Queue → First in First out

Operations :

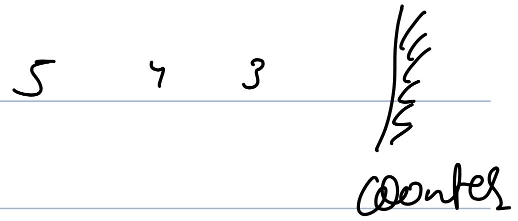
(insert) enqueue (x)

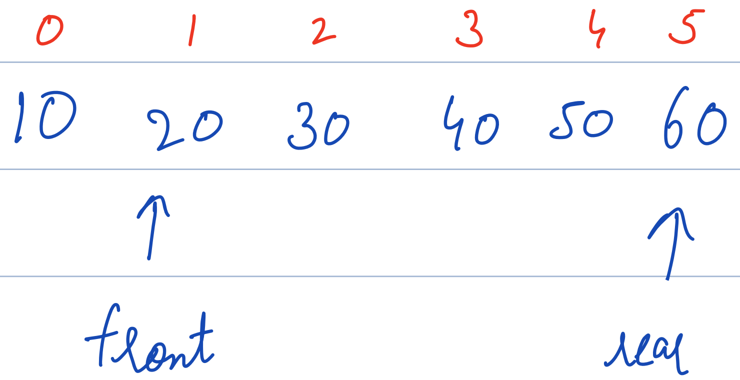
(remove) deque (x)

front ()

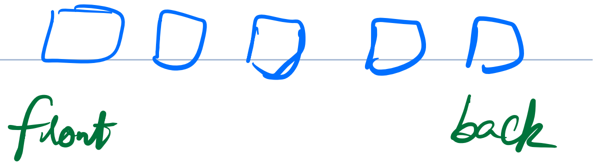
isEmpty ()

rear () back





Queue using array



```
class Queue {
 int size
 int arr[size]
 int front =                     starting idx of queue
 int rear = -1                   ending idx of queue

 enque(int x){
  if (rear == size-1) sout("Queue Full")
  else
     rear++          arr[rear] = x

 }
                                  size of the queue
     deque ( ){
 if ( rear-front+1   == 0 )
        sout ("Empty Queue")
 else
   front++

}
```

Q1 Given K. Series is made up of 1,2 . Return K^{th} num.

	1	2	3	4	5	6	7	8
Series ⇒	1	2	11	12	21	22	111	112

k=7 ans = 111

k=4 ans = 12

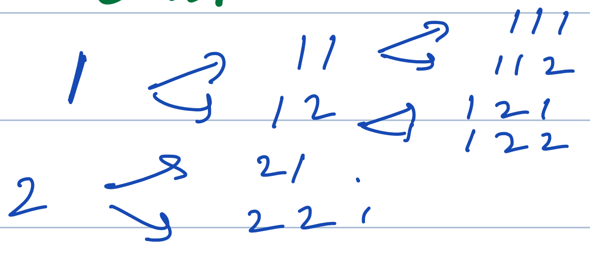
Brute Iterate from 1. check if num has only 1,2 . If yes, count++. when count = k , end.

