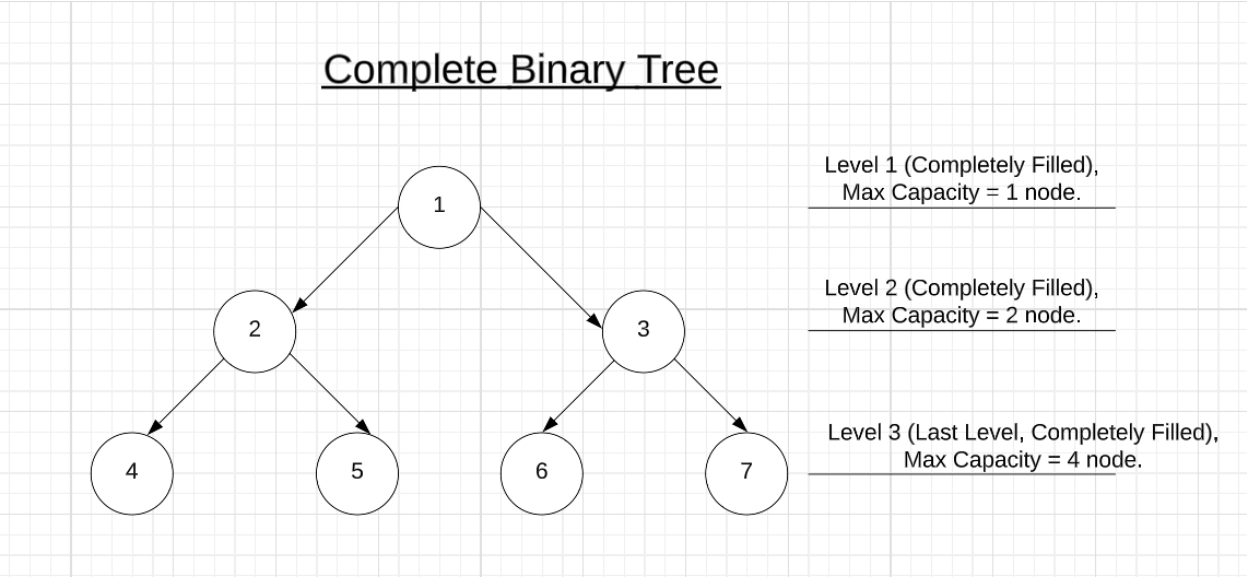
Heap Data Structure

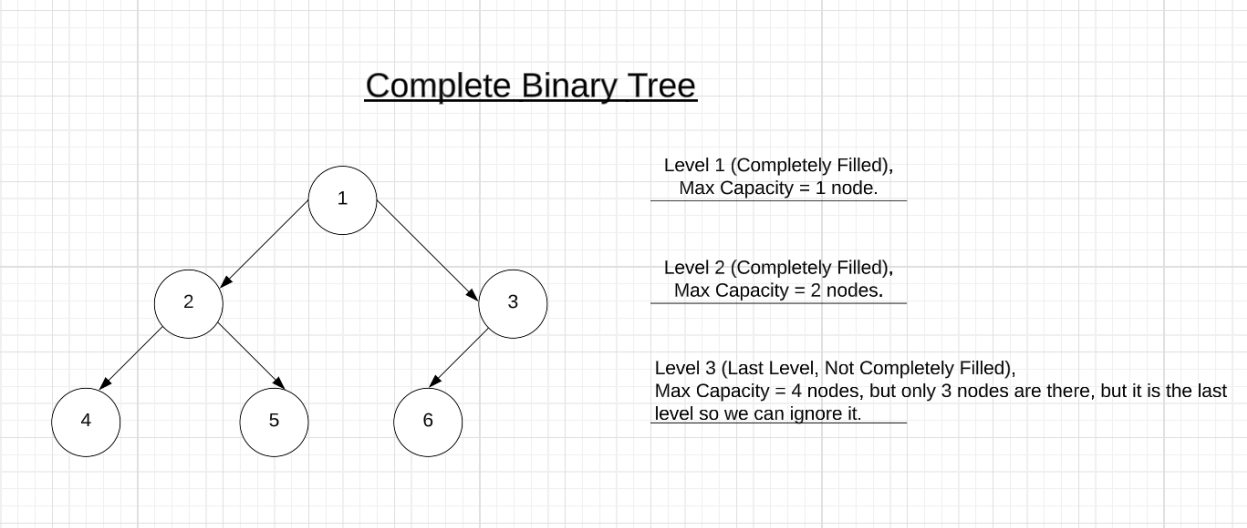
What is Heap ?

