Heap

Heap is a binary tree, which satisfies the following two properties :-

i) Structure Property = All the levels have maximum number of nodes except possibly the last level. In the last level, all the nodes occur to the left.

ii) Heap Order Property = The key value in any node N is greater than or equal to the key values in both its children.

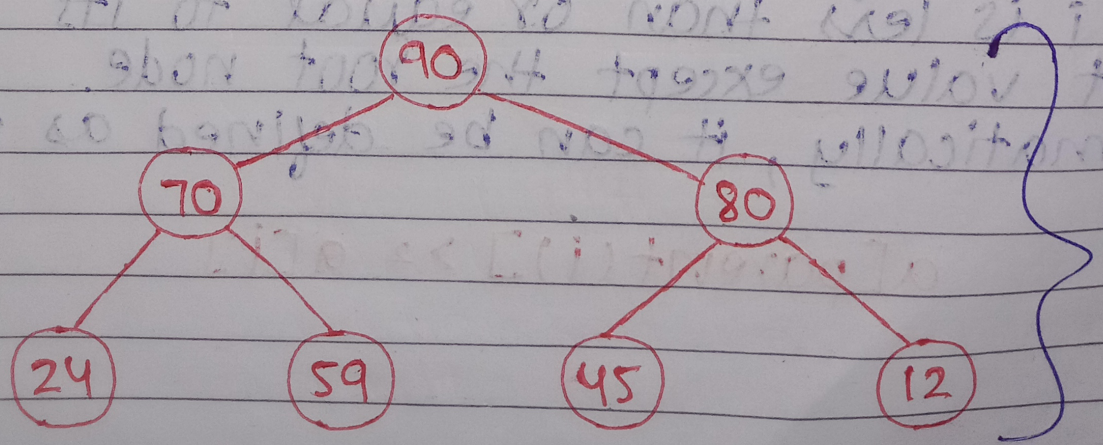
From the Structure Property, we can say that a heap is a complete binary tree.

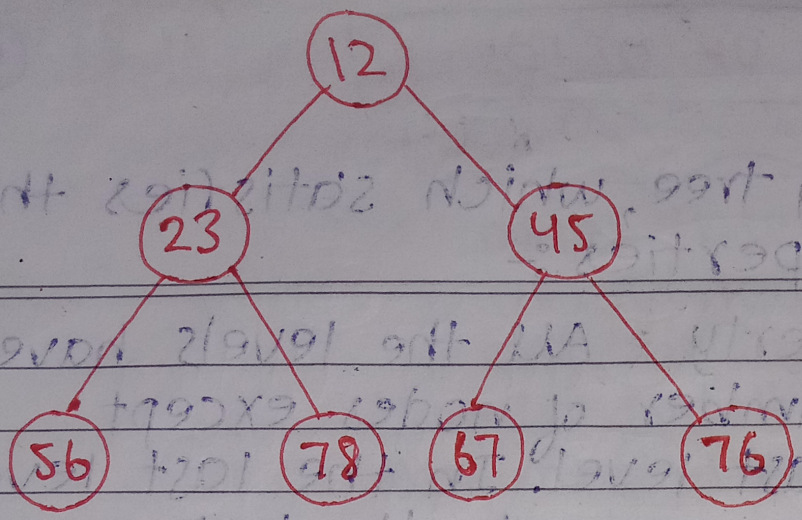
From the Heap Order Property, we can say that key value in any node N is greater than or equal to the key value in each successor of node N. This means that root node will have the highest key value.

Heap can be of two types :-

i) min heap ⟶ root will have min element.

ii) max heap ⟶ root will have max element.





* Min Heap = The value of the parent node should be less than or equal to either of its children.

In other words, the min heap can be defined as, for every node i, the value of node i is greater than or equal to its parent value except the root node.

Mathematically, it can be defined as :-

$$a[parent(i)] <= a[i]$$

* Max Heap = The value of the parent node should be greater than or equal to either of its children.

In other words, the max heap can be defined as, for every node i, the value of node i is less than or equal to its parent value except the root node.

