CS & IT ENGINEERING

Data Structure & Programming

Stack and Queues

Lec - 05

By- Pankaj Sharma sir

TOPICS TO BE COVERED

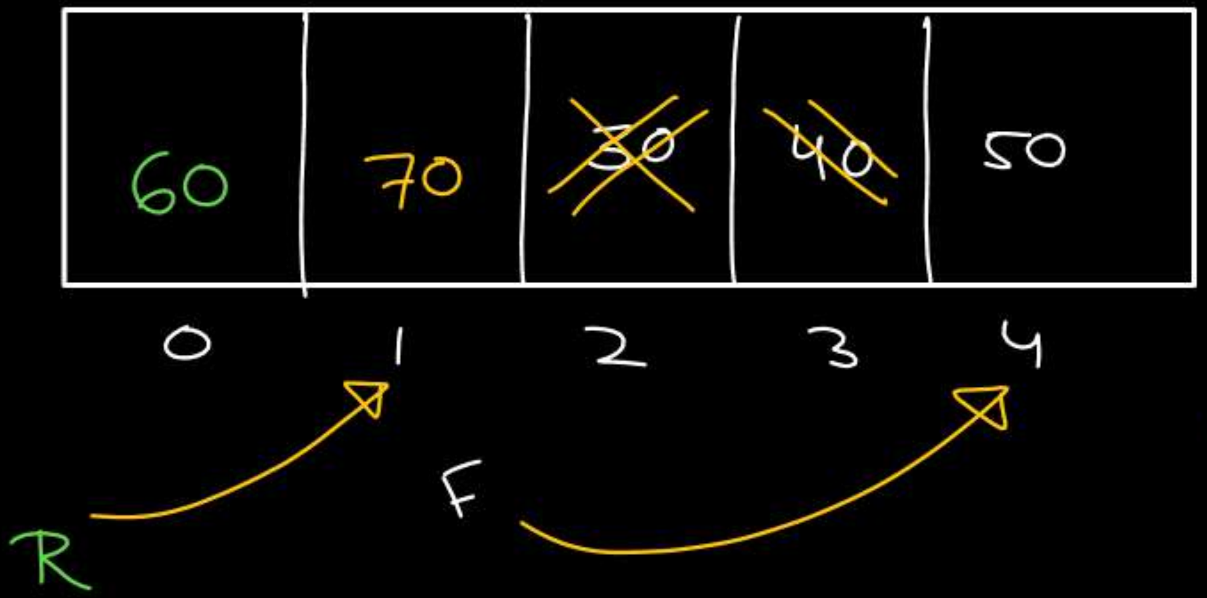
- Queues Part-2

~~30~~, 40, 50, 60, 70

Circular Queue

F = = (R+1) mod size

2 = = (1+1) mod 5 ✓



(i) F = 0 && R = size-1

(ii) F = = R+1

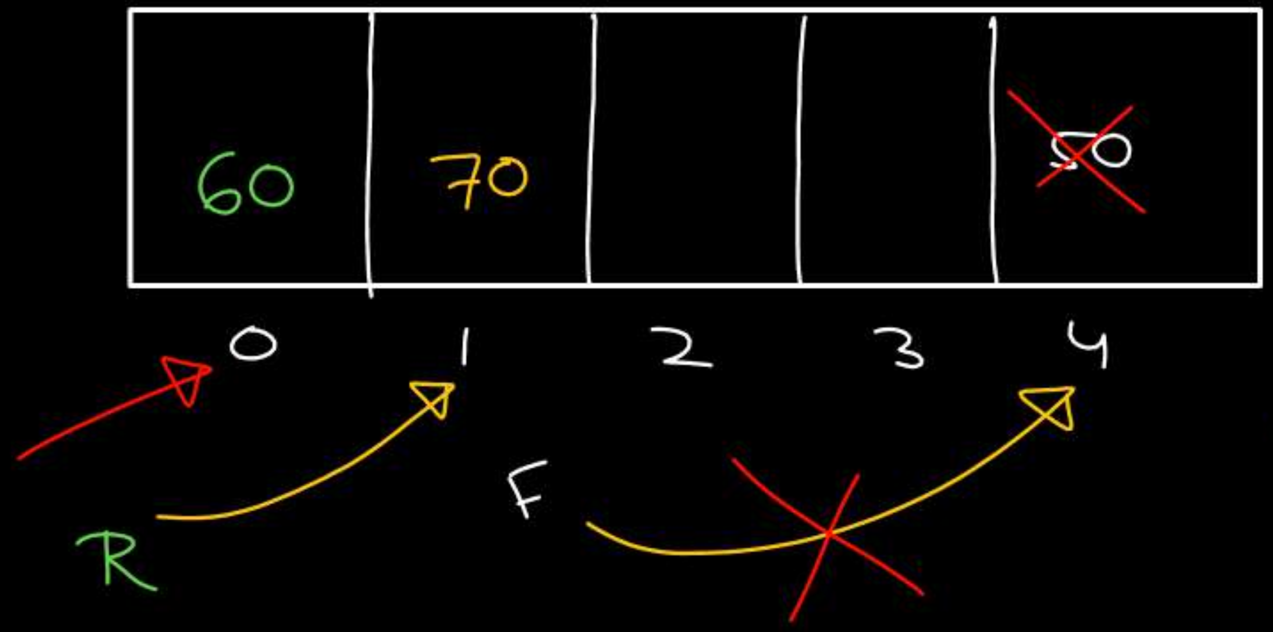
	F	R
	2	4
Insert(60)	2	60
Insert(70)	2	1
Delete( )	3	1
Delete( )	4	1

Circular Queue

~~30~~, ~~40~~, 50, 60, 70

$F == (R+1) \bmod size$

$2 == (1+1) \bmod 5$ ✓



(i) $F = 0$ && $R = size-1$

(ii) $F == R+1$

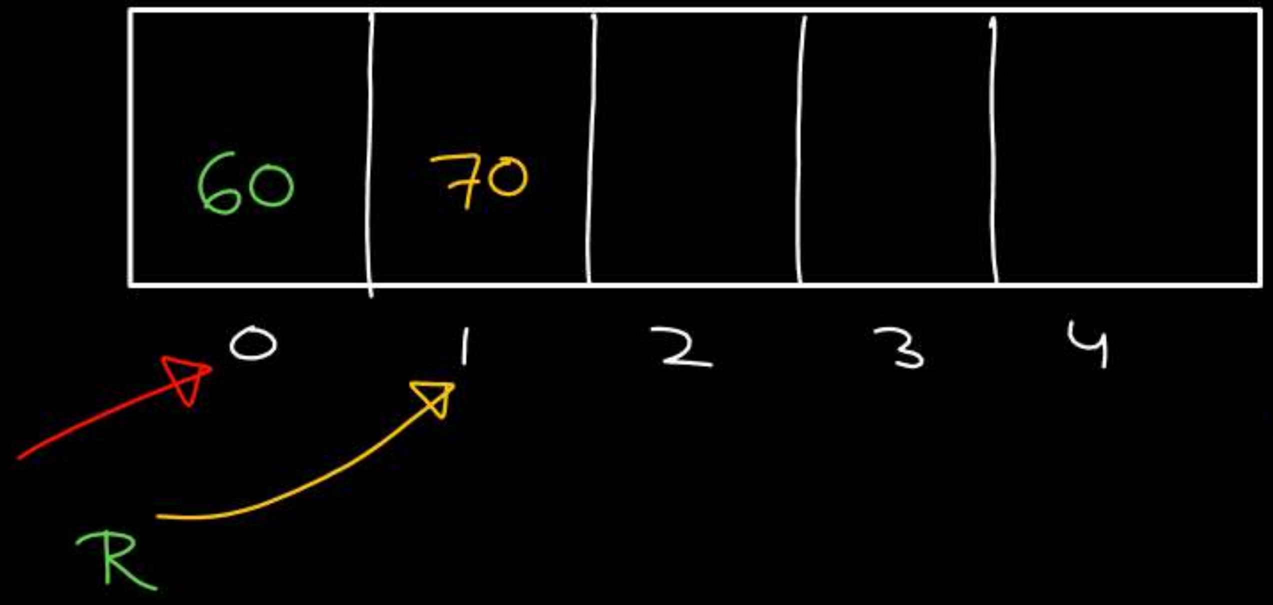
	F	R
	2	4
Insert(60)	2	60
Insert(70)	2	1
Delete( )	3	1
Delete( )	4	1
Delete( )	0	1

~~30~~, ~~40~~, ~~50~~, 60, 70

Circular Queue

$F == (R+1) \bmod size$

$2 == (1+1) \bmod 5$



$\left\{ \begin{array}{l} (i)\ F = 0\ \&\&\ R = size-1 \\ (ii)\ F == R+1 \end{array} \right\}$

	F	R
	2	4
Insert(60)	2	60
Insert(70)	2	1
Delete( )	3	1
Delete( )	4	1
Delete( )	0	1

```
void CQ_Enqueue(int x){
    if(Front==(Rear+1)%SIZE)
            return;

    else if(Rear==SIZE-1)
            Rear=0;

    else if( Front==-1 )
            Front=Rear=0;
    else
            Rear++;
        Queue[Rear]=x;
}
```

