Queues

Do something today that your future self will thank you for.

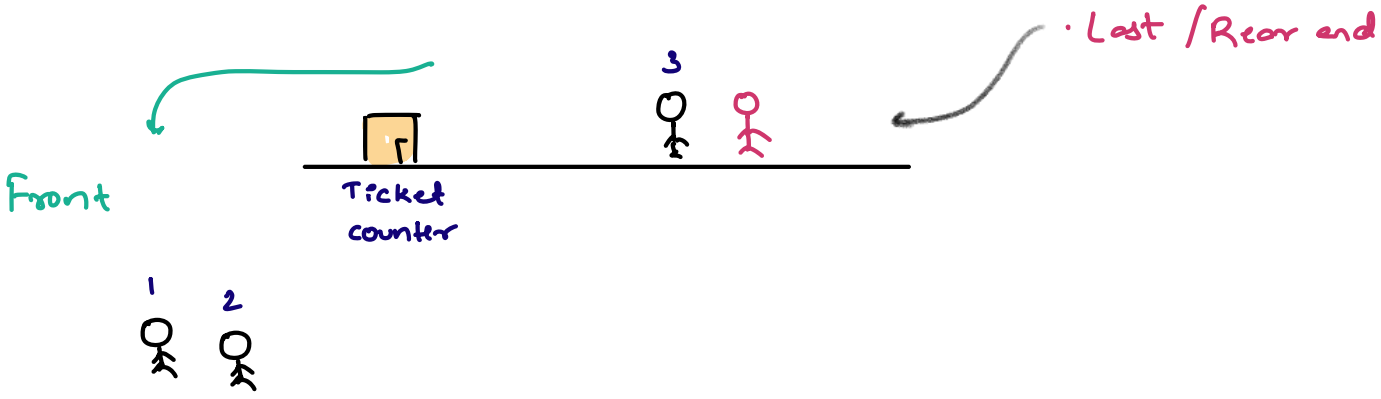
Good Evening

Today's content

01. Introduction to Queues
02. Implementation of Queues
    - → using Arrays
    - → using LinkedList
    - → using stacks
03. Generate K^{+} no. using 1 & 2
04. Max of every window (Deque)

Queue → Linear data structure

↳ First In First Out



Eg:- Water stream in pipe

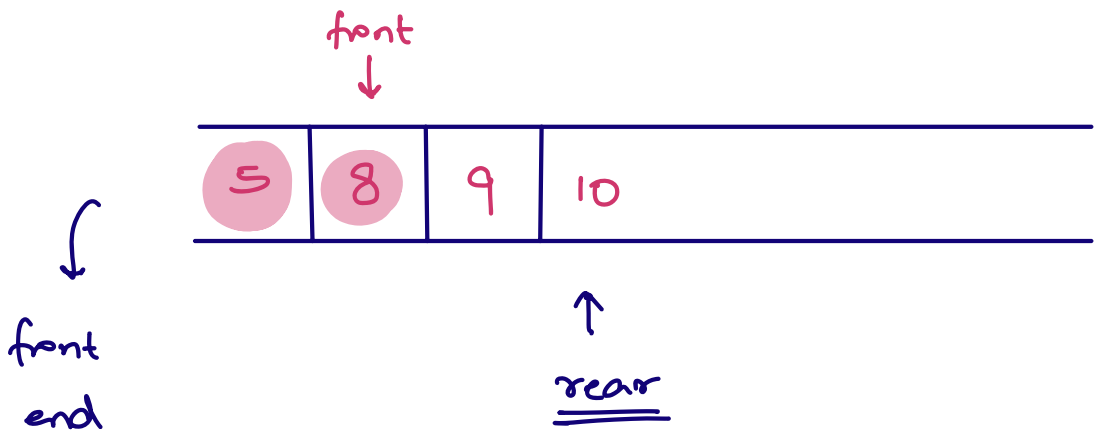
- → Printer
- → Message queue
- → Data transfer
- → Call center queue

Operations by Queue

01. Enqueue(x) → insert the data at rear end
02. Deque( ) → remove data from front
03. front()/peek() → data present at the front
04. size( ) → count of data present in queue

Enq(5) Enq(8) Enq(9) Enq(10) Deq() Deq() front()

