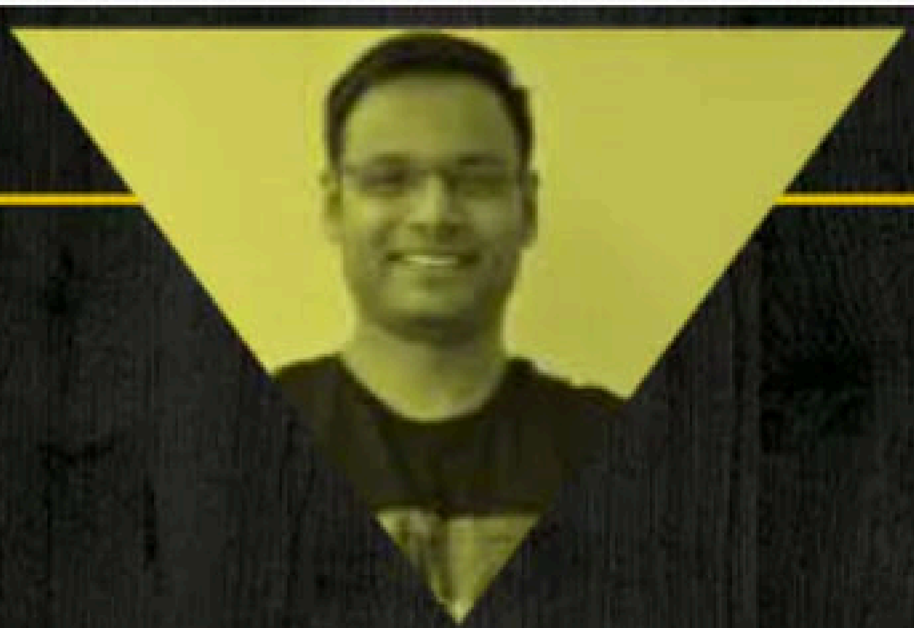
Data Structure: Queue & Stack

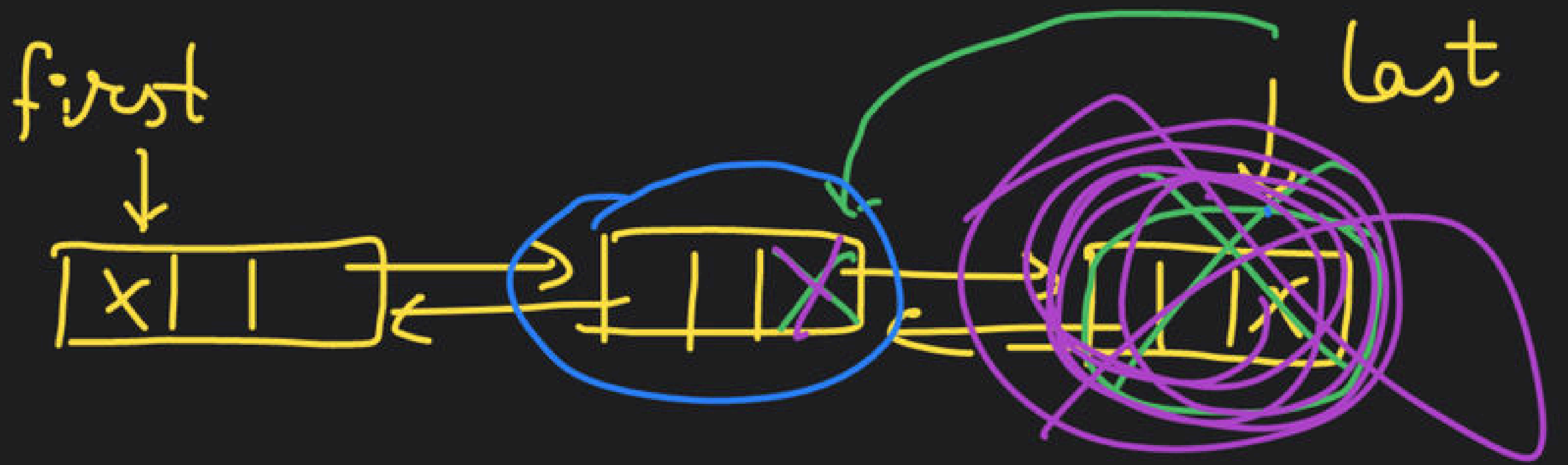
By: Vishvadeep Gothi



Hello!

