Binary Heap:

A Binary Heap is a complete Binary Tree which is used to store data efficiently to get the max or min element based on its structure.

What is Heap?

A heap is a data structure based on a unique binary tree designed to efficiently access the smallest or largest element in a collection of items. It follows a complete binary tree's property and satisfies the heap property. Therefore, it is also known as a binary heap.

As we all know, the complete binary tree is a tree with every level filled and all the nodes are as far left as possible. In the binary tree, it is possible that the last level is empty and not filled. Now, you must be wondering what is the heap property.

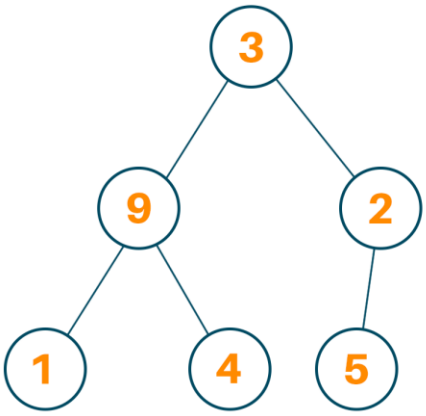
In the heap data structure, we assign key-value or weight to every node of the tree. Now, the root node key value is compared with the children's nodes and then the tree is arranged accordingly into two categories i.e., max-heap and min-heap.

The heap data structure is basically used as a heapsort algorithm to sort the elements in an array or a list. These algorithms can be used in priority queues, order statistics, Prim's algorithm or Dijkstra's algorithm, etc.

As learned earlier, there are two categories of heap data structure i.e. max-heap and min-heap. Let us understand them below but before that, we will study the heapify property to understand max-heap and min-heap.

What is Heapify?

The process of creating a heap data structure using the binary tree is called Heapify. The heapify process is used to create the Max-Heap or the Min-Heap.



Binary Tree

We will start the process of heapify from the first index of the non-leaf node as shown below:

Now we will set the current element “k” as “largest” and as we know the index of a left child is given by “2k + 1” and the right child is given by “2k + 2”.

Therefore, if the left child is larger than the current element i.e. kth index we will set the “largest” with the left child’s index, and if the right child is larger than the current element i.e., kth index then we will set the “largest” with right child’s index.

Lastly, we will swap the “largest” element with the current element(kth element).

We’ll repeat the above steps 3-6 until the tree is heaped.

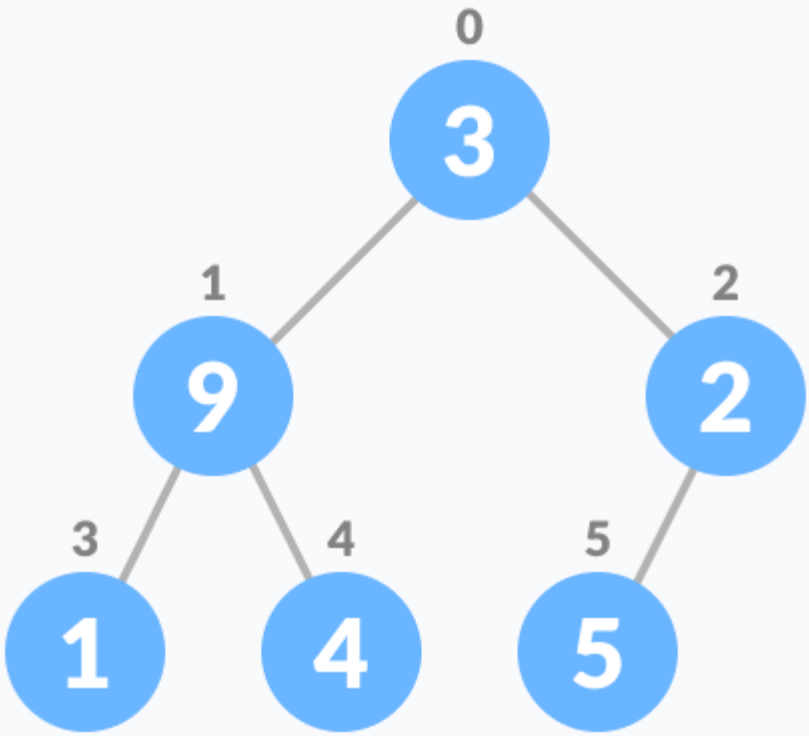
What is Max Heap?

When the value of each internal node is larger than or equal to the value of its children node then it is called the Max-Heap Property. Also, in a max-heap, the value of the root node is largest among all the other nodes of the tree.

