Expt No. 10.1 & Expt No 10.2

Aim: Static Implementation of Deque and dynamic implementation of Deque

Theory :

Double Ended Queue is a Queue data structure in which the insertion and deletion operations are performed at both the ends (**front** and **rear**). That means, we can insert at both front and rear positions and can delete from both front and rear positions.

Double Ended Queue can be represented in TWO ways, those are as follows:

1. Input Restricted Double Ended Queue : the insertion operation is performed at only one end and deletion operation is performed at both the ends.

2. Output Restricted Double Ended Queue: the deletion operation is performed at only one end and insertion operation is performed at both the ends.

Operations on Deque:

Mainly the following four basic operations are performed on queue:

insetFront(): Adds an item at the front of Deque.

insertRear(): Adds an item at the rear of Deque.

deleteFront(): Deletes an item from front of Deque.

deleteRear(): Deletes an item from rear of Deque.

Circular array implementation deque

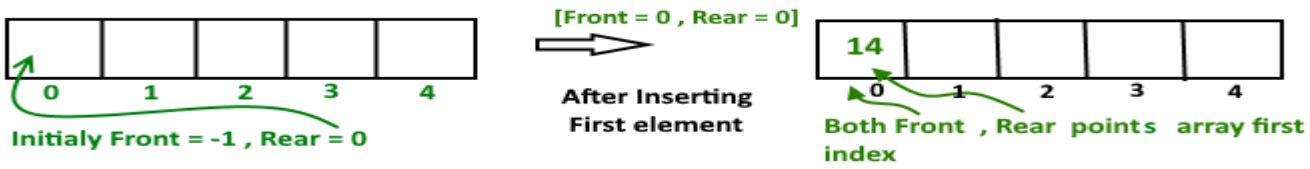
For implementing deque, we need to keep track of two indices, front and rear. We enqueue(push) an item at the rear or the front end of qedue and dequeue(pop) an item from both rear and front end.

Working :

1. Create an empty array 'arr' of size 'n'

initialize **front = -1** , **rear = -1**

Inserting First element in deque either front end or read end they both lead to the same result.



After insert **Front** Points to 0 and **Rear** points to 0

Algorithm:

1) Insert Elements at Rear end

a) First we check deque if Full or Not . if it is not full go to step b otherwise return

b) IF Rear == Size-1

then reinitialize Rear = 0 ;

Else

increment Rear by '1'

c) Store value into Arr[Rear] = val

2) Insert Elements at Front end

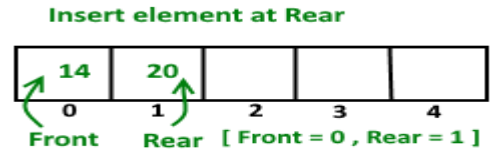
a) First we check deque if Full or Not. if it is not full go to step b otherwise return

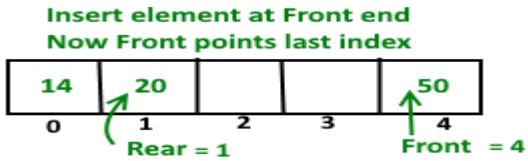
b). IF Front = 0 || initial position

move Front to point to last index of array

Else decremented front by '1'

c) Store value into Arr[Front] = val





3) Delete Element From Rear end

a) Check whether deque is Empty or Not. If empty return otherwise go to step b

b) Print the element at index Rear

c) If deque has only one element i.e.(front=rear)

SET front = -1 ; rear =-1 ;

Else IF Rear points to the first index of array then

move Rear to last index of the array

Else

Decrement Rear by '1'

4) Delete Element From Front end

a) Check whether deque is Empty or Not. If empty return otherwise go to step b

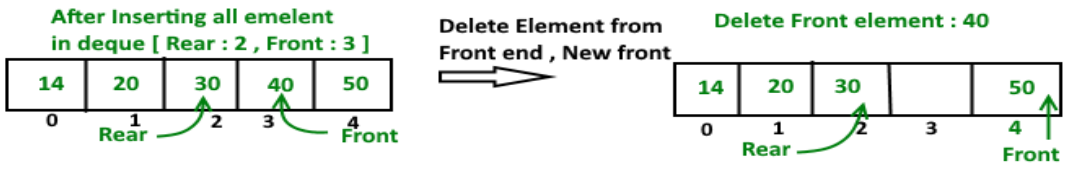
b) Print the element at index Front

c) If deque has only one element i.e.(front=rear)

SET front = -1 ; rear =-1 ;

Else IF front points to the last index of the array then move front to point to first index of array

Else increment Front by '1'



Post Lab Questions:

1) Discuss real world application of Deque.