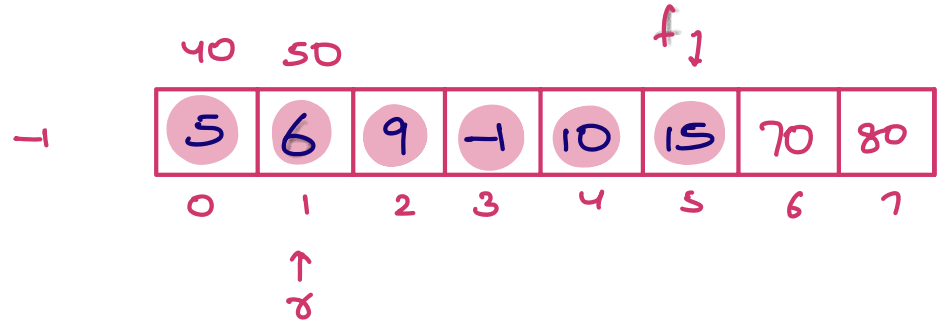
9



* Implementation of Queues using Array

f = -1 ⇒ idx of last ele which was just removed

r = -1 ⇒ idx of last ele which was just inserted



Enq(5) Enq(6) Enq(9) Enq(-1) Deq() Deq() Deq()

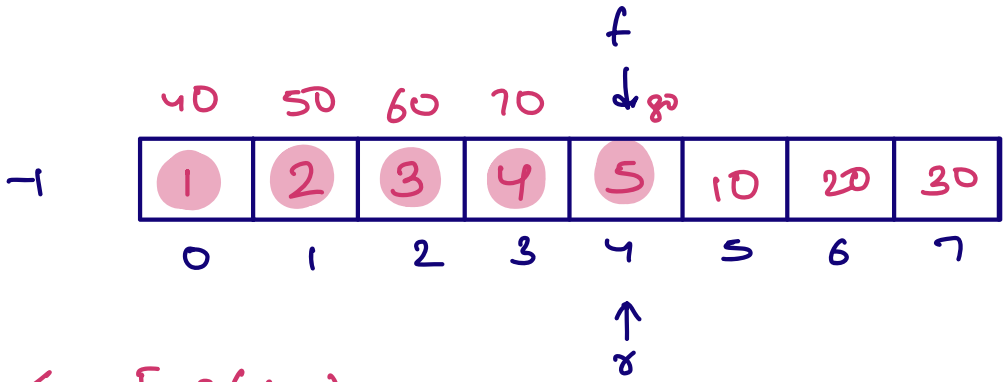
Enq(10) Enq(15) front() Deq() Deq() Deq()

n = 8

```
void enqueue(x){
    if (size==n){Queue is full}
    r=(r+1)%n
    A[r]=x
    size++;
}
```

```
void deque()
    if (size==0){Queue is empty}
    f=(f+1)%n
    size--;
}
```

```
front()
    if (size==0){Queue is empty}
    return A[(f+1)%n];
}
```



n=8

Enq(1) ✓ Enq(10) ✓

Enq(2) ✓ Enq(20) ✓

Enq(3) ✓ Enq(30) ✓

Enq(4) ✓ Enq(40) ✓

Enq(5) ✓ Enq(50) ✓

Deq() ✓ Enq(60) ✓

Deq() ✓ Enq(70) ✓

Deq() ✓ Enq(80) ✓

Deq() ✓

Deq() ✓ Enq(100)

if (f==r) → queue is empty ?

↳ queue is full ?

* Implement Queue using LinkedList

↳ First In First Out

Enque(10) ✓

Enque(20) ✓

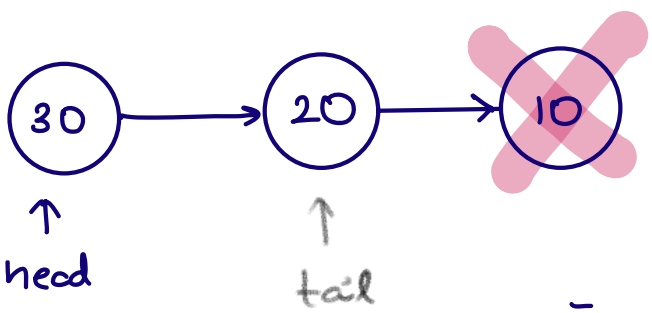
Enque(30) ✓

Deq() ✓

Deq() ✓

front() ✓

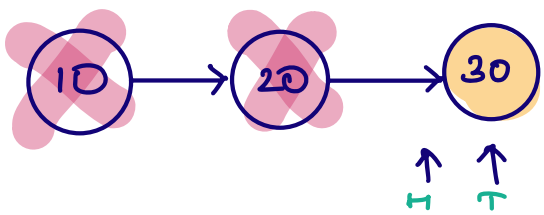
Insert at head & remove at tail → O(n)



