Article

Priority Queue

By - Akshat Singh Gour

Priority Queue is an abstract data type that performs operations on data elements per their priority. To understand it better, first analyze the real-life scenario of a priority queue. The hospital emergency queue is an ideal real-life example of a priority queue. In this queue of patients, the patient with the most critical situation is the first in a queue, and the patient who doesn’t need immediate medical attention will be the last. In this queue, the priority depends on the medical condition of the patients.

Priority queues are a type of container adapters, specifically designed such that the first element of the queue is the greatest of all elements in the queue and elements are in nonincreasing order (hence we can see that each element of the queue has a priority {fixed order}). Following is an example to demonstrate the priority queue and its various methods.
It is similar to the ordinary queue in certain aspects but differs in the following ways:

1. **In a priority queue, every element in the queue is associated with some priority, but priority does not exist in a queue data structure.**
2. **The element with the highest priority in a priority queue will be removed first while queue follows the FIFO(First-In-First-Out) policy means the element which is inserted first will be deleted first.**
3. **If more than one element exists with the same priority, then the order of the element in a queue will be taken into consideration.**

In this tutorial, you will learn what priority queue is. Also, you will learn about it's implementations in Python, Java, C, and C++.

A priority queue is a special type of queue in which each element is associated with a priority value. And, elements are served on the basis of their priority. That is, higher priority elements are served first.

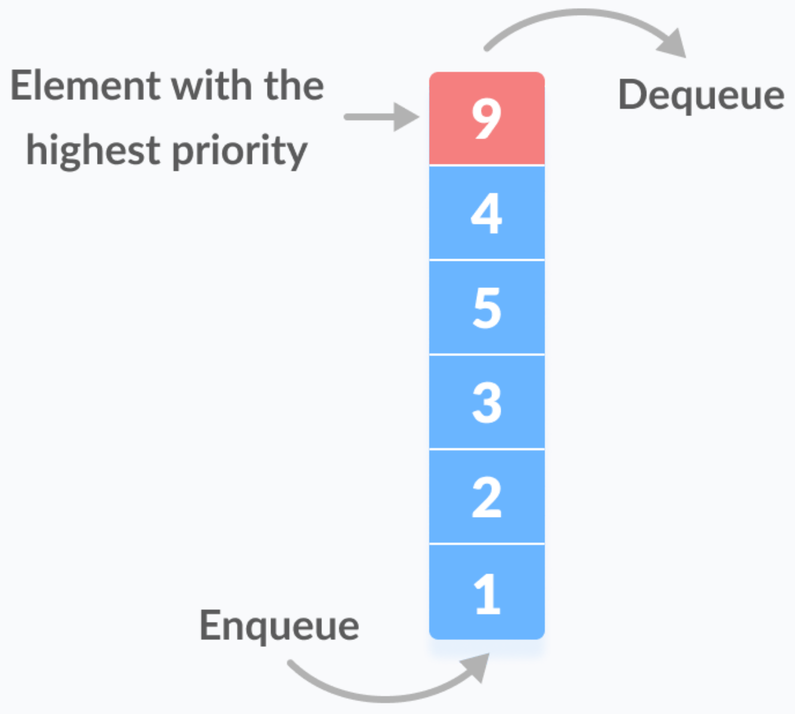
However, if elements with the same priority occur, they are served according to their order in the queue.

Assigning Priority Value

Generally, the value of the element itself is considered for assigning the priority. For example,

The element with the highest value is considered the highest priority element. However, in other cases, we can assume the element with the lowest value as the highest priority element.

We can also set priorities according to our needs.



Removing Highest Priority Element

❖ Difference between Priority Queue and Normal Queue

In a queue, the first-in-first-out rule is implemented whereas, in a priority queue, the values are removed on the basis of priority. The element with the highest priority is removed first.

❖ Implementation of Priority Queue

Priority queue can be implemented using an array, a linked list, a heap data structure, or a binary search tree. Among these data structures, heap data structure provides an efficient implementation of priority queues.

Hence, we will be using the heap data structure to implement the priority queue in this tutorial. A max-heap is implement is in the following operations. If you want to learn more about it, please visit max-heap and mean-heap.

