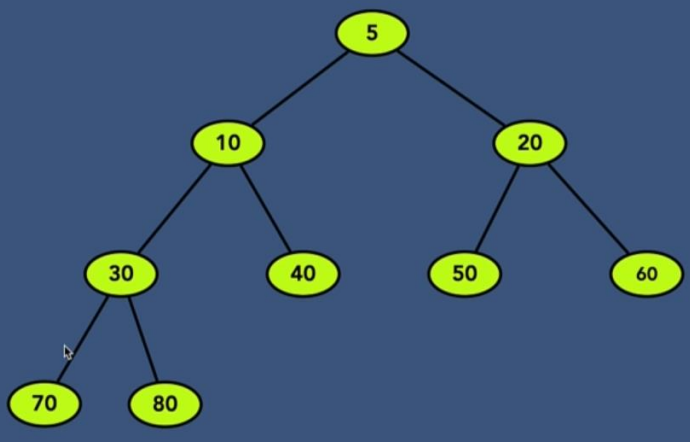
What is a Binary Heap?

A Binary Heap is a Binary Tree with following properties.

- 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.
- 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.



The above is the example of a min heap → the root element is the smallest when compared with the other two children.

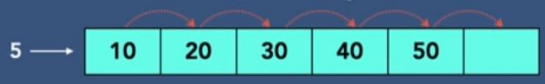
Max Heap → the root element is the largest when compared with the other two children.

Why we need a Binary Heap?

Find the minimum or maximum number among a set of numbers in logN time. And also we want to make sure that inserting additional numbers does not take more than O(logN) time

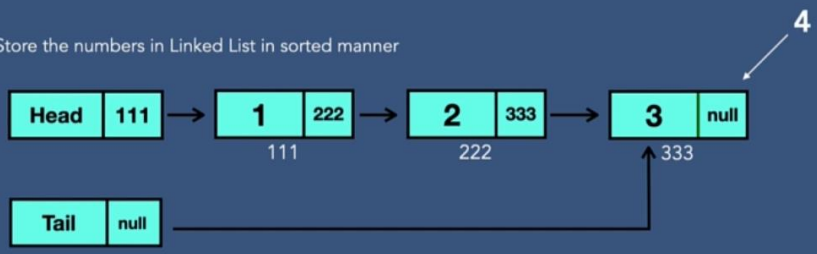
Possible Solutions

- Store the numbers in sorted Array



Find minimum: **O(1)**
Insertion: **O(n)**

- Store the numbers in Linked List in sorted manner



Why we need a Binary Heap?

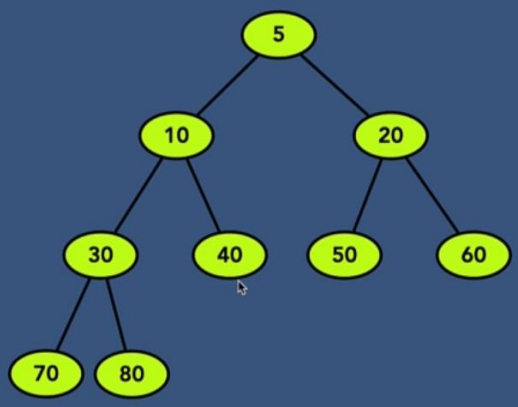
Find the minimum or maximum number among a set of numbers in logN time. And also we want to make sure that inserting additional numbers does not take more than O(logN) time

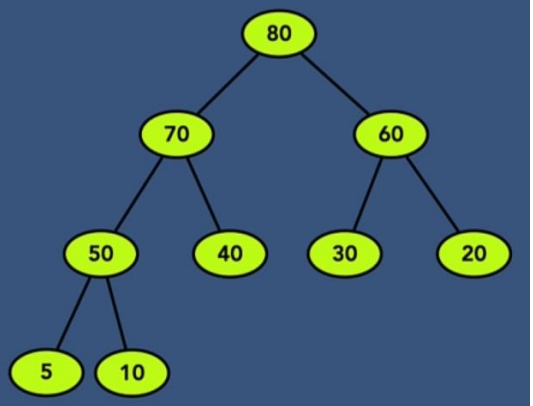
Practical Use

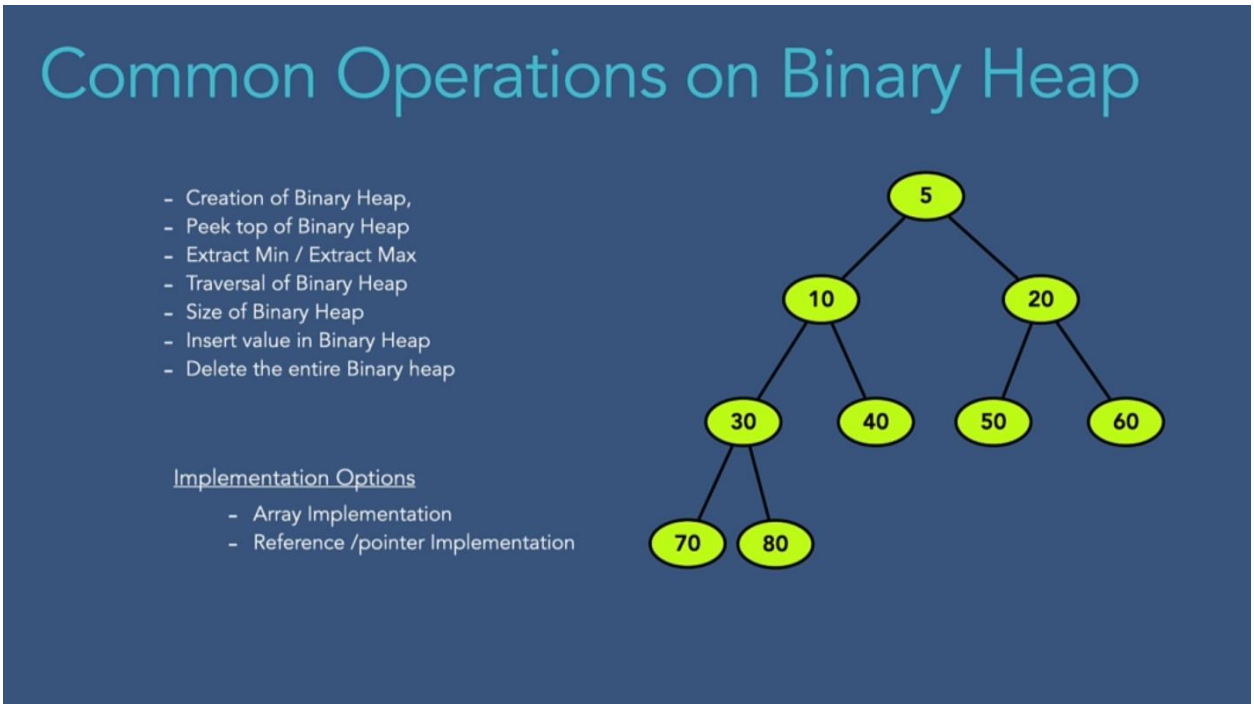
- Prim's Algorithm
- Heap Sort
- Priority Queue