Idea

First 2 no.s are 1, 2

From 1, what all num we can get directly ⇒

From 2, ⇒



21 22 111 112 121 122

From x ⇒ 1) $10^*x + 1$

Code 2) $10^*x + 2$

```
queue <int> q
q.enque (1)
q.enque (2)
```

~~1~~ ~~2~~ 3 4

21 22 111 112 121 122

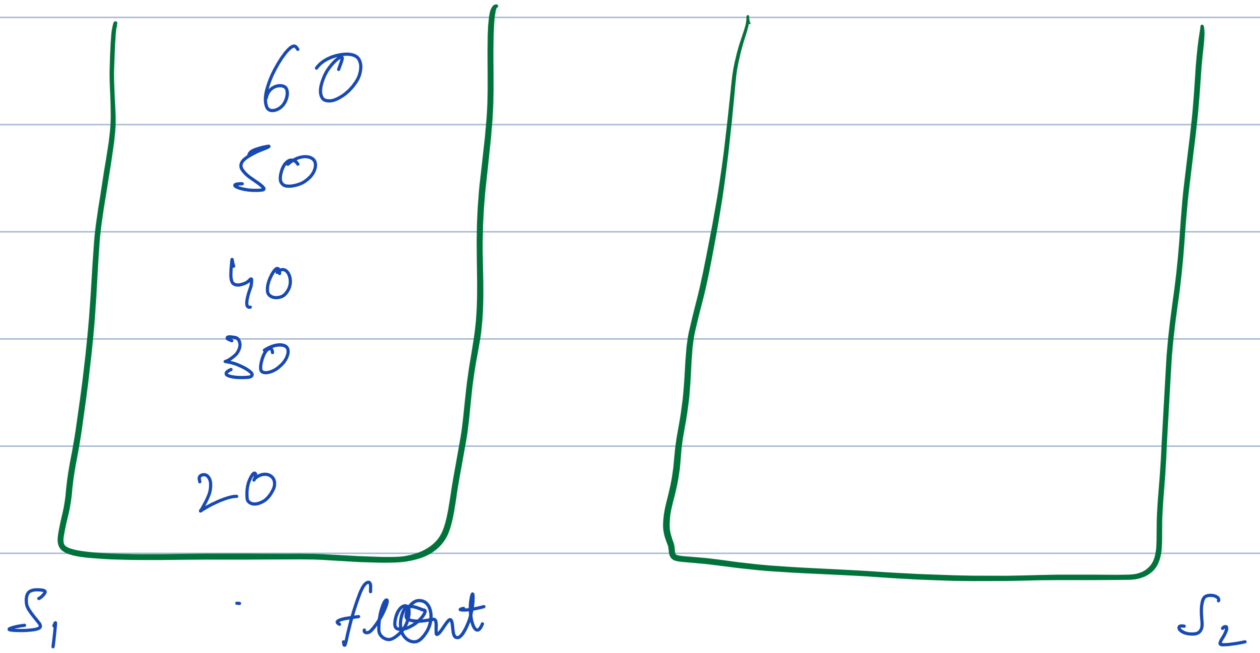
```
cnt = 1
while ( cnt != k ) {
    int x = q.front ()
    q.deque ()
    cnt ++
    q.enque ( 10x + 1 )
    q.enque ( 10x + 2 )
}
return q.front ()
```

TC: } O(k)
SC: }

MMT

Q3 Implement Queues using stacks (only)

Idea: Use 2 stacks

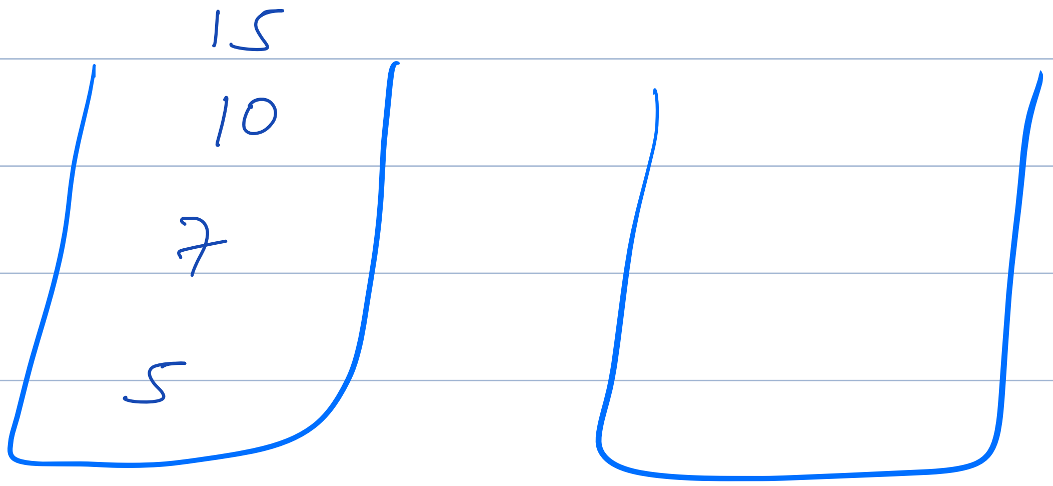


enqueue → Just put on top of S_1

deque → Need to remove bottom elem.

