**Course: Data Structures and Algorithms**

**Lab 10**

**Priority Queue using binary heap**

A Binary Heap is either Min Heap or Max Heap. In a Min Binary Heap, the key at the root must be minimum among all keys present in Binary Heap. The same property must be recursively true for all nodes in Binary Tree. Max Binary Heap is similar to MinHeap.

### Examples of Min Heap:

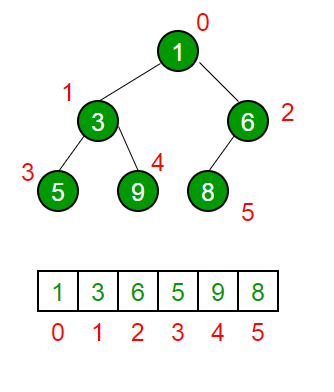
*10                       10  
         /      \                 /         \    
     20     100        15           30    
   /                        /    \         /    \  
30                     40   50   100   40*

## How is Binary Heap represented?

A Binary Heap is a **Complete Binary Tree**. A binary heap is typically represented as an array.

* The root element will be at Arr[0].
* The below table shows indices of other nodes for the ith node, i.e., Arr[i]:

|  |  |
| --- | --- |
| Arr[(i-1)/2] | Returns the parent node |
| Arr[(2\*i)+1] | Returns the left child node |
| Arr[(2\*i)+2] | Returns the right child node |



## Operations on Heap:

Below are some standard operations on min heap:

* **getMin():** It returns the root element of Min Heap. The time Complexity of this operation is **O(1)**. In case of a maxheap it would be **getMax()**.
* **extractMin():** Removes the minimum element from MinHeap. The time Complexity of this Operation is **O(log N)** as this operation needs to maintain the heap property (by calling **heapify()**) after removing the root.
* **insert():** Inserting a new key takes **O(log N)** time. We add a new key at the end of the tree. If the new key is greater than its parent, then we don’t need to do anything. Otherwise, we need to traverse up to fix the violated heap property.
* **delete():** Deleting a key also takes **O(log N)** time.

# Lab 10-PQ using binary heap

implement PQ using binary heap...  
Complete the following code and also add a driver class to run the code.  
  
  
class BinaryHeap{  
int[] heap;  
//boolean insertion(int value){  
//}  
  
//int deletion(){  
//}  
//private void heapifyUp()  
  
//private void heapifyDown()  
  
//int size(){ u can implement the same method frm binary tree }  
  
//void print( )  
  
}  
  
class PQ{  
  
//boolean enqueue(int value){  
//return insertion(value);  
}  
  
// int dequeue(){  
//return deletion();  
//}  
  
//print()  
}