I am Vishvadeep Gothi

I am here because I love to teach

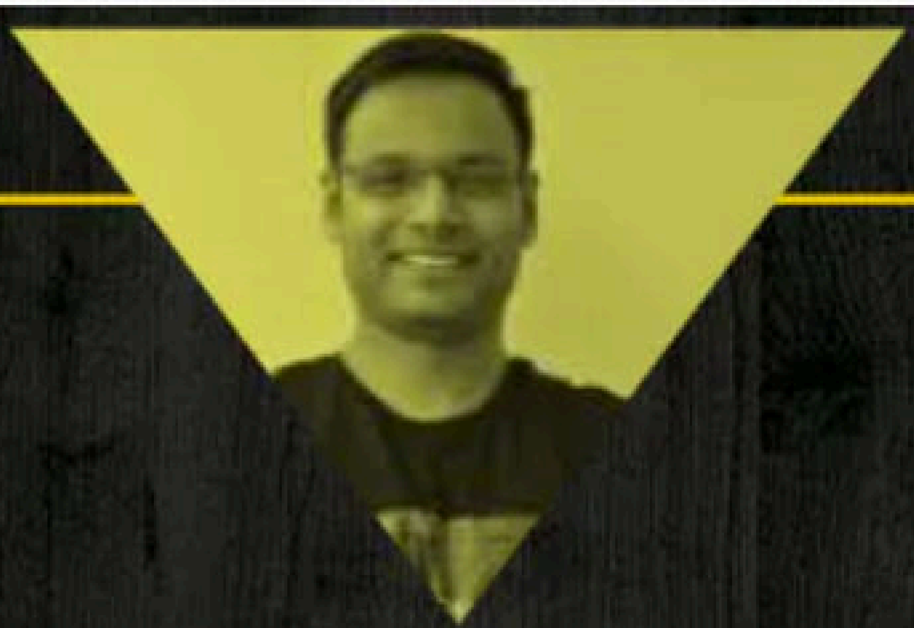


last = last → prev

last → next = NULL

Data Structure: Queue 1

By: Vishvadeep Gothi



Hello!

I am Vishvadeep Gothi

I am here because I love to teach

Queue

Basic operations

- EnQueue()
- DeQueue()

Implementation of Queue: Using Array

Implementation of Queue: Using Linked List

Question 1

- What is the content of queue after following operations on an empty queue?

1. Enqueue(a)
2. Enqueue(b)
3. Dequeue()
4. Enqueue(d)
5. Enqueue(e)
6. Dequeue()

Question 2

What is the content of queue after following operations on an empty queue?

4 insertions

2 deletions

(A) First inserted 2 elements

(B) Last inserted 2 elements

(C) Random 2 elements

(D) None

Question 3

A circular queue is implemented using an array A[1:8]. The array contains values [x, y, z, _, _, _, _, a]. In the queue 3 enqueue and 2 dequeue operations are performed in random arbitrary order. What is the value of front and rear indexes?

Other Functions on Queue

QueueFront() → It reads front element of Queue and returns it

QueueRear() → — " — rear — " — " — " —

IsEmptyQueue()
- → True when given Queue is empty
- → False — " — is not empty

	f				r
Q	a	c	e	b	d

QueueFront (Q) ⇒ a | QueueRear(Q) ⇒ d

Question 4

Consider a linear queue Q. What would be the content of queue Q after the following code is executed?

File = 4,7,5,9,0,4,7,8,3,5,0,7,0,8,9,0,5

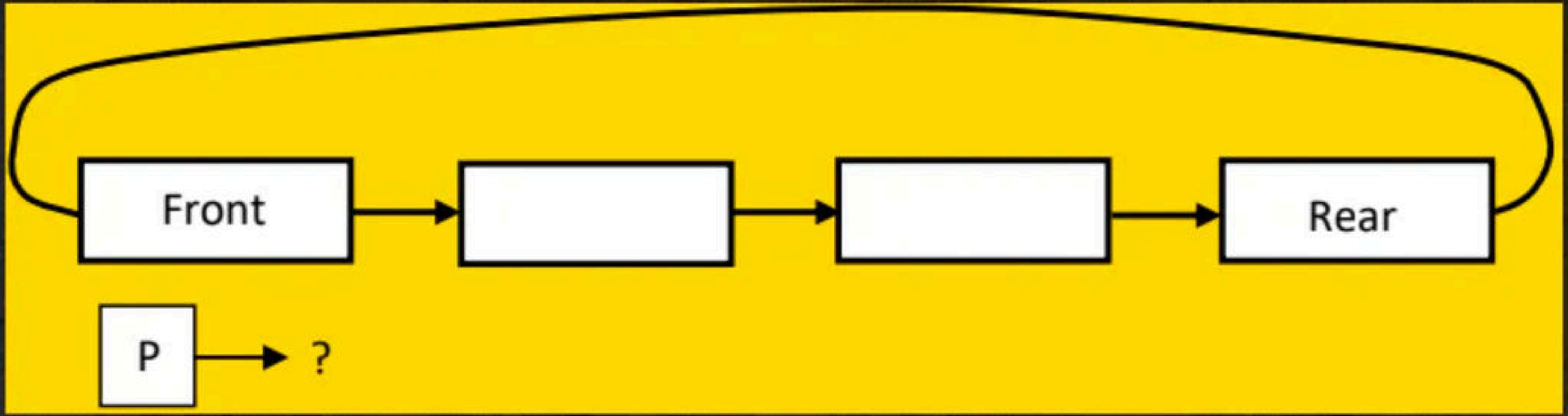
```
loop(till end of file)
X= read number
if(x ! = 0)
        enqueue(Q, X)
else
        a = queueRear(Q)
        enqueue(Q, a)
End Loop
```

