K.R. MANGALAM UNIVERSITY

THE COMPLETE WORLD OF EDUCATION

Data Structure

ENCS205

School of Engineering & Technology (SOET)
K.R. MANGALAM University

UNIT-2

Session 30: PRIORITY QUEUES Concept

Recap

Circular queues: Utilize a fixed-size array with a circular design for efficient data management.

Operations: Enqueue (insertion), Dequeue (deletion), isFull, isEmpty, Front, Rear.

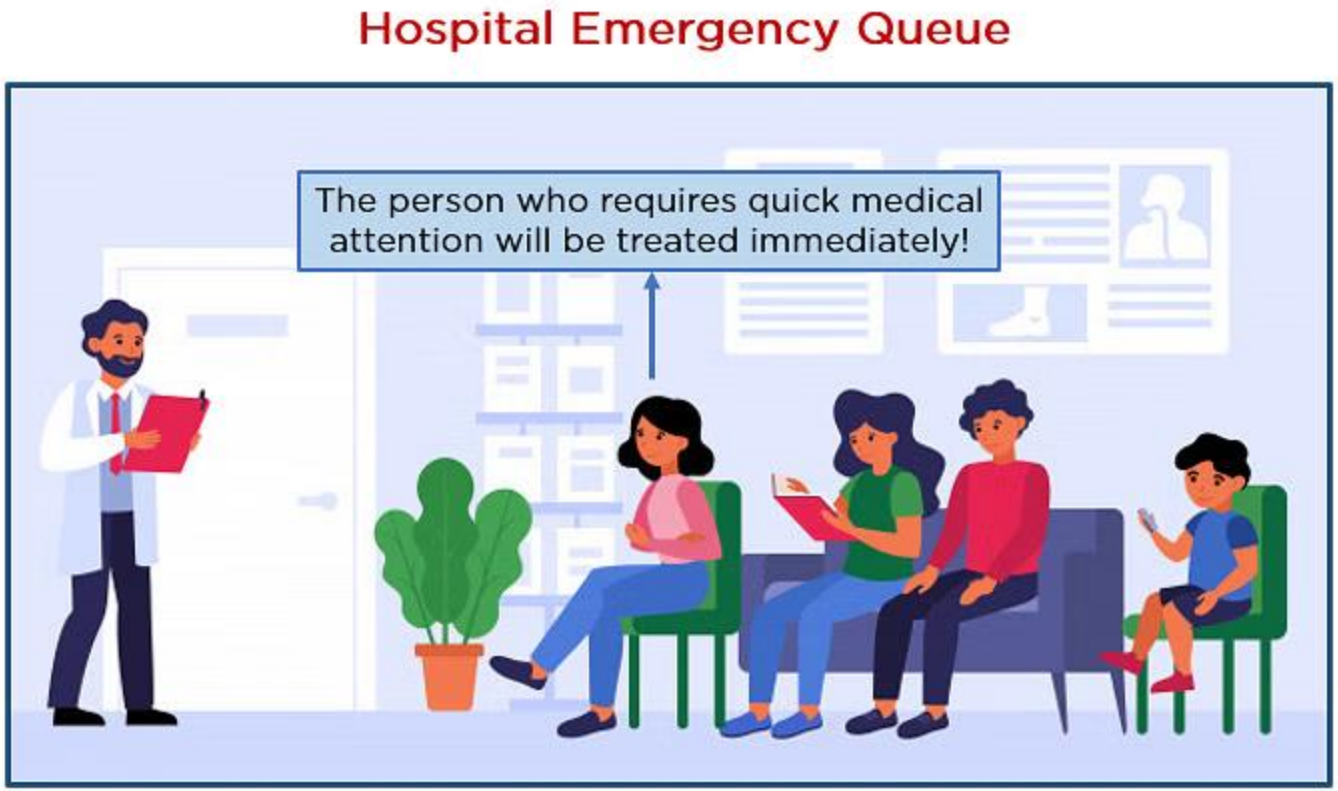
Insertion Algorithm: Check if full, increment rear pointer, insert element, handle wraparound.

