Data Structures & Operators

- Data structure
  - Classification of data structures
  - Operations on data structure
- Stack
  - Implementation
  - Operation
- Queue
  - Implementation
  - Operation
- Linked list
  - Implementation
  - Operation

Data structures

- Way of organizing similar or dissimilar logically related data items into a single unit
- Classification of data structure
  - Based on data type

Simple data structure

— Array
— Structure

[mixture]

Compound data structure

Linear

— Stack
— Queue
— Linked list

Non Linear

— Tree
— Graph

- Based on Memory

Static data structure	Dynamic data structure
— Array	— Linked list

- Static Data Structure
  - It is associated with primary (main) memory.
  - Memory allocation done before the execution of the program.
  - It is fixed.
  - Example : Array
- Dynamic Data Structure
  - It is associated with secondary (auxiliary) memory.
  - Memory allocation done during the execution of the program.
  - It is flexible
  - Example : linked list, Files

- Operations on Data Structure
  1. Traversing
  2. Searching
  3. Inserting
  4. Deleting
  5. Sorting
  6. Merging

1. Traversing

- Accessing / visiting / reading all elements of a data structure is called traversing
- Example : Reading all the element in an array

2. Searching

- Process of finding elements in a data structure is called searching
- Two methods :
    1) Linear searching
    2) Binary searching

3. Inserting

- Process of adding new elements at a particular location in a data structure is called inserting
- Example : Adding new data into an array

4. Deleting

- Process of removing a particular element from a data structure is called inserting
- Example : delete a data from an array