Q = 4, 7, 5, 9, 9, 4, 7, 8, 3, 5, 5, 7, 7, 8, 9, 9, 5

GATE Question

CS- 2004

◈ A circularly linked list is used to represent a Queue. A single variable p is used to access the Queue. To which node should p point such that both the operations enQueue and deQueue can be performed in constant time?



(A) Rear node

(B) Front node

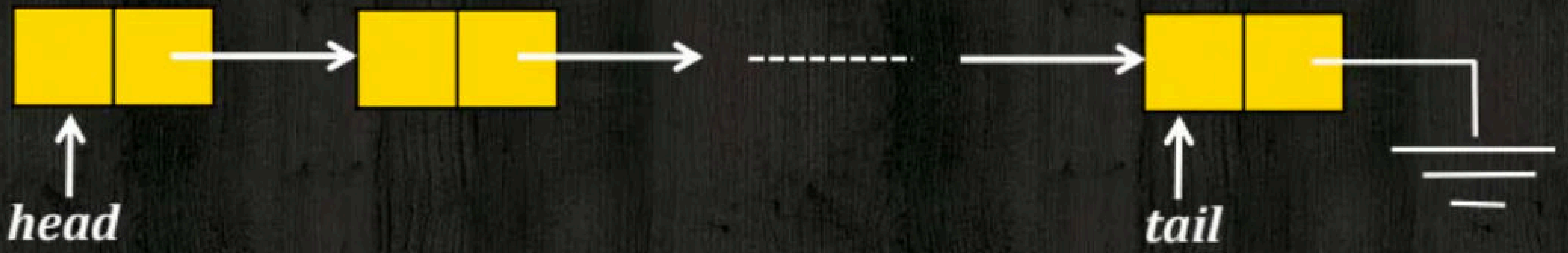
(C) Not possible with a single pointer

(D) Node next to front

GATE Question

CS- 2018

- A queue is implemented using a non-circular singly linked-list. The queue has a head pointer and a tail pointer, as shown in the figure. Let n denote the 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?



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

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

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

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

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

First & last node add. given

	EnQ.	DeQ.
EnQ => Rear, DeQ => front	$\Theta(1)$	$\Theta(1)$
EnQ => front, DeQ => Rear	$\Theta(1)$	$\Theta(n)$

only first node add. given

	EnQ	DeQ
EnQ => Rear, DeQ => front	$\Theta(n)$	$\Theta(1)$
EnQ => front, DeQ => Rear	$\Theta(1)$	$\Theta(n)$

Question 1 GATE-2007

Suppose you are given an implementation of a queue of integers. The operations that can be performed on the queue are:

isEmpty(Q) — returns true if the queue is empty, false otherwise.
delete(Q) — deletes the element at the front of the queue and returns its value.
insert(Q,i) — inserts the integer i at the rear of the queue.

Consider the following function:

```
void f (queue Q) {
int i ;
if (!isEmpty(Q)) {
   i = delete(Q);
   f(Q);
   insert(Q, i);
  }}
```

Queue 10, 5, 9, 6

Q

10	5	9	6

f f r

f(Q)
{
i = 10
f(Q)
insert(Q, 10)
}

Q

5	9	6

i = 5
f(Q)
insert(Q, 5)

Q

9	6

i = 9
f(Q)
insert(Q, 9)

6

i = 6
f(Q)
insert(Q, 6)

Q

6	9	5	10

Question 1 GATE-2007 cont..

What operation is performed by the above function f ?

A. Leaves the queue Q unchanged
B. Reverses the order of the elements in the queue Q
C. Deletes the element at the front of the queue Q and inserts it at the rear keeping the other elements in the same order
D. Empties the queue Q

There is function multideQ(k);
which deletes k elements from a Queue which is having $n \geq k$ elements.