A comparative analysis of different implementations of priority queue is given below.

Operations	peek	insert	delete
Linked List	`O(1)`	`O(n)`	`O(1)`
Binary Heap	`O(1)`	`O(log n)`	`O(log n)`

Binary Search Tree	`O(1)`	`O(log n)`	`O(log n)`

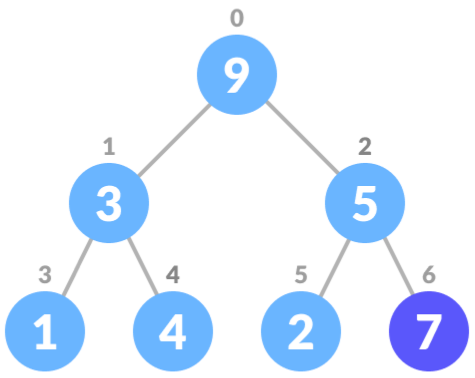
❖Priority Queue Operations

Basic operations of a priority queue are inserting, removing, and peeking elements.

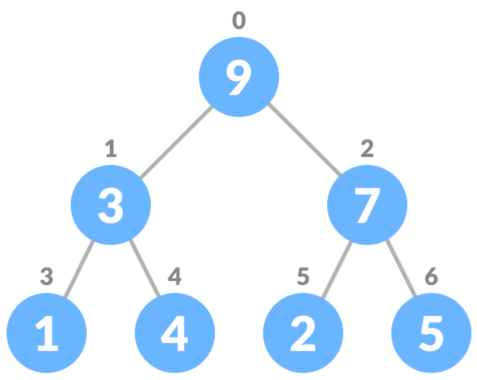
1. Inserting an Element into the Priority Queue

Inserting an element into a priority queue (max-heap) is done by the following steps.

- **Insert the new element at the end of the tree.**



- **Insert an element at the end of the queue**



- Heapify the tree.
- Heapify after insertion

Algorithm for insertion of an element into priority queue (max-heap)

For Min Heap, the above algorithm is modified so that `parentNode` is always smaller than `newNode`.

---

2. Deleting an Element from the Priority Queue

Deleting an element from a priority queue (max-heap) is done as follows:

-
- Select the element to be deleted.Select the element to be deleted
- Swap it with the last element.Swap with the last leaf node element
- Remove the last element.Remove the last element leaf
- Heapify the tree.Heapify the priority queue

Algorithm for deletion of an element in the priority queue (max-heap)

For Min Heap, the above algorithm is modified so that the both `childNodes` are smaller than `currentNode`.

---

3. Peeking from the Priority Queue (Find max/min)

Peek operation returns the maximum element from Max Heap or minimum element from Min Heap without deleting the node.

For both Max heap and Min Heap

---

4. Extract-Max/Min from the Priority Queue

Extract-Max returns the node with maximum value after removing it from a Max Heap whereas Extract-Min returns the node with minimum value after removing it from Min Heap.

In the above figure, we have inserted the elements by using a push() function, and the insert operation is identical to the normal queue. But when we delete the element from the queue by using a pop() function, then the element with the highest priority will be deleted first.

Function	Description
push()	It inserts a new element in a priority queue.
pop()	It removes the top element from the queue, which has the highest priority.
top()	This function is used to address the topmost element of a priority queue.
size()	It determines the size of a priority queue.
empty()	It verifies whether the queue is empty or not. Based on the verification, it returns the status.
swap()	It swaps the elements of a priority queue with another queue having the same type and size.

❖Simple program of priority queue-

```
#include <iostream>
#include<queue>
using namespace std;
int main()
{
 priority_queue<int> p;
 p.push(10);
 p.push(30);
 p.push(20);
 cout<<"Number of elements available in 'p' :"<<p.size()<<endl;
 while(!p.empty())
 {
    std::cout << p.top() << std::endl;
    p.pop();
 }
 return 0;
}
```

In the above code, we have created a priority queue in which we insert three elements, i.e., 10, 30, 20. After inserting the elements, we display all the elements of a priority queue by using a while loop.

Output:

```
Number of elements available in 'p' :3
30
20
10
```