Queues -1

Lecture-49

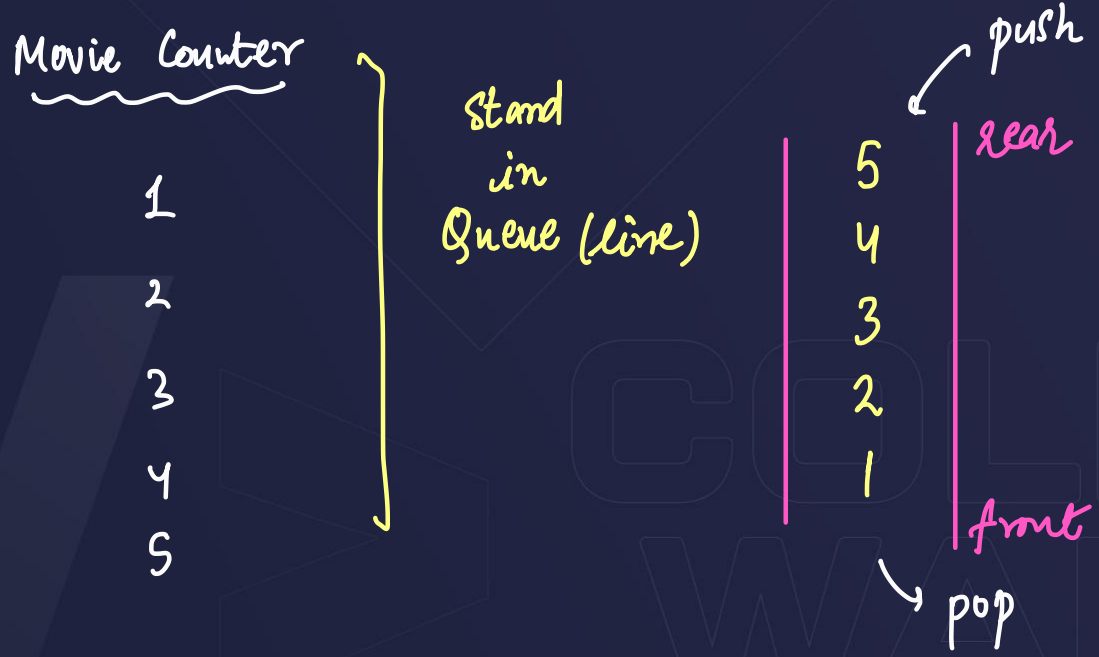
Raghav Garg

Today's checklist

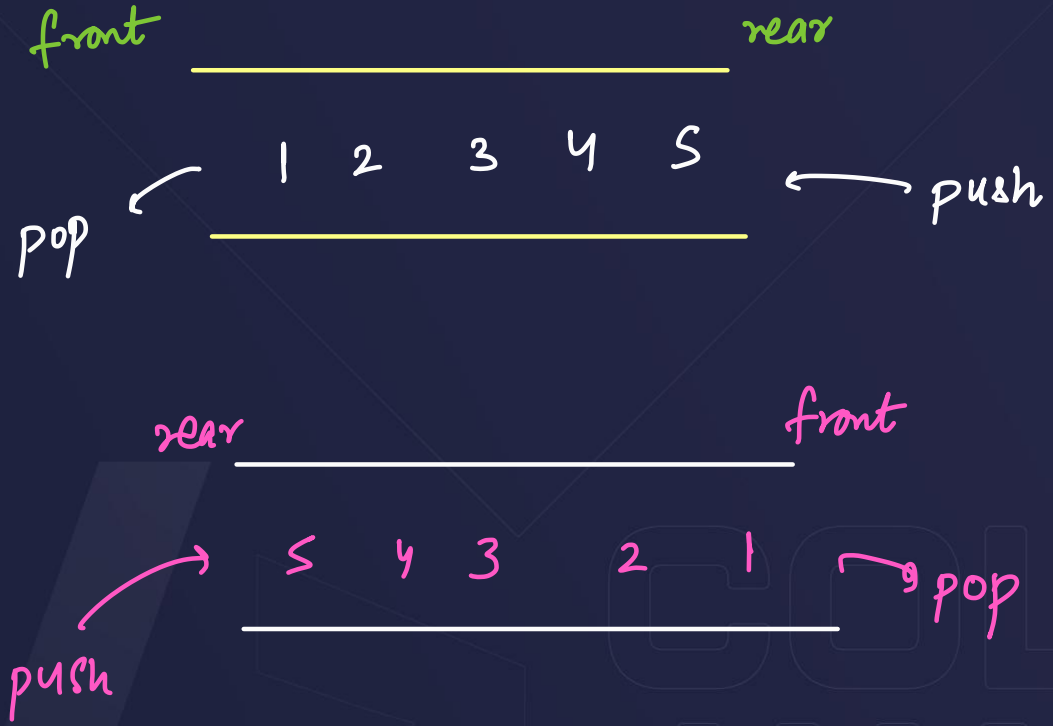
1) Introduction
2) Queue vs Stack
3) STL Operations performed on queue
4) Overflow vs Underflow
5) Array implementation of queue
6) Linked list implementation of queue
7) Introduction to Circular queue
8) Array implementation of circular queues
9) Deque introduction

