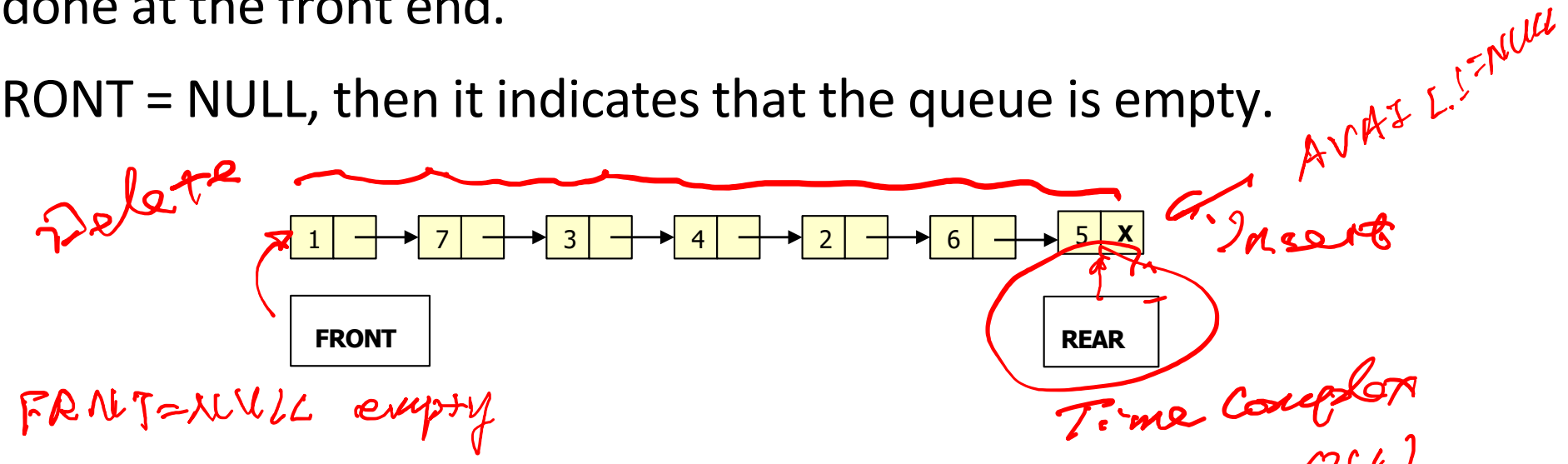
Linked Representation of Queues

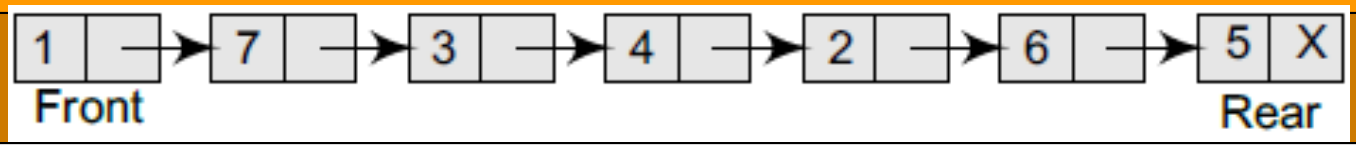
- In a linked queue, every element has two parts: one that stores data and the other that stores the address of the next element.
- The START pointer of the linked list is used as FRONT.
- We will also use another pointer called REAR which will store the address of the last element in the queue.
- All insertions will be done at the rear end and all the deletions will be done at the front end.
- If FRONT = NULL, then it indicates that the queue is empty.

