**EX - 3: Priority Queue**

**Program Description:** implementation of priority queue using heap trees.

Priority queues are used in such situations where you to remove an element from queue which is not at the moment at front of queue, but its priority is high. For example in daily life if in some hospital there is a queue of patients waiting for checkup/treatment from doctor. If some patient comes whose have light threatening symptoms and need urgent treatment then he should be at high priority in that queue and should be removed from queue on his priority, may be patient at front of queue at the moment has come for general checkup.

A **priority queue** is an [abstract data-type](https://en.wikipedia.org/wiki/Abstract_data_type) similar to a regular [queue](https://en.wikipedia.org/wiki/Queue_(abstract_data_type)) or [stack](https://en.wikipedia.org/wiki/Stack_(abstract_data_type)) data structure. Each element in a priority queue has an associated *priority.*

The priority queue is an abstract data type that contains the following methods:

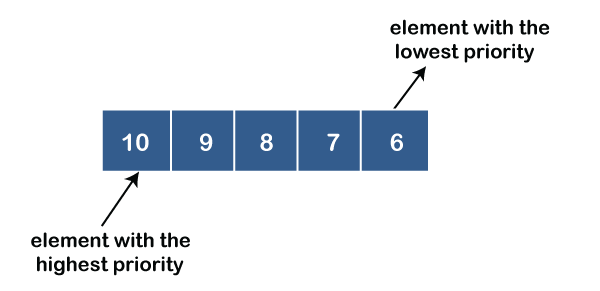
* insert(item, priority): Inserts item into the priority queue with priority value priority.
* poll(): Removes and returns the highest priority item in the priority queue.
* peek(): Returns the highest priority item.

It is similar to a Queue though the insert method will insert an item with a corresponding “priority value” and the poll method in the priority queue will remove the element with the highest priority, rather than the oldest element in the 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.
* 

**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.



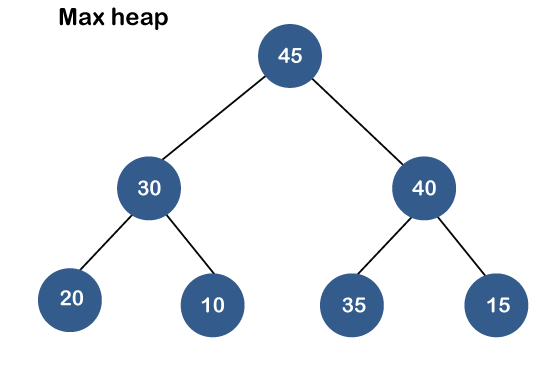
**Implementation of Priority Queue**

The priority queue can be implemented in four ways that include arrays, linked list, heap data structure and binary search tree. The heap data structure is the most efficient way of implementing the priority queue, so we will implement the priority queue using a heap data structure in this topic. Now, first we understand the reason why heap is the most efficient way among all the other data structures.

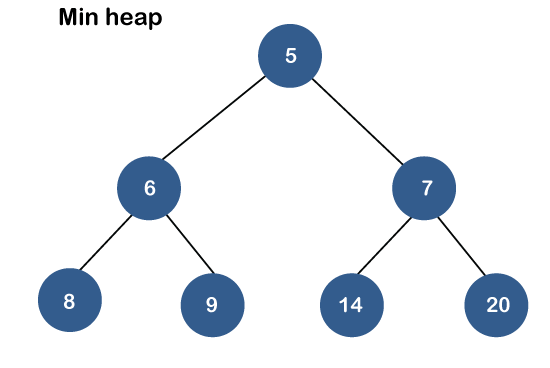
**Analysis of complexities using different implementations**

|  |  |  |  |
| --- | --- | --- | --- |
| **Implementation** | **add** | **Remove** | **peek** |
| Array | O(1) | O(n) | **peek** |
| Linked list | O(n) | O(1) | O(n) |
| Binary heap | O(logn) | O(logn) | O(1) |
| Binary search tree | O(logn) | O(logn) | O(1) |

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. Therefore, we can say that there are two types of heaps:

**Max heap:** The max heap is a heap in which the value of the parent node is greater than the value of the child nodes.  


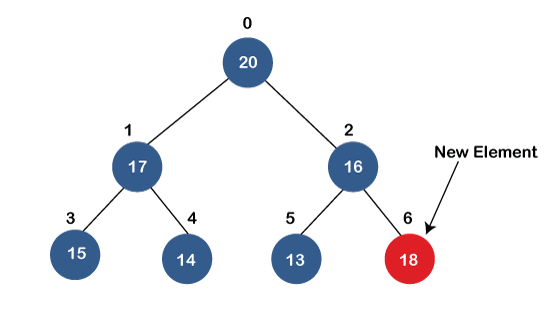
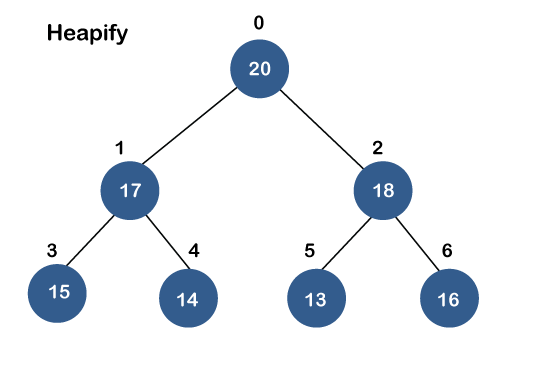
**Min heap:** The min heap is a heap in which the value of the parent node is less than the value of the child nodes.



