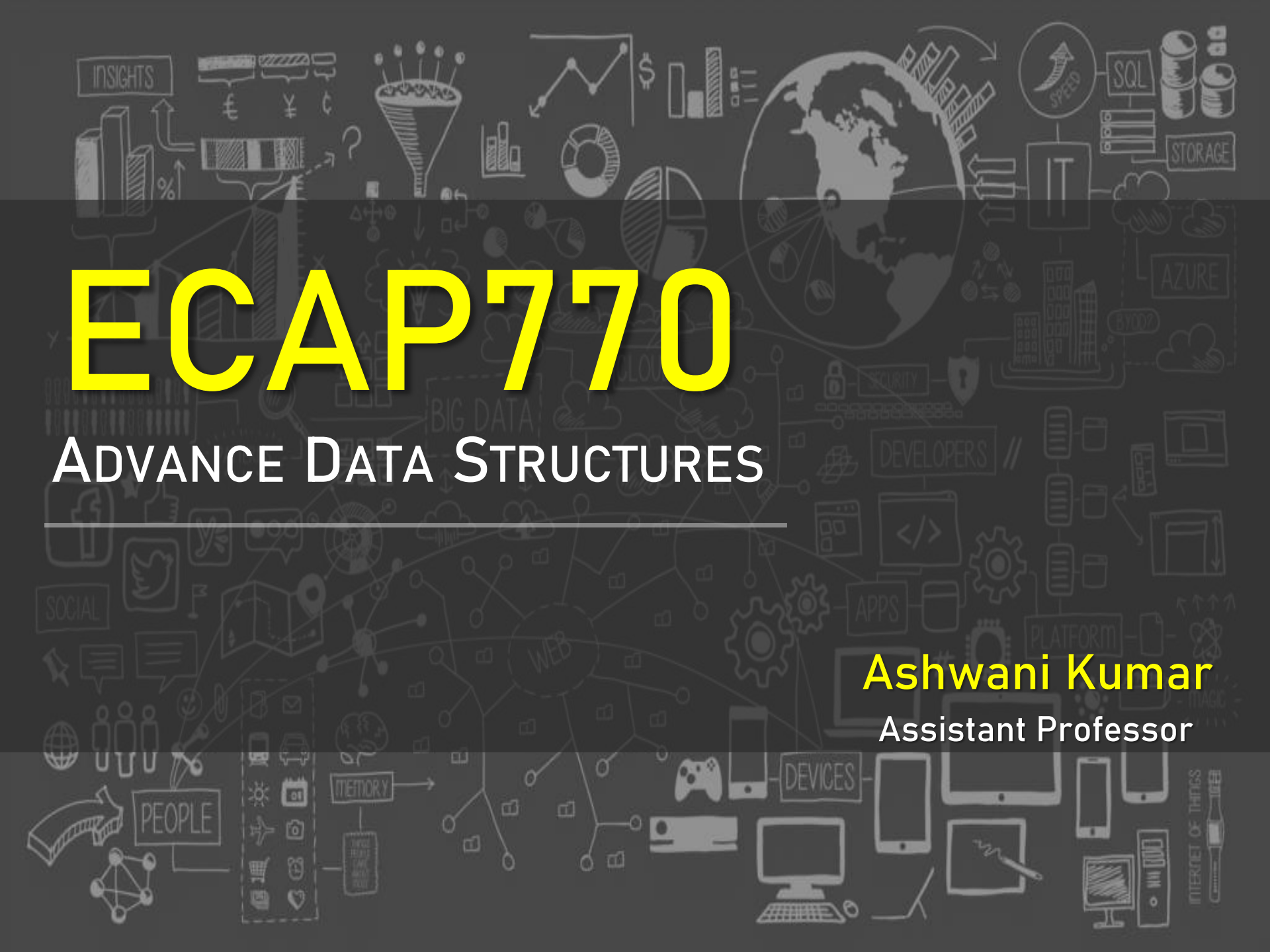


ECAP770

Advance Data Structures

Ashwani Kumar

Assistant Professor

Learning Outcomes

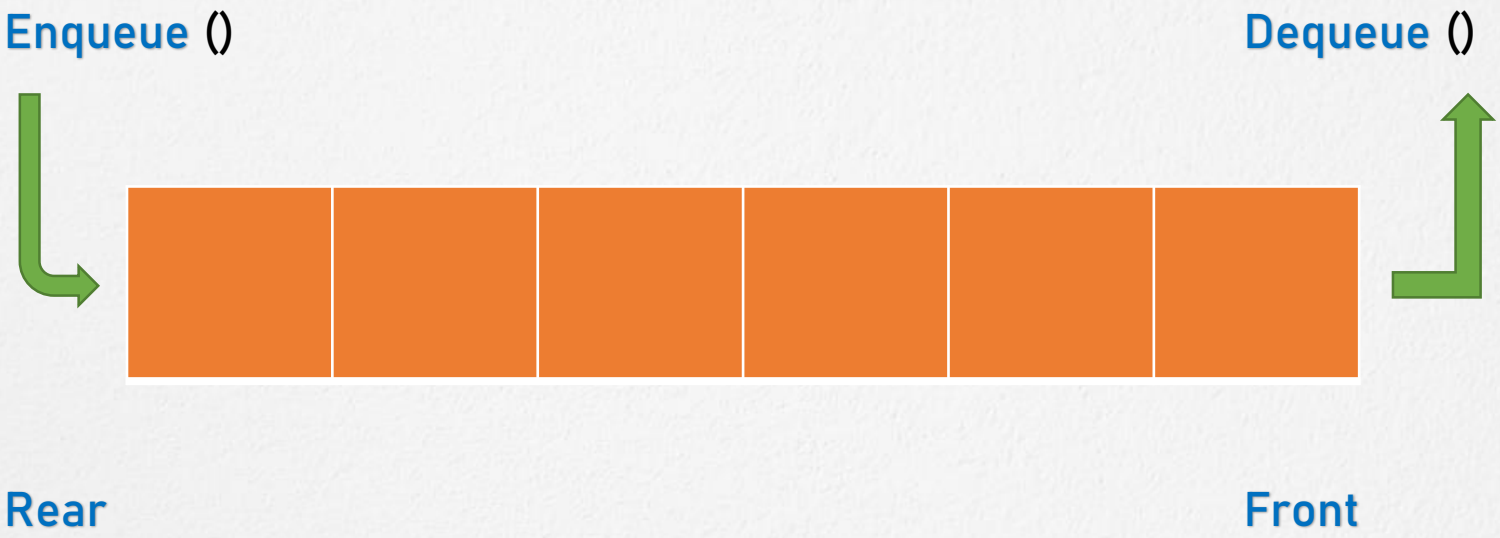
After this lecture, you will be able to

- Understand implementation of queue data structure

Queue

- Queue is a linear data structure and also called abstract data structure
- Queue follow First In First Out (FIFO) method
- It enables insert operations to be performed at one end called REAR and delete operations to be performed at another end called FRONT.

Queue representation



Implementation of queues

Queue can be implemented using:

- Array
- Linked List
- Stack

Queue implementation Using Array

- A queue can be implemented using one dimensional array.
- Queue implemented using array stores only fixed number of data values.
- Two variables front and rear, that are implemented in queue. Front and rear variables point to the position from where insertions and deletions are performed in a queue.

Queue implementation Using Array

- Initially both front and rear are set to -1.
- For insert a new value into the queue, increment rear value by one and then insert at that position.
- For delete a value from the queue, then delete the element which is at front position and increment front value by one.

Enqueue operation

- Enqueue() function is used to insert a new element into the queue
- In a queue, the new element is always inserted at rear position.
- The enQueue() function takes one integer value as a parameter and inserts that value into the queue.

Algorithm: Enqueue operation

Step 1: IF REAR = MAX - 1

Write OVERFLOW

Go to step

[END OF IF]

Step 2: IF FRONT = -1 and REAR = -1

SET FRONT = REAR = 0

ELSE

SET REAR = REAR + 1

[END OF IF]

Step 3: Set QUEUE[REAR] = NUM

Step 4: EXIT

Dequeue operation

- Dequeue() is a function used to delete an element from the queue.
- In a queue, the element is always deleted from front position.
- The Dequeue() function does not take any value as parameter.

