* A Binary Heap is a Binary Tree with following properties. It’s a complete tree (All levels are completely filled except possibly the last level and the last level has all keys as left as possible). This property of Binary Heap makes them suitable to be stored in an array.
* A Binary Heap is either Min Heap or Max Heap. In a Min Binary Heap, the key at 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.
* [Heap Sort](http://quiz.geeksforgeeks.org/heap-sort/): Heap Sort uses Binary Heap to sort an array in O(nLogn) time.
* Priority Queue: Priority queues can be efficiently implemented using Binary Heap because it supports insert(), delete() and extractmax(), decreaseKey() operations in O(logn) time.
* **Operations on Min Heap:**
* getMini(): It returns the root element of Min Heap. Time Complexity of this operation is O(1).
* extractMin(): Removes the minimum element from MinHeap in O(Logn) as this operation needs to maintain the heap property (by calling heapify()) after removing root.
* **3)** decreaseKey(): Decreases value of key. The time complexity of this operation is O(Logn). If the decreases key value of a node is greater than the parent of the node, then we don’t need to do anything. Otherwise, we need to traverse up to fix the violated heap property.
* **4)**insert(): Inserting a new key takes O(Logn) time. We add a new key at the end of the tree. IF 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.
* **5)** delete(): Deleting a key also takes O(Logn) time. We replace the key to be deleted with minimum infinite by calling decreaseKey(). After decreaseKey(), the minus infinite value must reach root, so we call extractMin() to remove the key.

In a heap left element stays at 2i + 1 and right stays at 2i+2. Any index i if it’s i >= (n - 2) / 2 then it is a leaf node