Priority Queue

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.

A priority queue is an abstract data type that behaves similarly to the normal queue except that each element has some priority, i.e., the element with the highest priority would come first in a priority queue. The priority of the elements in a priority queue will determine the order in which elements are removed from the priority queue.

The priority queue supports only comparable elements, which means that the elements are either arranged in an ascending or descending order.

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.

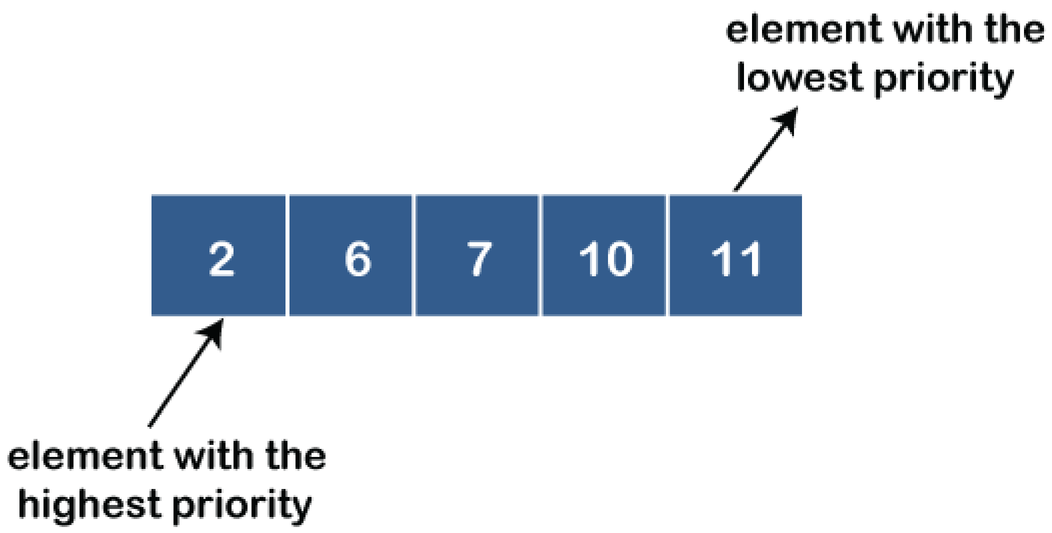
Characteristics of a Priority queue

- Every element in a priority queue has some priority associated with it.
- An element with the higher priority will be deleted before the deletion of the lesser priority.
- If two elements in a priority queue have the same priority, they will be arranged using the FIFO principle.

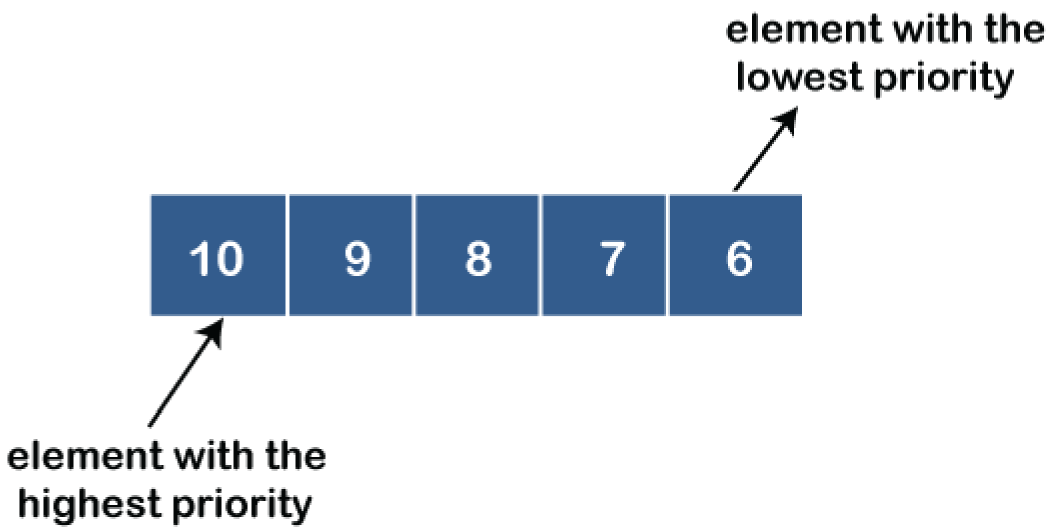
Types of Priority Queue

There are two types of priority queue:

Ascending order priority queue: In ascending order priority queue, a lower priority number is given as a higher priority in a priority. For example, we take the numbers from 1 to 5 arranged in an ascending order like 1,2,3,4,5; therefore, the smallest number, i.e., 1 is given as the highest priority in a priority queue.


element with the lowest priority
2
6
7
10
11
element with the highest priority

Descending order priority queue: In descending order priority queue, a higher priority number is given as a higher priority in a priority. For example, we take the numbers from 1 to 5 arranged in descending order like 5, 4, 3, 2, 1; therefore, the largest number, i.e., 5 is given as the highest priority in a priority queue.


element with the lowest priority
10
9
8
7
6
element with the highest priority

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.