Insert 10, 20, 30, 40

Insert 50

10	20	30	40	50
0	1	2	3	4

F

R

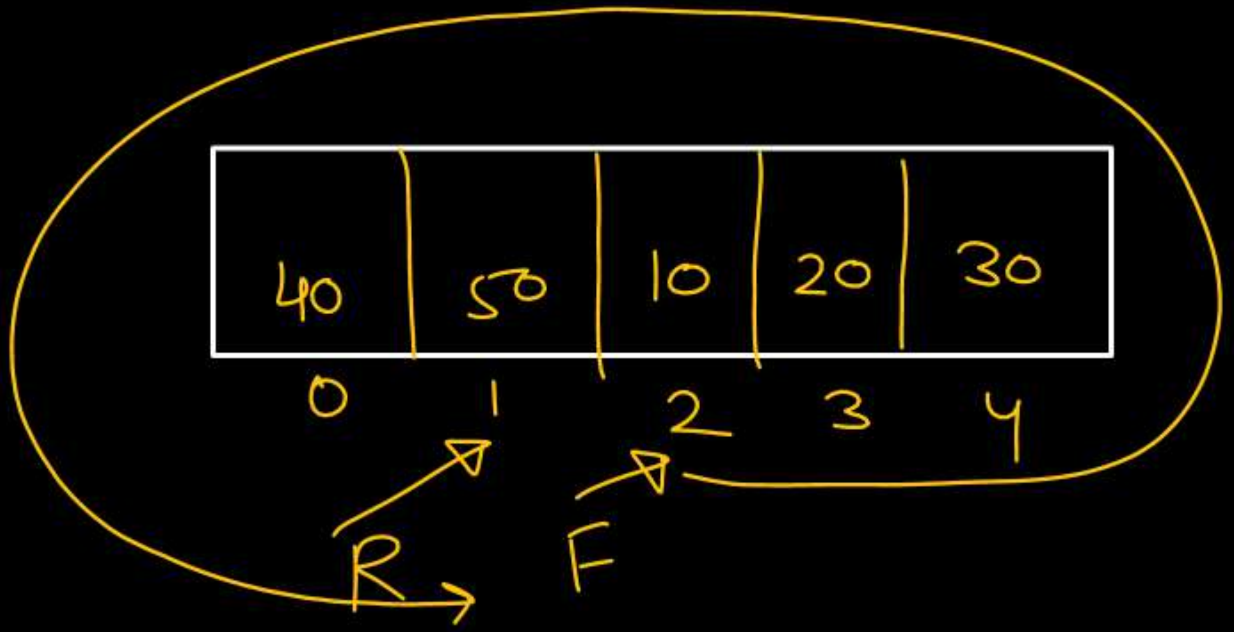
Queue $\Rightarrow$ Full

F $\Rightarrow$ 1st index

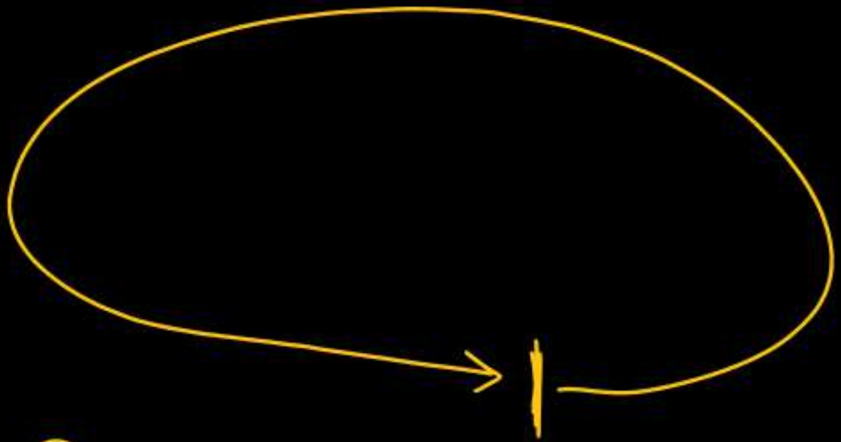
R $\Rightarrow$ last index

Insert 40

Insert 50



Full $\Longrightarrow$



F == Rear+1

int CQ_delete( )

1. Front = = -1

Underflow

return INT_MIN;

2. Front = = Rear // only 1 ele.

temp = Queue[Front];

Front = Rear = -1

return temp;

3. Front = = SIZE-1

temp = Queue[Front];

Front = 0;

return temp;

4. temp = Queue[Front];

Front++

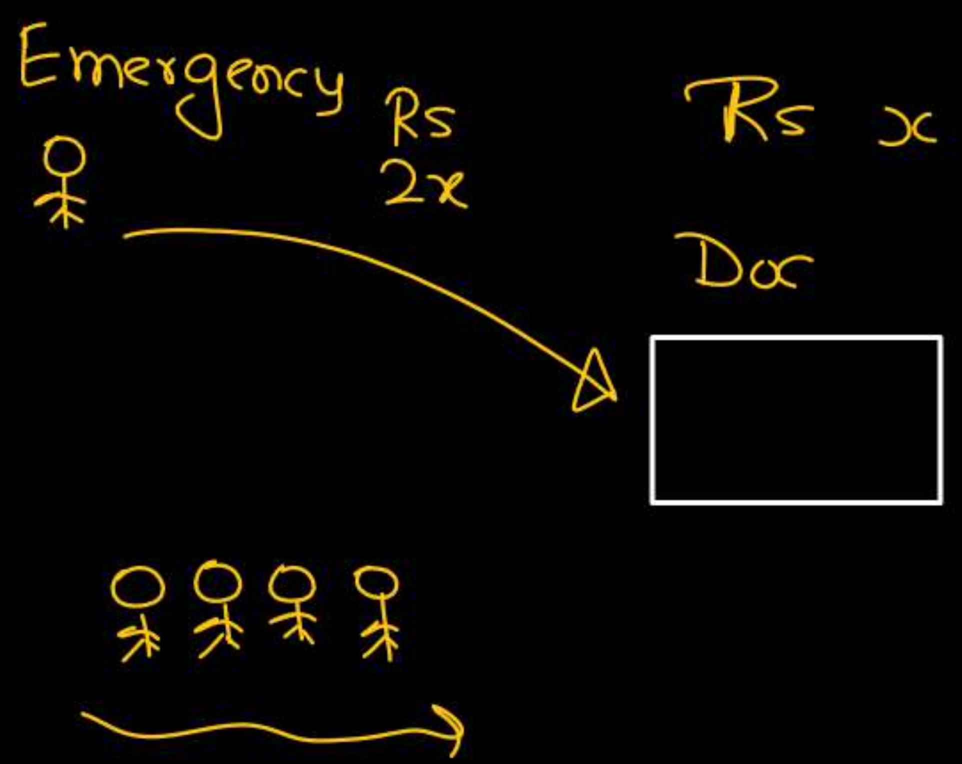
return temp;

O(1)

Priority Queue

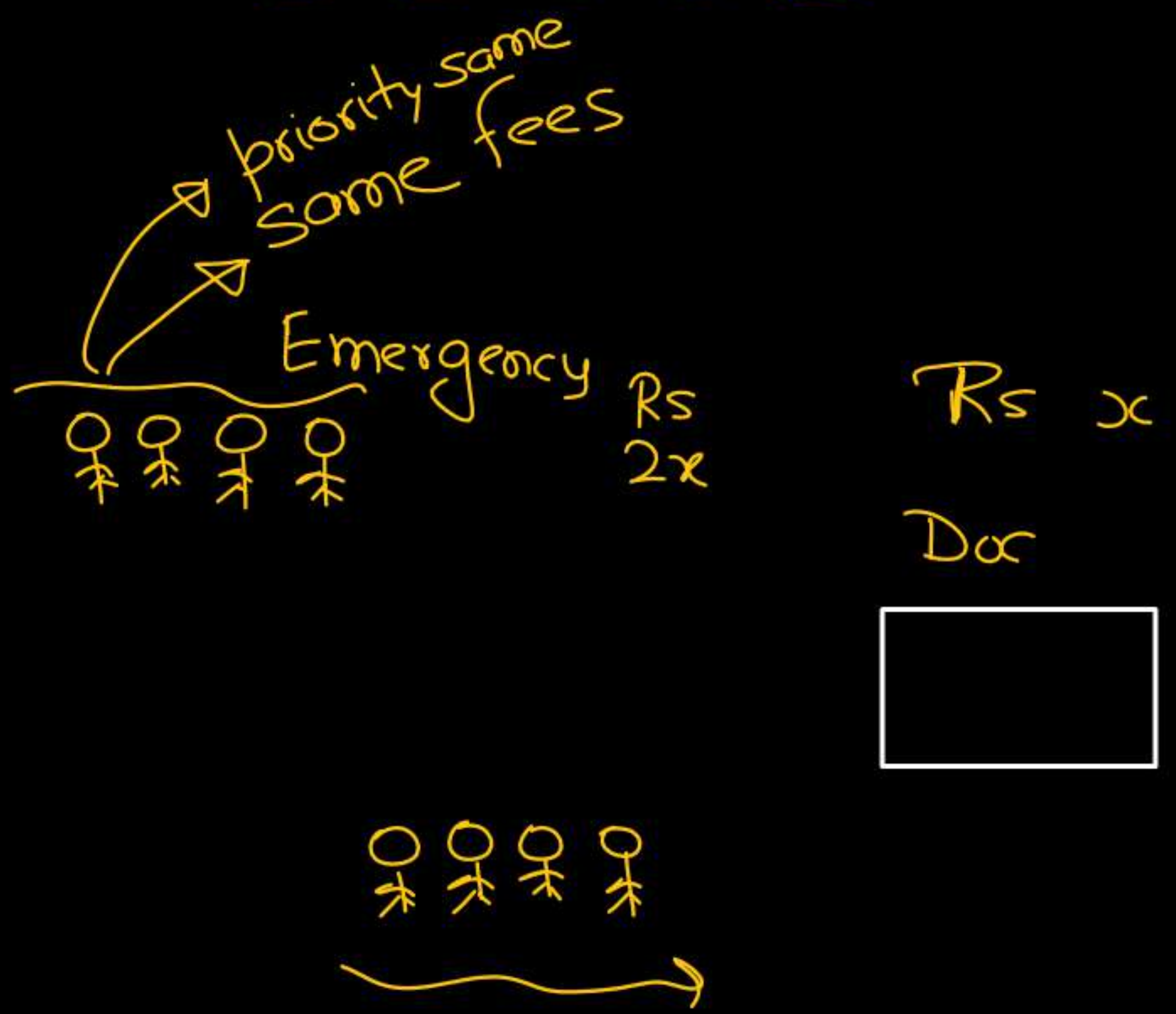
* A priority is associated with Every element.

* Elements will be processed as per priority.



