Queues

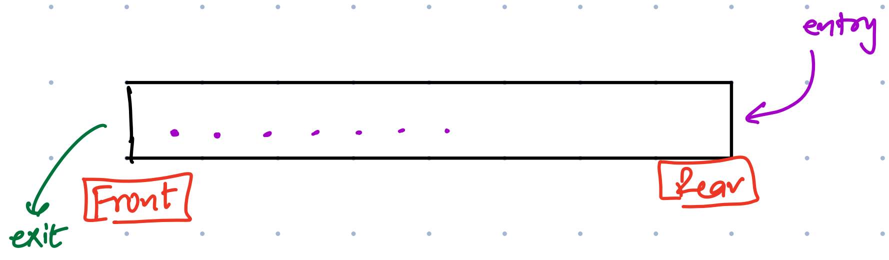
Mar 20, 2024

AGENDA

- what/why Queues
- Implementation of Queue
- Problems on Queue
    - Queue using stacks
    - Perfect no.
    - Sliding window maximum (** V.I. interview que)

Queue.

A linear data structure which supports operations in FIFO order.



e.g.

Queues in front of cinema hall, railway ticket counter.

Printers

Call centers

Operations supported by Queue

1. Enqueue (x) → Insert x from rear end of the queue.

2. Dequeue () → Remove from front end

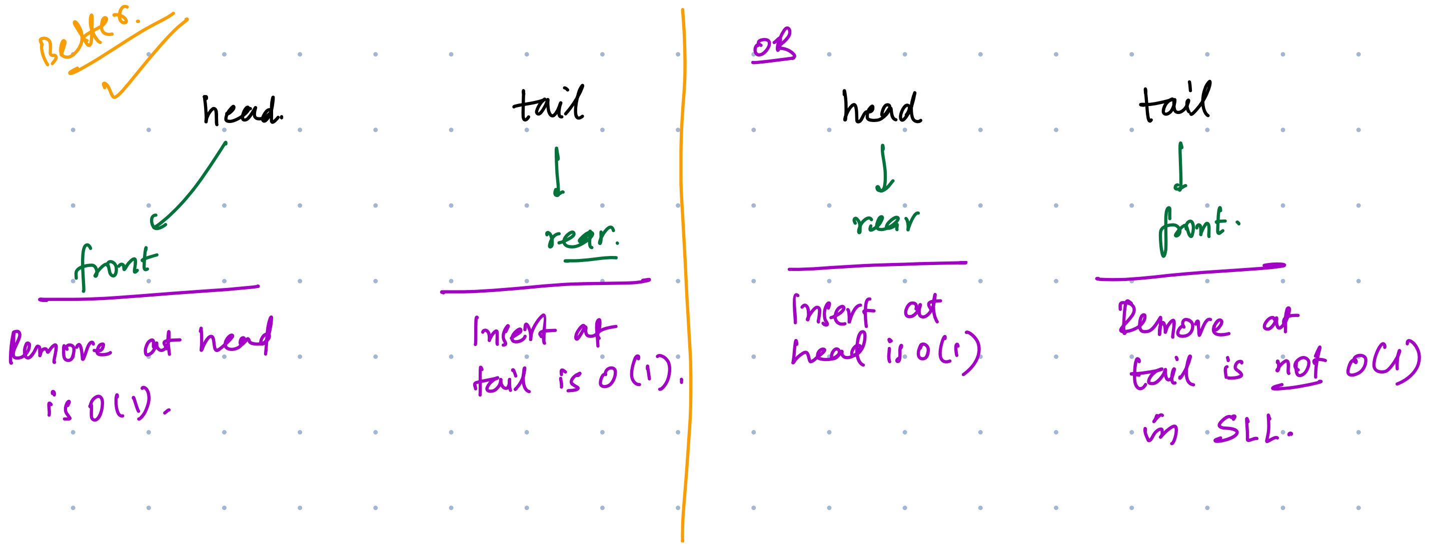
3. Peek or front() → Return the front-most elem of queue.

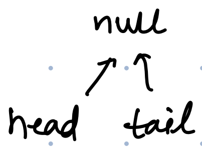
4. size .5. isEmpty()

Implementation of Queue

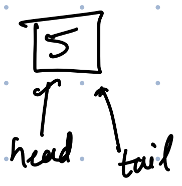
(Using LL)

Insert at rear
remove from front.





