Welcome ☺

Agenda: Queues
Implemation
DQ
1-2

Queue

FIFO → First In First Out

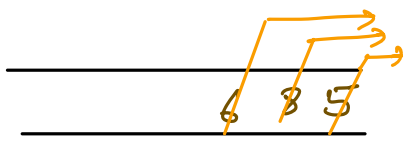
Operations

1) Enqueue (x) → Add x from the rear end.
2) Dequeue () → Remove element from front end.
3) isEmpty () → To check if queue is empty or not
4) front()/rear() → To get the element present at front / rear end

T.C
O(1)

Implement Queue using dynamic arrays

```
enqueue (5)
   "    (8)
   "    (6)
dequeue ()
   "    ()
   "    ()
```



r = ~~1~~ 0 1 2

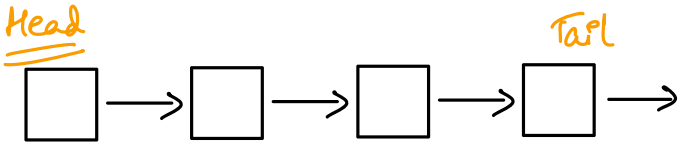
f = 0

```
void enqueue(x){
    r++
    A[r] = x
}
```

```
int dequeue(){
    if(isEmpty())
        return -1
    f++
    return A[f-1]
}
```

```
bool isEmpty(){
    return f>r
}
```

Q Implement Queue using L.L

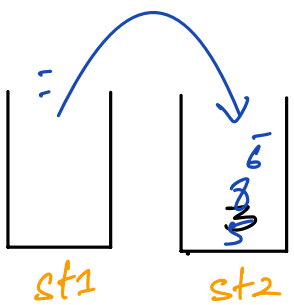


1) Insert at Head → O(1)
2) Delete at Head → O(1) ✓

1) Insert at Tail → O(1) ✓
2) Delete at Tail → O(N)

Q Implement Queue using Stacks..

```
enqueue(5)
   "    (8)
   "    (6)
dequeue()
   "    ()
   "    ()
enqueue(7)
   "    (3)
dequeue()
dequeue()
dequeue()
```



Insertion → O(1)

Deletion → ~~O(N)~~ → O(1)

N Insertions. ⇒ O(N)

1st dequeue. → moved all st1 → st2 O(N)

N Insertions O(1)

N-1 dequeues.

↳ O(N) for all remaining N-1 dequeues.

```
void enqueue(x) {
    st1.push(x)
}
```

```
bool isEmpty() {
    return st1.empty()
      && st2.empty()
}
```

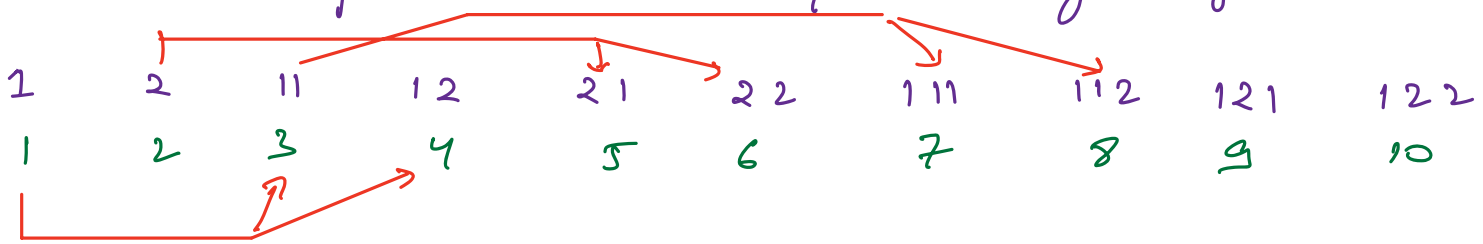
```
int dequeue() {
    if(isEmpty())
        return -1
    if(st2.empty()) {
        move()
    }
    return st2.pop()
}
```

```
void move() {
    while(!st1.empty()){
        st2.push(st1.pop())
    }
}
```