Priority Queue

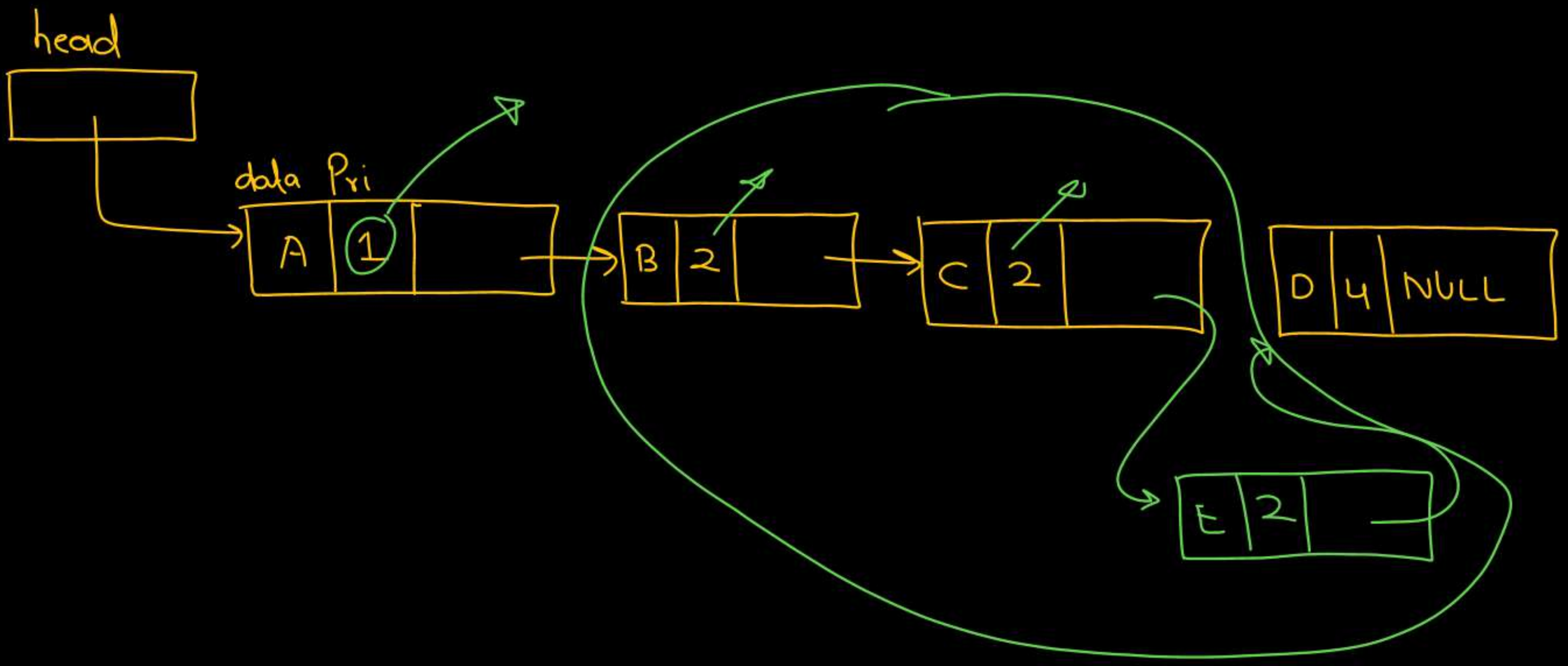
In case
2 elements have
same priority
↓
their order of
insertion



data Pri
(E, 2)

Small number : High priority

Large number : High priority



NAT

Consider a sequence a of elements $a_0 = 1, a_1 = 5, a_2 = 7, a_3 = 8, a_4 = 9$, and $a_5 = 2$. The following operations are performed on a stack S and a queue Q, both of which are initially empty.

I: push the elements of a from a_0 to a_5 in that order into S.

II: enqueue the elements of a from a_0 to a_5 in that order into Q.

III: pop an element from S.

IV: dequeue an element from Q.

V: pop an element from S.

VI: dequeue an element from Q.

VII: dequeue an element from Q and push the same element into S.

VIII: Repeat operation VII three times.

IX: pop an element from S.

X: pop an element from S.

The top element of S after executing the above operations is _____.

[GATE-2023-CS: 2M]

NAT

Consider a sequence a of elements $a_0 = 1$, $a_1 = 5$, $a_2 = 7$, $a_3 = 8$, $a_4 = 9$, and $a_5 = 2$. The following operations are performed on a stack S and a queue Q, both of which are initially empty.

I: push the elements of a from a_0 to a_5 in that order into S.

II: enqueue the elements of a from a_0 to a_5 in that order into Q.

III: pop an element from S.

IV: dequeue an element from Q.

V: pop an element from S.

VI: dequeue an element from Q.

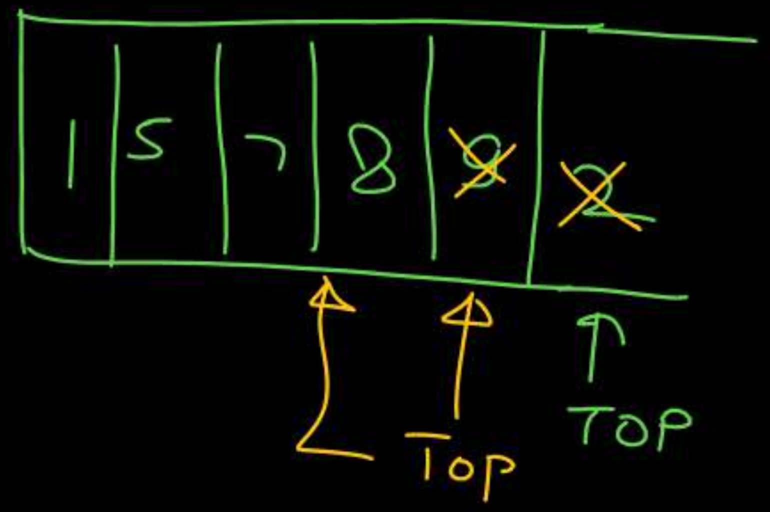
VII: dequeue an element from Q and push the same element into S.

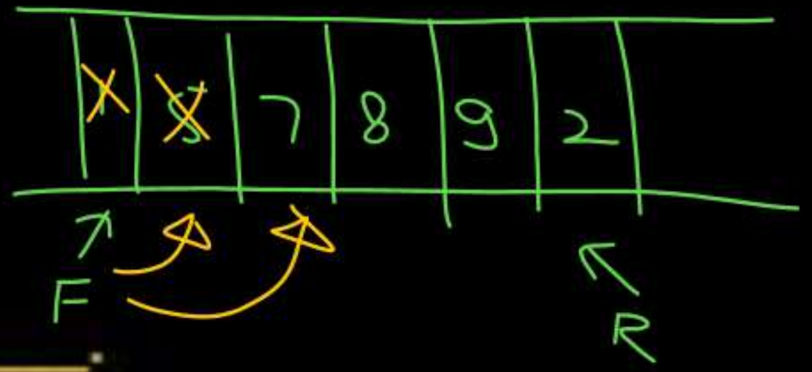
VIII: Repeat operation VII three times.

IX: pop an element from S.

X: pop an element from S.

The top element of S after executing the above operations is ______.





[GATE-2023-CS: 2M]

NAT

Consider a sequence a of elements $a_0 = 1$, $a_1 = 5$, $a_2 = 7$, $a_3 = 8$, $a_4 = 9$, and $a_5 = 2$. The following operations are performed on a stack S and a queue Q, both of which are initially empty.

I: push the elements of a from a_0 to a_5 in that order into S.

II: enqueue the elements of a from a_0 to a_5 in that order into Q.

III: pop an element from S.

IV: dequeue an element from Q.

V: pop an element from S.

VI: dequeue an element from Q.

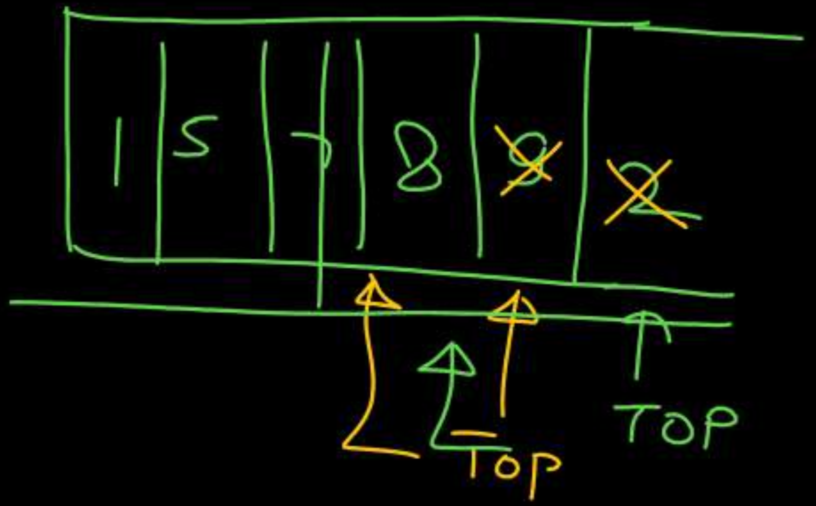
VII: dequeue an element from Q and push the same element into S.

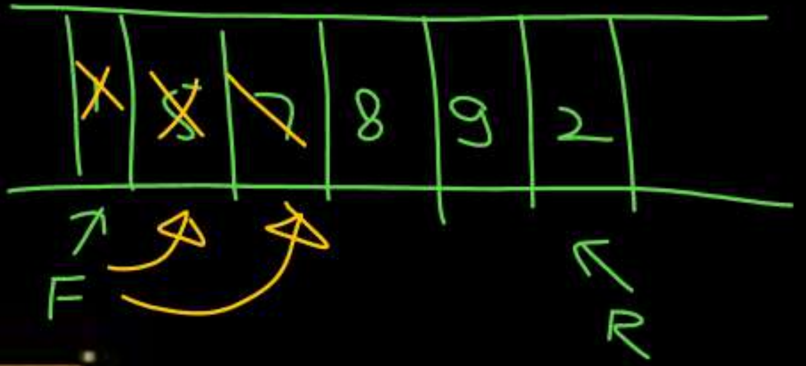
VIII: Repeat operation VII three times.

IX: pop an element from S.