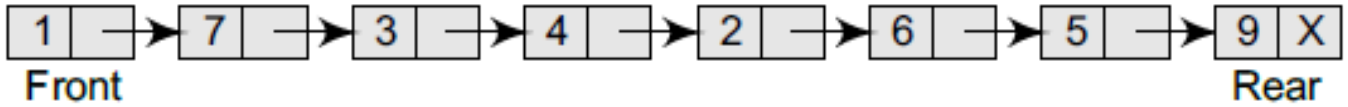


Inserting an Element in a Linked Queue

```
Algorithm to insert an element in a linked queue

Step 1: Allocate memory for the new node as PTR
Step 2: SET PTR->DATA = VAL
Step 3: IF FRONT = NULL, THEN
                SET FRONT = REAR = PTR
                SET REAR->NEXT = NULL
        Else
                SET REAR->NEXT = PTR
                SET REAR = PTR
                SET REAR->NEXT = NULL
        [END OF IF]
Step 4: END
```



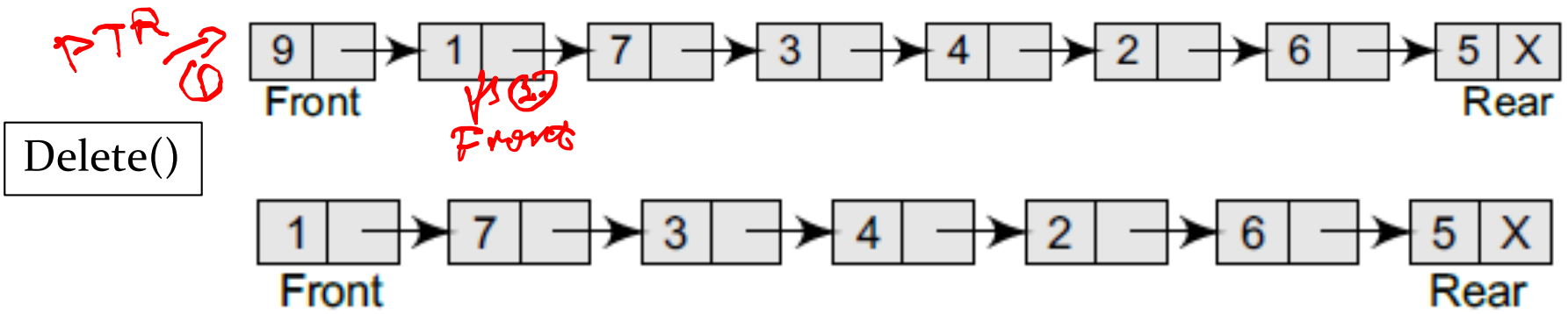


Insert(9)

Deleting an Element from a Linked Queue

```
Algorithm to delete an element from a linked queue

Step 1: IF FRONT = NULL, THEN
                WRITE "Underflow"
                GOTO Step 5
        [END OF IF]
Step 2: SET PTR = FRONT
Step 3: SET FRONT = FRONT->NEXT
Step 4: FREE PTR
Step 5: END
```