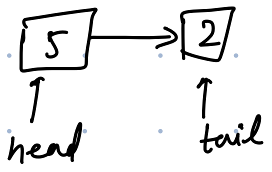
enqueue(5)



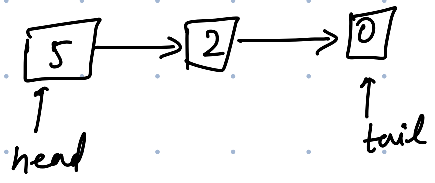
```
enqueue(int x)
{
    node = new Node(x);
    if (tail==null)  ← // Queue is empty right now.
    {
        head = node
        tail = node
        return
    }
```

enqueue(2);

```
        else
        {
              tail.next = node
              tail = tail.next

        }
}
```



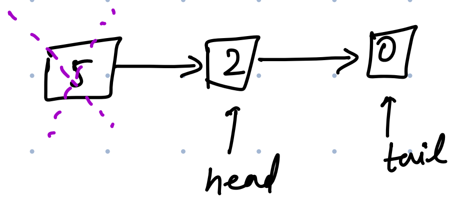
enqueue (0);



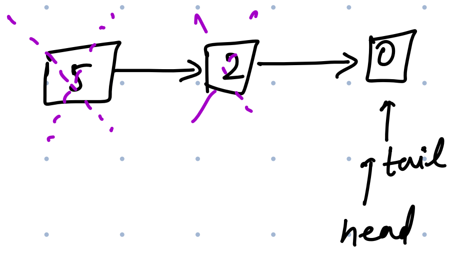
O(1)

```
deque ( )
{
        if ( head == null ) {

              //Throw error
              // Queue is empty

        }
        head = head.next
        if ( head == null )       //Resetting your LL.
              tail = null

}
```

dequeue()

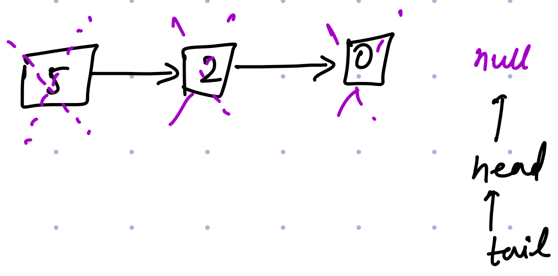


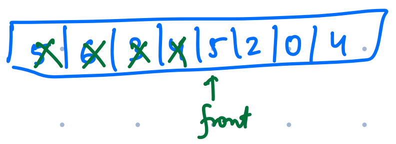
dequeue()



```
Int  front( )
{
        if ( head == null )
              // throw error.


        return    head.val

}
```

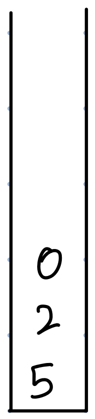
dequeue()





(Implement using array → H.W ☺)

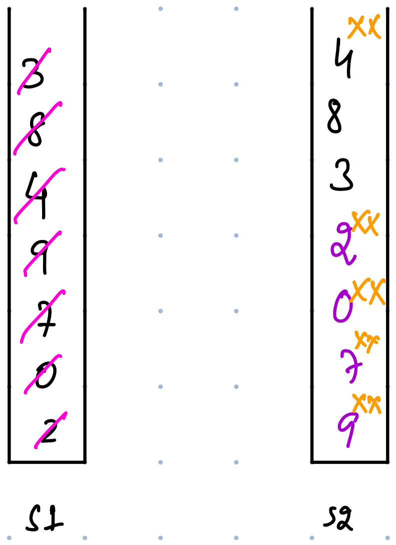
Q. Implement Queue using Stack(s).

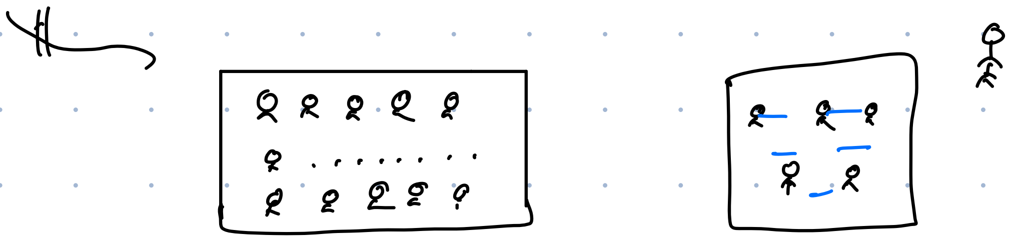


enqueue(5)
enqueue(2)
enqueue(0)
dequeue → 5

Use 2 stacks.