#todo

Insert at tail & remove at head → O(1)

→ More optimal



03. Implement queue using Stacks

Last In First Out

Stack

Queue operations

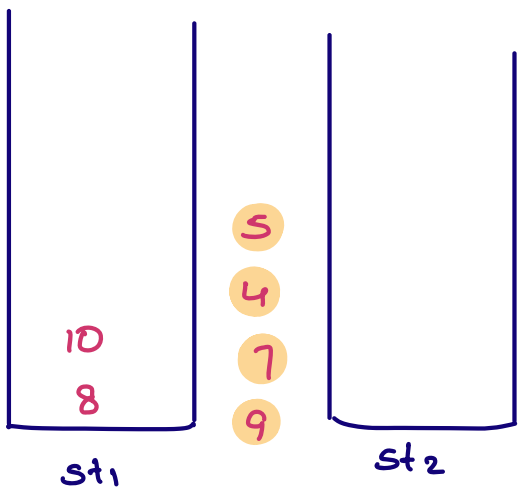
(a) Enqueue()
(b) Dequeue()
(c) Peek()/front()

} Every queue function should be implemented using stack function only

{ push() pop() peek() }

01 Enq(5) ✓
02 Enq(4) ✓
03 Enq(7) ✓
04. Enq(9) ✓
} O(1)

05. Deq() → O(n)
06 Enq(8) ✓
07 Enq(10) ✓
10. Deq() → O(1)
11 Deq() → O(1)
12 Deq() → O(1)



Enq(x) = st1.push(x) TC: O(1)

Deq() = if (st2.size() == 0) TC: O(1)

Transfer all elements from st1 to st2 & then st2.pop()

st2.pop();

S1.pop() S2.push

01 Enq(5) → O(1) ✓

02 Enq(4) → O(1) ✓

03 Enq(7) → O(1) ✓

04. Enq(9) → O(1) ✓

05. Deq() ✓

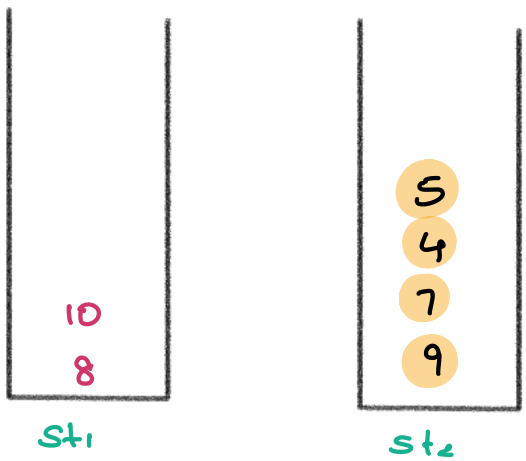
06 Enq(8) → O(1) ✓

07 Enq(10) → O(1) ✓

10. Deq() ✓

11 Deq() ✓

12 Deq() ✓



1st Deq() → Transfer all ele from S1 to S2 & S2.pop()

8ops + 1ops

2nd deq = 1 ops

3rd deq = 1 ops

4th deq = 1 ops

4 deq = 12 ops

1 deq = 12/4 = 3 ops ≈ O(1)

10:08 → 10:18 pm

01. Generate k^{th} perfect no in series using only 1 & 2 as digits. The series should be in increasing order

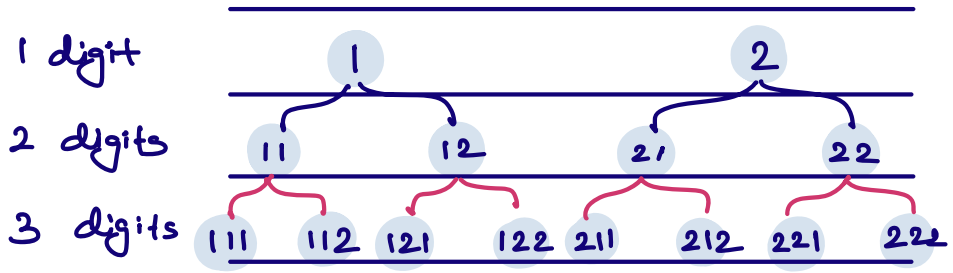
K = 5 → 1 2 11 12 21

K = 7 → 1 2 11 12 21 22 111

Bruteforce → start from 1 & keep going while keeping the track of count of no. that are only made up of 1 & 2 as digits.