1) Remove all elem & put in S_2
2) S_2.pop()
3) Put all contents of S_2 back in S_1

dry run.



Code

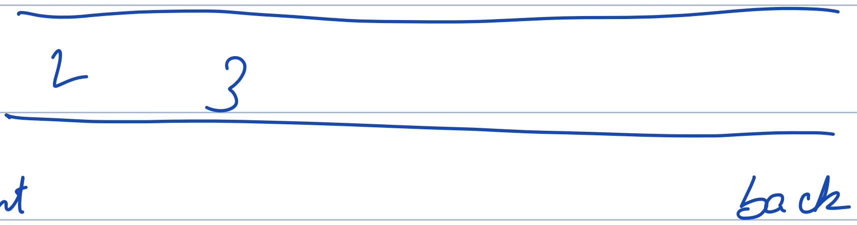
```
void enque (x) {
 s1. push (x)
}


void deque () {
      if ( s1. empty () )
            print ("Error")
    while (! s1. empty ())
          s2. push (s1. top ())
          s1. pop ()
}
```

```
S2. pop()
while(!S2.empty()){
    S1.push(S2.top())
    S2.pop()
}
}
```

TC enque → O(1)

deque → O(n)

Doubly ended queue Deque



Sliding window maximum

Deque

CRED

VVV Imp

Adobe GS Amazon Microsoft

Array of size N. Find max elem of every subarray of size K.

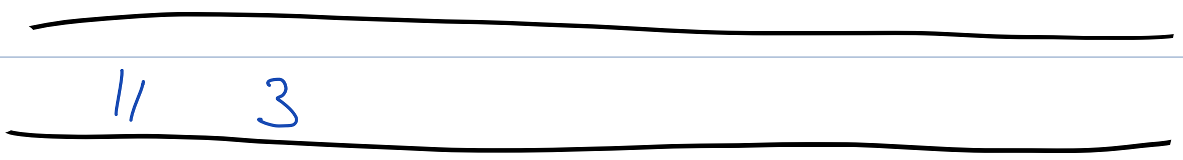
10 1 4 9 7 6 5 11 8

k=3 ans ⇒ {10 9 9 9 7 11 11}

Brute: Check all subarrays of size K & find max. TC: $O(n^2)$

Idea k=3

10 1 8 9 7 6 5 11 3



front

access last elem stack
access first elem queue

Deque (Doubly ended queue)

k=4

3 15 6 15 12 4 2 10 9 18

15 15 15 15 12 10 18

18

First prepare the deque for first window

new_elem

rear < new_elem

remove rear

repeat

rear ≥ new_elem

insert at rear

remove elem

cheek with front

if equal

remove from front

if not

do nothing

Max ⇒ front element

Code

```
Deque <int> dq
for (i=0; i<k; i++){
    while (!dq.empty() &&
                dq.rear() < arr[i])
          dq.pop_rear()
    dq.push_rear(arr[i])
}

ans. insert (dq.front())
int s=1          e= k
while (e<n){
    while (!dq.empty() &&
                dq.rear() < arr[e])
          dq.pop_rear()
    dq.push_rear(arr[e])

    if (dq.front == arr[s-1])
         dq.pop_front()
    ans. insert (dq.front())
```

TC: O(n)
SC: O(n)

Stt ett

y

1 insert

1 elem

1 delete

} 2

1 → 2 op

N → 2N op

<done>

Idea k=3

10 1 8 9 7 6 5 11 3

8

front