Deletion Algorithm: Check if empty, increment front pointer, remove element, manage wraparound.

Sessions 30 Outlook

- Priority Queues
- Insertion and Deletion Algorithm
- Implementations
- Applications

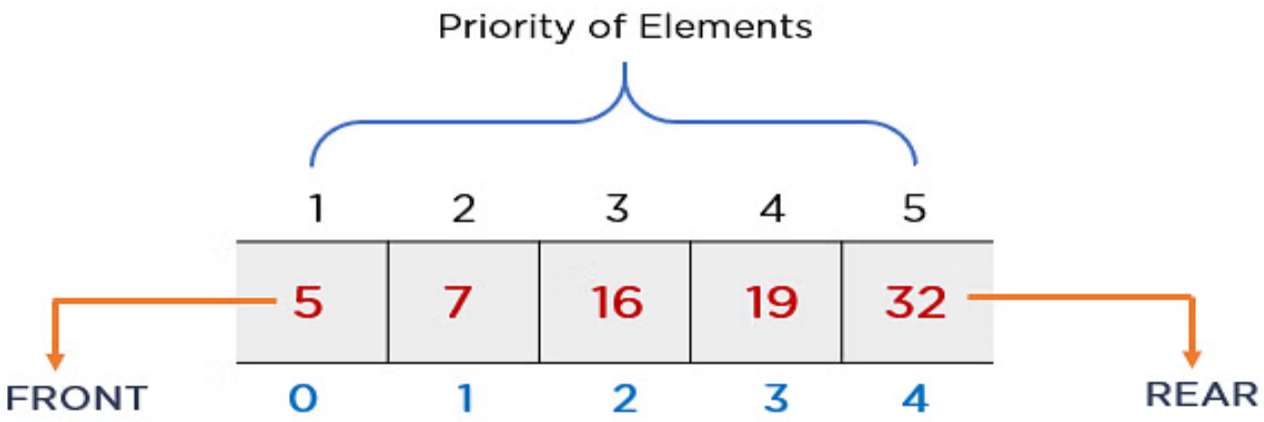
Priority Queues



https://www.simplilearn.com/tutorials/data-structure-tutorial/priority-queue-in-data-structure

Priority Queues

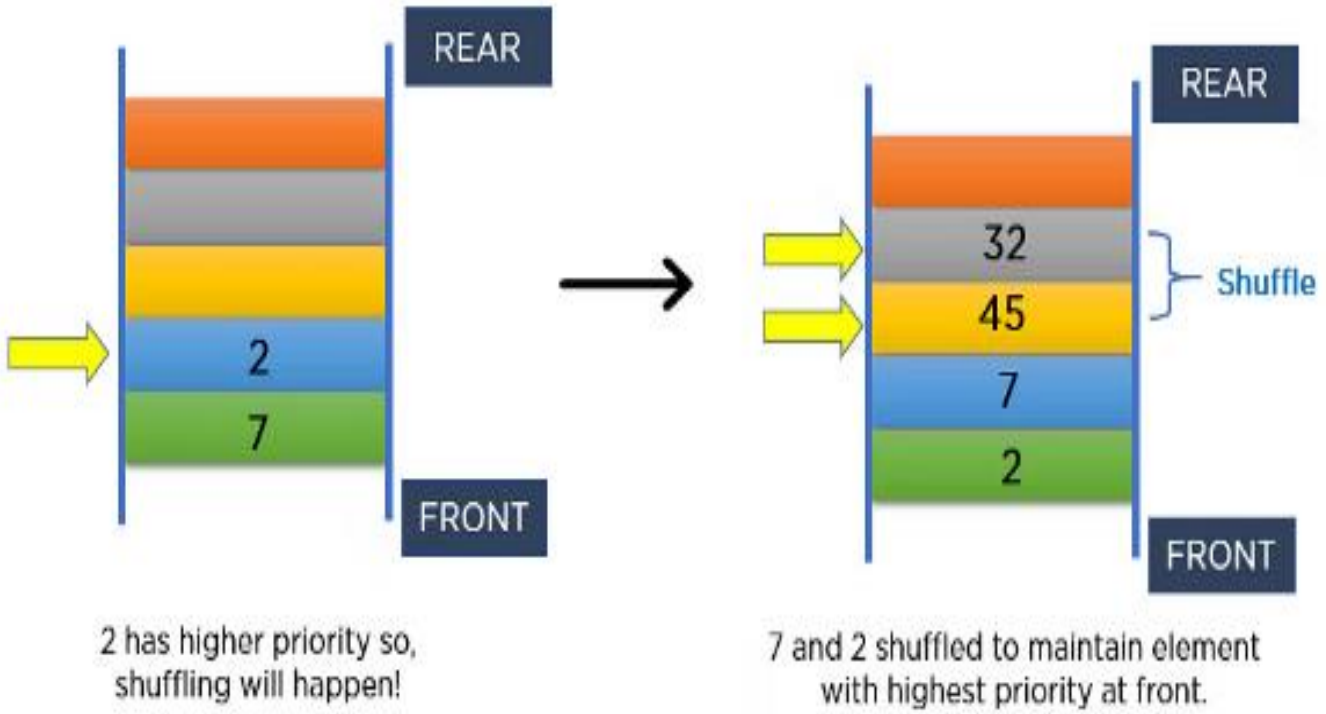
The priority of elements determines the order of removal in a queue, i.e., the element with higher priority will leave the queue first, whereas the element with the lowest priority at last