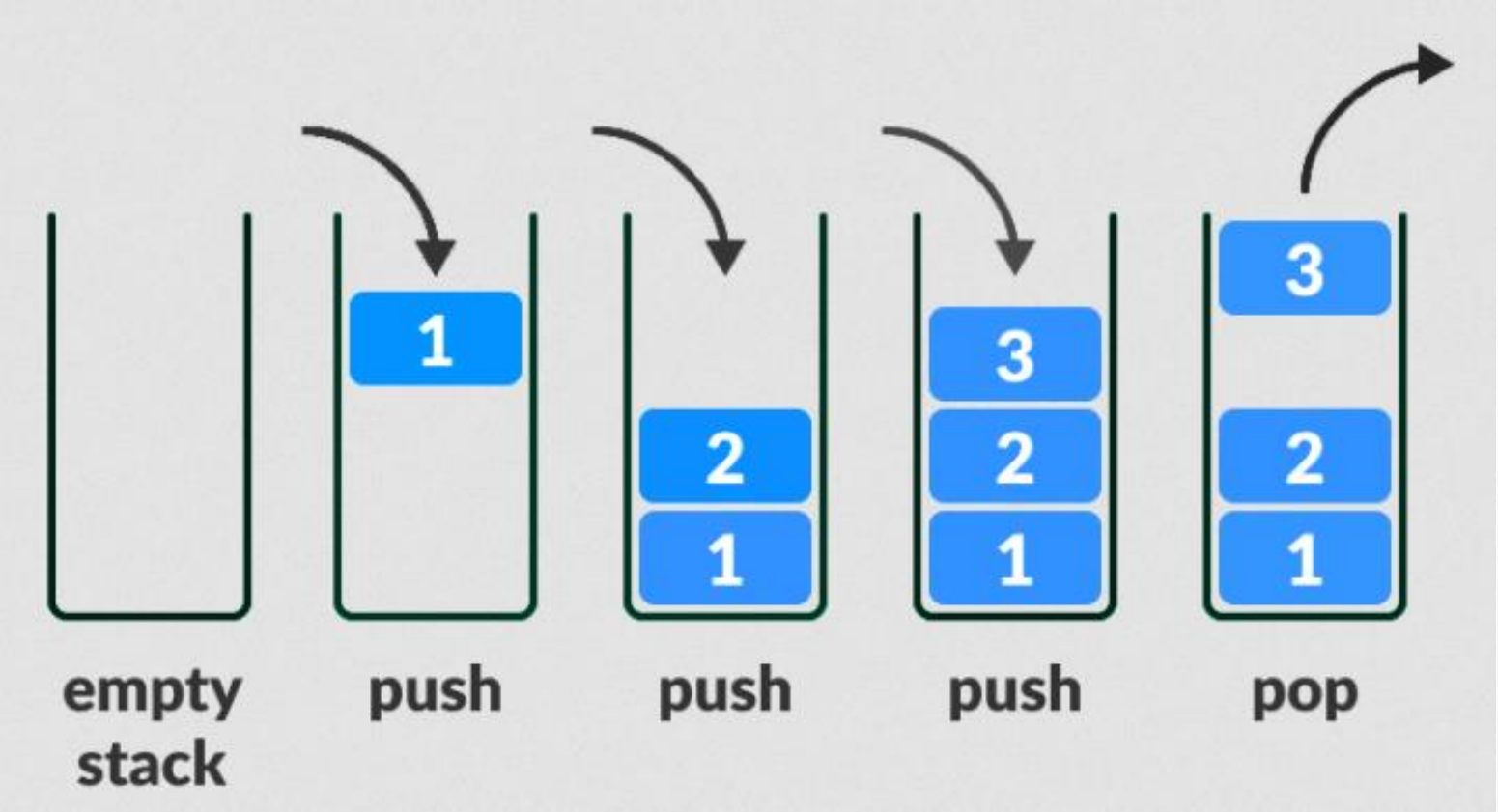
5. Sorting

- Arranging elements in a particular order is called sorting.
- Two methods :
    1) Bubble sorting
    2) Selection sorting

5. Merging

- Process of combining 2 sorted data structure and form a new one is called merging
- Example : combining 2 arrays.

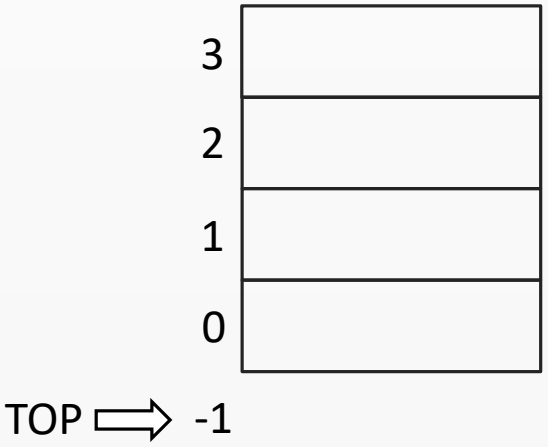
Stack

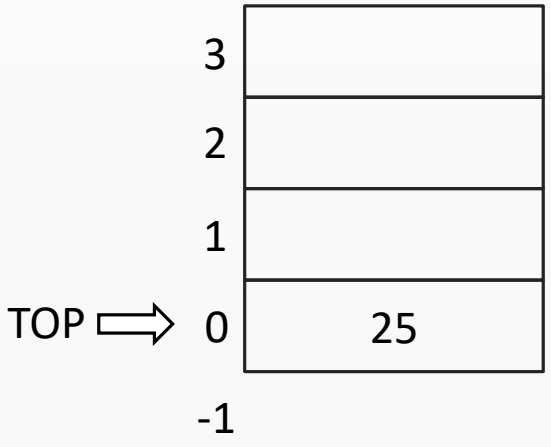


- Stack
  - Stack is a logical concept
  - It is a static data structure
  - The data structure follows “LIFO” principal is known as stack.
  - LIFO : Last In First Out.
  - Def :

    “ A stack is a linear structure in which items can be added or remove only at one end called TOP ”
  - Add an item into stack is called PUSH
  - Remove an item from the stack is called POP

- Implementation of stack
  - Stack can be implemented using array
  - Initially TOP value is set to -1 [denote stack is empty]
  - when a data added to the stack, TOP is incremented by 1
  - Index of 1st element is ‘0’ and last element is ‘N-1’.





