GATE

CS & DA

Data Structure through Python

Queues and Hash Tables

DPP: 1

Q1 In queue, Deletion is happens at the_______end.
(A) Rear
(B) Front
(C) Either rear or front
(D) More than one of the above
(E) None of the above

Q2 Carefully observe the below given queue

REAR	22	57	96	68	19	15	28	FRONT

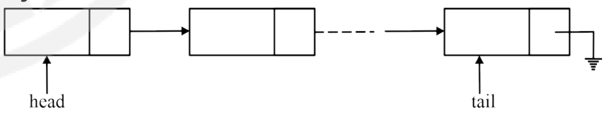
How many dequeue operations needed to delete the largest element from the queue?
(A) 1
(B) 3
(C) 5
(D) More than one of the above
(E) None of the above

Q3 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)?
(A) Both operations can be performed in 0(1) time
(B) At most one operation can be performed in 0(1) time but the worst case time for the other operation will be Ω(n)
(C) The worst-case time complexity for both operations will be Ω(n)
(D) Worst case time complexity for both operations will be Ω (log n)

Q4 A priority queue Q is used to implement a stack that stores characters. PUSH (C) is implemented 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

Q5 The best data structure to check whether an arithmetic expression has balanced parentheses is a
(A) Queue (B) Stack
(C) Tree (D) List

Q6 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 one of the following is the time complexity of the most time-efficient implementation of enqueue and dequeue, respectively, for this data structure?
(A) Θ(1), Θ1)
(B) Θ(1), Θ(n)
(C) Θ(n), Θ(1)
(D) Θ(n), Θ(n)

Answer Key

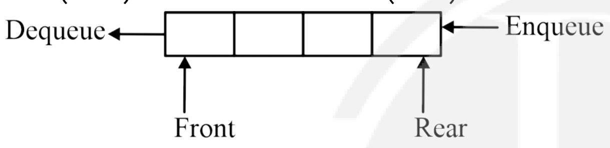
Q1	(B)
Q2	(C)
Q3	(A)
Q4	(A)
Q5	(B)
Q6	(B)

Hints & Solutions

Q1 Text Solution:

Queue:

- The queue is an abstract data structure, somewhat similar to Stacks.
- Insertion in a queue happens at the rear end and deletion happens at the front end.
- Unlike stacks, a queue is open at both ends. One end is always used to insert data (en queue) and the other is used to remove data (dequeue). Queue follows the First-In-First-Out methodology, i.e., the data item stored first will be accessed first.
- Queue follows the principle of First In First Out (FIFO) and Last In Last Out (LILO)



Hence the correct answer is front.

Q2 Text Solution:

- Insertion and deletion in queues take place from the opposite ends of the list.
- The insertion takes place at the rear of the list and the deletion takes place from the front of the list.
- Deletion takes place at the FRONT of the queue.

Analysis:-

96 is the largest element in the given queue.

1st deque operation:

REAR	22	57	96	68	19	15	FRONT

28 is deleted

2nd deque operation:

REAR	22	57	96	68	19	FRONT

15 is deleted

3rd operation:

REAR	22	57	96	63	FRONT

19 is deleted

4th operation:

REAR	22	57	96	FRONT

68 is deleted

5th operation:

REAR	22	57	FRONT

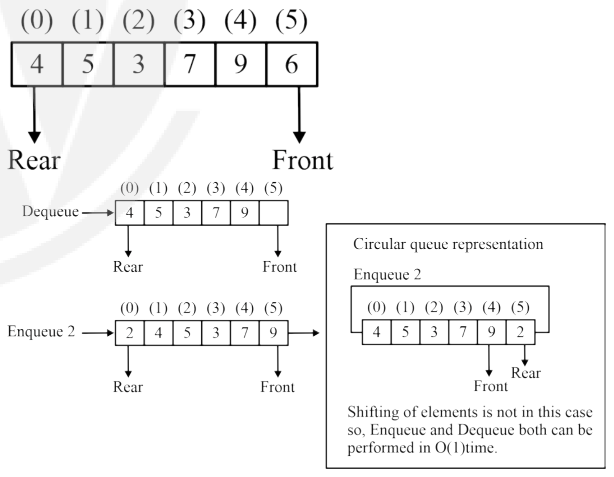
96 is deleted

Therefore, the number of deque operation needed to delete 96 is 5.

Q3 Text Solution:

When we consider a normal implementation of the queue using an array, in that case for enqueue and dequeue operation, we have to shift every other element. In this, every time we remove the item from the start of the queue, all of the rest of the items in the queue move down by one to fill the space made by the removal of other items.

But if we consider the circular array implementation of a queue, in this case, both enqueue and dequeue can be performed in 0(1) time.


(0) (1) (2) (3) (4) (5)
4 5 3 7 9 6
Rear
Front
Dequeue
4 5 3 7 9
Rear
Front
Enqueue 2
2 4 5 3 7 9
Rear
Front
Circular queue representation
Enqueue 2
4 5 3 7 9 2
Front
Rear
Shifting of elements is not in this case so, Enqueue and Dequeue both can be performed in O(1)time.

Q4 Text Solution:

Non-Increasing Order:

- By choosing keys in correct non-increasing order (decreasing order), each subsequent element pushed onto the stack will have a smaller key than the one before it.
- This ensures that when **POP** (implemented as **DELETEMIN(Q)**) is called, it removes the element with the smallest key, which

corresponds to the most recently pushed element.
- If keys were chosen in increasing order, **POP** would remove the oldest (first pushed) element, which would violate the stack's LIFO principle.

Q5 Text Solution:

A stack is the best data structure to check whether an arithmetic expression has balanced parentheses due to its Last In, First Out (LIFO) nature, which aligns perfectly with the problem requirements.

Q6 Text Solution:

New node to be inserted is P.

```
Enqueued {
P->Data=Data
P->Next=Head
Head=P
}
```

Time Complexity =O(1) Because only pointer manipulation is involved which takes constant time.

Delete Tail

```
Dequeued {
    temp=head
   While(temp->Next->Next!=NULL)
tenp=temp->next
temp->next=NULL
tail=terap
}
```

Time Complexity = Time for Traversing list, free the last node and keep track of Tail pointer = O(n)

Answer is B