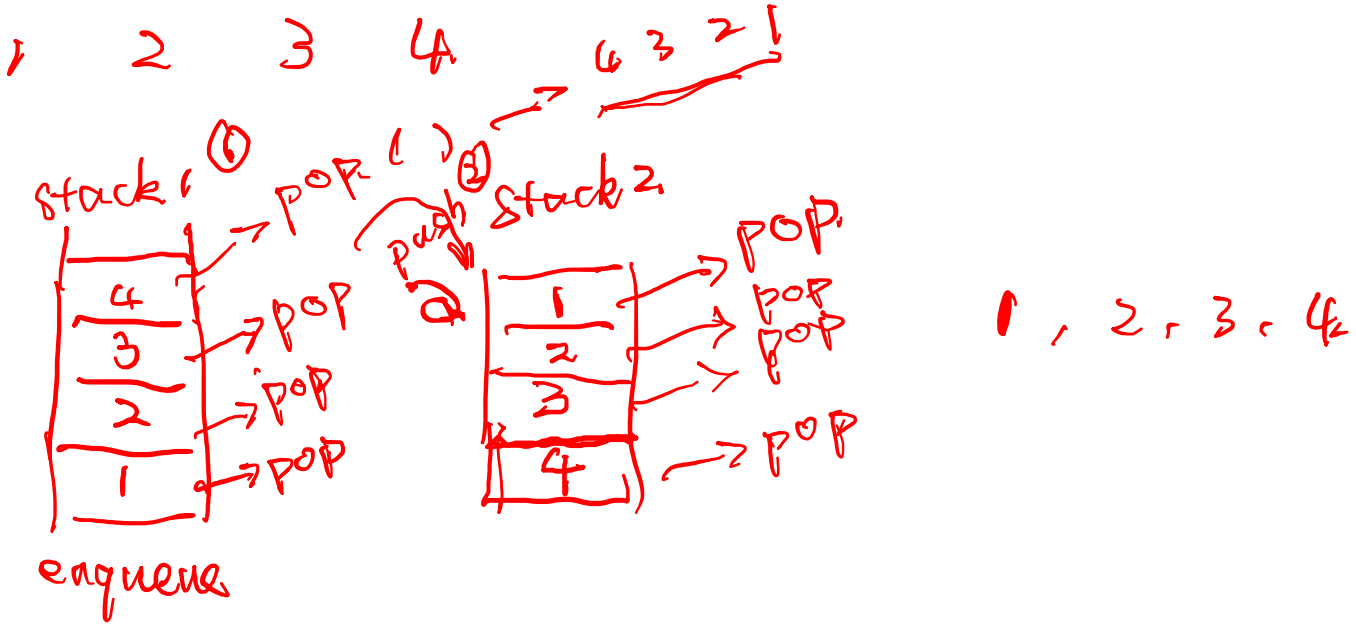


Time Complexity – Worst Case

Insertion/enqueue	O(1)
Deletion/dequeue	O(1)

Both linked list & Array:

How to implement a queue using two stacks



enqueue: 1 2 3 4 push() into stack1

dequeue: output: 1 2 3 4

How to implement a queue using two stacks

Method

In enqueue operation, the new element is entered at the top of stack1

In dequeue operation, if stack2 is empty then all the elements are moved to stack2 and finally top of stack2 is returned

enQueue(q, x)

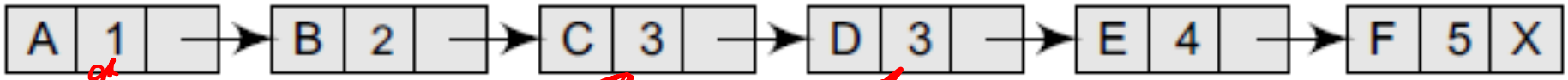
1) Push x to stack1 (assuming size of stacks is unlimited).

deQueue(q)

1) If both stacks are empty then error.
2) If stack2 is empty
   While stack1 is not empty, push everything from stack1 to stack2.
3) Pop the element from stack2 and return it.

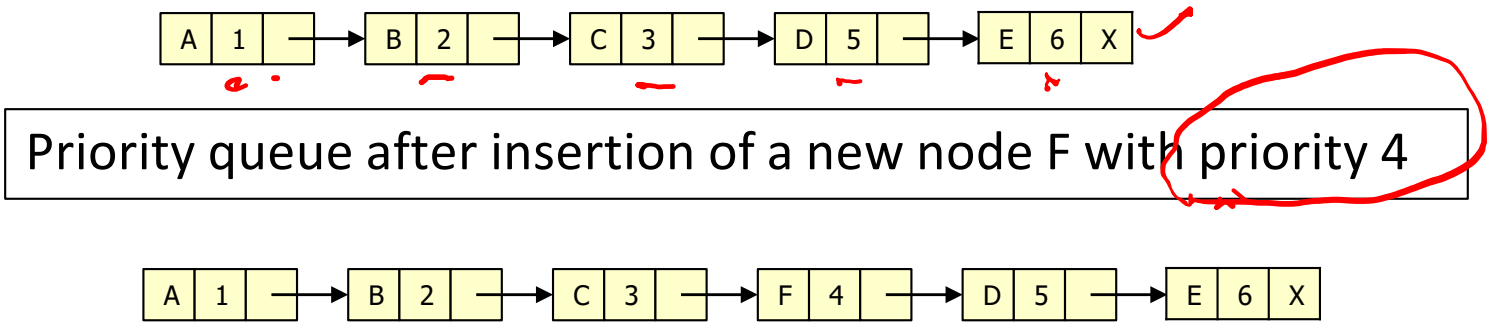
Priority Queues

- A priority queue is a queue in which each element is assigned a priority.
- The priority of elements is used to determine the order in which these elements will be processed.
- The general rule of processing elements of a priority queue can be given as:
  - An element with higher priority is processed before an element with lower priority
  - Two elements with same priority are processed on a first come first served (FCFS) basis
- Priority queues are widely used in operating systems to execute the highest priority process first.
- Priority queues can be represented using arrays or linked lists.