- Operations on stack
  1. PUSH operation
  2. POP operation

1. PUSH operation
   — It is the process of adding a new item into the stack
   — If the stack is full, it makes the stack “overflow”

- Algorithm of PUSH Operation

Step 1 : If TOP = N

then print “Overflow” and return

Step 2 : Set TOP = TOP + 1

Step 3 : Set Stack[TOP] = item

Step 4 : Stop

- Operations on stack
  1. PUSH operation
  2. POP operation

2. POP operation

— It is the process of deleting an item from the stack

— If the stack is empty, it makes the stack “underflow”

- Algorithm of POP Operation

Step 1 : If TOP = Null

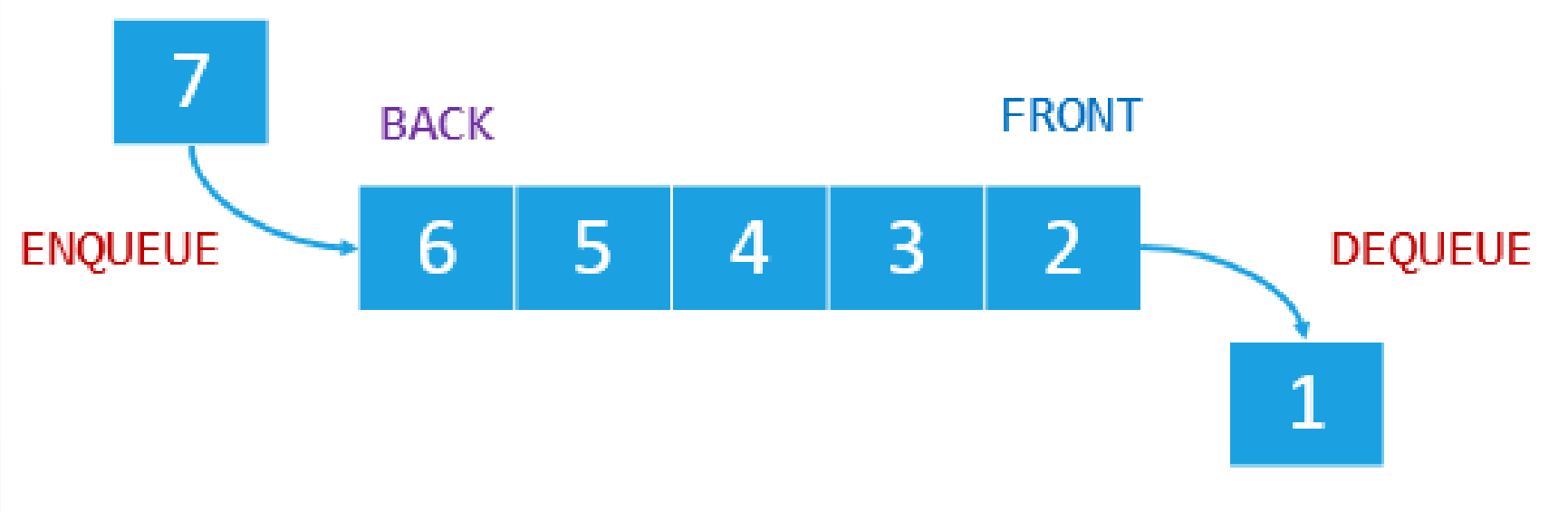
then print “Underflow” and return

Step 2 : Set item = Stack[TOP]

Step 3 : Set TOP = TOP - 1

Step 4 : Stop

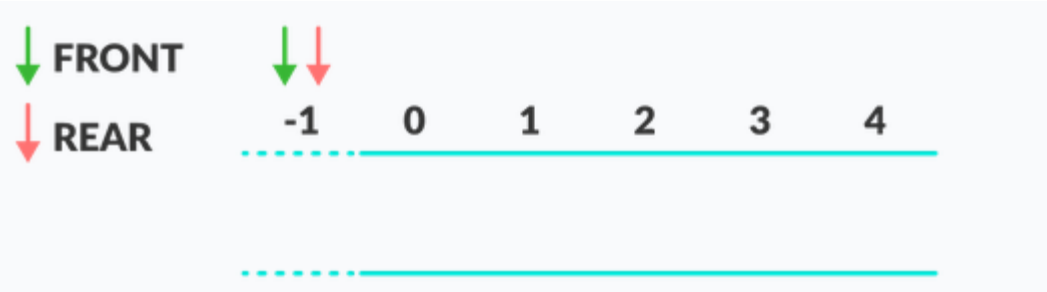
Queue



- Queue
  - Queue is a logical concept
  - It is a static data structure
  - The data structure follows “FIFO” principal is known as queue.
  - FIFO : First In First Out.
  - Queue have 2 ends
    1. REAR : adding a new item into the queue.
    2. FRONT : Removing an item from the queue.