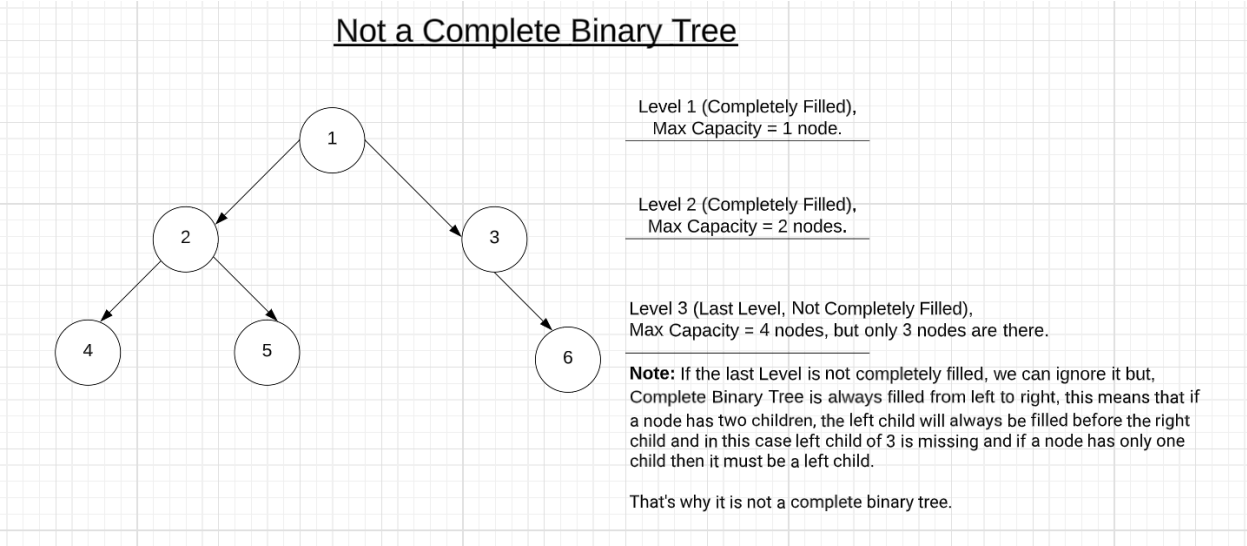
1. *A Heap is a special **Tree-based** data structure in which the **tree is a complete binary tree.***
2. *Heap comes with a special property called as **Heap Order property.***
3. In simple words, we can say that, A heap is a special tree-based data structure that satisfies the heap property and in which the tree is a Complete Binary Tree.

What is Complete Binary Tree (AKA CBT) ?

1. A binary tree is called complete binary tree **when all the level of binary tree is completely filled except the last level (last level can be completely filled or cannot be completely filled).**
2. A Complete Binary Tree is always filled from left to right. This means that if a node has two children, the left child will always be filled before the right child.
3. And if a node has only one child, then that child must be left child because of the order that we have just discussed in point 2.





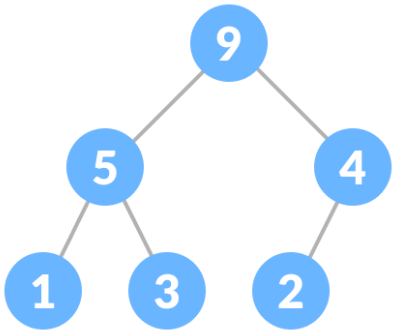


What is Heap Order Property ?

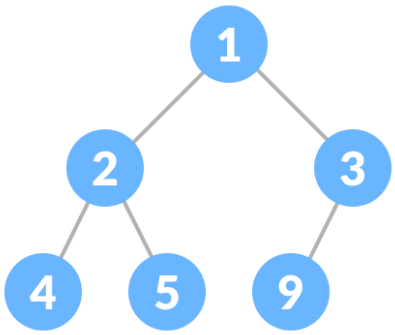
The heap order property says that, for every node, the value of the root node or parent node must be (≥ or ≤) than the value of its child nodes.