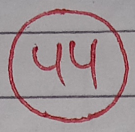
Mathematically, it can be defined as :-

$$a[parent(i)] >= a[i]$$

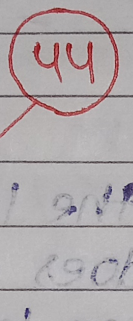
Ques Insertion in Max Heap Tree :-

44, 33, 77, 11, 55, 88, 66

Insert 44,

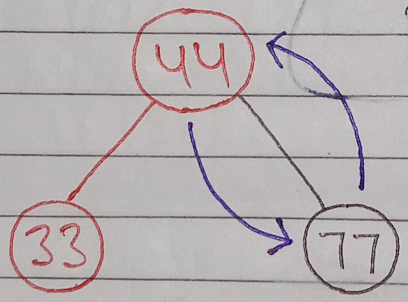


Insert 33,

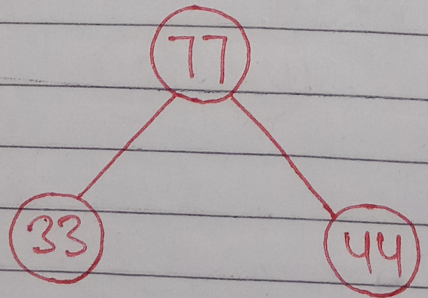


insertion in B-Tree always starts with left side, and Heap follows B-Tree properties. So, 33 will added in the left of 44.

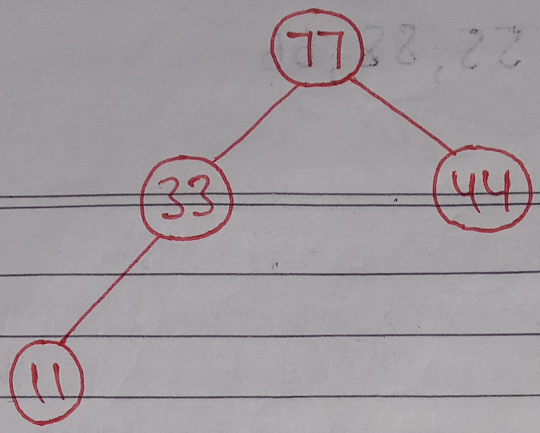
Insert 77,



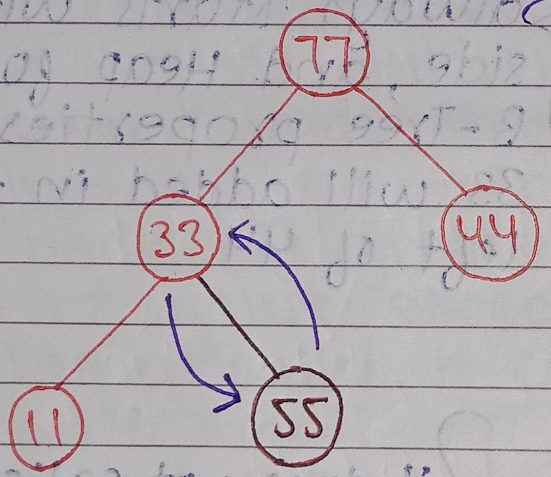
it does not satisfy the max heap property, because 44 is less than 77. So, we swap these values.



