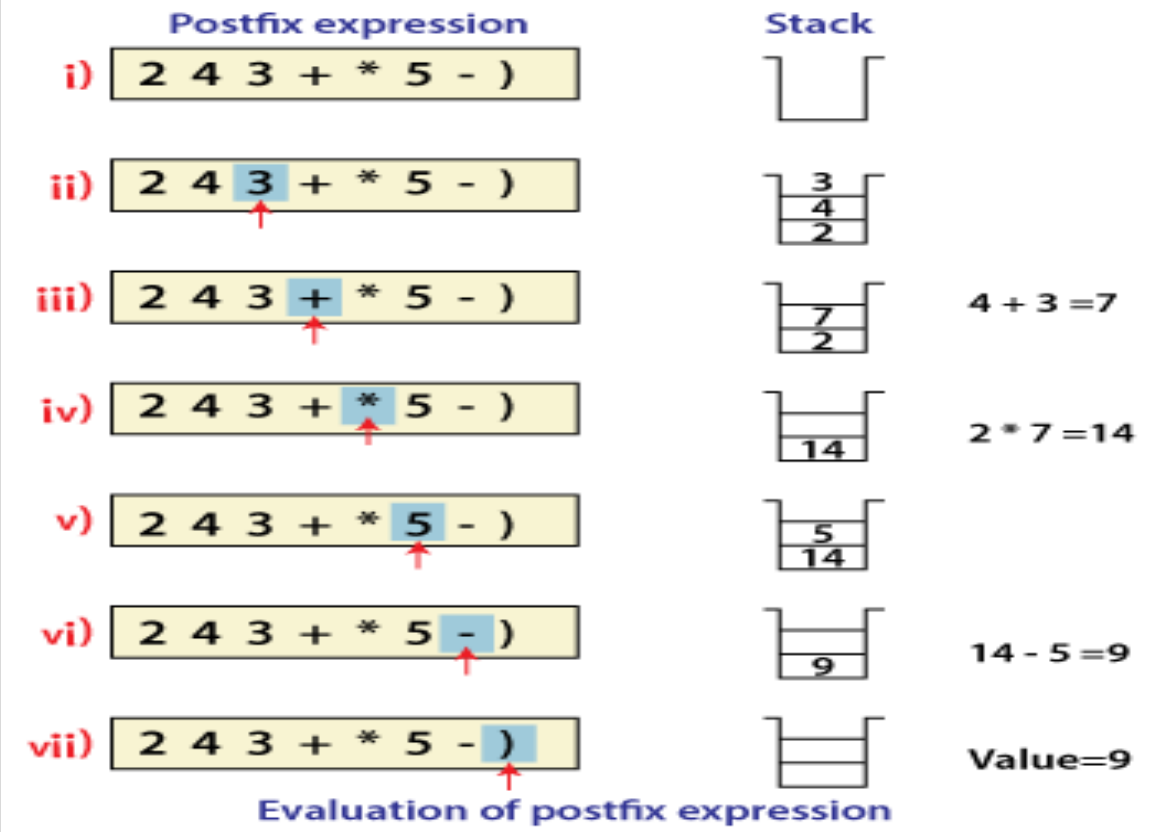
Different states of stack

Evaluating Postfix expression:

Stack is the ideal data structure to evaluate the postfix expression because the top element is always the most recent operand. The next element on the Stack is the second most recent operand to be operated on.

Postfix expression is 2 4 3 + * 5.

The following step illustrates how this postfix expression is evaluated.



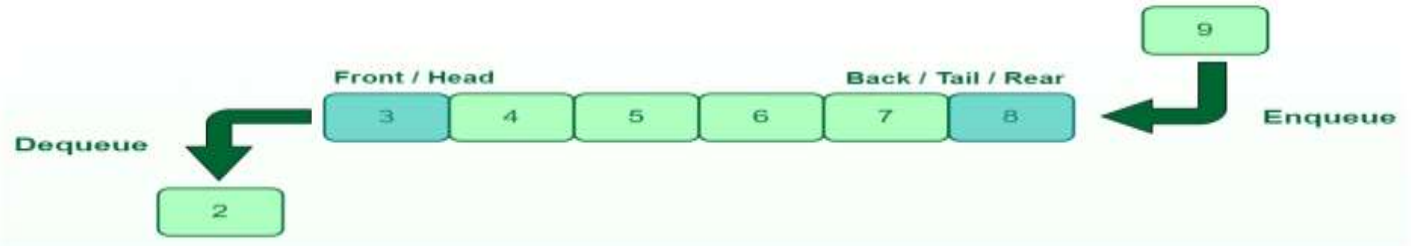
What is Queue Data Structure?