Types of Binary Heap

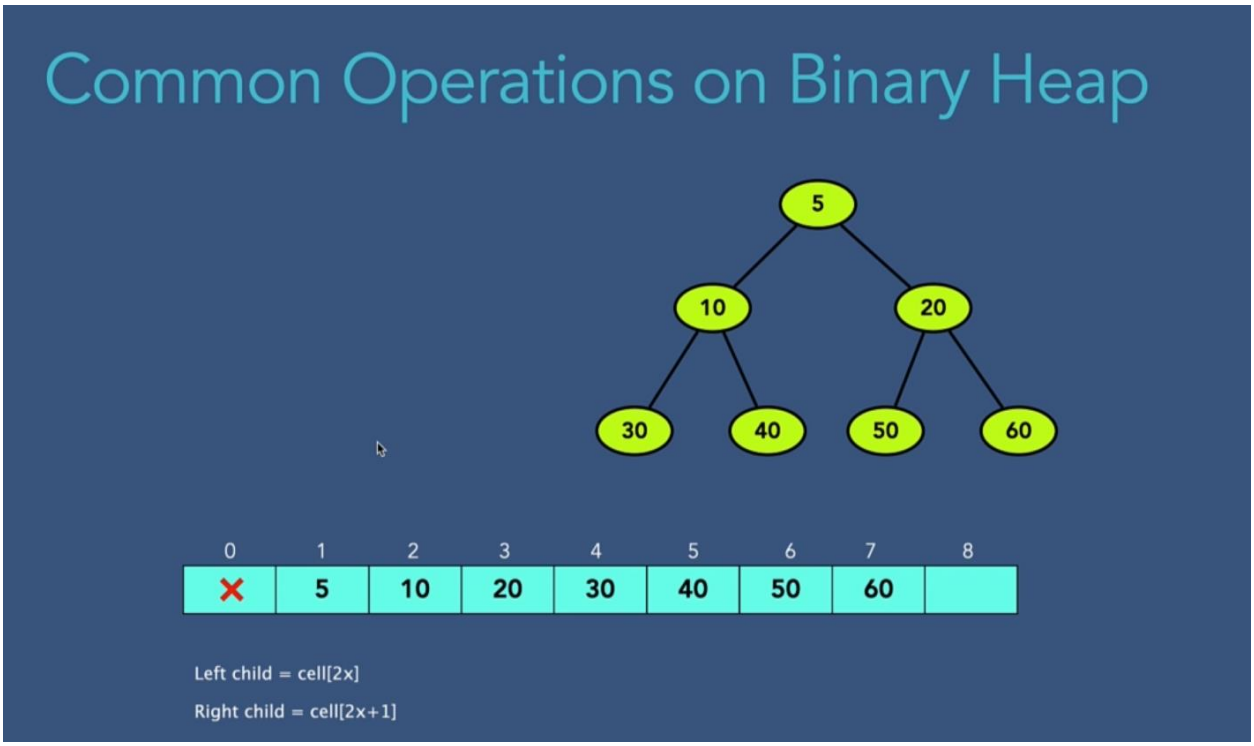
Min heap - the value of each node is less than or equal to the value of both its children.
Max heap - it is exactly the opposite of min heap that is the value of each node is more than or equal to the value of both its children.







We use the Array Implementation as it is the best fit for binary heap

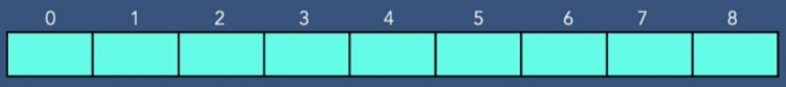


This is how the binary heap is stored in memory using the above formulas for left and the right child. Skipping the 0^{th} index which will make the mathematical implementation easy.

Common Operations on Binary Heap

- Creation of Binary Heap

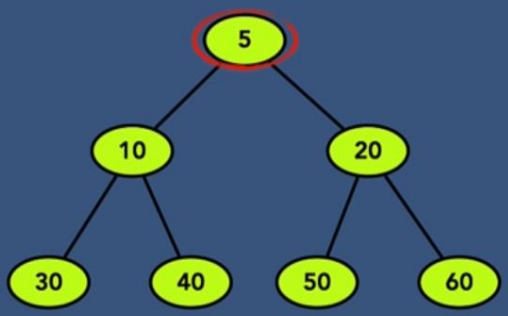
Initialize Array
set size of Binary Heap to 0

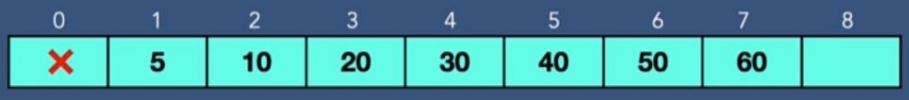


Common Operations on Binary Heap

- Peek of Binary Heap

Return Array[1]