Introduction

FIFO → first in first out



Introduction



Queue vs Stack

FIFO

↓

push & pop are done in opposite directions

push → front

pop →

LIFO

↓

push & pop are from same direction

↓

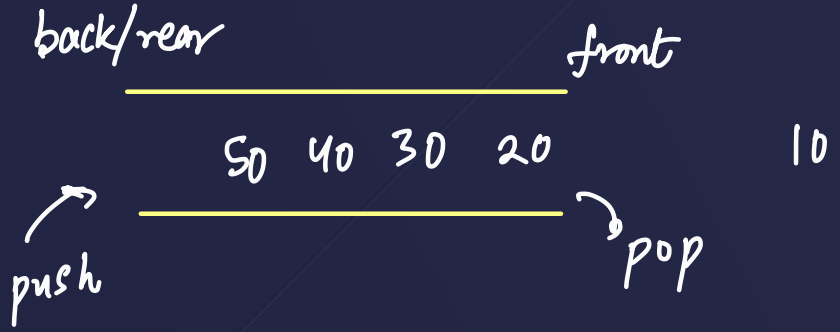
top

pop

push

STL Operations performed on queue

```
q.push(10);
q.push(20);
q.push(30);
q.push(40);
q.push(50);
cout<<q.front();
```




```
q.pop();
cout << q.front();
```

push() / add()

↓

insertion happens only at the back/rear. → O(1)

pop()

↓

only happens at front → O(1) T.C.

front() / top()

↓

we can access the front of the queue → O(1)

back()

↓

we can also access the rear element → O(1)

size()

↓

returns the size of queue

empty()

↓

it returns true if size() == 0

else it returns false

Overflow and Underflow

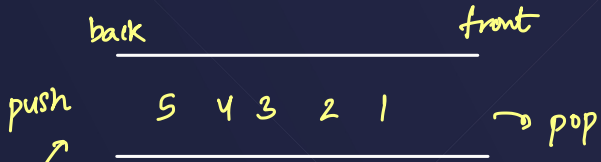
- Only happens if we implement the queue via an array if you fill the array
- You are out of memory

- Whenever the queue is empty & we try to use these functions
  - pop(), front(), back(),

Display:

T.C. = O(n)

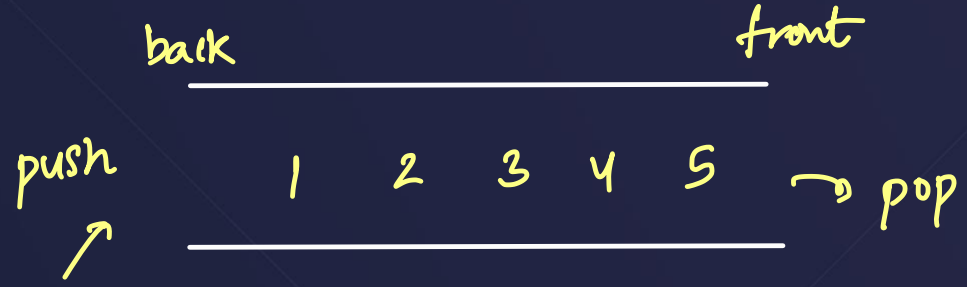
S.C. = O(1)



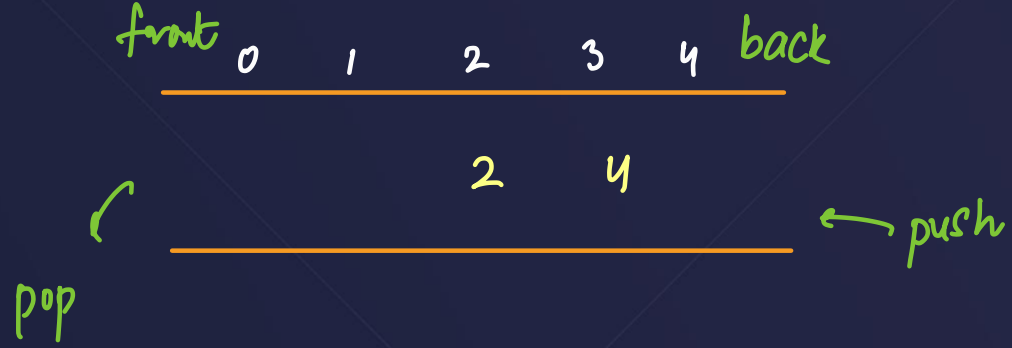
Output

1
2
3
4
5

Ques: Reverse the queue using a stack.