1. Let the input array be

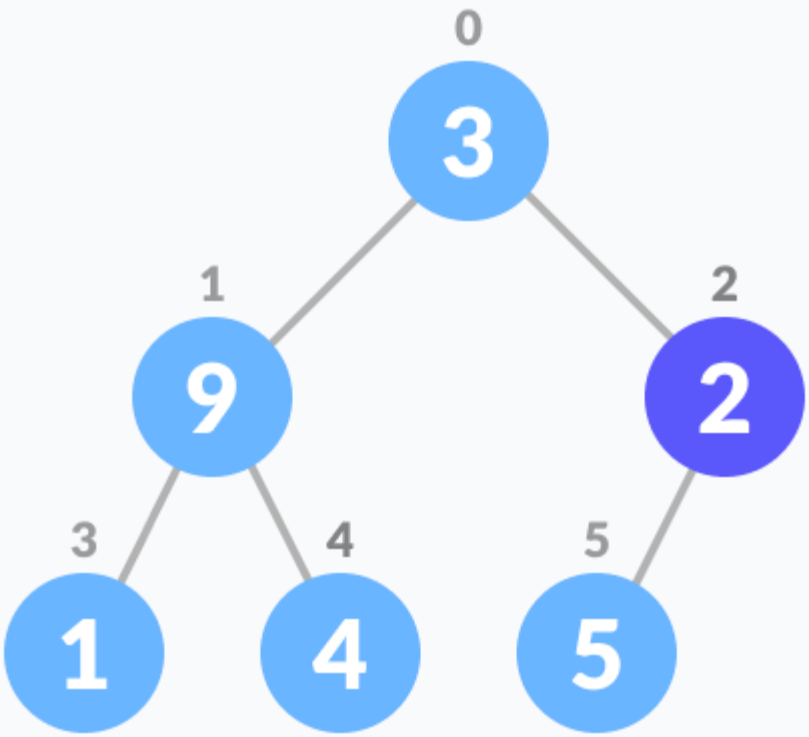
Initial Array

2. Create a complete binary tree from the array



Complete binary tree

3.Start from the first index of non-leaf node whose index is given by `n/2 - 1`.



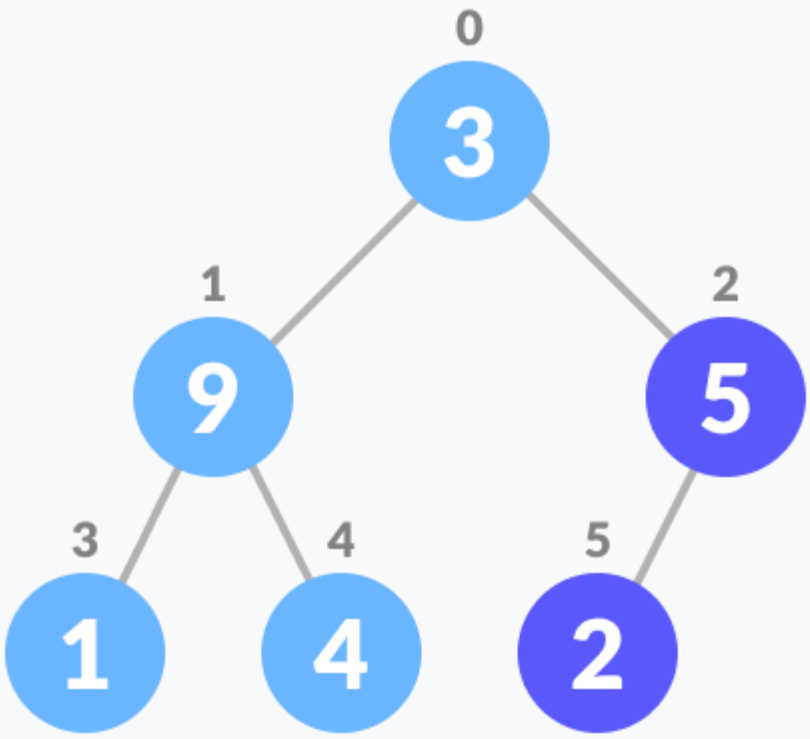
Start from the first on leaf node

3. Set current element `i` as `largest`.
4. 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`.

5. Swap `largest` with `currentElement`



Swap if necessary

6. Repeat steps 3-7 until the subtrees are also heapified.

What is Min Heap?

When the value of each internal node is smaller than the value of its children node then it is called the Min-Heap Property. Also, in the min-heap, the value of the root node is the smallest among all the other nodes of the tree.

Time complexity

The running time complexity of the building heap is O(n log(n)) where each call for heapify costs O(log(n)) and the cost of building heap is O(n). Therefore, the overall time complexity will be O(n log(n)).

Applications of Heap

A heap is used for a variety of purposes. It is a powerful tool used in sorting, searching, and graph traversal algorithms, as well as other applications requiring efficient management of a collection of ordered elements. Following are some of the main practical applications of it:

- Heap is used while implementing a priority queue. It is another data structure to access and remove the item in the highest priority.
- It is used in the Heap sort, selection algorithm, Prim's algo, and Dijkstra's algorithm.
- We can use max-heap and min-heap in the operating system for the job scheduling algorithm.
- It is used in order statistics, for tasks like how to find the median of a list of numbers.

Insert Element into Heap

Delete Element from Heap

```
# Priority Queue implementation in Python


# Function to heapify the tree
def heapify(arr, n, i):
    # Find the largest among root, left child and right child
    largest = i
    l = 2 * i + 1
    r = 2 * i + 2

    if l < n and arr[i] < arr[l]:
```

```
        largest = l

    if r < n and arr[largest] < arr[r]:
        largest = r

    # Swap and continue heapifying if root is not largest
    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]
        heapify(arr, n, largest)


# Function to insert an element into the tree
def insert(array, newNum):
    size = len(array)
    if size == 0:
        array.append(newNum)
    else:
        array.append(newNum)
        for i in range((size // 2) - 1, -1, -1):
            heapify(array, size, i)


# Function to delete an element from the tree
def deleteNode(array, num):
    size = len(array)
    i = 0
    for i in range(0, size):
        if num == array[i]:
            break

    array[i], array[size - 1] = array[size - 1], array[i]

    array.remove(size - 1)

    for i in range((len(array) // 2) - 1, -1, -1):
        heapify(array, len(array), i)
```

```
arr = []

insert(arr, 3)
insert(arr, 4)
insert(arr, 9)
insert(arr, 5)
insert(arr, 2)

print ("Max-Heap array: " + str(arr))

deleteNode(arr, 4)
print("After deleting an element: " + str(arr))
```