A **Queue** is defined as a linear data structure that is open at both ends and the operations are performed in First In First Out (FIFO) order.

FIFO Principle of Queue:

A Queue is like a line waiting to purchase tickets, where the first person in line is the first person served. (i.e. First come first serve).

- Position of the entry in a queue ready to be served, that is, the first entry that will be removed from the queue, is called the **front** of the queue(sometimes, **head** of the queue), similarly, the position of the last entry in the queue, that is, the one most recently added, is called the **rear** (or the **tail**) of the queue. See the below figure.

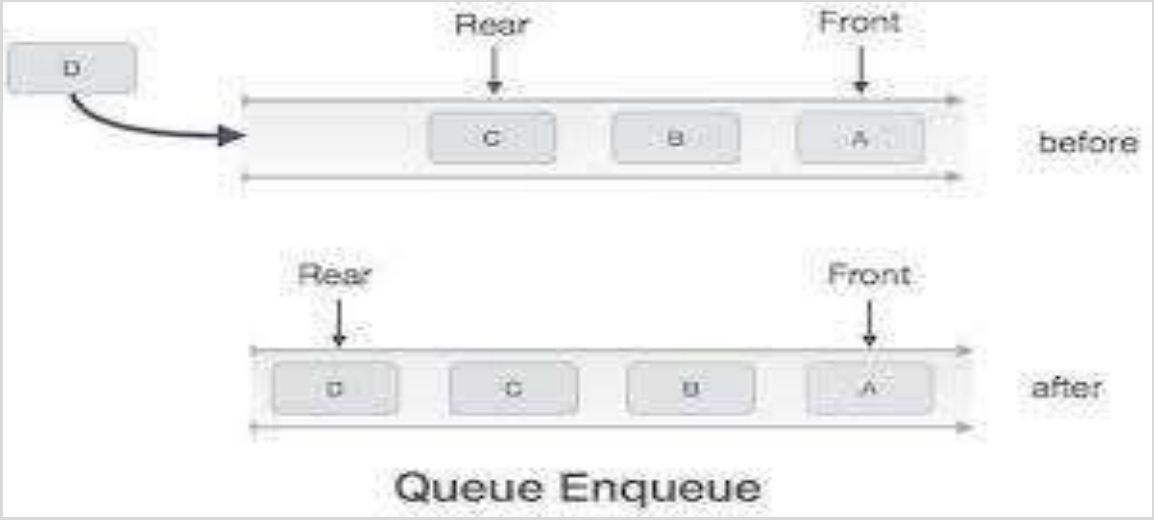


Queue Data Structure

Enqueue Operation

Algorithm : Enqueue (Q,FRONT,REAR,MAXSIZE,ITEM)

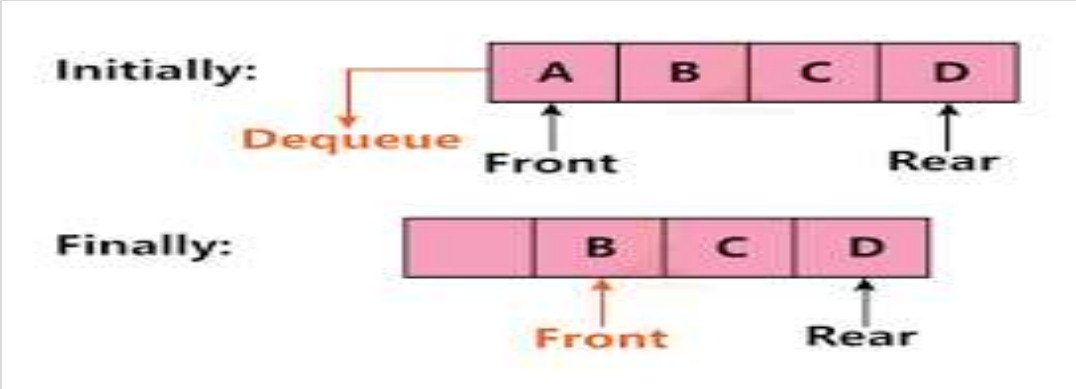
1. [check for queue overflow]
   If REAR=MAXSIZE then
   Write “Overflow”
   return
2. REAR <- REAR+1 [Increment REAR]
3. Q [REAR] <- ITEM [Insert ITEM]
4. Return