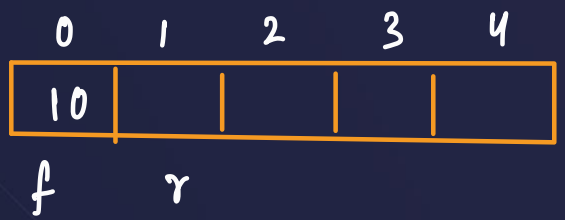


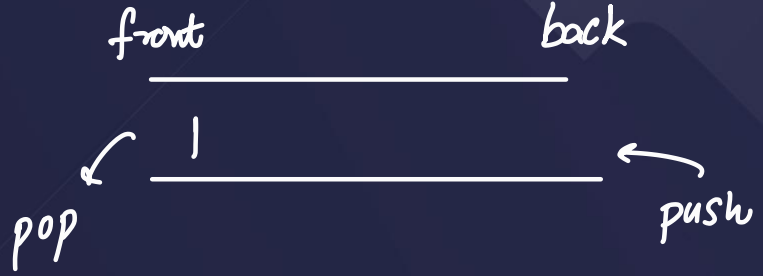
Ques: Remove all the elements present at even positions in queue. Consider 0-based indexing.




```
for(i=0; i<q.size(); i++){
    if(i%2 == 0) q.pop();
    else {
        int x = q.front();
        q.pop()
        q.push()
    }
```

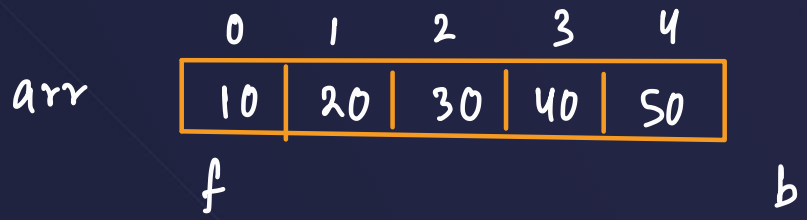
Array implementation of queue





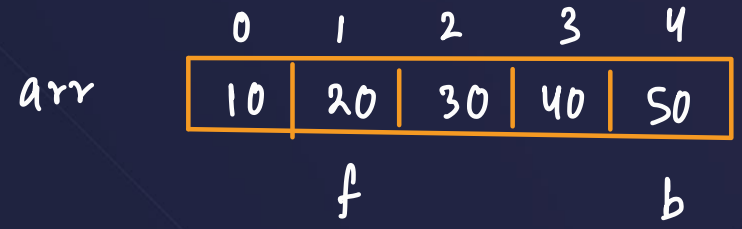

```
q.push(10) ; // push happens at rear
```

Implementing push()




```
void push(int val){
    if(b == arr.size()) cout << "queue is Full"; return
    arr[b] = val;
    b++;
}
```

Implementing pop()




```
void pop(){
    if(size()==0) return;
    f++;
}
```

Implementing front() , back()

```
int front(){
    if(size()==0) return -1;
    return arr[f];
}
```

```
int back(){
    if(size()==0) return -1;
    return arr[b-1]
}
```

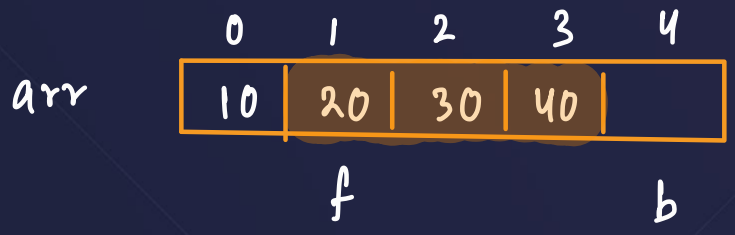
Implementing size()

```
int size(){
    return b-f;
}
```

Implementing empty()

```
bool empty(){
    if(size()==0) return true;
    else return false;
}
```

Implementing display()




```
for (int i=f;  i<b ; i++){
    cout << arr[i] << " ";
}
```

