**Data Structures Lab**

***Session 10***

**Course:** Data Structures (CL2001) **Semester:** Fall 2024

**Instructor:** Shafique Rehman  **T.A:** N/A

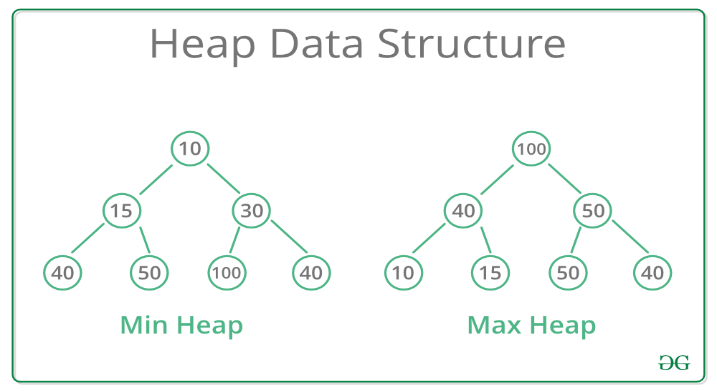
**Note:**

* Maintain discipline during the lab.
* Listen and follow the instructions as they are given.
* Just raise hand if you have any problem.
* Completing all tasks of each lab is compulsory.
* Get your lab checked at the end of the session.

**HEAP Data Structures**

A Heap is a special Tree-based data structure in which the tree is a complete binary tree. Operations of Heap Data Structure:

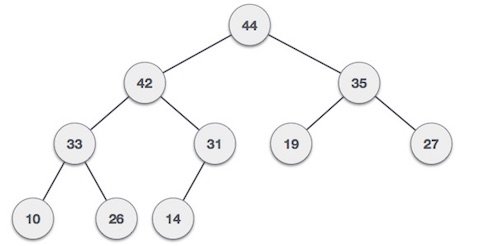
1. Heapify: a process of creating a heap from an array.
2. Insertion: process to insert an element in existing heap time complexity O(log N).
3. Deletion: deleting the top element of the heap or the highest priority element, and then organizing the heap and returning the element with time complexity O(log N).
4. Peek: to check or find the most prior element in the heap, (max or min element for max and min heap).



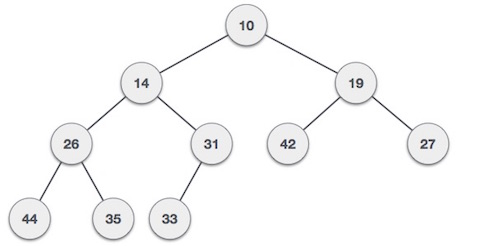
**Types of Heap Data Structure**

Generally, Heaps can be of two types:

Max-Heap: In a Max-Heap the key present at the root node must be greatest among the keys present at all of it’s children. The same property must be recursively true for all sub-trees in that Binary Tree.



Min-Heap: In a Min-Heap the key present at the root node must be minimum among the keys present at all of it’s children. The same property must be recursively true for all sub-trees in that Binary Tree.



**How to construct a HEAP:**

We shall use the same example to demonstrate how a Max Heap is created. The procedure to create Min Heap is similar but we go for min values instead of max values.

We are going to derive an algorithm for max heap by inserting one element at a time. At any point of time, heap must maintain its property. While insertion, we also assume that we are inserting a node in an already heapified tree.

**Step 1** − Create a new node at the end of heap.

**Step 2** − Assign new value to the node.

**Step 3** − Compare the value of this child node with its parent.

**Step 4** − If value of parent is less than child, then swap them.

**Step 5** − Repeat step 3 & 4 until Heap property holds.

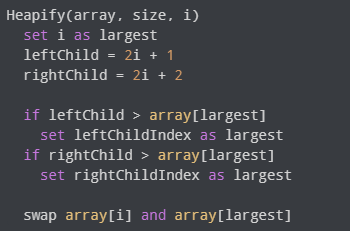
**Note** − In Min Heap construction algorithm, we expect the value of the parent node to be less than that of the child node.

Let's understand Max Heap construction by an animated illustration. We consider the same input sample that we used earlier.

Graphical user interface, text, application

Description automatically generated

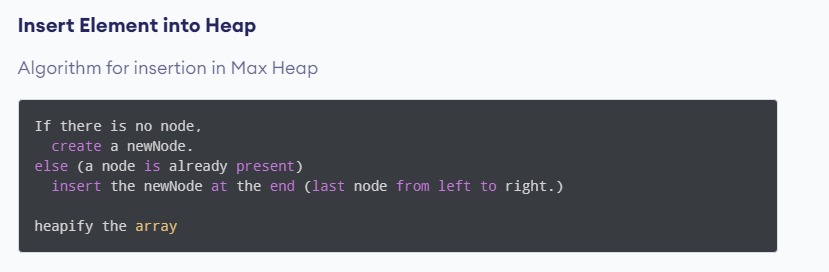
**Heapify Algorithm**

****

**Heapify:**

* Start from the first index of non-leaf node whose index is
* given by n/2 – 1
* Set current element i as largest.
* The index of left child is given by 2i + 1 and the right child is
* given by 2i + 2.
* If leftChild is greater than currentElement (i.e. element
* at ith index), set leftChildIndex as largest.
* If rightChild is greater than element in largest,
* set rightChildIndex as largest.
* Swap largest with currentElement.

**Insertion In Heap:**

****

**Max Heap Deletion Algorithm**

Let us derive an algorithm to delete from max heap. Deletion in Max (or Min) Heap always happens at the root to remove the Maximum (or minimum) value.

**Step 1** − Remove root node.

**Step 2** − Move the last element of last level to root.

**Step 3** − Compare the value of this child node with its parent.

**Step 4** − If value of parent is less than child, then swap them.

**Step 5** − Repeat step 3 & 4 until Heap property holds.

**Task-1**: Suppose you are building a program to manage a delivery company's package distribution system. The packages are assigned priority levels based on their importance and delivery time. The priority levels range from 1 to 5, where 1 represents the highest priority and 5 represents the lowest.

* To efficiently manage package delivery, you decide to use a binary heap. The heap will be initialized as a min-heap, which means that the package with the highest priority level (i.e., priority level 1) will have the highest priority and be the first to be delivered.
* Now, suppose a new package arrives with priority level 2 that needs to be added to the heap. Write code that adds the new package to the heap and displays the updated heap

**Task-2:** Suppose you are working on a project that requires a priority queue to manage the processing of incoming data. The data is sorted by priority level, and the highest priority data is processed first. You decide to use a binary heap to implement the priority queue.

Now, imagine that the heap has been initialized as a max-heap, and you need to remove the data with the highest priority (i.e., the root node) from the heap. Write code that removes the root node and displays the updated heap.

**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.

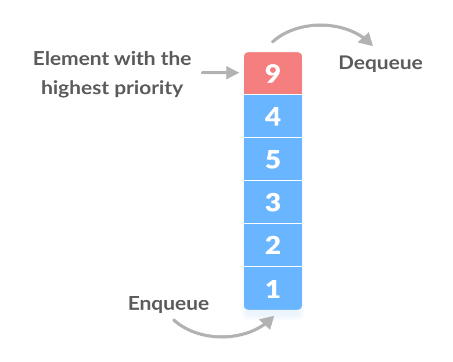
However, if elements with the same priority occur, they are served according to their order in the queue.

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.



**Task-3:** Using the array 25 30 35 11 15 19 18 55 78 36

1. Create a Max Heap and Delete the Value 55
2. Create a Min Heap and Delete the Value 18
3. Sort the max heap and print it