https://www.simplilearn.com/tutorials/data-structure-tutorial/priority-queue-in-data-structure

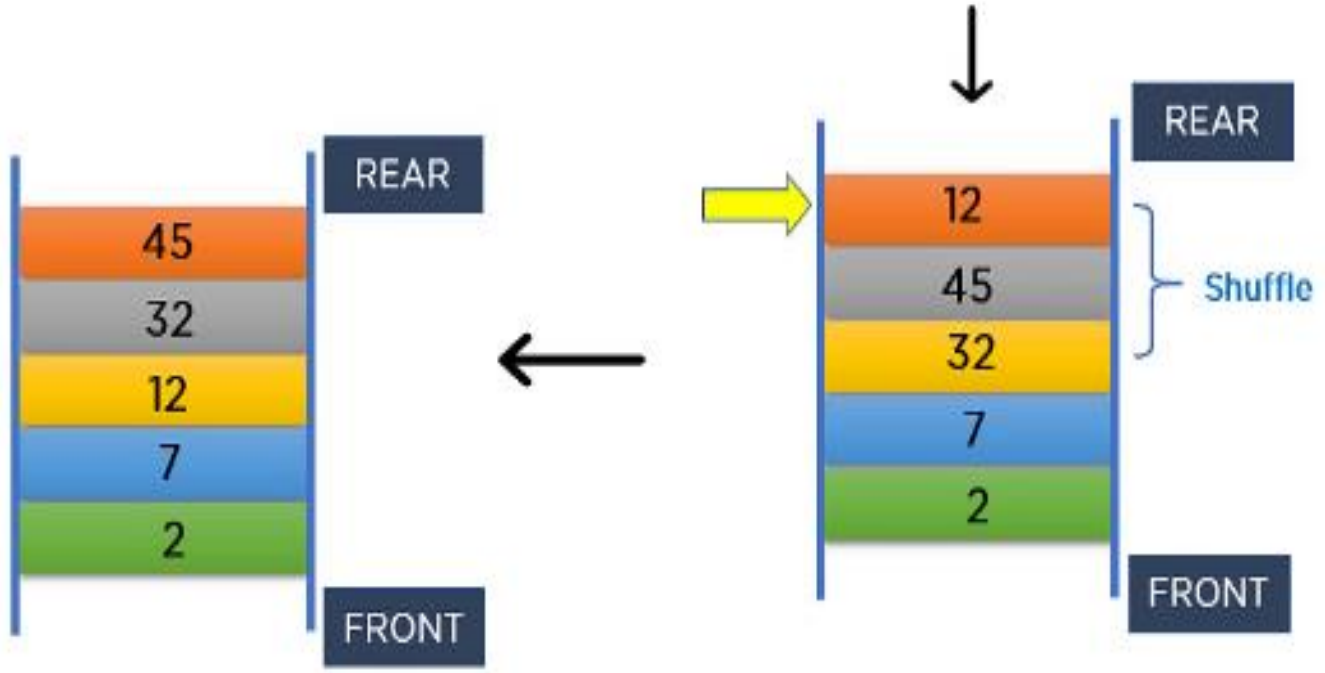
Priority Queues

Priority Queues



https://www.simplilearn.com/tutorials/data-structure-tutorial/priority-queue-in-data-structure

Priority Queues



https://www.simplilearn.com/tutorials/data-structure-tutorial/priority-queue-in-data-structure

Priority Queue Applications

- Operating systems
- Time-activated events
- Managing responsibilities
- Sorting
- Djikstra's Algorithm
- Prim's Algorithm

Implementation of Priority Queue

- Using Linked List
- Using Binary Heap
- Using Array
- Binary Search Tree

Operation on Priority Queue

Insertion:

Insertion operation is used to insert a new element into the queue. Whenever a new element is inserted,

- The new element is moved to an empty slot top to bottom and left to right.
- If the inserted element is not in the correct position. It will be compared with the parent element.
- And if the element is not in the correct order, then elements are swapped until the inserted element gets to the correct position.