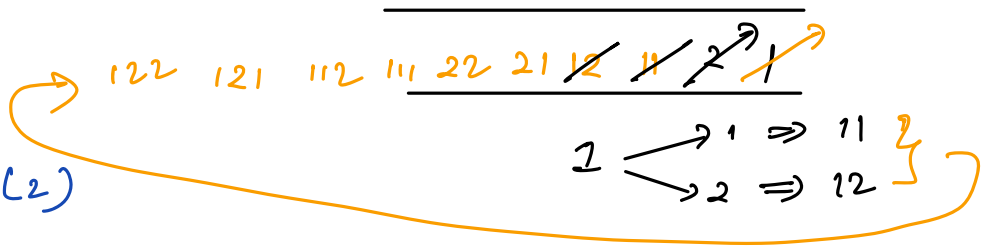
T.C ⇒ O(N)

T.C ⇒ O(1)

Q Find N^{th} perfect number i.e formed by digits 1 & 2




```
int solve(int N)
{
    if(N<=2) return N
    // Queue q
    q.enqueue(1)   q.enqueue(2)
    i=3
    while(i <= N)
    {
        x = q.dequeue()
        a = x*10+1
        b = x*10+2
        if(i==N) return a
        if(i+1==N) return b
        q.enqueue(a)   q.enqueue(b)
        i = i+2
    }
```

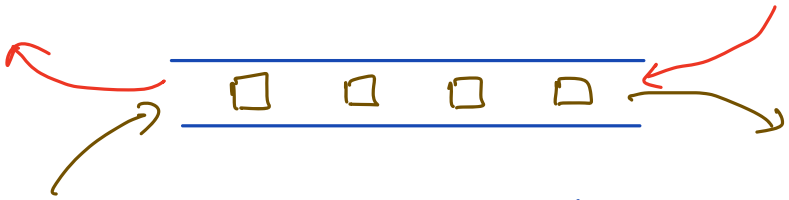


i → 3 5 7 9 11
x → 1 2 11 12
a → 11 21 111 121
b → 12 22 112 122

T.C → O(N)
S.C → O(N)

H.W ⇒ bit manip

Doubly Ended Queue

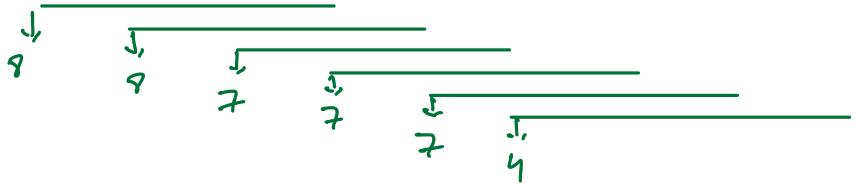


D.S to use to implement ⇒ DLL → doubly L.L

Q Given an integer array A, ∀ window of size K, find max. element.

eg: [1 8 5 6 7 4 2 0 3]

K=4

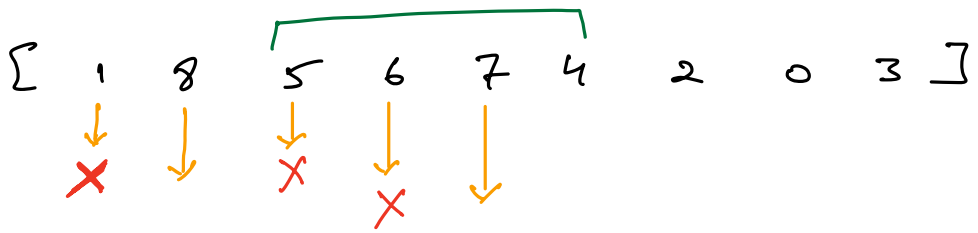


1) Brute force → ∀ subarray of size K, find max.

T.C ⇒ O(N*K) SC ⇒ O(1)

[1 8 5 6 7 4 2 0 3]

K=4



✓ stack + queue ⇒ Doubly Ended Queue

Code

```
if ( f == (i-K) ) q.dequeueFront()
while ( !q.empty() && A[r] < A[i] ) q.dequeueRear()
q.enqueue( i )
print ( A[f] )
```

T.C → O(N)

S.C → O(K)

1	8	5	6	7	4	2	0	3
0	1	2	3	4	5	6	7	8

Ans ⇒ 8 8 7