A priority queue is a type of queue that arranges elements based on their priority values. Elements with higher priority values are typically retrieved before elements with lower priority values.

In a priority queue, each element has a priority value associated with it. When you add an element to the queue, it is inserted in a position based on its priority value. For example, if you add an element with a high priority value to a priority queue, it may be inserted near the front of the queue, while an element with a low priority value may be inserted near the back.

There are several ways to implement a priority queue, including using an array, linked list, heap, or binary search tree. Each method has its own advantages and disadvantages, and the best choice will depend on the specific needs of your application.

Priority queues are often used in real-time systems, where the order in which elements are processed can have significant consequences. They are also used in algorithms to improve their

efficiencies, such as Dijkstra's algorithm for finding the shortest path in a graph and the A* search algorithm for pathfinding.

Properties of Priority Queue

So, a priority Queue is an extension of the queue with the following properties.

Every item has a priority associated with it.

An element with high priority is dequeued before an element with low priority.

If two elements have the same priority, they are served according to their order in the queue.

In the below priority queue, an element with a maximum ASCII value will have the highest priority. The elements with higher priority are served first.

How is Priority assigned to the elements in a Priority Queue?

In a priority queue, generally, the value of an element is considered for assigning the priority.

For example, the element with the highest value is assigned the highest priority and the element with the lowest value is assigned the lowest priority. The reverse case can also be used i.e., the element with the lowest value can be assigned the highest priority. Also, the priority can be assigned according to our needs.

Operations of a Priority Queue:

A typical priority queue supports the following operations:

1) Insertion in a Priority Queue

When a new element is inserted in a priority queue, it moves to the empty slot from top to bottom and left to right. However, if the element is not in the correct place then it will be compared

with the parent node. If the element is not in the correct order, the elements are swapped. The swapping process continues until all the elements are placed in the correct position.

2) Deletion in a Priority Queue

As you know that in a max heap, the maximum element is the root node. And it will remove the element which has maximum priority first. Thus, you remove the root node from the queue. This removal creates an empty slot, which will be further filled with new insertion. Then, it compares the newly inserted element with all the elements inside the queue to maintain the heap invariant.

3) Peek in a Priority Queue

This operation helps to return the maximum element from Max Heap or the minimum element from Min Heap without deleting the node from the priority queue.

Types of Priority Queue:

1) Ascending Order Priority Queue

As the name suggests, in ascending order priority queue, the element with a lower priority value is given a higher priority in the priority list. For example, if we have the following elements in a priority queue arranged in ascending order like 4,6,8,9,10. Here, 4 is the smallest number, therefore, it will get the highest priority in a priority queue and so when we dequeue from this type of priority queue, 4 will remove from the queue and dequeue returns 4.

2) Descending order Priority Queue

The root node is the maximum element in a max heap, as you may know. It will also remove the element with the highest priority first. As a result, the root node is removed from the queue. This deletion leaves an empty space, which will be filled with fresh insertions in the future. The heap invariant is then maintained by comparing the newly inserted element to all other entries in the queue.

Types of Priority Queues

Types of Priority Queues

Difference between Priority Queue and Normal Queue?

There is no priority attached to elements in a queue, the rule of first-in-first-out(FIFO) is implemented whereas, in a priority queue, the elements have a priority. The elements with higher priority are served first.

How to Implement Priority Queue?

Priority queue can be implemented using the following data structures:

Arrays

Linked list

Heap data structure

Binary search tree

Let’s discuss all these in detail.

1) Implement Priority Queue Using Array:

A simple implementation is to use an array of the following structure.

```
struct item {
    int item;
    int priority;
}
```

enqueue(): This function is used to insert new data into the queue.

dequeue(): This function removes the element with the highest priority from the queue.

peek()/top(): This function is used to get the highest priority element in the queue without removing it from the queue.

Arrays

enqueue()

dequeue()

peek()

Time Complexity

O(1)

O(n)

O(n)