What is Heap?

A heap is a tree-based data structure that forms a complete binary tree, and satisfies the heap property. If A is a parent node of B, then A is ordered with respect to the node B for all nodes A and B in a heap. It means that the value of the parent node could be more than or equal to the value of the child node, or the value of the parent node could be less than or equal to the value of the child node.

Heap Sort Algorithm

Heap Sort is a popular and efficient sorting algorithm in computer programming. Learning how to write the heap sort algorithm requires knowledge of two types of data structures - arrays and trees.

The initial set of numbers that we want to sort is stored in an array e.g. `[10, 3, 76, 34, 23, 32]` and after sorting, we get a sorted array `[3,10,23,32,34,76]`.

Heap sort works by visualizing the elements of the array as a special kind of complete binary tree called a heap.

Relationship between Array Indexes and Tree Elements

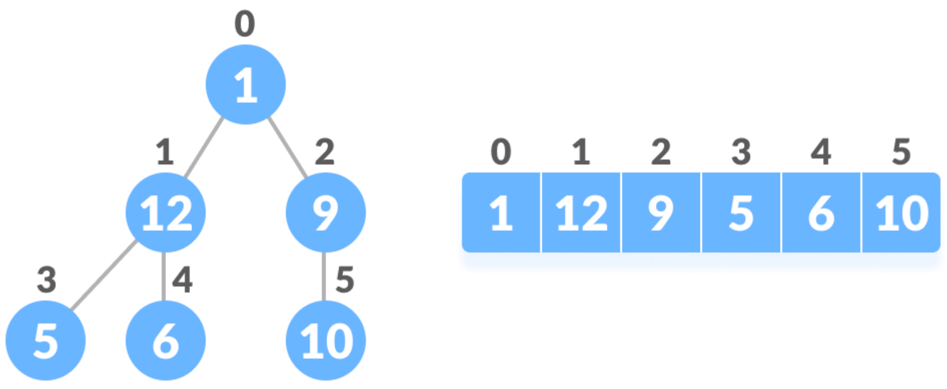
A complete binary tree has an interesting property that we can use to find the children and parents of any node.

A complete binary tree is a binary tree in which all the levels are completely filled except possibly the lowest one, which is filled from the left.

A complete binary tree is just like a full binary tree, but with two major differences

1. All the leaf elements must lean towards the left.
2. The last leaf element might not have a right sibling i.e. a complete binary tree doesn't have to be a full binary tree.

If the index of any element in the array is `i`, the element in the index `2i+1` will become the left child and element in `2i+2` index will become the right child.

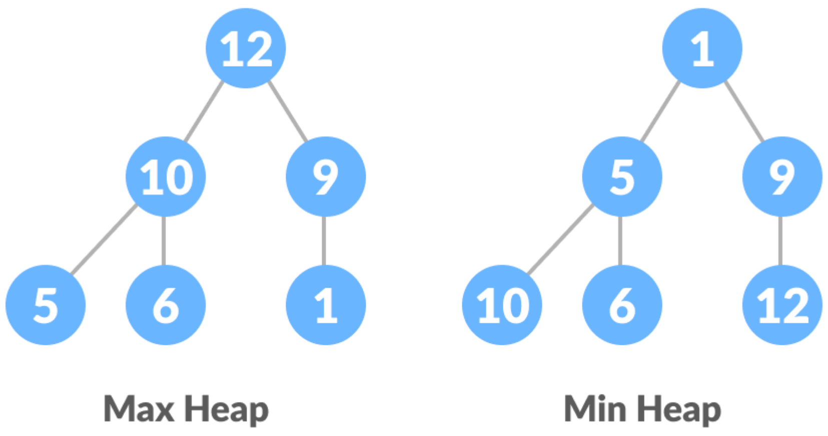


What is Heap Data Structure?

Heap is a special tree-based data structure. A binary tree is said to follow a heap data structure if

- it is a complete binary tree
- All nodes in the tree follow the property that they are greater than their children i.e. the largest element is at the root and both its children and smaller than the root and so on. Such a heap is called a max-heap. If instead, all nodes are smaller than their children, it is called a min-heap

The following example diagram shows Max-Heap and Min-Heap.



Build max-heap

To build a max-heap from any tree, we can thus start heapifying each sub-tree from the bottom up and end up with a max-heap after the function is applied to all the elements including the root element