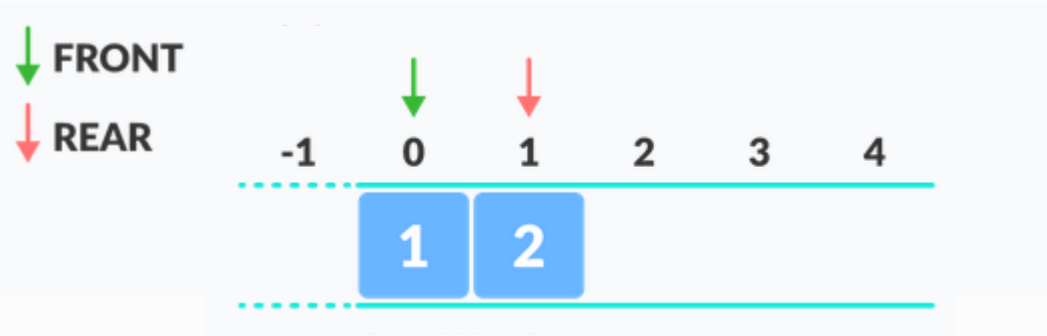
- Implementation of queue
  - Queue can be implemented using array
  - New data added at “REAR” end and removed at “FRONT” end.
  - Initially the value of FRONT and REAR set to -1 [denote queue is empty]
  - A new data is added to a queue, the REAR is incremented by 1.
  - A data removed from the queue, the FRONT is incremented by 1.