Dequeue Operation

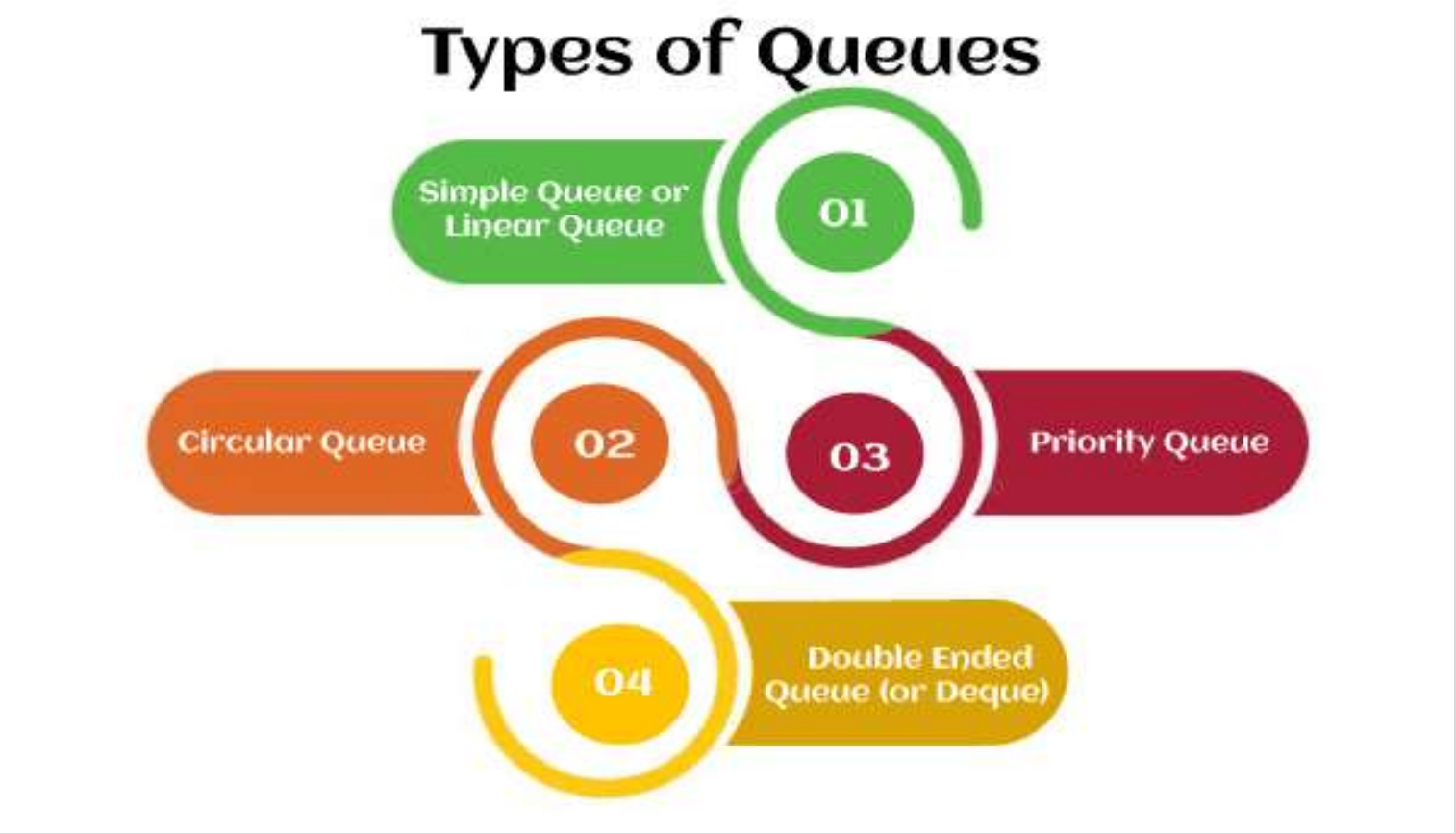
Algorithm : Dequeue (Q,FRONT,REAR,MAXSIZE)