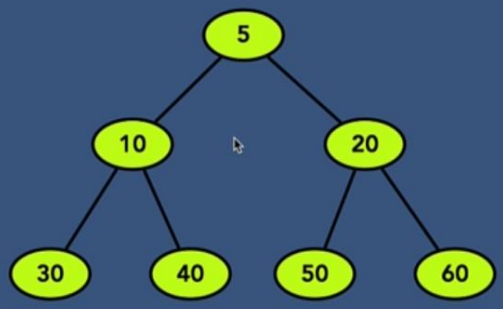




Common Operations on Binary Heap

- Size Binary Heap

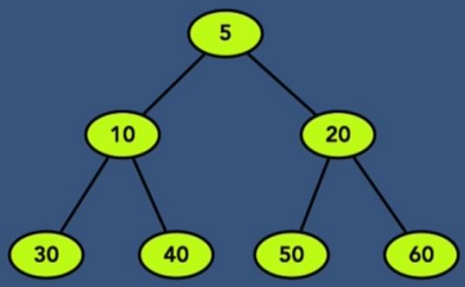
Return number of filled cells



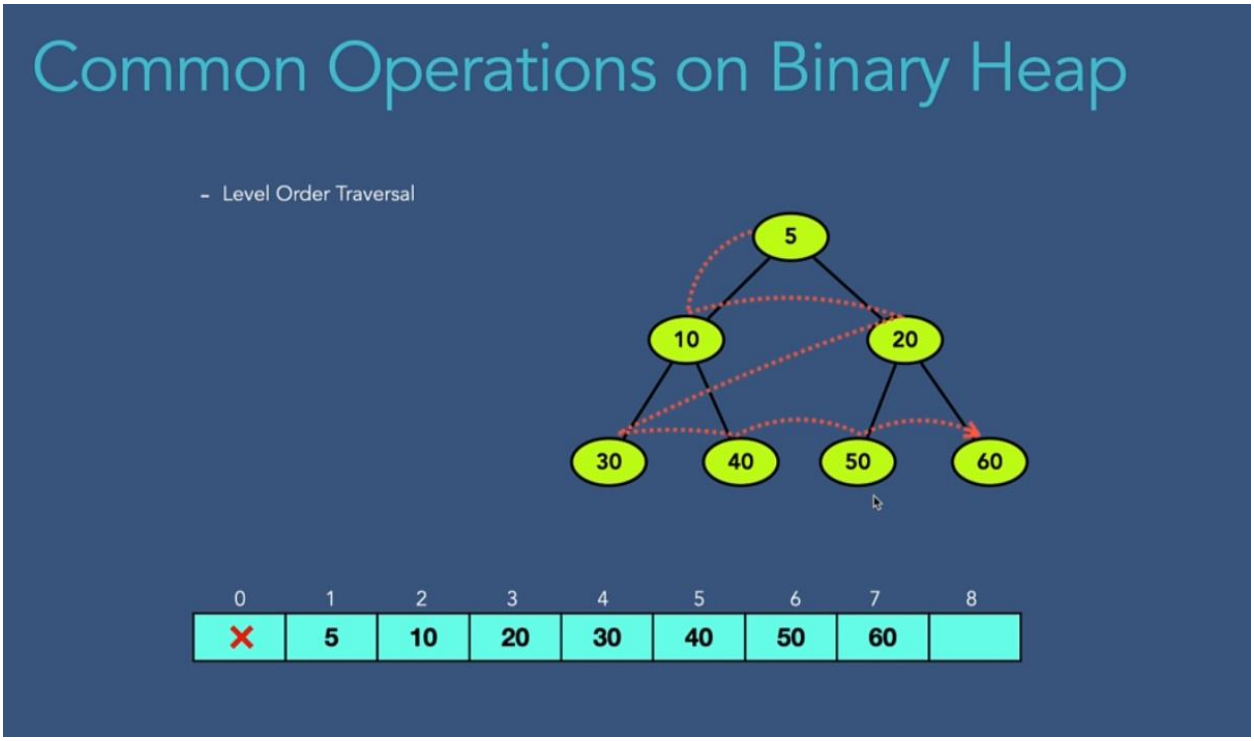
0	1	2	3	4	5	6	7	8
✗	5	10	20	30	40	50	60	

Common Operations on Binary Heap