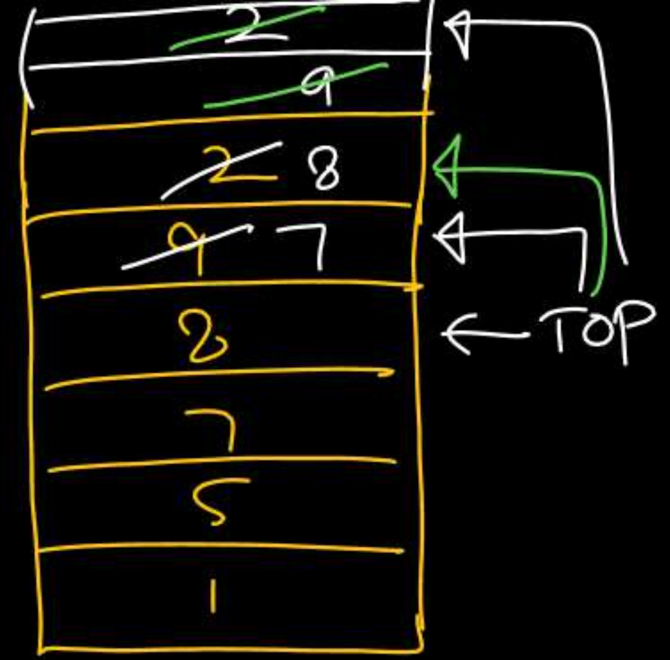
X: pop an element from S.

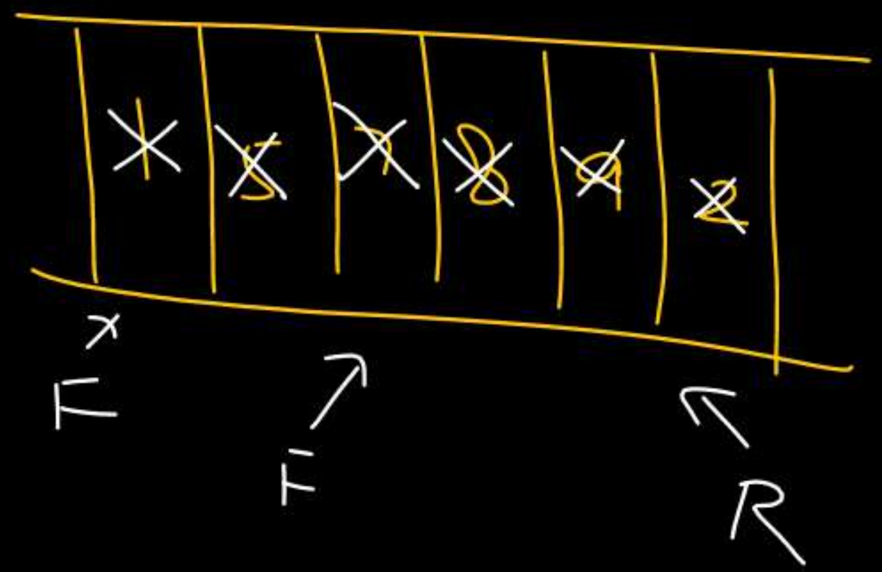
The top element of S after executing the above operations is 8.





[GATE-2023-CS: 2M]


TOP


F
F
R

MCQ

Let A be a priority queue for maintaining a set of elements. Suppose A is implemented using a max-heap data structure. The operation EXTRACT-MAX(A) extracts and deletes the maximum element from A. The operation INSERT(A, key) inserts a new element key in A. The properties of a max-heap are preserved at the end of each of these operations. When A contains n elements, which one of the following statements about the worst case running time of these two operations is TRUE? [GATE-2023-CS: 2M]

A. Both EXTRACT-MAX(A) and INSERT(A, key) run in $O(1)$.

B. Both EXTRACT-MAX(A) and INSERT(A, key) run in $O(\log(n))$.

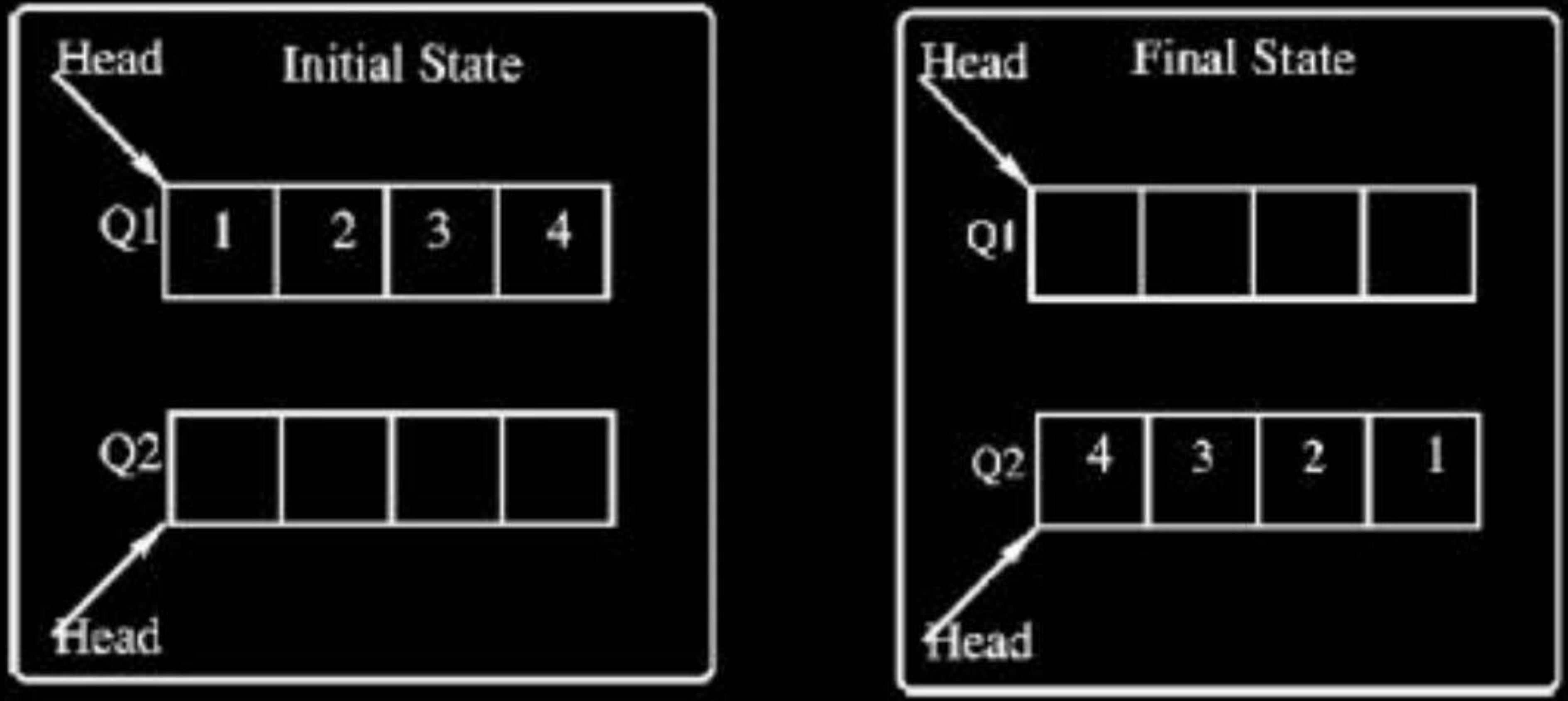
C. EXTRACT-MAX(A) runs in $O(1)$ whereas INSERT(A, key) runs in $O(n)$.

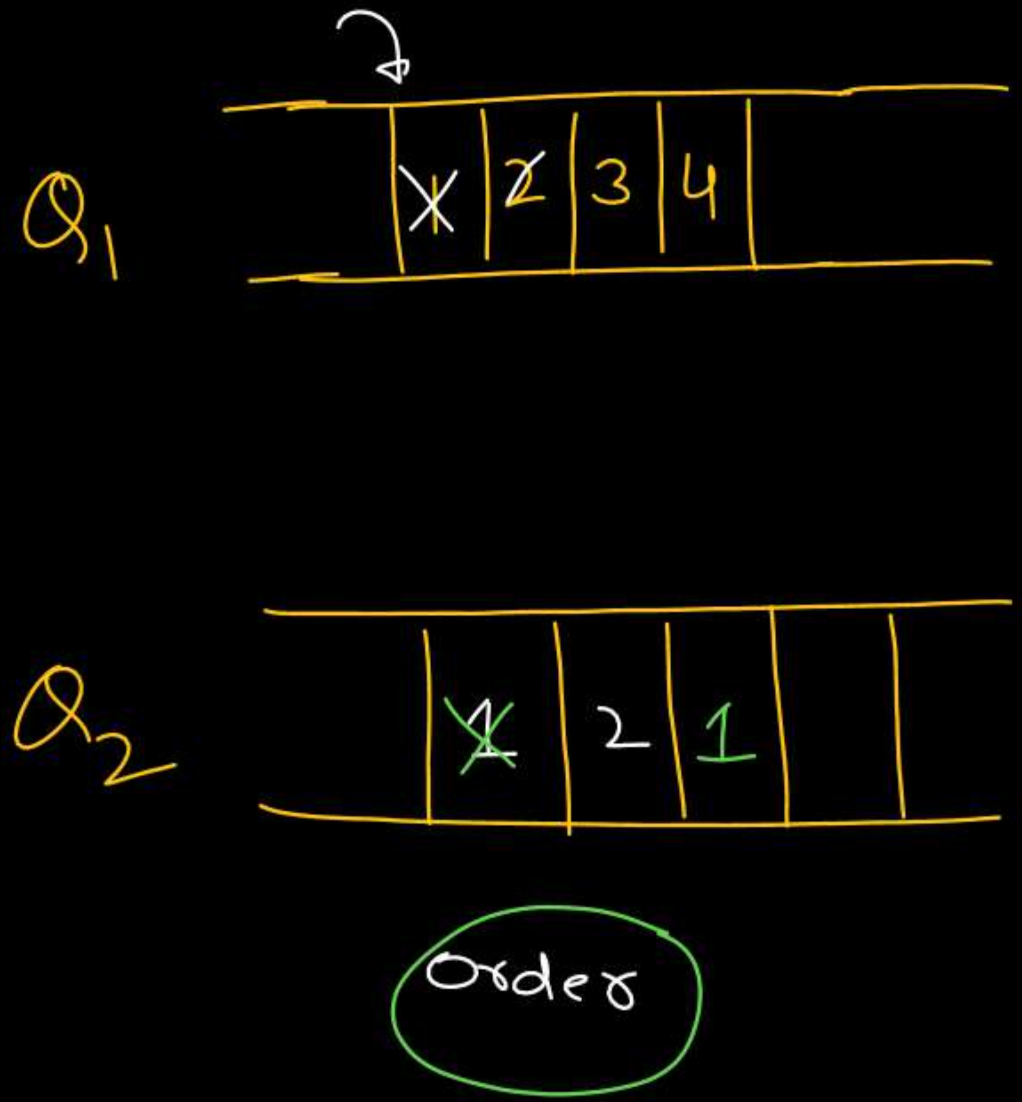
D. EXTRACT-MAX(A) runs in $O(1)$ whereas INSERT(A, key) runs in $O(\log(n))$.