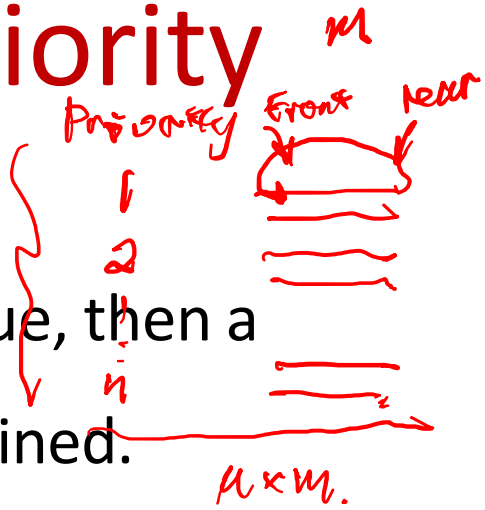
Linked Representation of Priority Queues

- When a priority queue is implemented using a linked list, then every node of the list contains three parts: (1) the information or data part, (ii) the priority number of the element, (iii) and address of the next element.
- If we are using a sorted linked list, then an element having higher priority will precede the element with lower priority.



- On deletion, the first node will be deleted, and the data processed.

Array Representation of Priority Queues

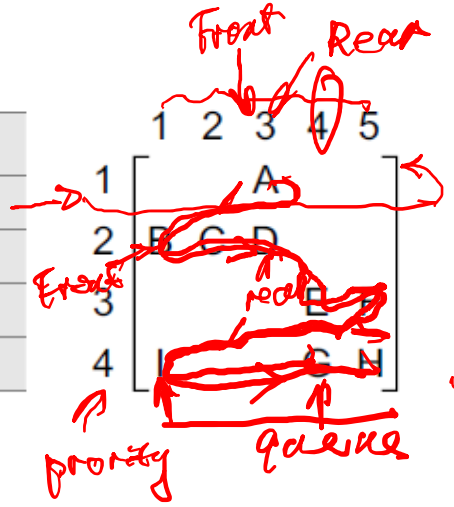


- When arrays are used to implement a priority queue, then a separate queue for each priority number is maintained.
- Each of these queues will be implemented using circular arrays or circular queues. Every individual queue will have its own FRONT and REAR pointers.
- We can use a two-dimensional array for this purpose where each queue will be allocated same amount of space.
- Given the front and rear values of each queue, a two-dimensional matrix can be formed.

Array Representation of Priority Queues

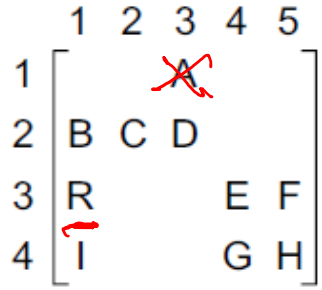
- Insertion – insert R with priority 3

Priority	FRONT	REAR
1	3	3
2	1	3
3	4	5
	4	1



=>

FRONT	REAR
3	3
1	3
4	1
4	1



- Deletion – find the first non-empty queue, delete the first element, and process the element
  - In the above example, the queue with priority 1 has a front element, A, which would be deleted.

Homework 2

📌 **Content:** Stack & Queue

📖 Exercises from the 1st textbook

⌛ **Deadline:** 11:59 PM, March 4 (Tuesday), 2025

⚠️ **Late Submission Penalty:** 10% per day

1st Midterm Exam

- **Midterm Exam**
- March 10th (Monday) in class
- Chapters: 2 (complexity Analysis), 6(LinkedList), 7(Stack)

- **Review and quiz**
- March 7th (Friday) in class
- Chapters 2, 6, 7

- **Lab on stack & queue**
- The week of March 3rd