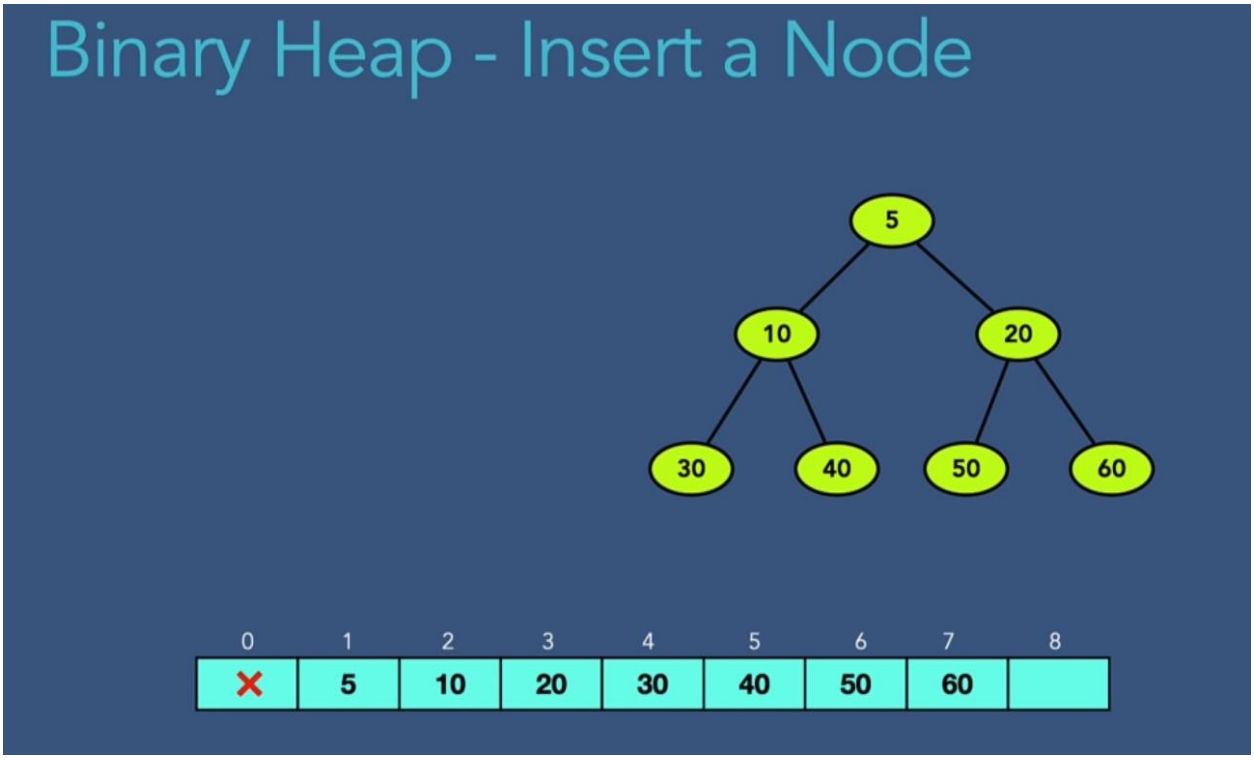
- Level Order Traversal



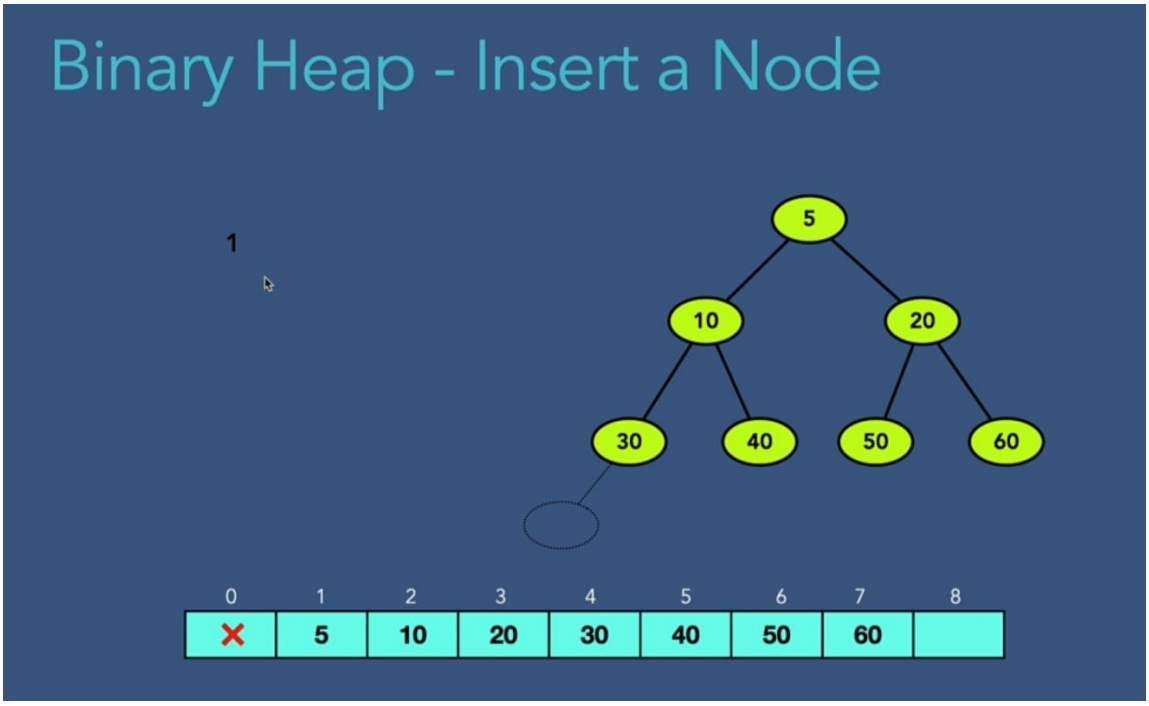
0	1	2	3	4	5	6	7	8
✗	5	10	20	30	40	50	60	



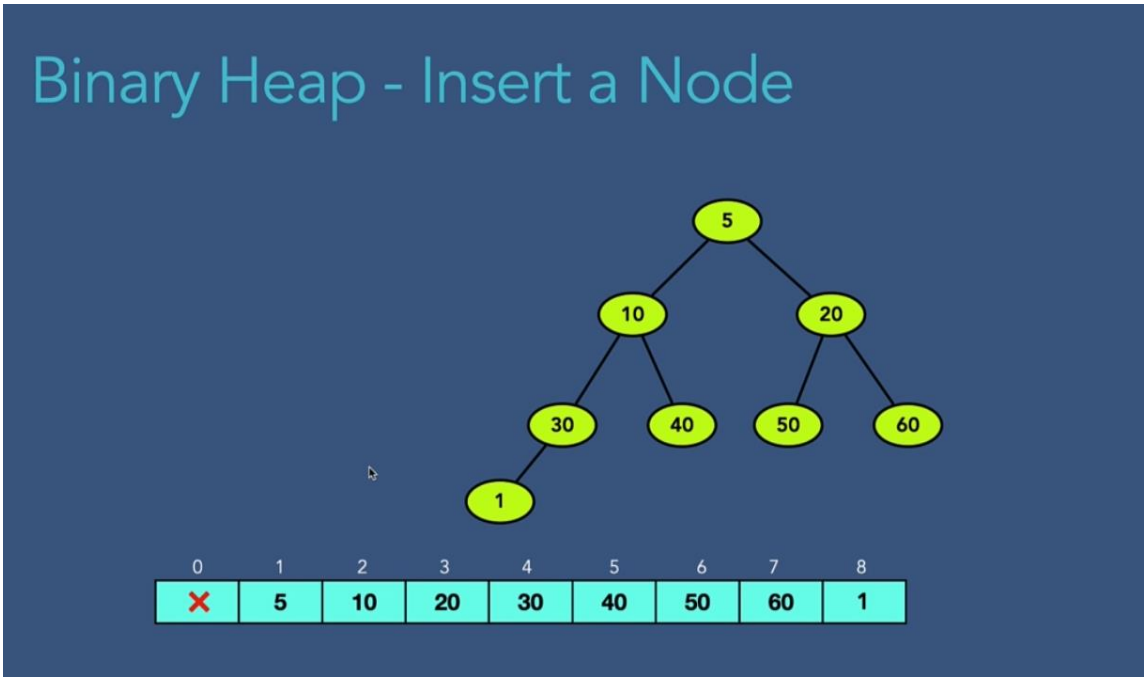
Level Order Traversal in array implementation of Binary Heap is very easy as the array already has the elements in Level Order