eq(2)
eq(0)
eq(7)
eq(9)
dq
dq
dq
eq(4)
eq(8)
dq
eq(3)
dq()





Approach:

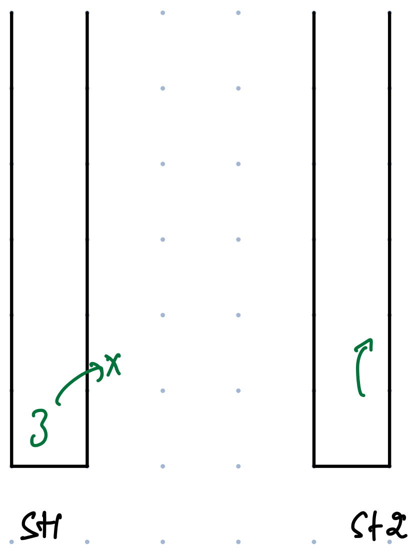
1. Enqueue in stack 1
2. Dequeue from stack 2.

```
class Queue
{
        stack<int> st1 ;
        stack<int> st2 ;

        void enqueue (int x)    ← O(1) ✓
        {
              st1.push (x)
        }

        void dequeue ()
        {
              if( st2.isEmpty () )
              {
```

```
            while ( ! st1.isEmpty () )
            {
                    int  x = st1.peek()
                     st1.pop()
                     st2.push (x)
             }
     }
     if( st2. is Empty())
                // Throw error
     else
           st2.pop()
}

 int  front ( )
 {
        if( st2.isEmpty ()
        {

             while ( ! st1.isEmpty ()
             {                                        <—Same as above.
                    int  x = st1.peek()
                     st1.pop()
                     st2.push (x)
             }
        }
        if( st2. is Empty())
                // Throw error
        else
             st2.peek()
}
```

T.C. Analysis

eq(3)
dq → 2 ops [shift 1, pop 1]
eq(2)
eq(6)
dq → 3 ops [shift 2, pop 1]
dq → 1 op [pop 1]
eq(2)
eq(6)
eq(9)
eq(6)
eq(5)
eq(7)
dq → 7 ops [shift 6, pop 1]
dq → 1 op
dq → 1 op
dq → 1 op
dq → 1 op
dq → 1 op

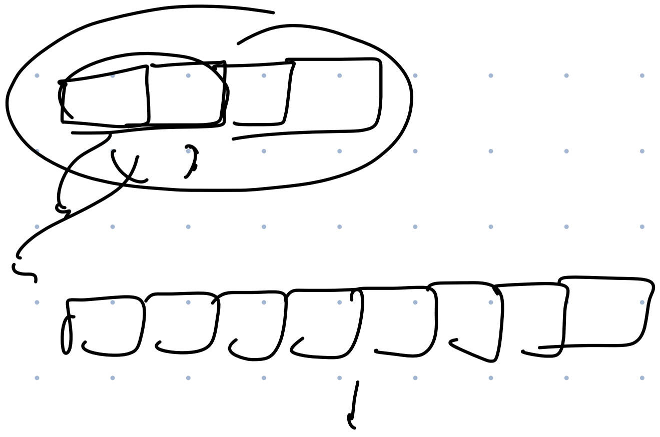


On average, every dequeue has 2 operations.

T.C : O(1)