→ Implementatⁿ ⇒ A loop runs k times & deletes one element in each iteratⁿ.

⇓

R.T. Complexity of multideQ(k) ⇒ Theta(k)

or

O(n)

Question 2 GATE-2013

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
    }
}
```

n no of operatⁿs → EnQ → DeQ → multideQ

What is the worst case time complexity of a sequence of n queue operations on an initially empty queue?

A. $\Theta(n)$ B. $\Theta(n+k)$ C. $\Theta(nk)$ D. $\Theta(n^2)$

$n-1 \Rightarrow$ EnQ

$1 \Rightarrow$ MultideQ

$k = n-1$

$\frac{n}{2}-1$ EnQ (2 times)

2 multidequeue

$k = n/2 - 1$

- - -

option 1:-

$n-1$ EnQ $\Rightarrow \theta(n)$

1 multiDeQ $(k=n-1)$ $\Rightarrow \theta(n)$

$\rightarrow \theta(n)+\theta(n)$

$\Rightarrow \theta(n)$

---

option 2:-

$\frac{n}{2}-1$ EnQ

1 M. DeQ $\left(k=\frac{n}{2}-1\right)$

$\frac{n}{2}-1$ EnQ

1 M. DeQ $\left(k=\frac{n}{2}-1\right)$

$\rightarrow n-2$

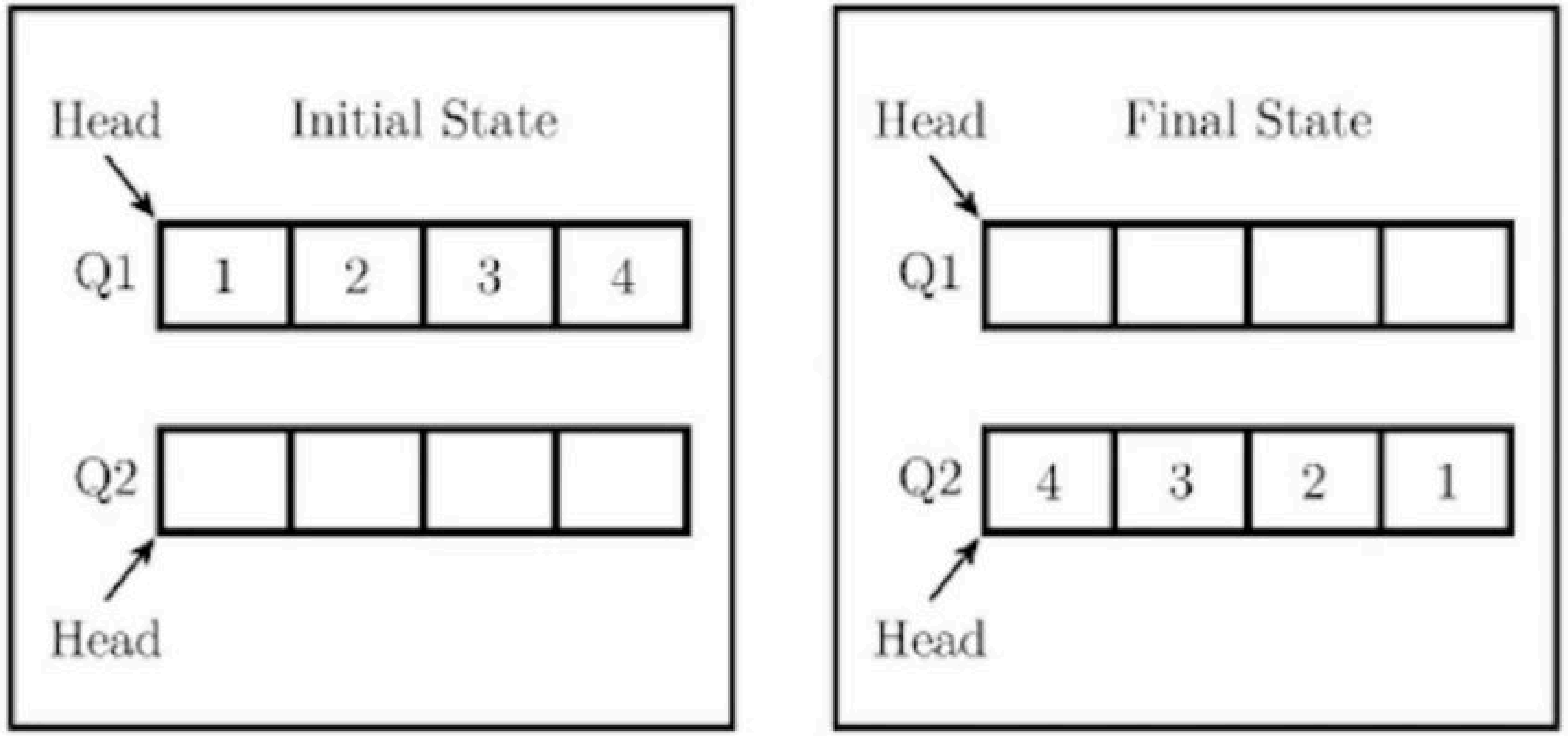
$\rightarrow n-2$

$\rightarrow 2n-4$

$\Rightarrow \theta(n)$

GATE - 2022

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______________.



Ans = 0

Q1 :- 1, 2, 3, 4

Q2 :-

Q1 2, 3, 4

Q2 1

Q1 3, 4

Q2 1, 2

Q1 3, 4

Q2 2, 1

Q1 4

Q2 2, 1, 3

Q1 4

Q2 1, 3, 2

Q1 4

Q2 3, 2, 1

Q1 =

Q2 = 3, 2, 1, 4

Q1 =

Q2 = 2, 1, 4, 3

Q1

Q2 = 1, 4, 3, 2

Q1

Q2 = 4, 3, 2, 1

Double Ended Queue

Priority Queue

Happy Learning

Double Ended Queue

Priority Queue

Priority Queue Implementation

Stack

Implementation of Stack: Using Array

Implementation of Stack: Using Linked List

Operations on Stack

Peep(Stack, top, i)

Change (Stack, top, i, item)

Question GATE-1991

The following sequence of the operations is performed on a stack

PUSH(10), PUSH(20), POP, PUSH(10), PUSH(20), POP, POP, POP, PUSH(20), POP

The sequence of values popped out is?

a) 20,10,20,10,20
b) 20,20,10,10,20
c) 10,20,20,10,20
d) 20,20,10,20,10

Question GATE-2005

A function f defined on stacks of integers satisfies the following properties.

$f(\phi) = 0$ and $f(push\ (S, i)) = max\ (f(S), 0) + i.$

for all stacks *S* and integers *i*. if a stack *S* contains the integers 2, -3, 2, -1, 2 in order from bottom to top, what is *f(S)*?

(A) 6
(B) 4
(C) 3
(D) 2

Stack Permutations

Question GATE-1994