NAT

Consider the queues Q_1 containing four elements and Q_2 containing none (shown as the Initial State in the figure). The only operations allowed on these two queues are Enqueue(Q, element) and Dequeue(Q). The minimum number of Enqueue operations on Q_1 required to place the elements of Q_1 in Q_2 in reverse order (shown as the Final State in the figure) without using any additional storage is________.

[GATE-2022-CS: 2M]





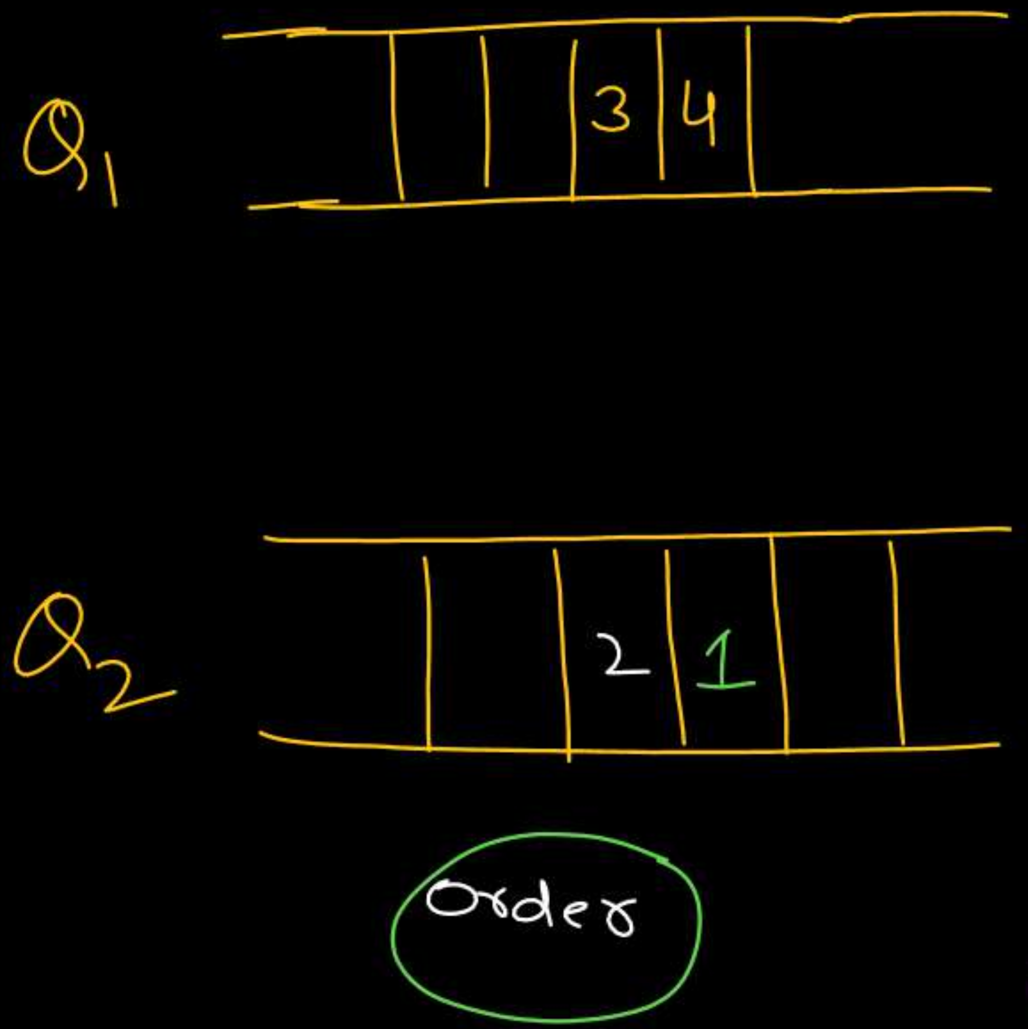
(i)

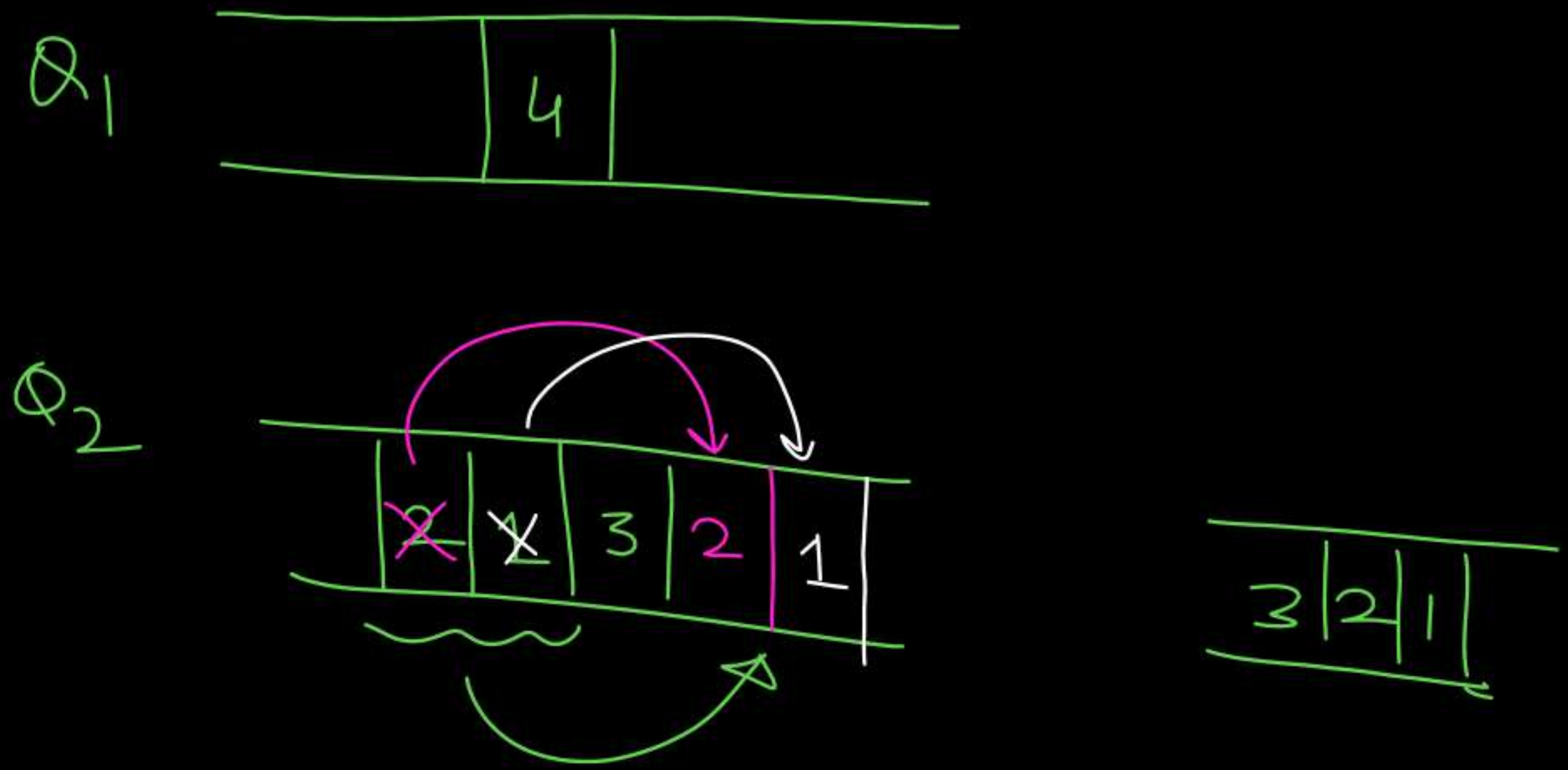
Enqueue(Q_2, Dequeue(Q_1))

Enqueue(Q_2, Dequeue(Q_1))

Enqueue(Q_2, $\overset{1}{\text{Dequeue}}$($Q_2$))

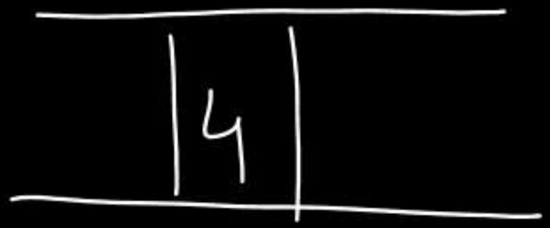
$Enqueue(Q_2, Dequeue(Q_1))$

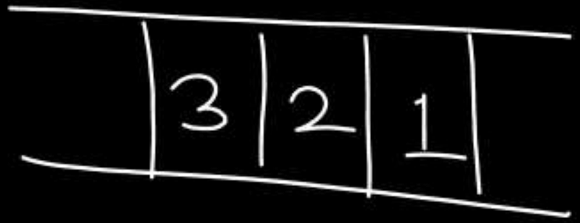




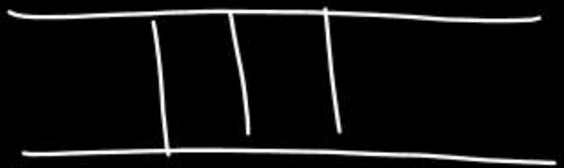
$Enqueue(Q_2, Dequeue(Q_2))$

$Enqueue(Q_2, Dequeue(Q_2))$

Q_1 

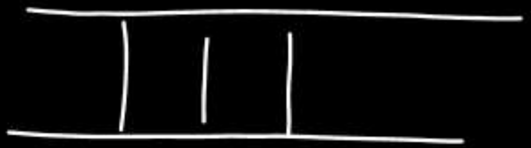
Q_2 

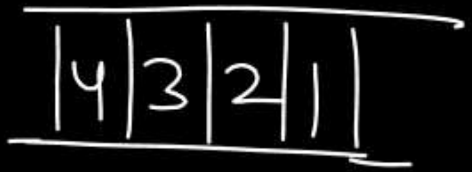
Enqueue(Q_2, Dequeue(Q_1))