1. [check for underflow state of queue]
   If REAR = 0 then
   Write “Underflow”
   return
2. ITEM <- Q[FRONT] [Copy element at FRONT to ITEM]
3. If FRONT < REAR then
   FRONT <- FRONT + 1 [Increment FRONT by 1]
   Else if FRONT = REAR [Queue contains only 1 element}
   FRONT <- 1, REAR <- 0 [Queue becomes empty]
4. Return (ITEM)



Types of Queue

There are four different types of queue that are listed as follows –

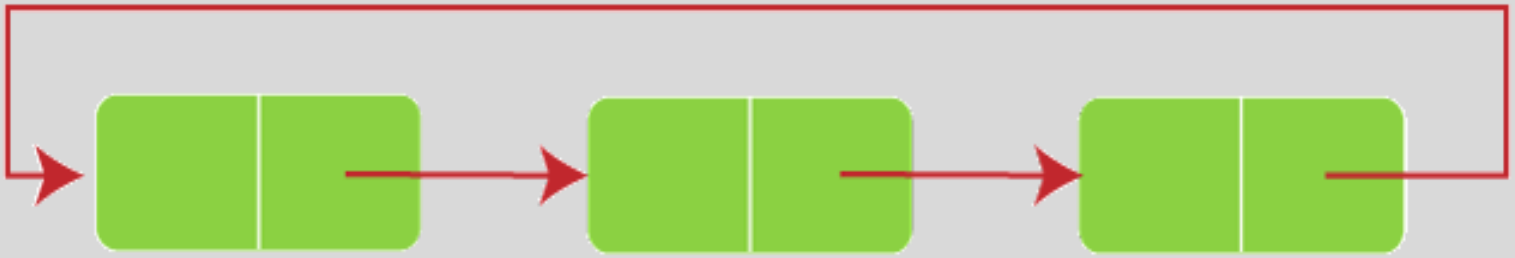


Simple Queue or Linear Queue

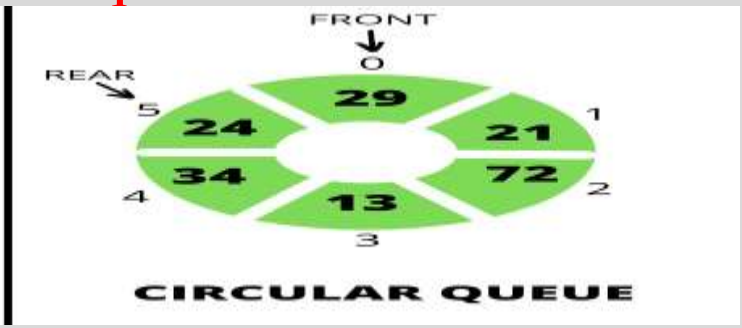
In Linear Queue, an insertion takes place from one end while the deletion occurs from another end. The end at which the insertion takes place is known as the rear end, and the end at which the deletion takes place is known as front end. It strictly follows the FIFO rule.