Which of the following permutations can be obtained in the output (in the same order) using a stack assuming that the input is the sequence 1, 2, 3, 4, 5 in that order?

a) 3, 4, 5, 1, 2
b) 3, 4, 5, 2, 1
c) 1, 5, 2, 3, 4
d) 5, 4, 3, 1, 2

Question GATE-1997

A priority queue Q is used to implement a stack S that stores characters. PUSH(C) is implemented as INSERT(Q, C, K) where K is an appropriate integer key chosen by the implementation. POP is implemented as DELETEMIN(Q). For a sequence of operations, the keys chosen are in

(A) Non-increasing order

(B) Non-decreasing order

(C) strictly increasing order

(D) strictly decreasing order

Question GATE-1997

A priority queue Q is used to implement a stack S that stores characters. PUSH(C) is implemented as INSERT(Q, C, K) where K is an appropriate integer key chosen by the implementation. POP is implemented as DELETEMIN(Q). For a sequence of operations, the keys chosen are in

(A) Non-increasing order

(B) Non-decreasing order

(C) strictly increasing order

(D) strictly decreasing order

DPP

Question 1 GATE-1996

Consider the following statements:

i. First-in-first out types of computations are efficiently supported by STACKS.

ii. Implementing LISTS on linked lists is more efficient than implementing LISTS on an array for almost all the basic LIST operations.

iii. Implementing QUEUES on a circular array is more efficient than implementing QUEUES on a linear array with two indices.

iv. Last-in-first-out type of computations are efficiently supported by QUEUES.

A. (ii) and (iii) are true

B. (i) and (ii) are true

C. (iii) and (iv) are true

D. (ii) and (iv) are true

Question 2

Which of the following permutations can be obtained in the output (in the same order) using a stack assuming that the input is the sequence 1, 2, 3, 4, 5 in that order?

a) 1, 2, 4, 5, 3
b) 4, 5, 3, 1, 2
c) 3, 5, 4, 2, 1
d) 1, 2, 5, 3, 4

Question 3 GATE-2014

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*)?

A. A queue cannot be implemented using this stack.

B. A queue can be implemented where $ENQUEUE$ takes a single instruction and $DEQUEUE$ takes a 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.

Question 4

How many valid and invalid stack permutations are there with a sequence of 6 inputs?

Happy Learning