Operation on Priority Queue

Algorithm:

if (the queue is empty)

Create a node

else //when node is already present

Insert the new node at the end of the queue (last node from left to right) heapify the array

Operation on Priority Queue

Deletion:

Deletion operation is used to remove any element from the queue, For deleting any element,

- Choose the element which needs to be deleted.
- Swap it with the last element.
- Remove the last element
- Heapify the array.

Operation on Priority Queue

Deletion:

Algorithm:

if (node== leaf node)

delete node

else //when node is not leaf node

Swap node with leaf node

Delete the leaf node

heapify the array

Implementation of Priority Queue Using Array:

The basic implementation using array is:

```
struct node
{
   int node:
   int priority:
}
```

Implementation of Priority Queue Using Array:

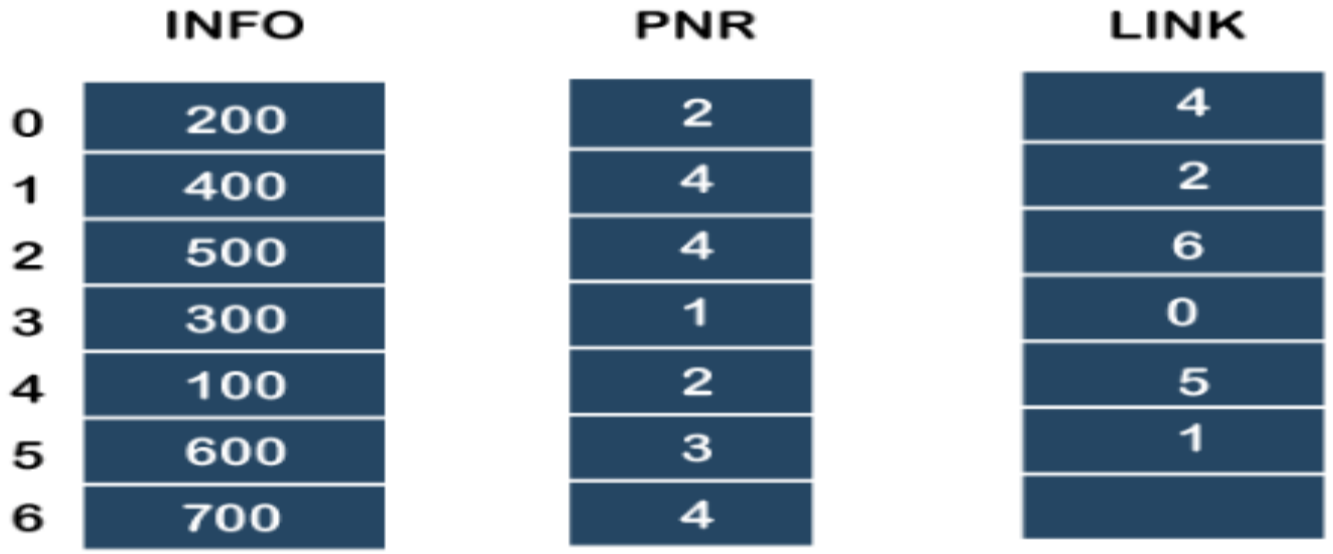
Operations in array:

enqueue(): It is used to insert new element into the priority queue.