Amortized T.C

Dynamic Array



$\boxed{\text{Break till 8:15 AM}}$

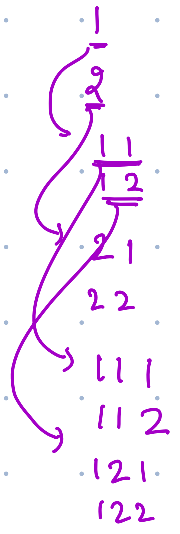
Q. <u>N</u>

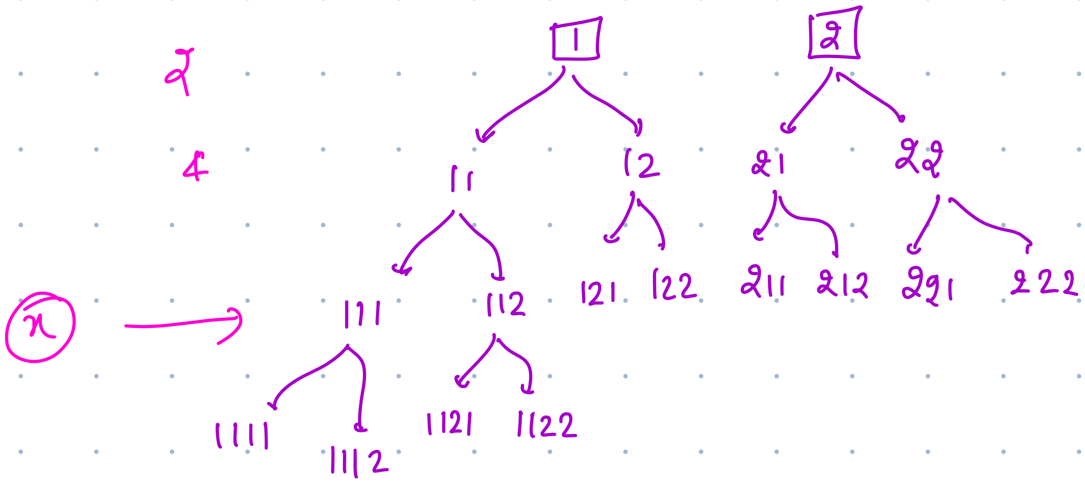
Nth perfect no.

Return the Nth no. formed using only digits 1 and 2.

1, 2, 11, 12, 21, 22, 111, 112, 121, 122, 211, 212, 221, 222, 1111....

N=5th no. ✓





1, 2

3, 4, 5, 6

7, 8, 9, 10, 11, 12, 13, 14.

2^x

2^{x+1}

~~1~~, ~~2~~, ~~11~~, 12, 21, 22, 111, 112

Code.

```
Queue<int> q;
q.enqueue(1);
q.enqueue(2);
cnt = 0
while( true )
{
        int x = q.dequeue()
        cnt ++;
        if(cnt == N)
                return x

        q.enqueue( x * 10 + 1)
        q.enqueue ( x * 10 + 2)
}
```

$\boxed{N=10}$

~~1~~, ~~2~~, ~~11~~, ~~12~~, 21, 22, 111, 112
121, 122

$x = 1$	cnt = 1
$x = 2$	cnt = 2
$x = 11$	cnt = 3
$x = 12$	cnt = 4
$x = 21$	cnt = 5
$x = 22$	cnt = 6
$x = 111$	cnt = 7
$x = 112$	cnt = 8
$x = 121$	cnt = 9
$x = 122$	$\boxed{\text{cnt} = 10}$

T.C $\rightarrow$ $O(N)$

S.C $\rightarrow$ $O(N)$

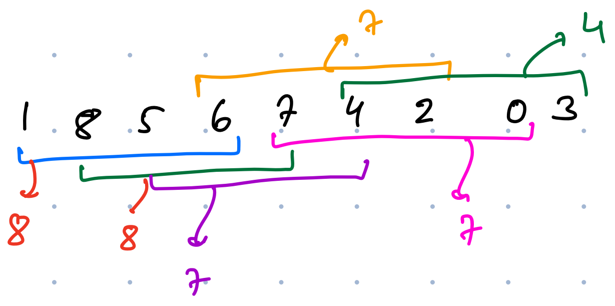
level = $\log_2 N$

no. of pending children in queue $= 2^{\text{level}}$

$= N$

Q. Sliding window maximum

Given an integer array, find the max element in every window of size k.



K=4

B.F.

go to every window $\xrightarrow{}$ $n-k+1$

and find the max in each window. $\rightarrow O(k)$

T.C. $(n-k+1) * k$

$= O(N^2)$

$$\begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 8 & 5 & 6 & 7 & 4 & 2 & 0 & 3 \end{array}$$

$\underbrace{1 \quad 8 \quad 5 \quad 6}$

1st window

$\boxed{s=0, e=3}$

Box

$\boxed{\underline{8} \quad 6}$

Ans. ✓

Mystery box.

~~1~~ 8 ~~5~~ ~~6~~ 7

2nd window

s=1, e=4

out = 1

inc = 7

$\boxed{\underline{8} \quad 7}$

Ans ✓

3rd window

s=2, e=5

$$\begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 8 & 5 & 6 & 7 & 4 & 2 & 0 & 3 \end{array}$$

out = 8

inc = 4

$\boxed{\cancel{8} \quad \underline{7} \quad 4}$

ans ✓

4th window

s=3, e=6

$$\begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 8 & 5 & 6 & 7 & 4 & 2 & 0 & 3 \end{array}$$

out = 5

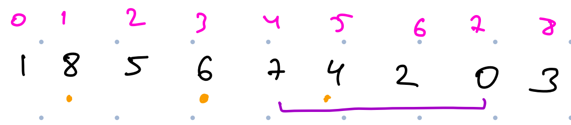
inc = 2

$\boxed{\underline{7} \quad 4 \quad 2}$

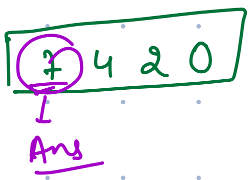
ans.

