Heapify

Wat is Heapify ?

Heapify is a process of converting an array into a heap data structure. A heap is a complete binary tree where the value of each node is greater than or equal to (in a max heap) or less than or equal to (in a min heap) the values of its children nodes.

Basically, in input, you have been given a Complete Binary Tree which is represented using arrays. So, heapify is the process of converting that Complete Binary Tree (CBT is represented using an array) into the heap data structure.

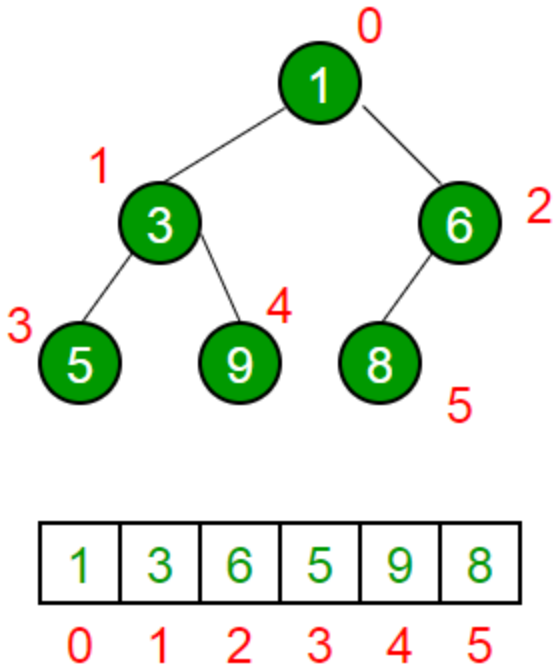
Heapify can be performed in two ways:

1. **Bottom-up heap construction:** In this method, the heap is constructed from the bottom up by starting with the leaf nodes and working upward to the root node. This is also known as the Floyd's algorithm or the heapify down algorithm.
2. **Top-down heap construction:** In this method, the heap is constructed from the top down by starting with the root node and working downward to the leaf nodes. This is also known as the heapify up algorithm.

Both methods have a time complexity of **O(nlogn)**, where n is the size of the array.

Leaf Nodes indexes in CBT:

Example for leaf nodes in 0-based indexing:



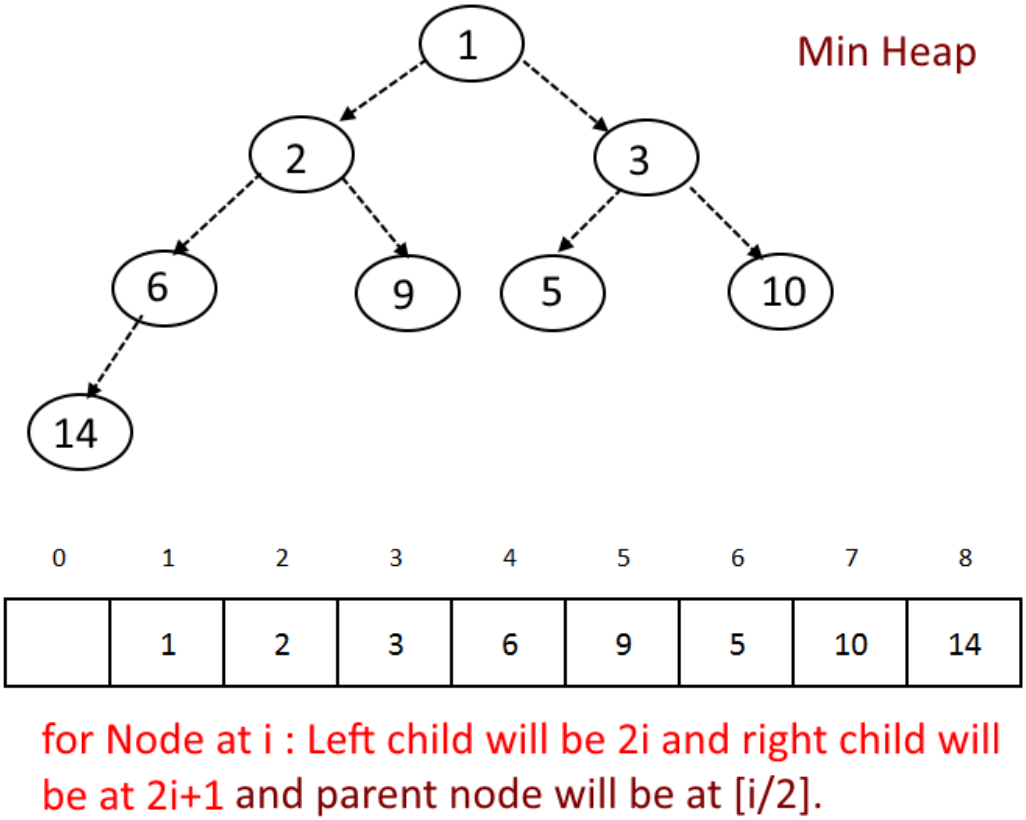
Above image is representation of MIN HEAP.