- Queue is empty

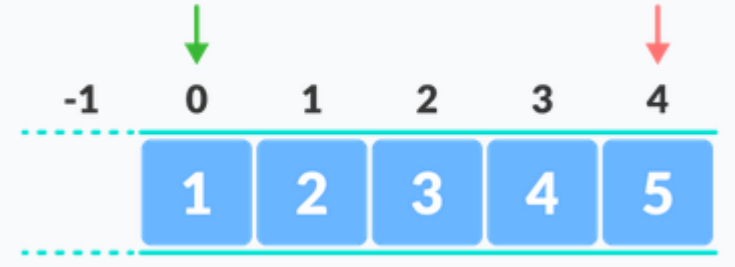


- Add First element to the Queue

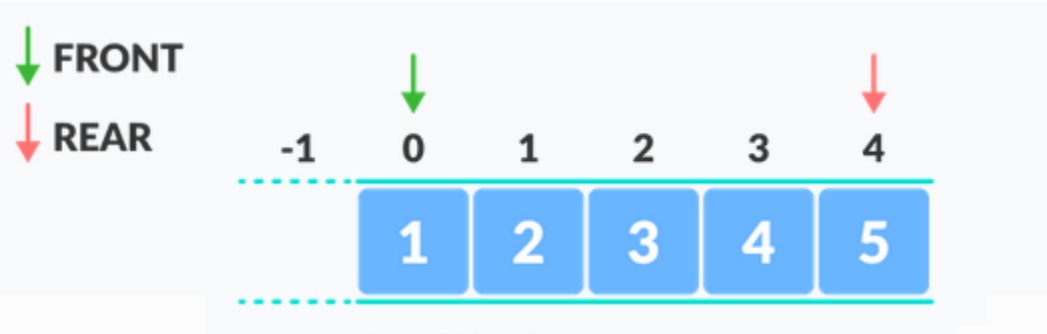
- Add second element to the queue



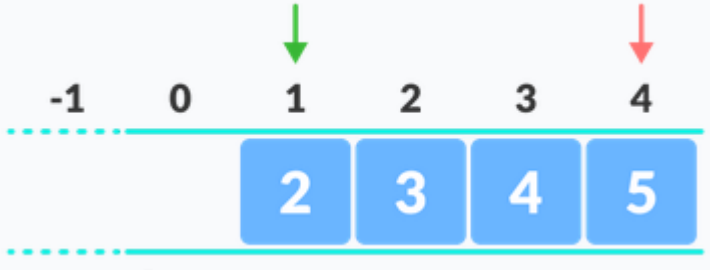
- Add more element to the Queue



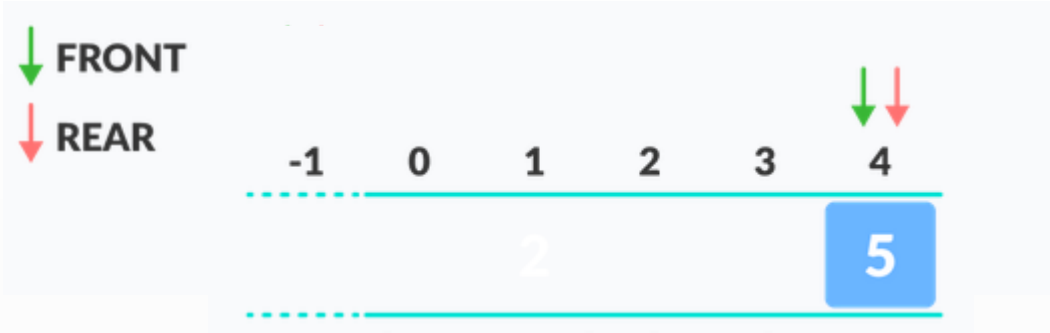
- Queue is full



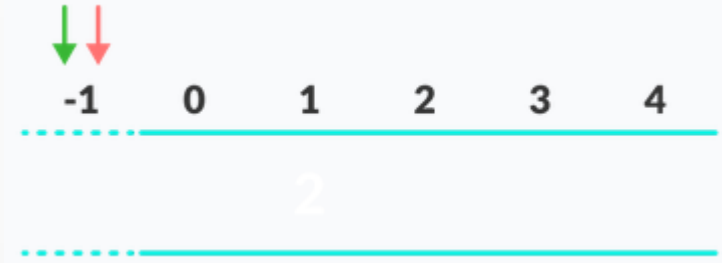
- Remove one item from the Queue
- [First added item]



- Remove more items [left 1 item]



- If last item deleted [empty queue]



- Operations on Queue
  1. Insertion operation
  2. Deletion operation

1. Insertion operation
   — It is the process of adding a new item into the queue at the ‘REAR end
   — If the queue is full, it makes the stack “overflow”

Algorithm of Insertion Operation

Step 1 : If REAR = N-1 or FRONT = REAR + 1
print “Overflow”
goto Step 4
End if

Step 2 : If FRONT = -1 and REAR = -1
Set FRONT = REAR = 0
Else
Set REAR = REAR + 1
End If