5th window

s=4, e=7

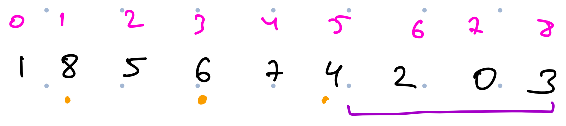


out = 6
inc = 0

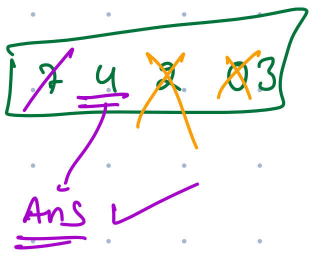


6th window

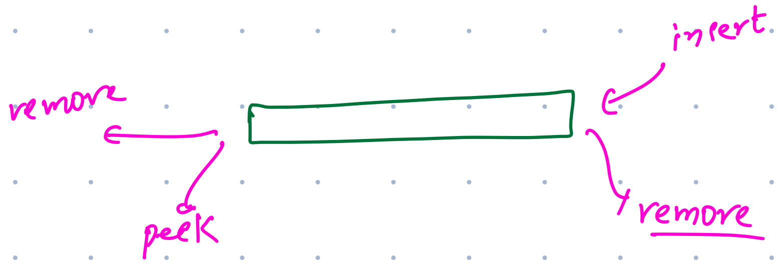
s=5, e=8.



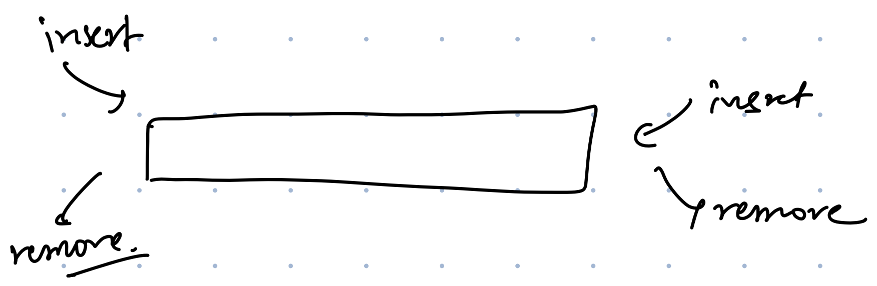
out = 7
inc = 3

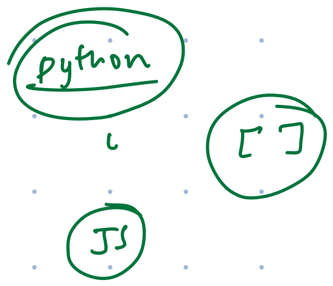


what is this mystery Box?

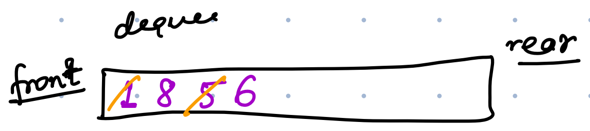


Deque → Double-ended-queue.





Code.



```
Deque<int> dq;
```

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

```
for(int i=0 ;i< K ;i++)
{
      while ( !dq.isEmpty()  &&  arr[i] >  dq.peekRear())
      {
              dq.popRear()
      }
      dq.insertRear(arr[i])
}
print( dq.peekFront())
```

→ Get the rear element.

s = 1
e = K

```
while ( e < n)
{
        inc = arr[e]
        Out = arr[s-1]
        // Handle outgoing.
        if (out == dq.peekFront())
        {
              dq.popFront()
        }

        // Handle incoming.

        while ( !dq.isEmpty()  &&  inc > dq.peekRear())
        {
              dq.popRear()
        }
        dq.insertRear(inc)

        print (dq.peekFront())

        s++
        e++
}
```

3 15 6 15 $\underbrace{12\ 4\ 2\ 10}$ 9 3

~~3~~ ~~15~~ ~~6~~ 15 12 4

T.C → O(N)

S.C → O(K)