Insertion of node 1 to the Binary Heap

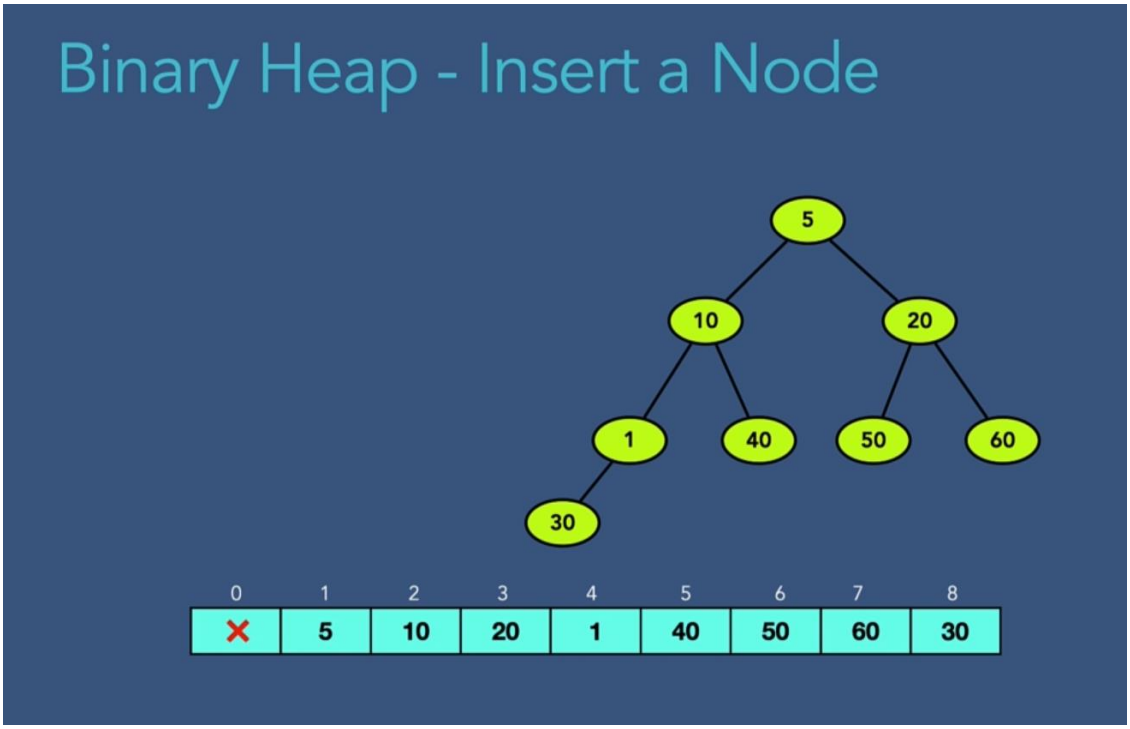


After the insertion of Node 1 to the Binary Heap

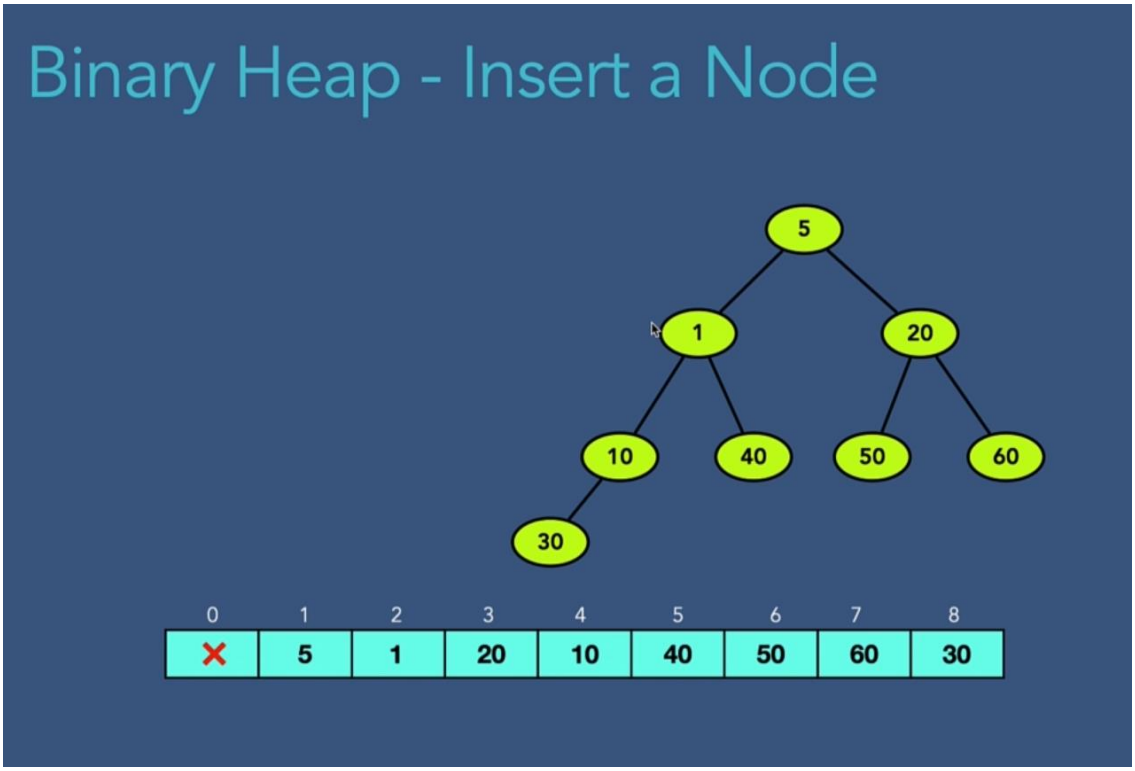


This violates the property of the Binary Heap as the Node 1 is inserted as the leaf node and as the child of the node 30. As this is an example of Min-Heap so the value inserted here should be greater than or equal to the parent node. That's the violation of the property of the Binary Heap. So, we have to heapify the Binary Heap from Bottom to Top.

