Binary Heap

**Input**: arr[] = {4, 10, 3, 5, 1}

**Output**: Corresponding Max-Heap:

10

/ \

5 3

/ \

4 1

**Input**: arr[] = {1, 3, 5, 4, 6, 13, 10, 9, 8, 15, 17}

**Output**: Corresponding Max-Heap:

17

/ \

15 13

/ \ / \

9 6 5 10

/ \ / \

4 8 3 1

The corresponding complete binary tree for this array of elements [4, 10, 3, 5, 1] will be:

4

/ \

10 3

/ \

5 1

**Note**:

Root is at index 0 in array.

Left child of i-th node is at (2\*i + 1)th index.

Right child of i-th node is at (2\*i + 2)th index.

Parent of i-th node is at (i-1)/2 index.

**Simple Approach:** Suppose, we need to build a Max-Heap from the above-given array elements. It can be clearly seen that the above complete binary tree formed does not follow the Heap property. So, the idea is to heapify the complete binary tree formed from the array in reverse level order following top-down approach.

That is first heapify, the last node in level order traversal of the tree, then heapify the second last node and so on.

**Time Complexity:** Heapify a single node takes O(Log N) time complexity where N is the total number of Nodes. Therefore, building the entire Heap will take N heapify operations and the total time complexity will be **O(N\*logN)**.

**Optimized Approach:** The above approach can be optimized by observing the fact that the leaf nodes need not to be *heapified* as they already follow the heap property. Also, the array representation of the complete binary tree contains the level order traversal of the tree.

So, the idea is to find the position of the last non-leaf node and perform the **heapify**operation of each non-leaf node in reverse level order.

**Last non-leaf node** = parent of last-node.

or, Last non-leaf node = parent of node at (n-1)th index.

or, Last non-leaf node = Node at index ((n-1) - 1)/2.

= (n/2) - 1.

**Illustration**:

Array = {1, 3, 5, 4, 6, 13, 10, 9, 8, 15, 17}

Corresponding Complete Binary Tree is:

1

/ \

3 5

/ \ / \

4 6 13 10

/ \ / \

9 8 15 17

***The task to build a Max-Heap from above array***.

Total Nodes = 11.

Last Non-leaf node index = (11/2) - 1 = 4.

Therefore, last non-leaf node = 6.

To build the heap, heapify only the nodes:

[1, 3, 5, 4, 6] in reverse order.

**Heapify 6**: Swap 6 and 17.

1

/ \

3 5

/ \ / \

4 17 13 10

/ \ / \

9 8 15 6

**Heapify 4**: Swap 4 and 9.

1

/ \

3 5

/ \ / \

9 17 13 10

/ \ / \

4 8 15 6

**Heapify 5**: Swap 13 and 5.

1

/ \

3 13

/ \ / \

9 17 5 10

/ \ / \

4 8 15 6

**Heapify 3**: First Swap 3 and 17, again swap 3 and 15.

1

/ \

17 13

/ \ / \

9 15 5 10

/ \ / \

4 8 3 6

**Heapify 1**: First Swap 1 and 17, again swap 1 and 15,

finally swap 1 and 6.

17

/ \

15 13

/ \ / \

9 6 5 10

/ \ / \

4 8 3 1