dequeue(): It removes an element with the highest priority from the priority queue.

peek(): It is used to determine the highest priority element in the queue without removing it.

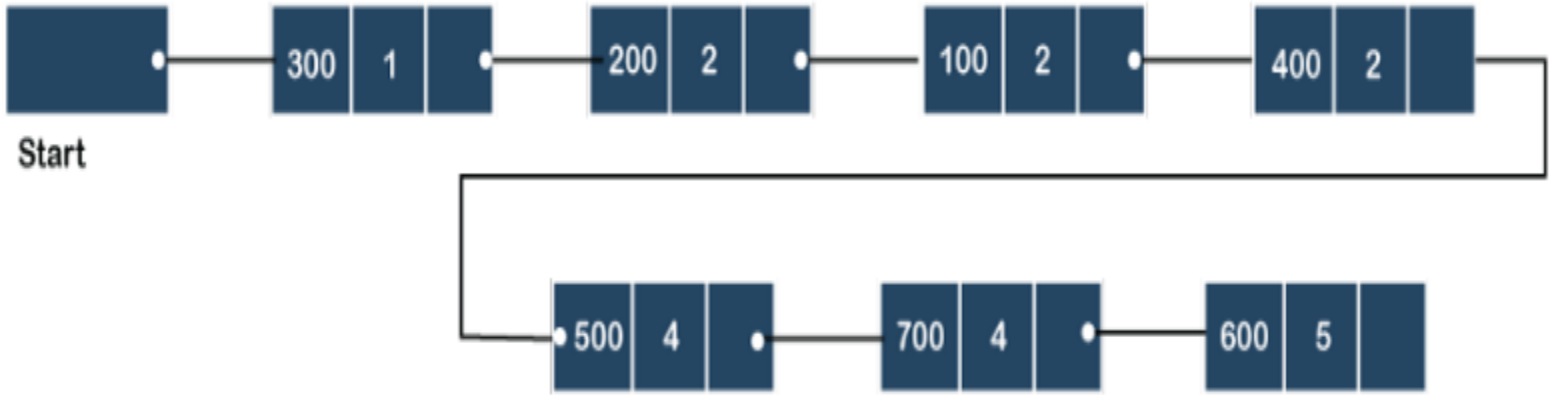
Implementation of Priority Queue



https://www.javatpoint.com/ds-priority-queue

Implementation of Priority Queue

Using link list



https://www.javatpoint.com/ds-priority-queue

Algorithm for Insertion at rear end

Step-1: [Check for overflow]

```
if(rear==MAX)
        Print("Queue is Overflow”);
        return;
```

Step-2: [Insert Element]

```
else
        rear=rear+1;
        q[rear]=no;
[Set rear and front pointer]
if rear=0
        rear=1;
if front=0
front=1;
```

Step-3: return

Algorithm for Insertion at front end

```
Step-1 :  [Check for the front position]
                if(front<=1)
                        Print("Cannot add item at the front”);
                        return;
Step-2 :  [Insert at front]
                else
                        front=front-1;
                        q[front]=no;
Step-3  : Return
```

Algorithm for Deletion from front end

Step-1 [Check for front pointer]

if front=0

print(" Queue is Underflow”);

return;

Step-2 [Perform deletion]

else no=q[front];

print(“Deleted element is”,no);

[Set front and rear pointer]

if front=rear

front=0;

rear=0;

else front=front+1;