Count == K ⟶ return no.

Idea 2 → data in levels / data in digits



K = 6

1 2 ~~11~~ ~~12~~ ~~21~~ ~~22~~ 111 112 121 122 211 212

~~CNT = 1 2 3 4 5~~ 6 → Ans = 22

```
String kthNum(int k)
{
    Queue<String> q = new ArrayDeque<>();

    q.add(1)
    q.add(2)

    for(i=1; i<=k-1; i++){
        String ele = q.deque();
        String p= ele+"1";
        q.enque(p);
        String r= ele+"2";
        q.enque(r);
    }

    return q.front();
}
```

TC: O(K)
SC: O(K)

* **Deque** : Double Ended queue
  ↳ Addition & removal can be done from both the ends

Functionality **TC: O(1)**

01. addfirst() ✓ addlast()

02 removeFirst() removeLast() ✓ } DLL

03. getFirst() getLast() }

Stack → FILO or LIFO

Queue → FIFO

Deque can act as stack & as queue.

* Given ar[N] & window of size K. Find max ele in every window of size K.

	0	1	2	3	4	5	6	7	8	
ar[] : {	10	1	9	3	7	6	5	11	8	} K=4

return = 10 9 9 7 11 11

Brute force → Generate all the subarrays of size K, iterate on subarray & get max

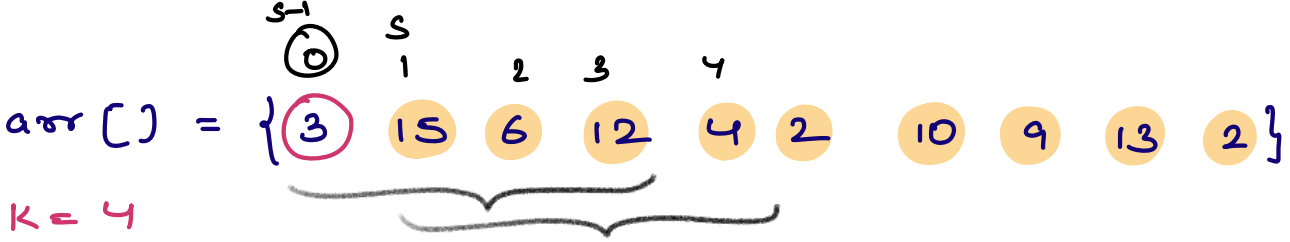
TC : (n-k+1) * k

TC : O(n²)
SC : O(1)

K = 1 TC : O(n)

K = n TC : O(n)

K = n/2 TC : O(n²)



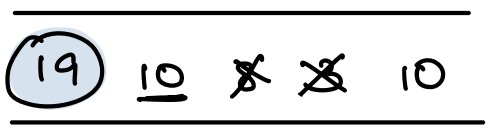
01. Remove from last
02 Front ele
03. Remove from front

Ans = 15 15 12 12 10 13 13

arr[] = { 10 1 9 3 7 6 5 11 8 } K=4

arr[] = { 19 3 10 8 10 10 7 6 } K=3

arr[] = { 19 10 8 3 10 7 6 5 } K=4



For duplicates

↓

check if we have to remove it or not.

19

```
void subarrmax ( int [] ar, int k)

    Deque <I> dq = new ArrayDeque <>();

    // Insert first window in dq

    for ( i=0 ; i<k ; i++){
    |
    |   while ( dq.size()>0  &&  ar[i] > dq.getlast())
    |   |
    |   |     dq.removelast();
    |   3
    |
    |   dq.addlast ( ar[i]);
    3

    print ( dq.getfirst());

// For all other windows

    s=1       e=k
    while ( e < n )
    |
    |   // remove ar[s-1]
    |   if ( ar[s-1] == dq.getfirst())
    |   |
    |   |     dq.removefirst();
    |   3
    |
    |   // add ar[e]
    |   while ( dq.size() >0 && ar[e] > dq.getlast()){
    |   |
    |   |     dq.removelast();
    |   3
    |   dq.addlast (ar[e]);
    |   print ( dq.getfirst());
    |
    |   s=s+1    e=e+1
    3
```

Contest → Realtempts