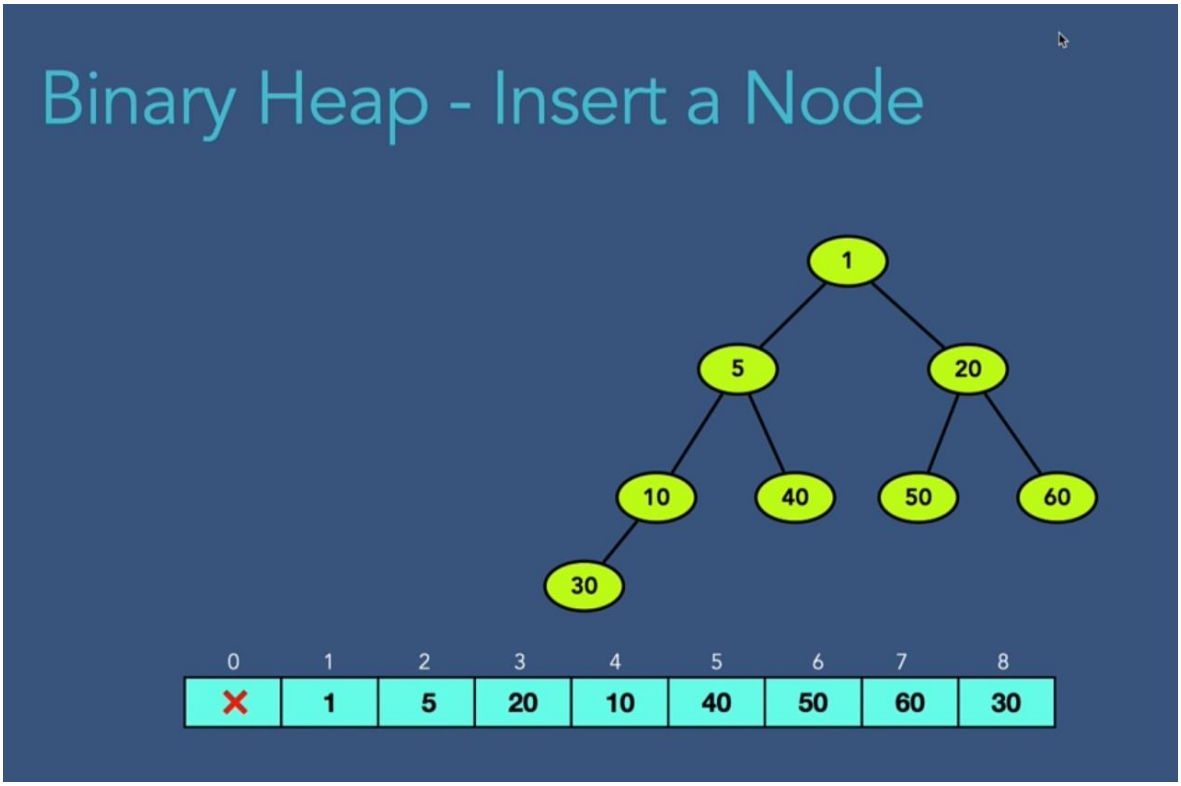
After 1st Heapify Operation



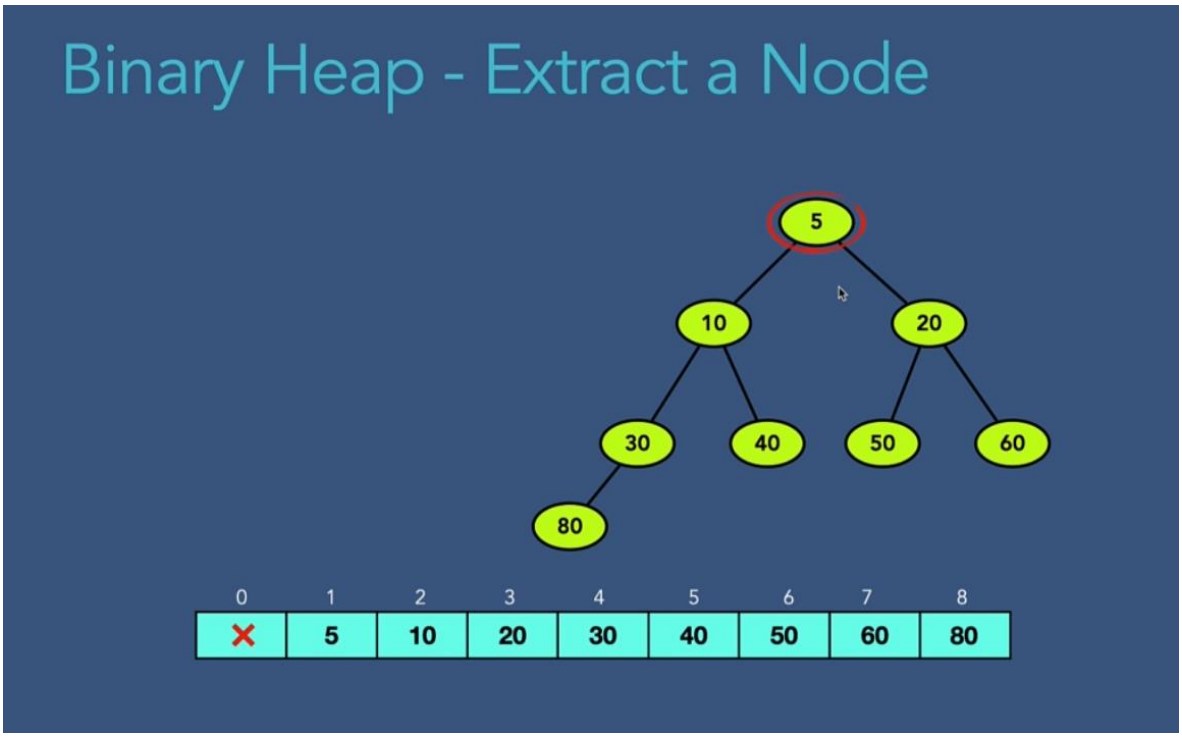
After 2nd Heapify Operation