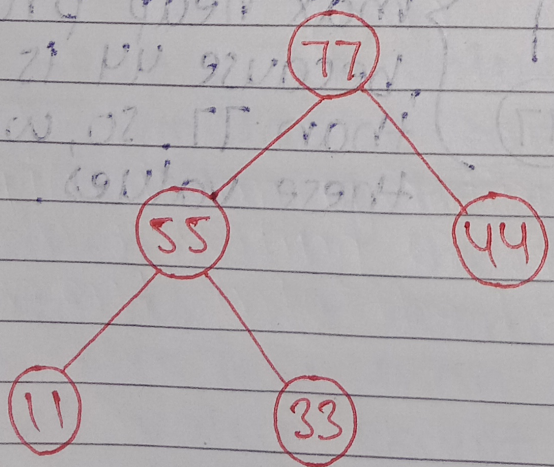
Insert 11,



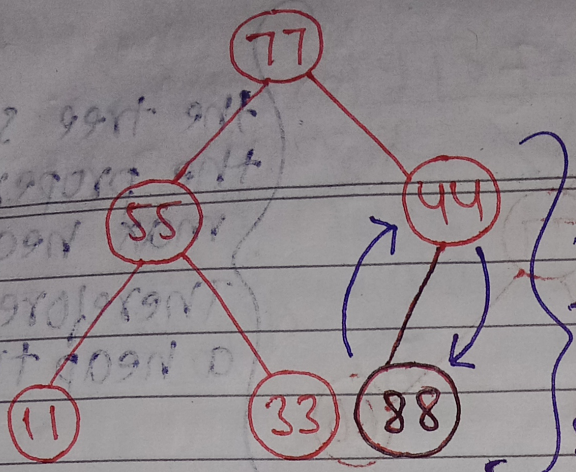
Insert 55,



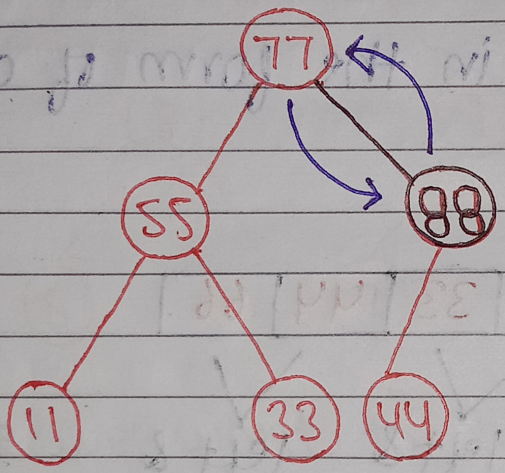
the left subtree of 77 does not satisfy the max heap property, as 33 is less than 55. So, we swap these values.



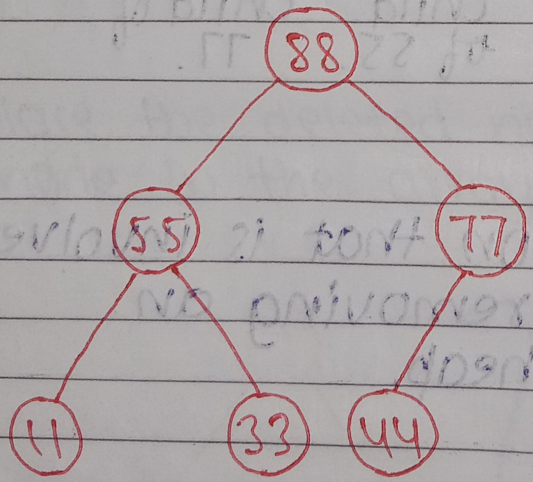
Insert 88,



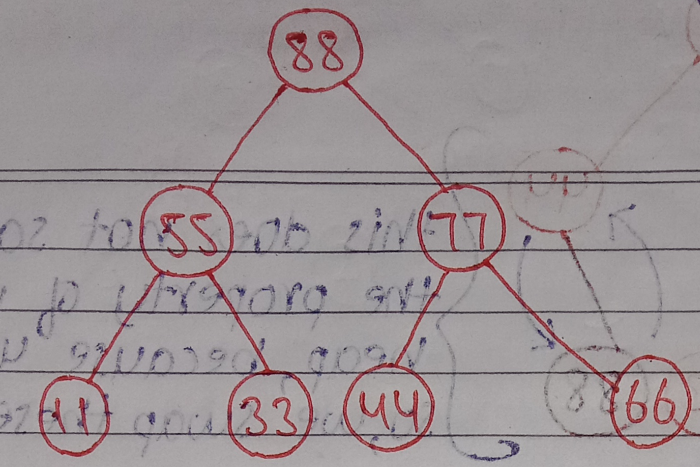
this does not satisfy the property of max heap, because 44 < 88. so, we swap these values.



again its voilating the max heap property. so, we swap 88 with 77.