In Simple Words, Heap Order property is,

1. For every node,
    a. The value of root node or parent node ≥ the value of its child nodes or,



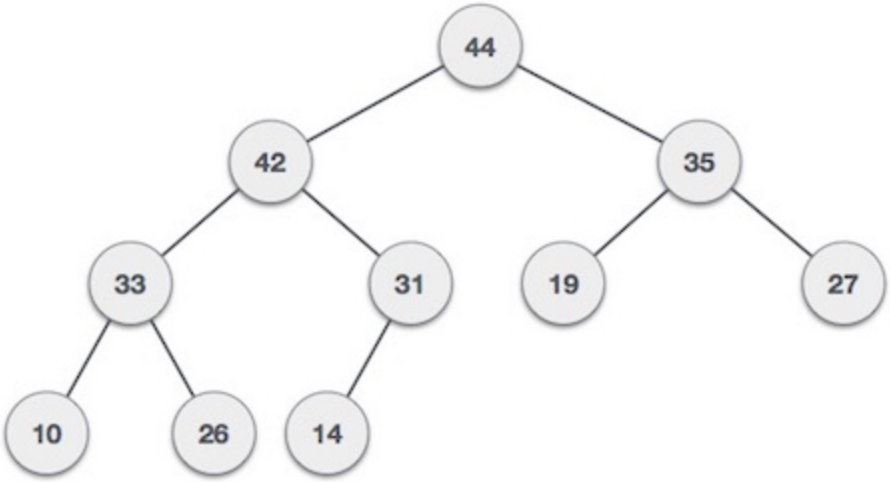
b. The value of root node or parent node ≤ the value of its child nodes .



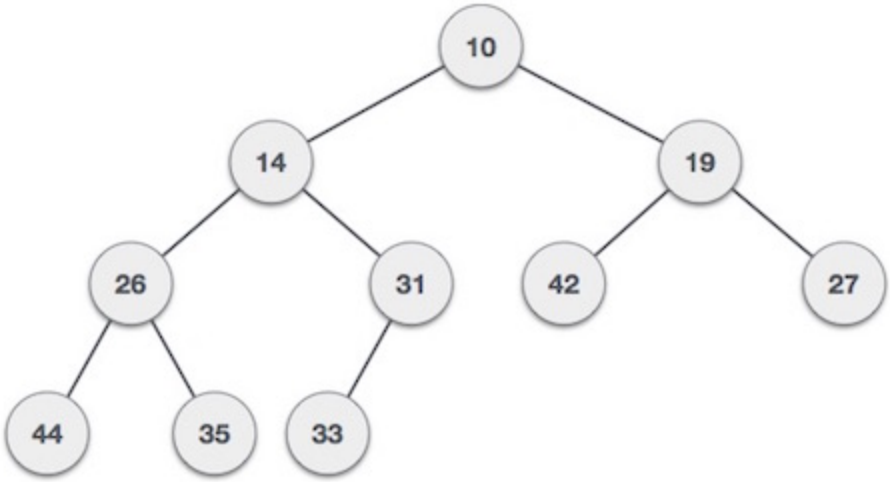
Types of Heap ?

Based on the property of heap (Heap Order property), we can classify heaps into two types,

1. **Max Heap**
    a. The value of root node or parent node ≥ the value of its child nodes.



2. **Min Heap**
    a. The value of root node or parent node ≤ the value of its child nodes .



How to implement Heap ?

Arrays are commonly used to implement heaps because heap are complete binary tree so there will be not wastage of memory.

In an array-based representation of a heap, the elements of the heap are stored in an array, where the root of the tree is located at index 0 and each child of a node at index I is located at indices 2i+1 and 2i+2.

In Short,

1. **In 0-based Indexing**
    a. Left Child = 2*i+1
    b. Right Child = 2*i+2
    c. Parent Node = ((i-1) / 2).
    d. Leaf Nodes = [n/2, n-1], n is the size of array in integers.
2. **In 1-based Indexing**
    a. Left Child = 2*i
    b. Right Child = 2*i+1
    c. Parent Node = (i / 2).
    d. Leaf Nodes = [((n/2)+ 1), n], n is the size of array in integers.