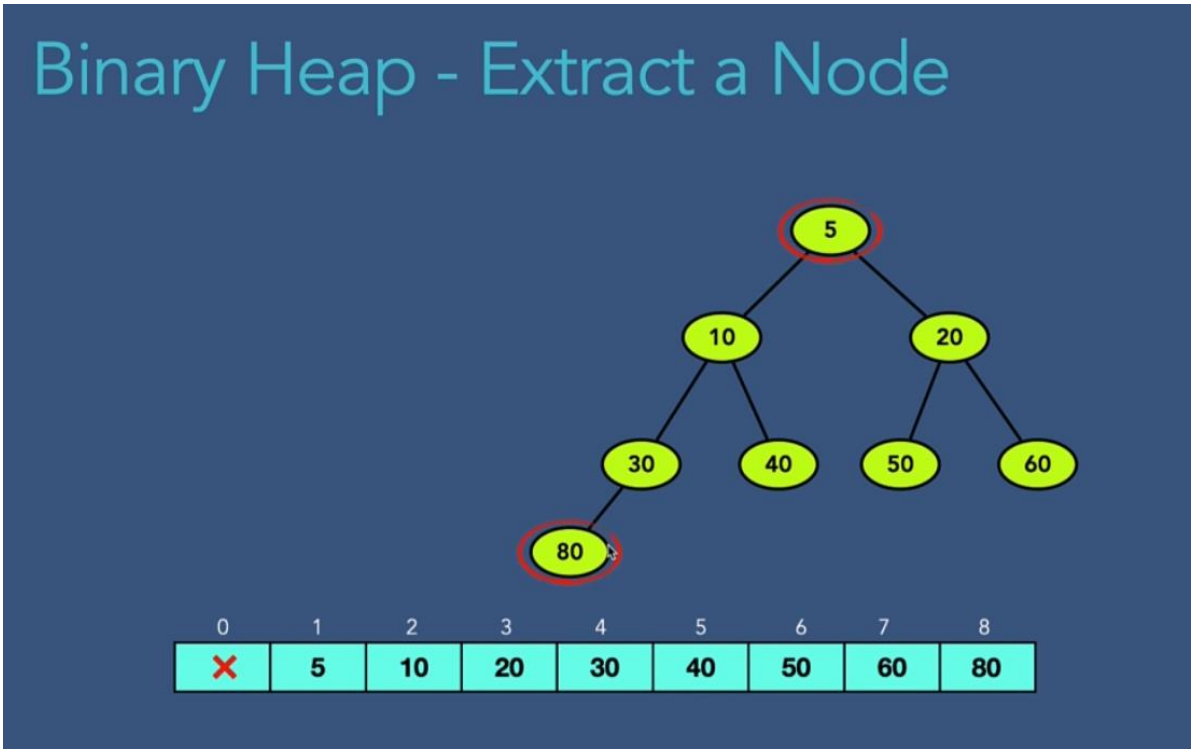
After 3rd Heapify Operation



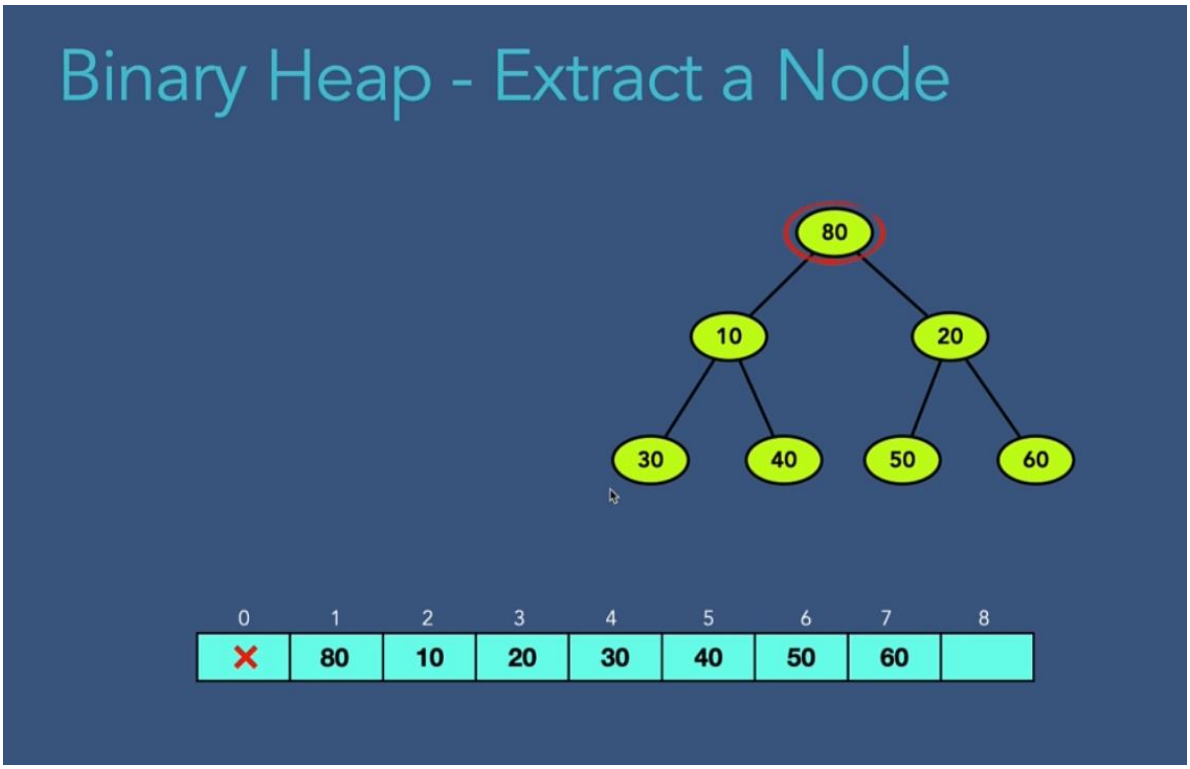
And now all the properties of the Binary Heap are satisfied