Algorithm: Dequeue operation

- Step 1: IF FRONT = -1 or FRONT > REAR
- Write UNDERFLOW
- ELSE
- SET VAL = QUEUE[FRONT]
- SET FRONT = FRONT + 1
- [END OF IF]
- Step 2: EXIT

display() - Displays the elements of a Queue

- Step 1 - Check whether queue is EMPTY. (front == rear)
- Step 2 - If it is EMPTY, then display "Queue is EMPTY!!!" and terminate the function.
- Step 3 - If it is NOT EMPTY, then define an integer variable 'i' and set 'i = front+1'.
- Step 4 - Display 'queue[i]' value and increment 'i' value by one (i++). Repeat the same until 'i' value reaches to rear (i <= rear)

Disadvantages of array implementation

- Memory wastage
- Array size declaration

Implementation of Queue using Linked List

- Due to the drawbacks of array. The array implementation can not be used for the large scale applications where the queues are implemented.
- The alternative of array implementation is linked list implementation of queue.

Implementation of Queue using Linked List

- In a linked queue, each node of the queue consists of two parts i.e. data part and the link part.
- Each element of the queue points to its immediate next element in the memory.

Implementation of Queue using Linked List

- In the linked queue, there are two pointers maintained in the memory i.e. front pointer and rear pointer.
- The front pointer contains the address of the starting element of the queue while the rear pointer contains the address of the last element of the queue.

Linked Queue: Insert operation

- There can be the two scenario of inserting this new node ptr into the linked queue.
- In the first scenario, we insert element into an empty queue. In this case, the condition front = NULL becomes true.
- In the second case, the queue contains more than one element. The condition front = NULL becomes false.

Algorithm

Step 1: Allocate the space for the new node PTR

Step 2: SET PTR -> DATA = VAL

Step 3: IF FRONT = NULL

SET FRONT = REAR = PTR

SET FRONT -> NEXT = REAR -> NEXT = NULL

ELSE

SET REAR -> NEXT = PTR

SET REAR = PTR

SET REAR -> NEXT = NULL

[END OF IF]

Step 4: END

Linked Queue: Delete operation

- Deletion operation removes the element that is first inserted among all the queue elements
- The condition front == NULL becomes true if the list is empty.
- Otherwise, we will delete the element that is pointed by the pointer front.

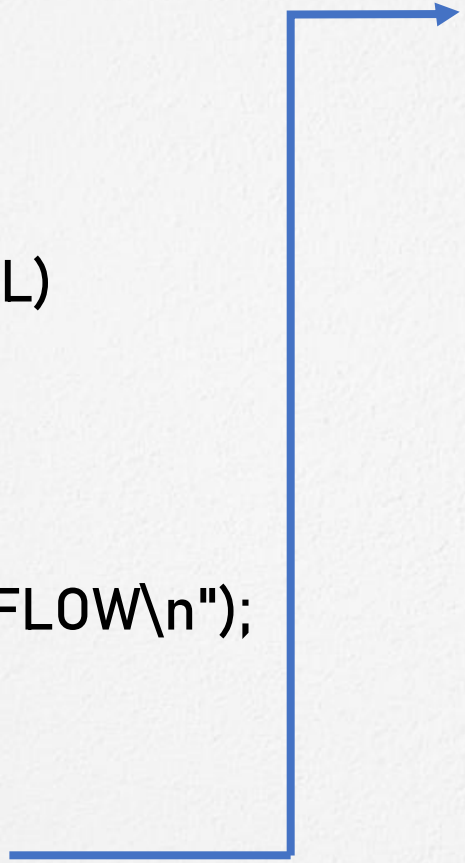
Algorithm

- Step 1: IF FRONT = NULL
- Write " Underflow "
- Go to Step 5
- [END OF IF]
- Step 2: SET PTR = FRONT
- Step 3: SET FRONT = FRONT -> NEXT
- Step 4: FREE PTR
- Step 5: END

Linked Queue: Delete operation

```
void delete (struct node
*ptr)
{
    if(front == NULL)
    {

printf("\nUNDERFLOW\n");
        return;
    }
else
    {
        ptr = front;
        front = front -> next;
        free(ptr);
    }
}
```



That's all for now...