The number of nodes in above Min Heap is 6 and the size of array is also 6.

In 0-Based indexing:

1. Leaf Nodes starts from index **n/2 to n-1**, where n is the size of the array or n is the number of nodes in Heap.
2. In above Images, n = 6, n/2 ⇒ 6/2 ⇒ 3, and n-1 ⇒ 6-1 ⇒ 5, so from index 3 to index 5 all nodes are leaf nodes.
3. Nodes from index 3 to 5 are ⇒ 5, 9, and 8, so these nodes are leaf node.

Example for leaf nodes in 1-based indexing:



Above image is representation of MIN HEAP.

The number of nodes in above Min Heap is 8 and the size of array is 9 because in 1-based indexing we don't use the 0th index.

In 1-Based indexing:

1. **If n = size of array, then,**
    a. Leaf Nodes starts from index ((**n/2) + 1) to n-1**, where n is the size of the array.
    b. In above Images, size of array = 9, n = 9, ((n/2) + 1) ⇒ ((9/2) + 1) ⇒ 4 + 1 ⇒ 5, and n-1 ⇒ 9-1 ⇒ 8, so from index 5 to index 8 all nodes are leaf nodes.
    c. Nodes from index 5 to 8 are ⇒ 9, 5, 10 and 14, so these nodes are leaf node.

2. **If n = number of nodes in Heap, then,**

a. Leaf Nodes starts from index ((**n/2) + 1) to n**, where n is the number of nodes in Heap.

b. In above Images, number of nodes in heap = 9, n = 8, ((8/2) + 1) ⇒ ((8/2) + 1) ⇒ 4 + 1 ⇒ 5, and n = 8, so from index 5 to index 8 all nodes are leaf nodes.

c. Nodes from index 5 to 8 are ⇒ 9, 5, 10 and 14, so these nodes are leaf node.

```
/*
------------------------------------ What is Heapify -------------------------------
What is Heapify Algorithm ?
Heapify is a process of converting an array into a heap data structure.
A heap is a complete binary tree where the value of each node is greater than or
equal to (in a max heap) or less than or equal to (in a min heap) the values of its
children nodes.

Basically, in input, you have been given a Complete Binary Tree which is represented
using arrays. So, heapify is the process of converting that Complete Binary Tree
(CBT is represented using an array) into the heap data structure.


--------------------------------------- Leaf Nodes ---------------------------------
Leaf Nodes:
In 0-Based indexing:
    1. Leaf Nodes starts from index n/2 to n-1, where n is the size of the array or
       n is the number of nodes in Heap.

In 1-Based indexing:
1. If n = size of array, then,
    1.1. Leaf Nodes starts from index ( (n/2) + 1 ) to n-1,
         where n is the size of the array.

2. If n = number of nodes in Heap, then,
    2.1. Leaf Nodes starts from index ( (n/2) + 1 ) to n,
         where n is the number of nodes in Heap.


------------------------------------- Important Point ------------------------------
So, in Heapify what we are doing is, a Complete Binary Tree is given which is represented
using an array and our task is to convert this CBT(which is represented using an array)
into a heap either Max Heap or Min Heap.

I am using a Bottom-Up approach to convert the given array into the heap either Max Heap
or Min Heap.

So, In the Bottom-Up approach, we start traversing the array from the back and skip all
the leaf nodes and start the construction of heaf from the first non-leaf nodes.

Question: Why we are skipping the leaf nodes ?
```

```
Answer: We are skipping the leaf nodes because the leaf nodes do not have any children,
they cannot violate the heap property. Hence, we can safely skip them during the heap
construction process.

In Short, all leaf nodes are already statisfies the heap order preperty.
All leaf nodes in a heap data structure already satisfy the heap order property
by definition and hence there is no need to perform any heapify operation on them.

*/
```