Step-3 : Return

Algorithm for Deletion from rear end

Step-1 : [Check for the rear pointer]

if rear=0

print("Cannot delete value at rear end");

return;

Step-2: [perform deletion]

else no=q[rear];

[Check for the front and rear pointer]

if front= rear

front=0;

rear=0;

else rear=rear-1;

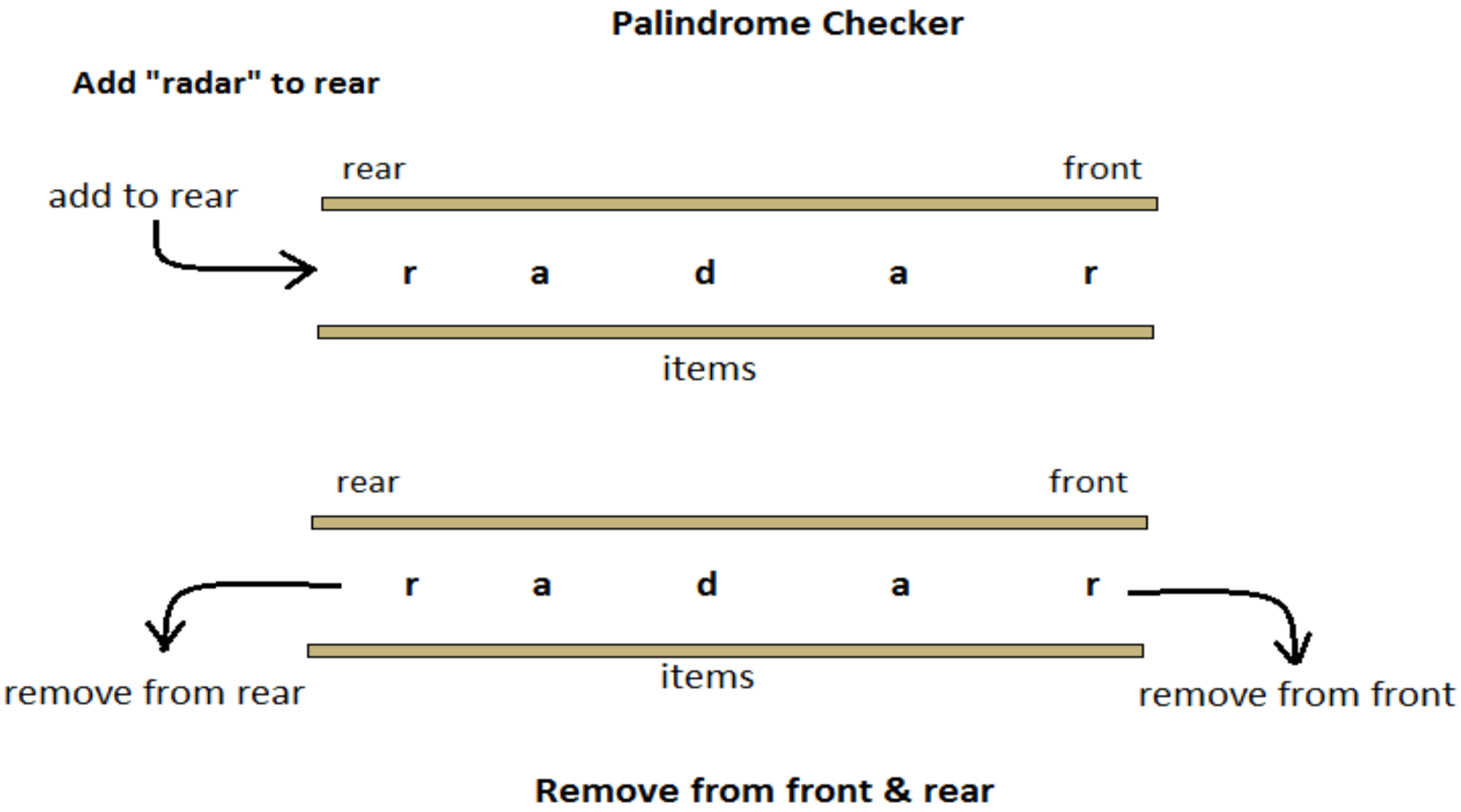
print("Deleted element is",no);

Step-3 : Return

Complexity

- Insertion or deletion in the middle is O(n)
- The time complexity of random access by index is O(1)
- Time complexity of all enque (insert)/deque(delete) operations is O(1)

Applications



Applications

A-Steal job scheduling algorithm

The A-Steal algorithm implements task scheduling for several processors(multiprocessor scheduling).