**Priority Queue Operations**

The common operations that we can perform on a priority queue are insertion, deletion and peek.

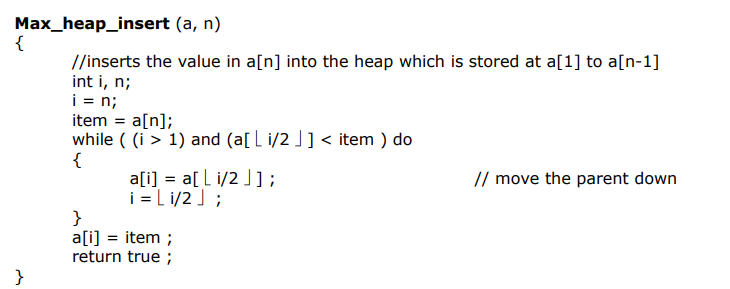
**Inserting the element in a priority queue (max heap):** Inserting an element in a priority queue, it will move to the empty slot by looking from top to bottom and left to right. If the element is not in a correct place then it is compared with the parent node; if it is found out of order, elements are swapped. This process continues until the element is placed in a correct position.

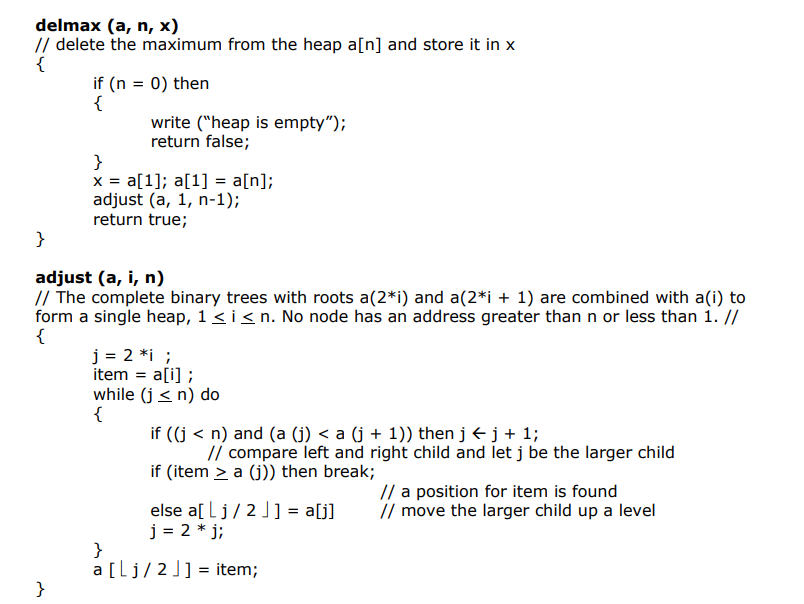
  


**Removing the minimum element from the priority queue**

As we know that in a max heap, the maximum element is the root node. When we remove the root node, it creates an empty slot. The last inserted element will be added in this empty slot. Then, this element is compared with the child nodes, i.e., left-child and right child, and swap with the smaller of the two. It keeps moving down the tree until the heap property is restored.

**Pseudo code:**





heapify

**Program Code:**

**Construct a priority queue using heap data structure, and perform operations like**

1. **insertion**
2. **Deletion**
3. **Merging**

**Also analyze the time complexity of each operation.**

**Input and Output:**

**List of Application programs:**

1. **Implement heap sort procedure. Also analyze the complexity for sorting**
2. Given an unordered sequence of N numbers (a1, a2, ... aN), select the k-th largest number
3. Implement Shortest job first (SJF) scheduling using Heap

**Hacker Ranker problems:**

1. <https://leetcode.com/problems/top-k-frequent-words/>
2. <https://leetcode.com/problems/kth-largest-element-in-a-stream/>
3. <https://leetcode.com/problems/maximum-average-pass-ratio/>
4. <https://leetcode.com/problems/maximum-ice-cream-bars/>

Programs using STL library:

1. Write a program to store 10 employees records in priority queue using STL C++ Priority\_Quque. Retrieve the employee information based on their ages in increasing order.

The employee record should maintain fields like

EMPID

Ename

Designation

Age

Height

**Sample input :**

{100,’aaa’,’manager’,56,5.6}

{100,’aaa’,’clerk’,30,5.5}

**Output:**

100,’aaa’,’clerk’,30,5.5

100,’aaa’,’manager’,56,5.6

**Hint: Use structure or class concept**