Q_1 

Q_2 

Enqueue(Q_2, Dequeue(Q_2)) $\Rightarrow$ 3 times

Q_1 

Q_2 

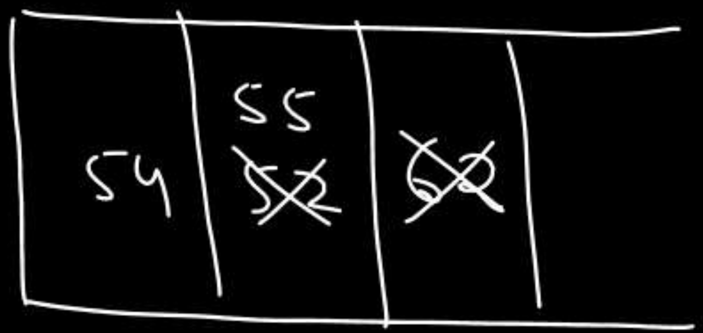
NAT

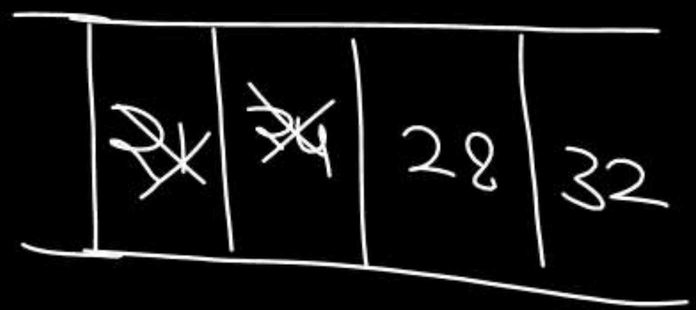
Consider the following sequence of operations on an empty stack.

push(54); push(52); pop(); push(55); push(62); s = pop();

Consider the following sequence of operations on an empty queue.
enqueue(21); enqueue(24); dequeue(); enqueue(28); enqueue(32);
q = dequeue();
The value of s + q is ______ 62+24=86 s=62

[GATE-2021-Set1-CS: 1M]

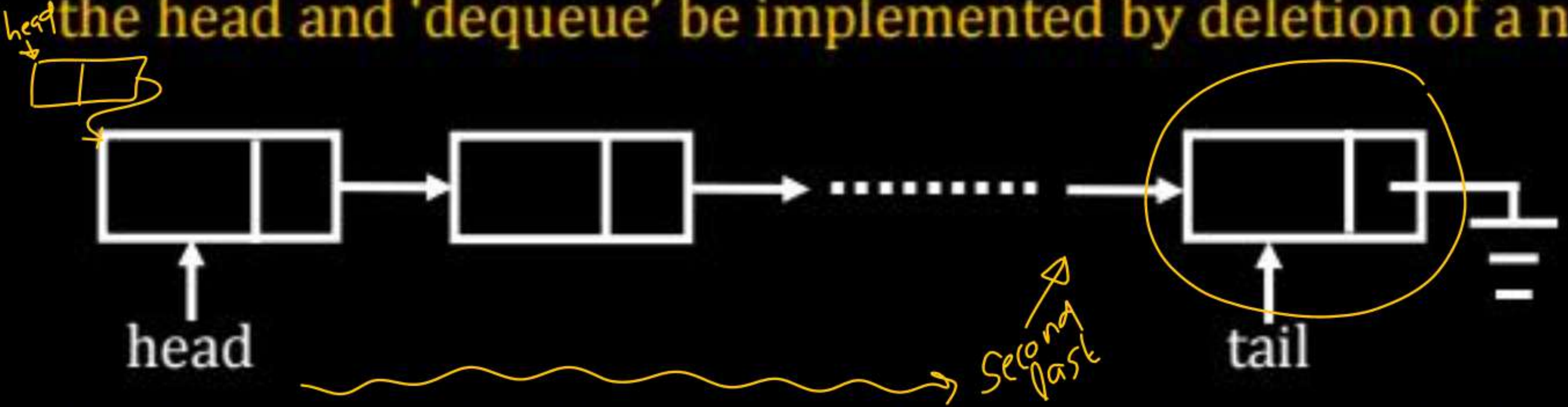




q = 24

MCQ

A queue is implemented using a non-circular singly linked list. The queue has a head pointer and tail pointer, as shown in the figure. Let n denote of number of nodes in the queue. Let 'enqueue' be implemented by inserting a new node at the head and 'dequeue' be implemented by deletion of a node from the tail.



$\Theta(n)$

Which one of the following is the time complexity of the most time-efficient implementation of enqueue and dequeue, respectively, for this data structure?

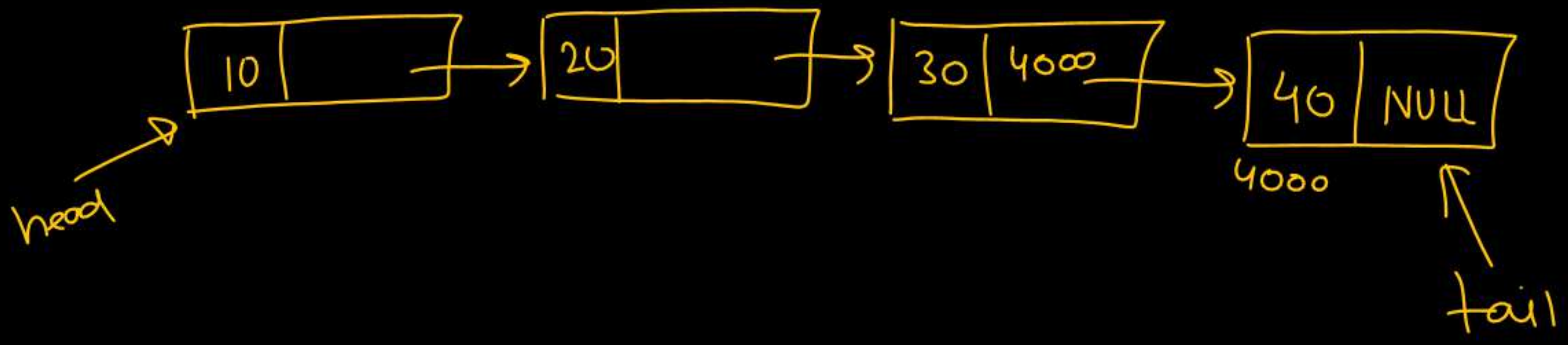
[GATE-2018 - CS: 1M]

(A) $\theta(1), \theta(1)$

(B) $\theta(1), \theta(n)$

(C) $\theta(n), \theta(1)$

(D) $\theta(n), \theta(n)$


10
20
30
4000
40
NULL
4000
head
tail

MCQ

A circular queue has been implemented using a singly linked list where each node consists of a value and a single pointer pointing to the next node. We maintain exactly two external pointers FRONT and REAR pointing to the front node and the rear node of the queue, respectively. Which of the following statements is/are CORRECT for such a circular queue, so that insertion and deletion operations can be performed in O(1) time?

I. Next pointer of front node points to the rear node.

II. Next pointer of rear node points to the front node [GATE-2017 - CS: 1M]

(A) I only

(B) II only

(C) Both I and II

(D) Neither I nor II

MCQ

Consider the following New-order strategy for traversing a binary tree:

Visit the root;

Visit the right subtree using New-order;

Visit the left subtree using New-order;

The New – order traversal of the expression tree corresponding to the reverse polish expression

3 4 * 5-2^6 7 * 1 + - is given by

Postorder

LT, RT, Root

[GATE-2016 - CS: 2M]

Root, RT, LT (Reverse of Postorder)

A. + - 1 6 7 * 2 ^ 5 – 3 4 *

B. - + 1 * 6 7 ^2 – 5 * 3 4

C. - + 1 * 7 6 ^ 2 – 5 * 4 3

D. 1 7 6 * + 2 5 4 3 * - ^ -

NAT

Let Q denote a queue containing sixteen numbers and S be an empty stack. Head(Q) returns the element at the head of the queue Q without removing it from Q. Similarly Top(S) returns the element at the top of S without removing it from S. Consider the algorithm given below.