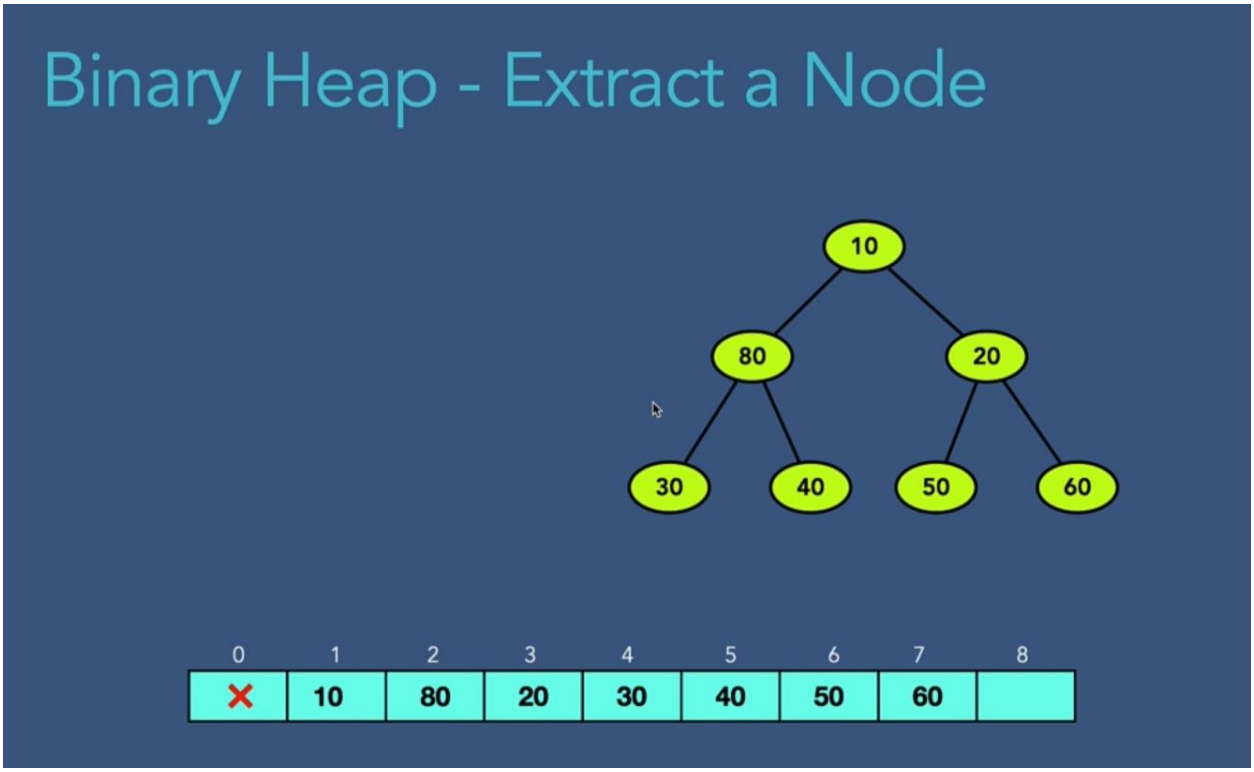
In Binary Heap we can only extract the elements in order. The root node is always extracted first.



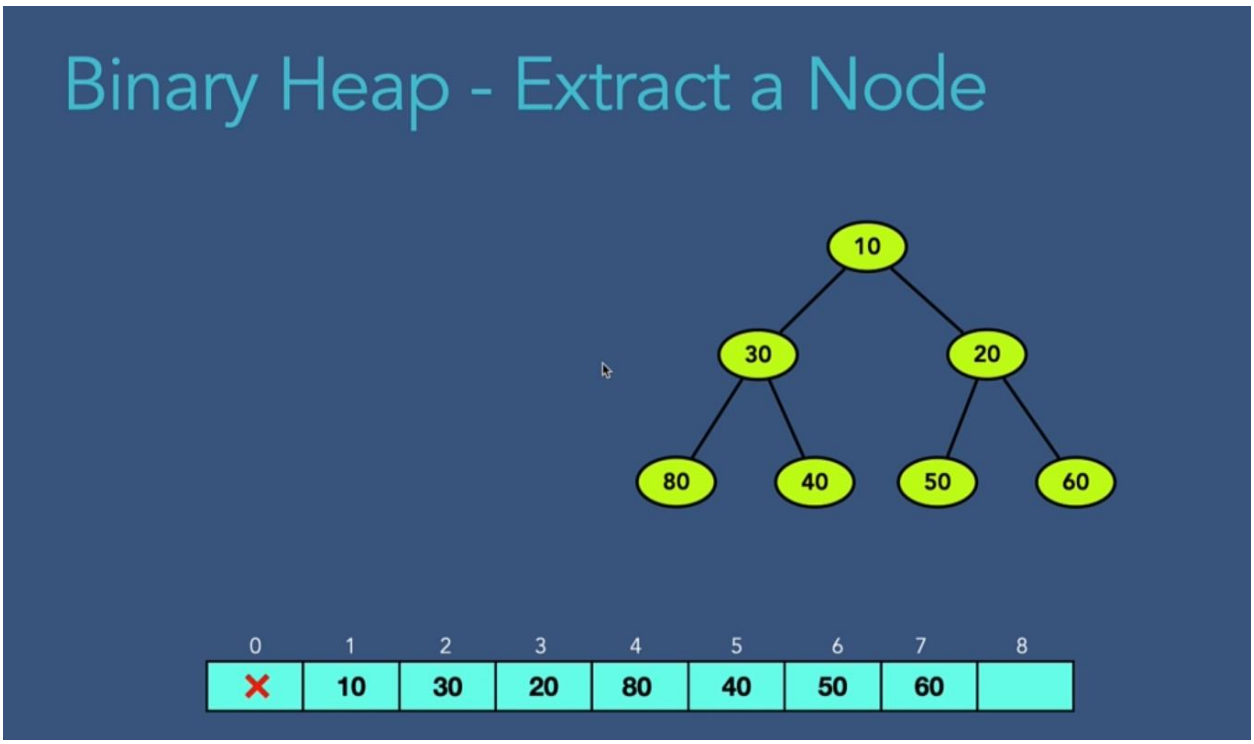
We first find the last node of the Binary Heap and the change the last node as the root node and delete the last node from the Binary Heap



After the deletion of the last node from the Binary Heap and now the Binary Heap property is violated. So, we have to Heapify the Binary Heap from Top to Bottom.



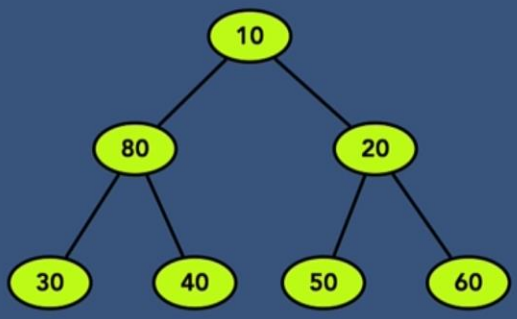
As this is a Min-Heap so the root node should be the smallest so we choose Node 10 as the root node. And again, this Binary Heap violates the property of the Binary Heap.



Continuing the above process until the Binary Heap does not violate the property. Now, this Binary Heap does not violate the property.

Binary Heap - Delete

array = Null



Binary Heap - Delete

array = Null

Time and Space Complexity of Binary Heap

	Time complexity	Space complexity
Create Binary Heap	O(1)	O(N)
Peek of Heap	O(1)	O(1)
Size of Heap	O(1)	O(1)
Traversal of Heap	O(N)	O(1)
Insert a node to Binary Heap	O(logN)	O(logN)
Extract a node from Binary Heap	O(logN)	O(logN)
Delete Entire Binary Heap	O(1)	O(1)