- The processor gets the first element from the deque.
- When one of the processor completes execution of its own threads it can steal a thread from another processor.
- It gets the last element from the deque of another processor and executes it.

Undo-Redo operations in Software applications.

Code

```
// C++ program to demonstrate the use of priority queue
#include <iostream>
#include <queue>
using namespace std;
// driver code
int main()
{
        int arr[6] = { 10, 2, 4, 8, 6, 9 };
```

Code

```
        // defining priority queue
        priority_queue<int> pq;
// printing array
        cout << "Array: ";
        for (auto i : arr) {
                cout << i << ' ';
        }
```

Code

```
        cout << endl;
        // pushing array sequentially one by one
        for (int i = 0; i < 6; i++) {
                pq.push(arr[i]);
        }
// printing priority queue
        cout << "Priority Queue: ";
        while (!pq.empty()) {
                cout << pq.top() << ' ';
                pq.pop();}
return 0;
```

Find the Complete code here.

Points to Remember

- A queue is a FIFO data structure in which the element that is inserted first is the first one to be taken out.
- The elements in a queue are added at one end called the rear and removed from the other end called the front.
- In the computer ’s memory, queues can be implemented using both arrays and linked lists.

Points to Remember

- The storage requirement of linked representation of queue with n elements is O(n) and the typical time requirement for operations is O(1).
- In a circular queue, the first index comes after the last index.
- Multiple queues means to have more than one queue in the same array of sufficient size.

Points to Remember

- A queue is a FIFO data structure in which the element that is inserted first is the first one to be taken out.
- The elements in a queue are added at one end called the rear and removed from the other end called the front.
- In the computer ’s memory, queues can be implemented using both arrays and linked lists.

Points to Remember

- A deque is a list in which elements can be inserted or deleted at either end. It is also known as a head tail linked list because elements can be added to or removed from the front (head) or back (tail). However, no element can be added or deleted from the middle. In the computer's memory, a deque is implemented using either a circular array or a circular doubly linked list.
- In an input restricted deque, insertions can be done only at one end, while deletions can be done from both the ends. In an output restricted deque, deletions can be done only at one end, while insertions can be done at both the ends.

Points to Remember

- A priority queue is a data structure in which each element is assigned a priority. The priority of the element will be used to determine the order in which the elements will be processed.
- When a priority queue is implemented using a linked list, then every node of the list will have three parts:

(a) the information or data part,

(b) the priority number of the element, and

(c) the address of the next element

Review

- Definition: A queue is a linear data structure following the FIFO (First In, First Out) principle.
- Operations: Enqueue (addition), Dequeue (removal), isEmpty, isFull, Front, Rear.
- Implementations: Array-based, Linked List-based.
- Applications: Task scheduling, BFS (Breadth-First Search), Printer queues.
- Advantages: Efficient FIFO management, simple implementation.
- Limitations: Fixed size (in array-based implementation), inefficiency in random access.
- Conclusion: Queues are essential for managing data in a sequential manner, crucial in various computing applications for efficient data processing.

Review

- Definition: A queue is a linear data structure following the FIFO (First In, First Out) principle.
- Operations: Enqueue (addition), Dequeue (removal), isEmpty, isFull, Front, Rear.
- Implementations: Array-based, Linked List-based.
- Applications: Task scheduling, BFS (Breadth-First Search), Printer queues.

Review

- Advantages: Efficient FIFO management, simple implementation.
- Limitations: Fixed size (in array-based implementation), inefficiency in random access.
- Conclusion: Queues are essential for managing data in a sequential manner, crucial in various computing applications for efficient data processing.


THANK
YOU