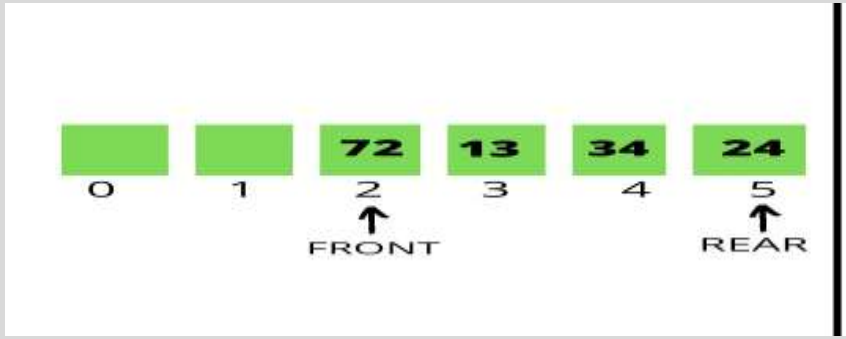
Circular Queue

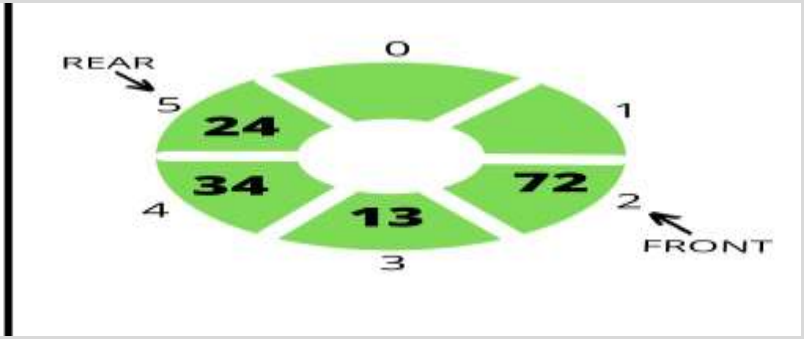
In Circular Queue, all the nodes are represented as circular. It is similar to the linear Queue except that the last element of the queue is connected to the first element. It is also known as Ring Buffer, as all the ends are connected to another end. The representation of circular queue is shown in the below image -