Q :

```
While Q is not Empty do
 If S is Empty OR Top (S) ≤ Head (Q) then
  x:=Dequeue (Q);
  Push (S, x):
 Else
  x := Pop (S);
  Enqueue (Q,x);
 End
End
```

[GATE-2016 - CS: 2M]

The maximum possible number of iterations of the while loop in the algorithm is ____

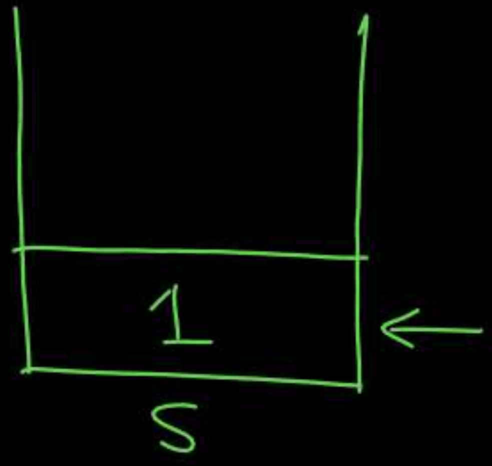
```
While (Q is not Empty)
    {
if S is Empty OR TOP(s) ≤ head(Q)
        {
            x = Dequeue(Q)
            Push(s,x);
        }
    else {
            x = Pop(s);
            Enqueue(Q,x);
        }
    }
```

```
While (Q is not Empty)
    {
if S is Empty OR TOP(S) ≤ head(Q)
            {
              x = Dequeue(Q)
              Push(S, x);
            }
        else {
              x = Pop(S);
              Enqueue(Q, x);
             }
    }
```

$n = 3$

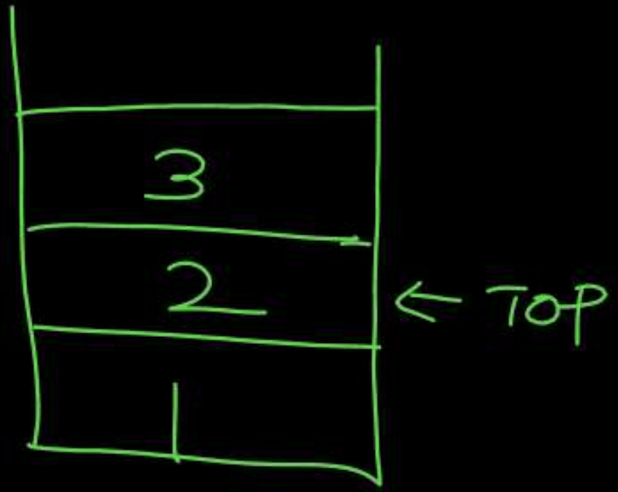
(i) 1, 2, 3

Q: ~~1~~, 2, 3

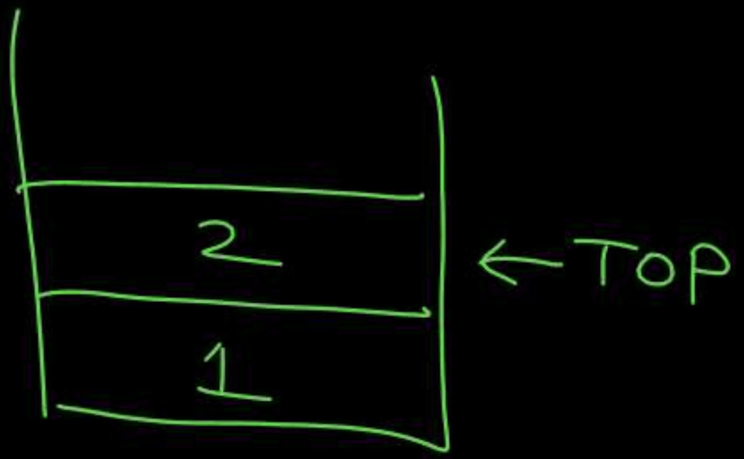


if ✓ if ✓ if ✓

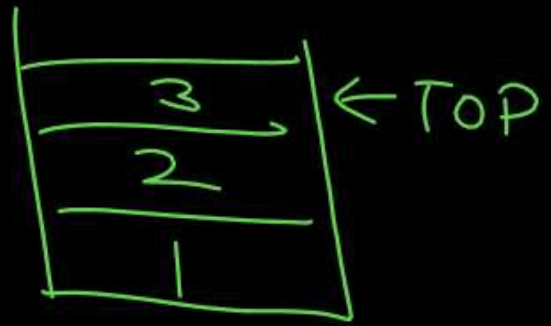
Q: ~~3~~



Q: ~~2~~, 3



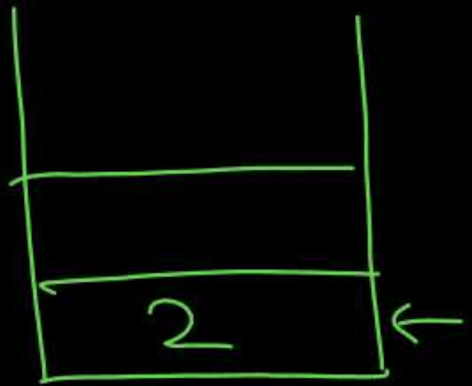
Q:



$1, 2, 3 \Rightarrow 3$

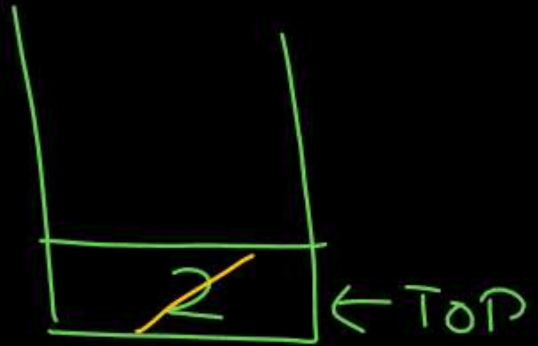
$n = 3$

Q: ~~2~~, 1, 3



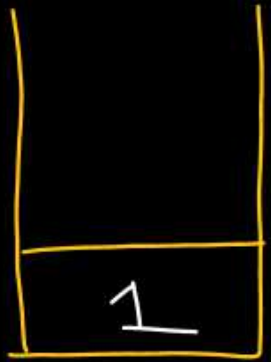
After 1 iteration:

Q: 1, 3



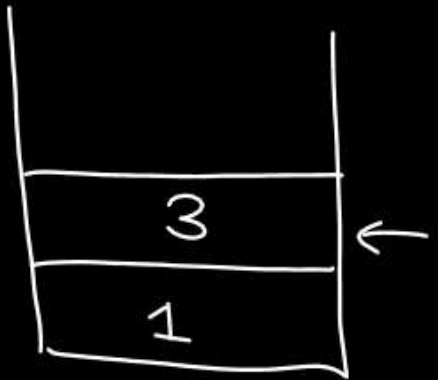
if else if if

Q: ~~1~~, 3, 2



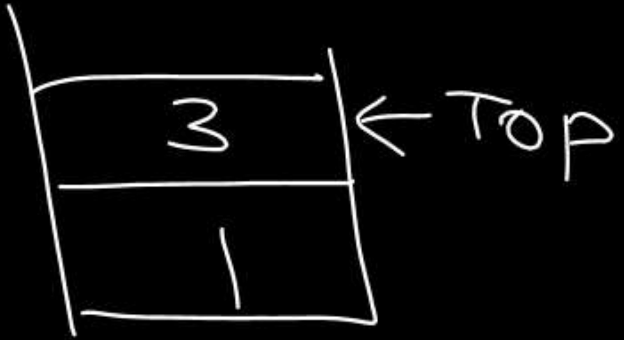
After 3 iterations:

Q: ~~3~~, 2

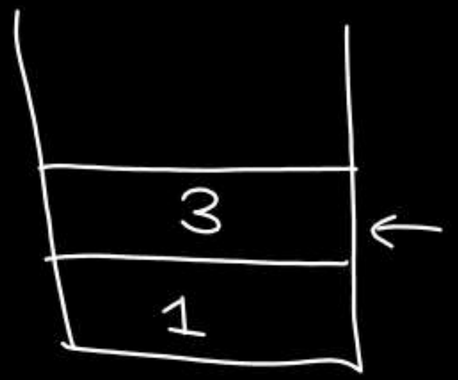


After 4 iterations

Q: 2



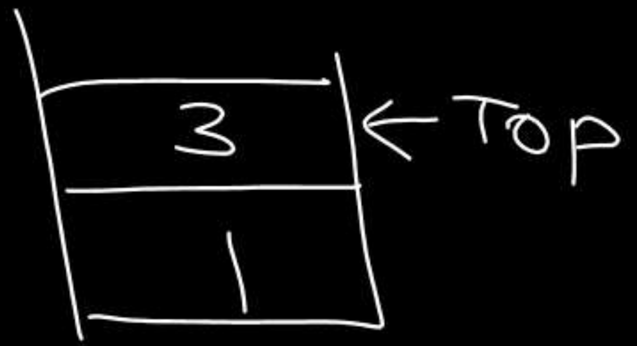
5th
else



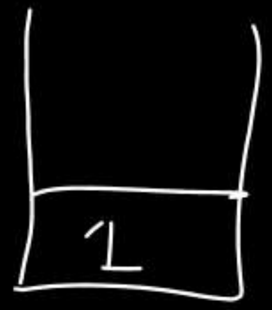
2 more iterations

After 4 iterations

Q : 2



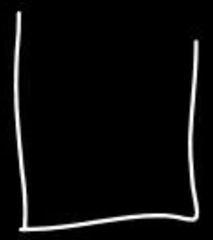
Q : 2,3



Q : 2,3

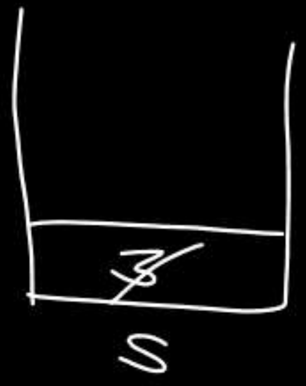
Q : 2,1,3 $\Rightarrow$ 7 iterations

Q: ~~3~~, 2, 1



1st ✓

Q: 2, 1



Q: 2, 1, 3

2nd else

After 2 iterations:

Q: 2, 1, 3

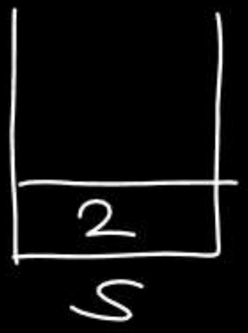
Q: 3, 2, 1 $\xrightarrow{\text{2 iteration}}$ Q: 2, 1, 3