Step 3 : Set Queue[REAR] = item

Step 4 : Stop

- Operations on Queue
  1. Insertion operation
  2. Deletion operation

2. Deletion operation

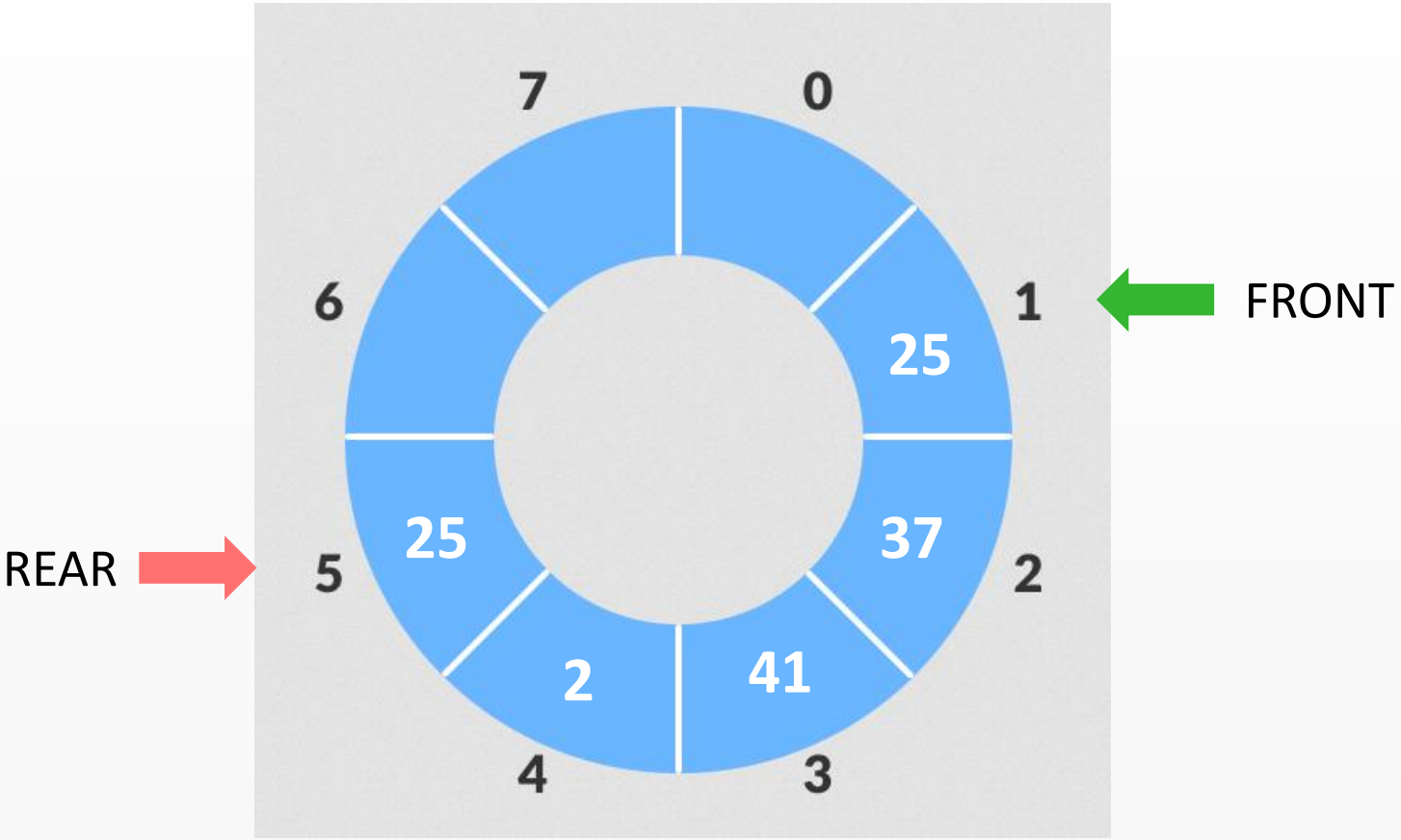
— It is the process of removing an item from the queue at the ‘FRONT’ end

— If the queue is empty, it makes the stack “Underflow”

Algorithm of Deletion Operation

```
Step 1   :   If FRONT= -1 or FRONT > REAR
                 print “Underflow”
             Else
                 Set item = Queue[ FRONT ]
                 Set FRONT = FRONT + 1
             End If
Step 2   :   Stop
```