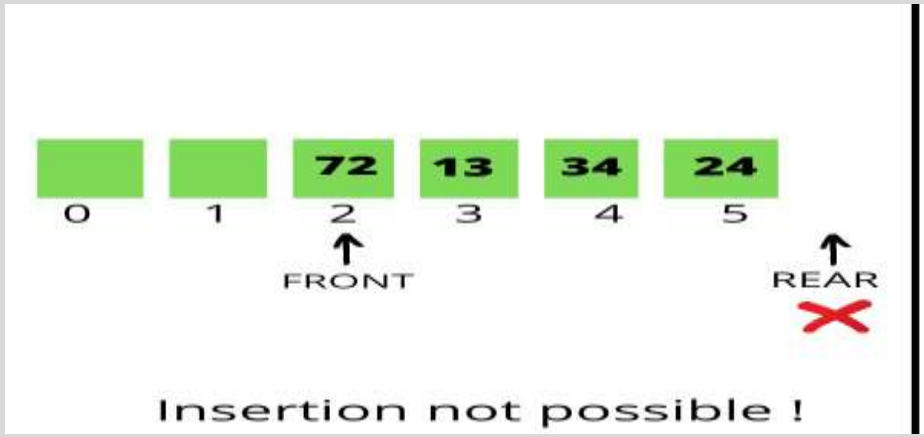


Linear V/s Circular queue





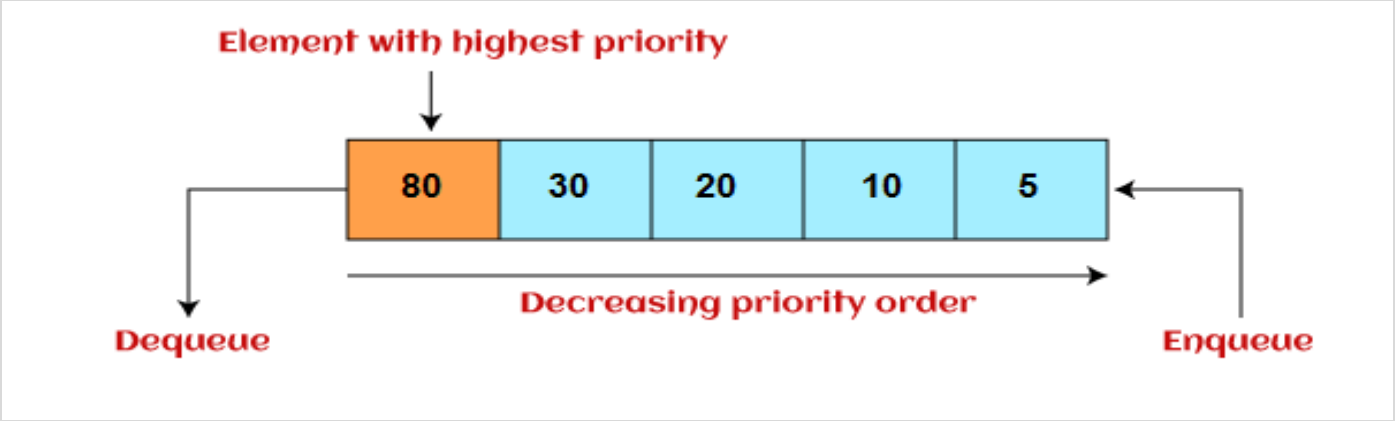




Priority Queue

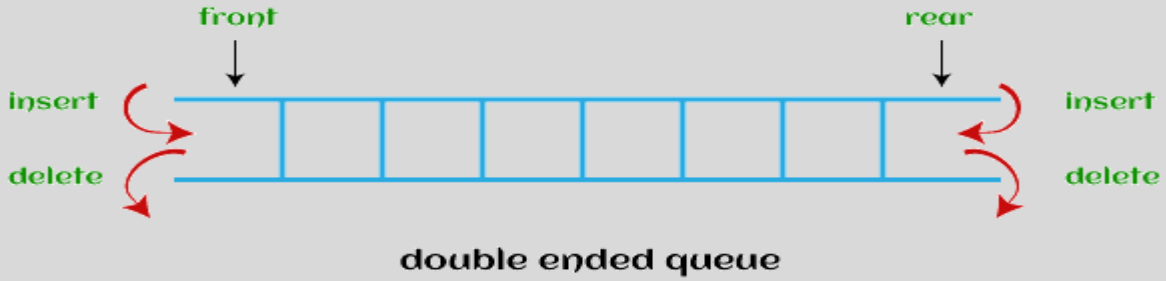
It is a special type of queue in which the elements are arranged based on the priority. It is a special type of queue data structure in which every element has a priority associated with it. Suppose some elements occur with the same priority, they will be arranged according to the FIFO principle. There are two types of priority queue that are discussed as follows –

- **Ascending priority queue -** In ascending priority queue, elements can be inserted in arbitrary order, but only smallest can be deleted first.
- **Descending priority queue -** In descending priority queue, elements can be inserted in arbitrary order, but only the largest element can be deleted first.



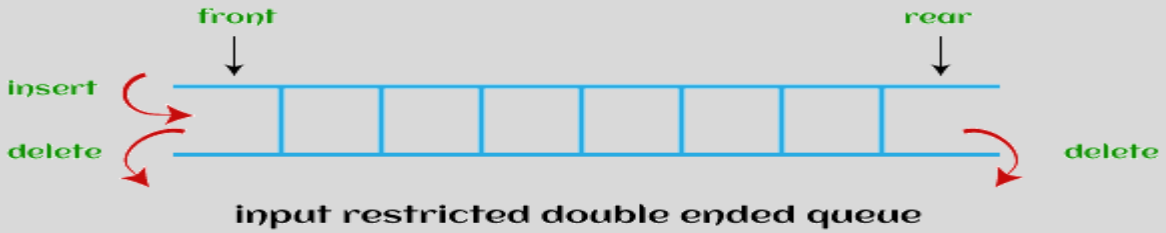
Deque (or Double Ended Queue)

In Deque or Double Ended Queue, insertion and deletion can be done from both ends of the queue either from the front or rear. It means that we can insert and delete elements from both front and rear ends of the queue.

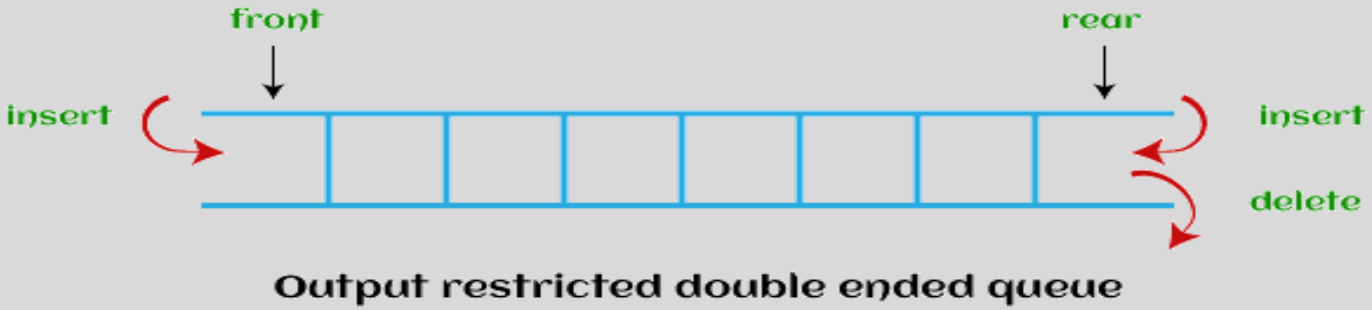


There are two types of deque that are discussed as follows −

Input restricted deque - As the name implies, in input restricted queue, insertion operation can be performed at only one end, while deletion can be performed from both ends.



Output restricted deque - As the name implies, in output restricted queue, deletion operation can be performed at only one end, while insertion can be performed from both ends.



THANKS