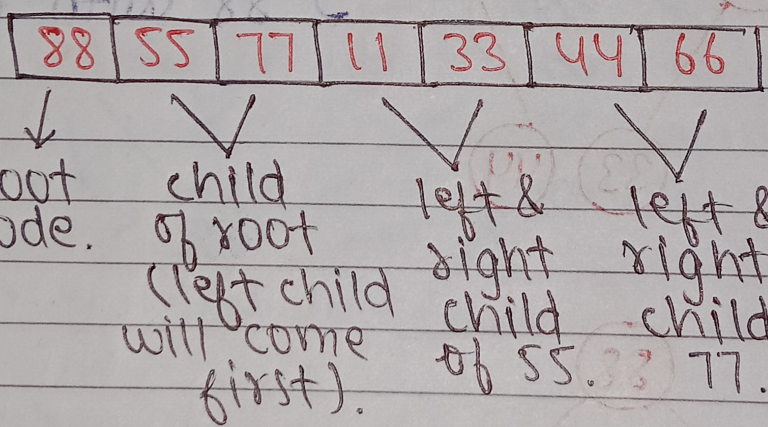


Insert 66,



the tree stisfy the property of max heap. Therefore, it is a heap tree.

If we store this tree in the form of array, then its like this,



Heapify is an operation that is involved when inserting or removing an element into the heap.

Ques The array below stores the Max Heap. Draw the tree version of Max Heap.

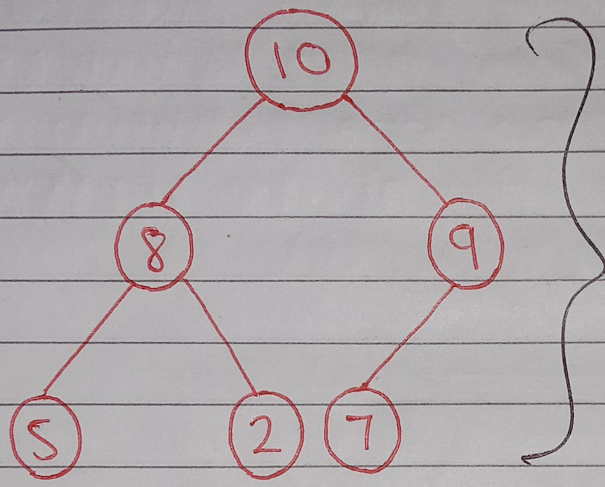
10	8	9	5	2	7

first one is always root

left & right of root

left & right of node 8

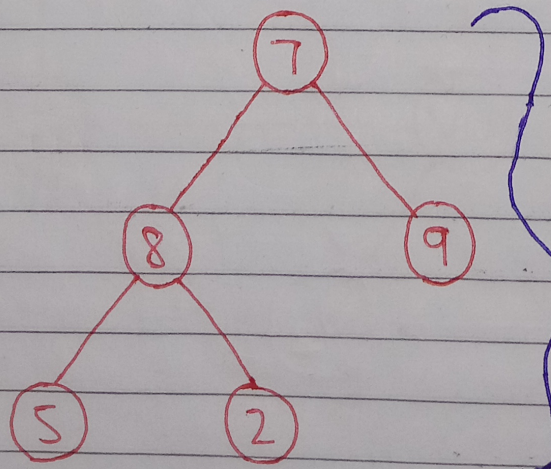
only left of node 9.



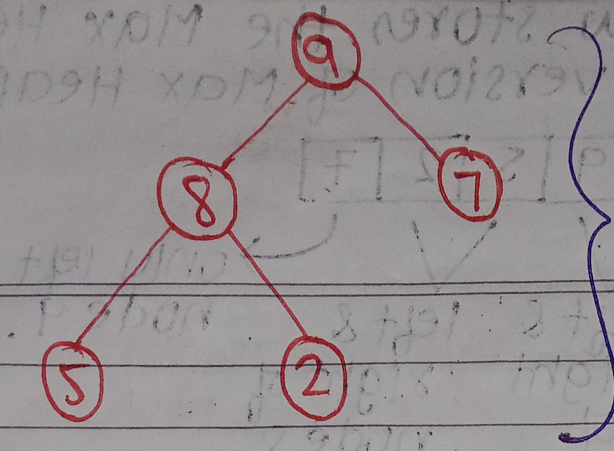
Delete the maximum element from the tree.

The node we want to delete from this tree is maximum element node, i.e., the root node, i.e., 10.

we replace the deleted node with the farthest right node in the array, i.e., we replace 10 with 7.



now we do Heapify. Heapify means Fix the Heap. 7 is less than both of the childs. So, we swap 7 with the largest child, i.e., 9.



now, the heap property is satisfy so, its a heap tree.