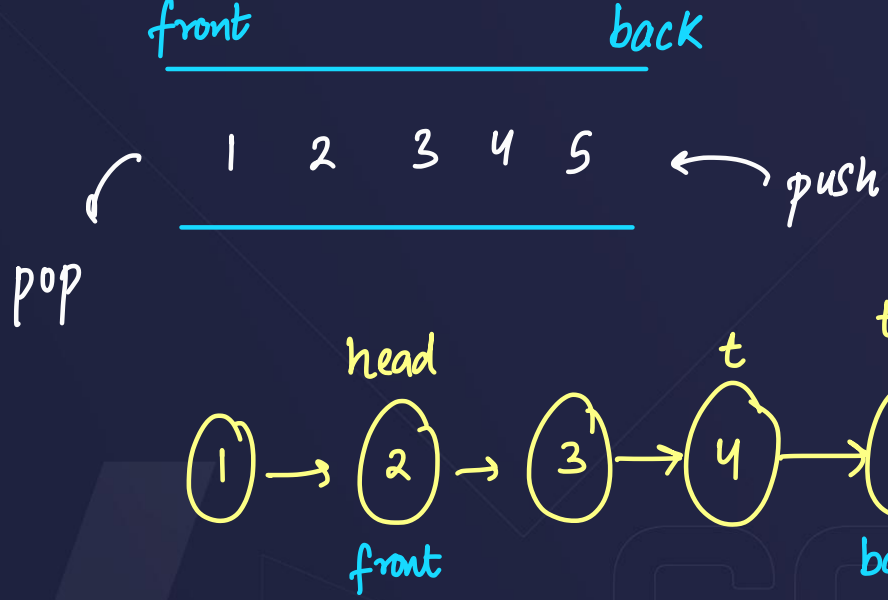
3

Problems in Array Implementation of Queue

wastage of space

Solution : Circular Array

Linked list implementation of queue



LL → push → insertAt Tail

pop → deleteAtHead

front = head → val

back = tail → val

size → LL size

push(4)

push(5)

Implementing push()

↓

Insert At Tail

Implementing pop()

DeleteAtHead

Implementing front() / back()

head → val

tail → val

Implementing size()

L

size of LL

Implementing empty()

Implementing display()

Advantage of Linked list implementation of queue over array implementation:

1) Unlimited size

2) Natural, It is like LL implementation

3) Wastage of size is not there

Disadvantage of Linked list implementation of queue over array implementation:

1) For each element we have a Node → val , *Next

Introduction to Deque → DLL se implement

d

doubly ended queue

O(1) T.C me hoti hai

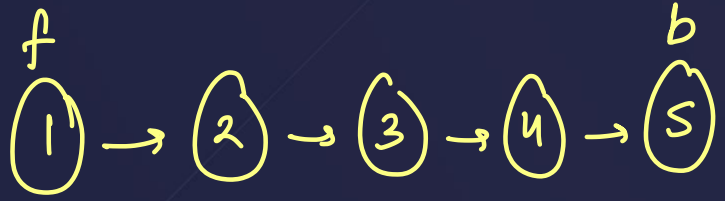
Design a Deque

push Front() Back()

Implement addFront(), addRear(), getFront(), getRear(), deleteFront(), deleteRear(), size()

↑ pop ↓

Ek doubly LL



SLL

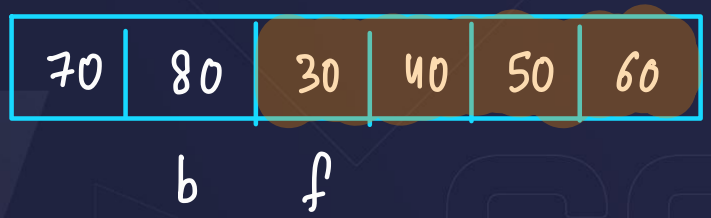
push front	:	O(1)
push back	:	O(1)
pop front	:	O(1)
pop back	:	O(n)

DLL

push front	:	O(1)
push back	:	O(1)
pop front	:	O(1)
pop back	:	O(1)

Introduction to Circular queue : We fully utilize the array

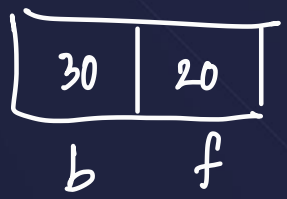
Array Implementation → back → array ke end
↓
jabki aage khali thi



30 40 50 60 70 80
f b

Ques: Design Circular Queue [Leetcode - 622]

↳ I will be given size (capacity) using vector/array



K = 2
C = 2
S = 0 1 2 1 2

push(10)
push(20)
pop()
push(30)
pop()

20 30

Implementing enQueue()

↓

push back

Implementing deQueue()

pop front

Implementing Front()

Implementing Rear()

Implementing isEmpty()

Implementing isFull()

THANK YOU!

↓

Next Lecture → Questions

Stack se Queue

Queue se Stack