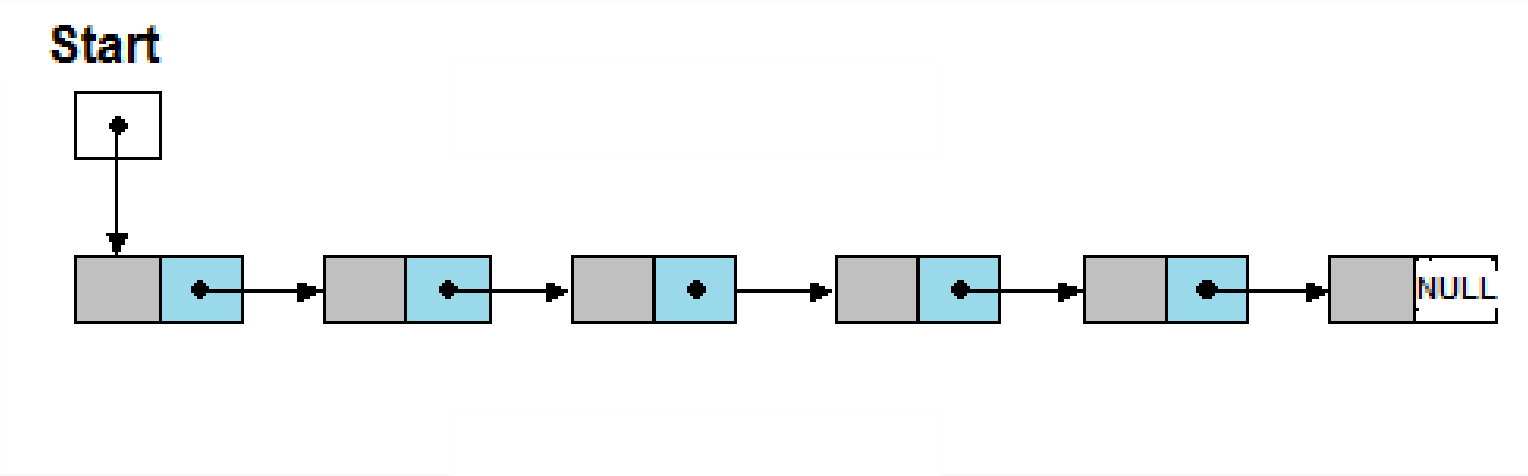
Circular Queue



- Circular Queue
  - Queue is a logical concept
  - It is a static data structure
  - Circular in shape
  - Def :

    “Circular queue is a queue in which the two end points are connected”
  - Limitations of linear queue can overcome by circular queue.

Linked List



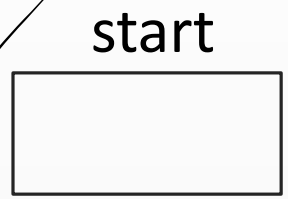
- Linked List
  - It is a dynamic data structure
  - It grow and shrinks as when the new items are added and remove respectively
  - An element in a linked list is called Node
  - A node have 2 part : data and address.
  - Linked list is a collection of nodes
  - A node contains data and address of next node .
  - Last node contains data and a null pointer [no address]

- Implementation of Linked List
  - A node consist of data and address of next node
  - Address is a pointer
  - Linked list is created with the help of self referential structure

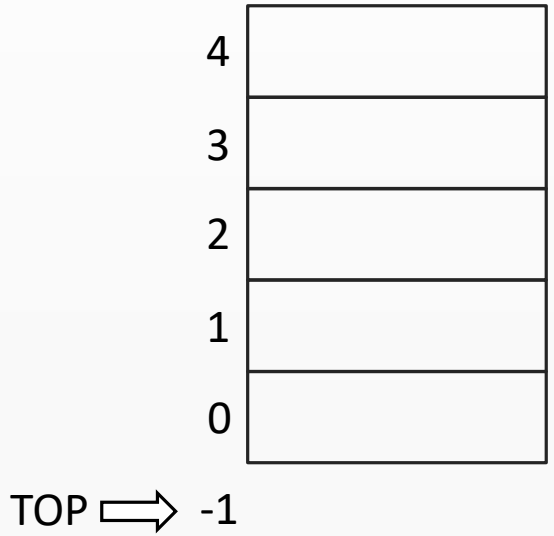
```
struct node
    {
        int data;
        node *link;
    };
```

- 1st node is called “Start”. [Special pointer]
- 1st node is creating using

struct node * start Or node * start



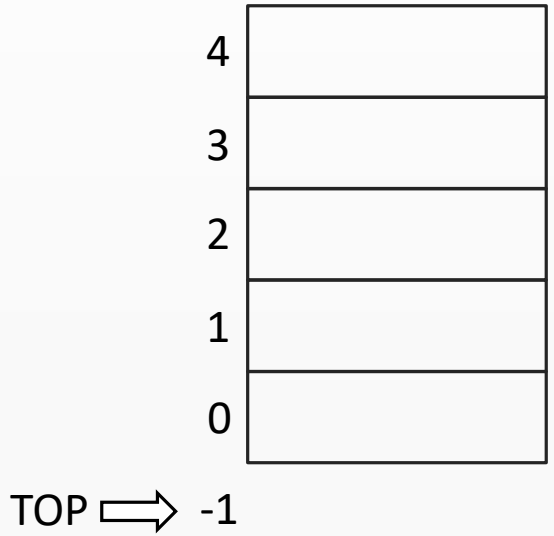
- when a data added to the stack, TOP is incremented by 1




```
#include <iostream>
using namespace std;

int main()
{
    int array[5];
    int TOP = -1;
}
```

- Implementation of stack
  - Stack can be implemented using array
  - Step 1 : Initially TOP value is set to -1 [denote stack is empty]




```
#include <iostream>
using namespace std;

int main()
{
    int array[5];
    int TOP = -1;
}
```