Push

Pop

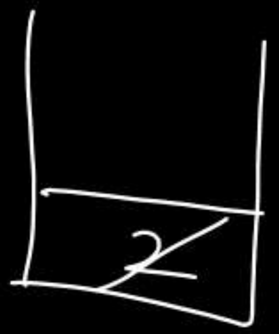
3rd if

Q: ~~2~~, 1, 3



4th else

Q: 1, 3

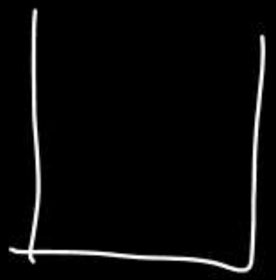


Q: 1, 3, 2

After 2 more iteration

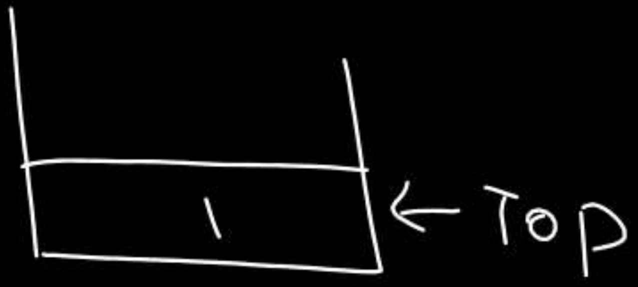
Q: 2, 1, 3 $\longrightarrow$ Q: 1, 3, 2

Q: 1, 3, 2

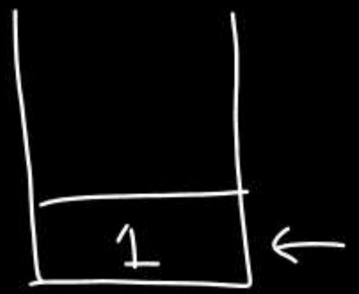


5th iteration

Q: 3, 2



Q : 3, 2



1 will never pop out

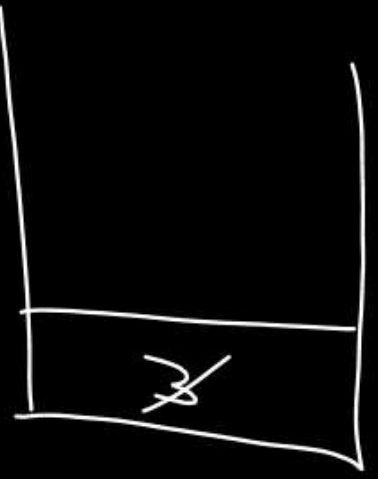
3 element $\xrightarrow{\text{5 Iteration}}$ 2 element

3,2,1 $\longrightarrow$

3,2 $\longrightarrow$

Q : 2

6th ✓



7th

Pop

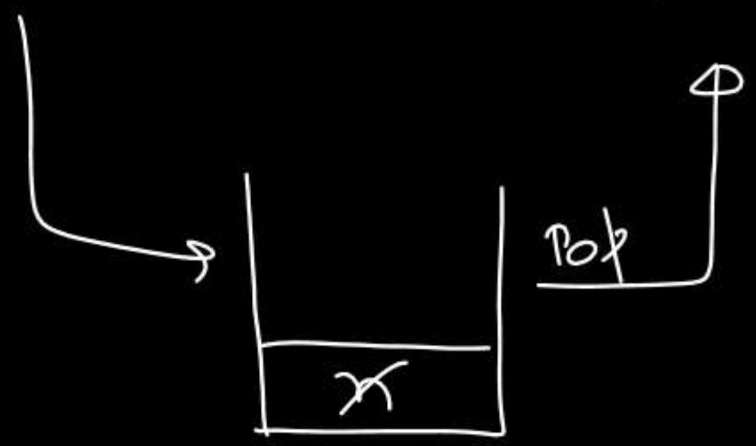
$\boxed{Q : 2, 3}$

2 more iteration

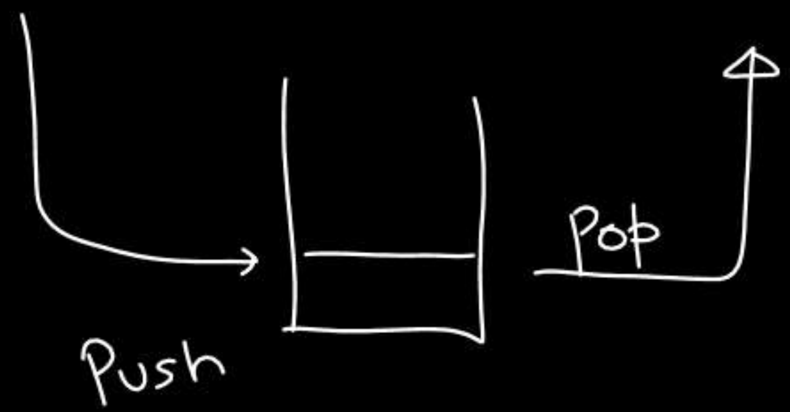
$\Rightarrow$ 9 iterations

n^2 iteration $\Rightarrow 16^2 = 256$

$\cancel{n}, n-1, n-2, \ldots 3, 2, 1, n$



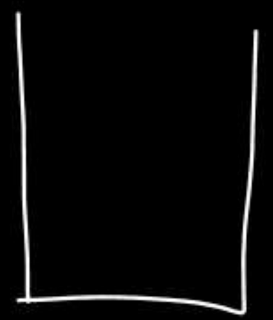
$n-1, n-2, \ldots 3, 2, 1, n$

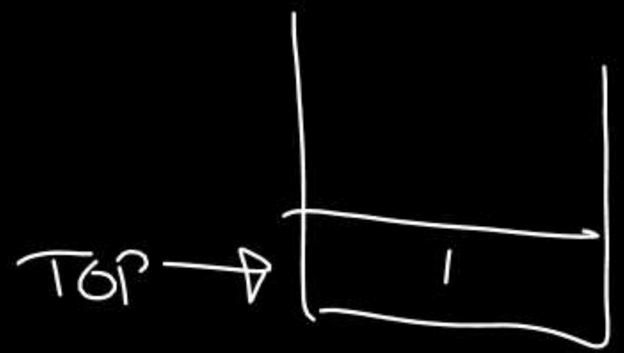


$n-2, \ldots 3, 2, 1, n, n-1$

for all 1^{st} $(n-1)$ elem $\Rightarrow$ 2 iterations

$n, n-1, n-2, \ldots 3, 2, 1 \xrightarrow[\text{iterations}]{2(n-1)} 1, n, n-1, n-2, \ldots 3, 2 \xrightarrow[\text{iteration}]{\text{1 more}}$





n size

$n, n-1, n-2, \ldots 3, 2, 1 \xrightarrow{2(n-1)+1} n, n-1, n-2, \ldots 3, 2 \xrightarrow{2(n-2)+1}$

$$2(n-1)+1,\ 2(n-2)+1,\ 2(n-3)+1\ \ldots\ 2(1)+1,\ 2(0)+1$$

$$\Rightarrow \quad 2(1+2+3+\cdots(n-1))+n$$

$$= \cancel{2}\frac{(n-1)(n)}{\cancel{2}}+n$$

$$= n^2-\cancel{n}+\cancel{n}$$

$$= n^2$$

MCQ

A queue is implemented using an array such that ENQUEUE and DEQUEUE operations are performed efficiently. Which one of the following statements is CORRECT (n refers to the number of items in the queue)? [GATE-2016 - CS: 1M]

(A) Both operations can be performed in O(1) time

Circular Queue
mod concept

(B) At most one operation can be performed in O(1) time but the worst case time for the other operation will be $\Omega(n)$

(C) The worst case time complexity for both operations will be $\Omega(n)$

(D) Worst case time complexity for both operations will be $\Omega(\log n)$

MCQ

The result evaluating the postfix expression 10 5 + 60 6 / * 8 – is

[GATE-2015 - CS: 1M]

(A) 284

(B) 213

(C) 142

(D) 71

NAT

Consider the C program below.

```
#include<stdio.h>
int * A, stkTop;
int stkFunc (int opcode, int val)
{
static int size = 0, stkTop = 0;
switch (opcode)
{
case -1: size = val; break;
case 0: if (stkTop < size)
      A[stkTop++] = val; break;
default: if (stkTop)
      return A[--stkTop];
}
return-1;
}
int main ()
{
   int B[20]; A = B; stkTop = -1;
   stkFunc (-1, 10);
   stkFunc (0, 5);
   stkFunc (0, 10);
   printf("%d\n",stkFunc(1,0) + stkFunc (1,0));
}
```

The value printed by the above program is ____

[GATE-2015 - CS: 2M]

MCQ

Suppose a stack implementation supports an instruction REVERSE, which reverses the order of elements on the stack, in addition to the PUSH and POP instructions. Which one of the following statements is TRUE with respect to this modified stack?

[GATE-2014 - CS: 2M]

(A) A queue cannot be implemented using this stack.

(B) A queue can be implemented where ENQUEUE takes a single instruction and DEQUEUE takes sequence of two instructions.

(C) A queue can be implemented where ENQUEUE takes a sequence of three instructions and DEQUEUE takes a single instruction

(D) A queue can be implemented where both ENQUEUE and DEQUEUE take a single instruction each.

MCQ

Consider the following operation along with Enqueue and Dequeue operations on queues, where k is a global parameter.

```
MultiDequeue(Q){
 m = k;
 while (Q is not empty and m>0) {
  Dequeue (Q);
  m = m-1;
 }
}
```

What is the worst case time complexity of a sequence of n MultiDequeue operations on an initially empty queue? [GATE-2013 - CS: 2M]

(A) $\Theta(n)$

(B) $\Theta(n+k)$

(C) $\Theta(nk)$

(D) $\Theta(n^2)$

MCQ

Suppose a circular queue of capacity (n-1) elements is implemented with an array of n elements. Assume that the insertion and deletion operations are carried out using REAR and FRONT as array index variables, respectively. Initially, REAR = FRONT = 0. The conditions to detect queue full and queue empty are

[GATE-2012 - CS: 2M]

(A) Full: (REAR + 1) mod n = = FRONT
Empty: REAR = = FRONT

(B) Full: (REAR + 1) mod n = = FRONT
Empty: (FRONT + 1) mod n = = REAR

(C) Full: REAR = = FRONT
Empty:(REAR + 1) mod n = = FRONT

(D) Full:(FRONT + 1) mod n = = REAR
Empty: REAR = = FRONT


THANK YOU GW
SOLDIERS !