DATA STRUCTURES (017013292)
Semester – II
Chapter Name: TREE DATA STRUCTURE

Contents

1. Introduction To Heap

In computer science, a heap is a specialized tree-based data structure that satisfies the heap property:

A Binary Heap is a Binary Tree with following properties.

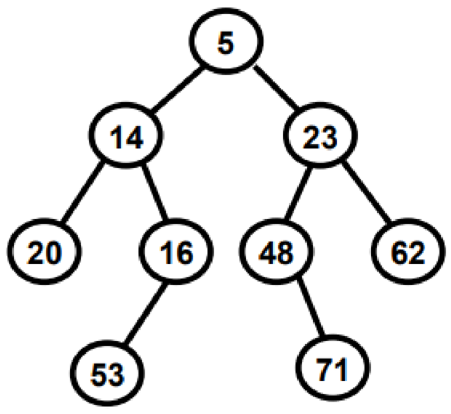
1) It's a complete tree (All levels are completely filled except possibly the last level and the last level has all keys as left as possible). This property of Binary Heap makes them suitable to be stored in an array.
2) A Binary Heap is either Min Heap or Max Heap. In a Min Binary Heap, the key at root must be minimum among all keys present in Binary Heap. The same property must be recursively true for all nodes in Binary Tree. Max Binary Heap is similar to Min Heap.

A Heap is Almost Complete Binary Tree. (ACBT)

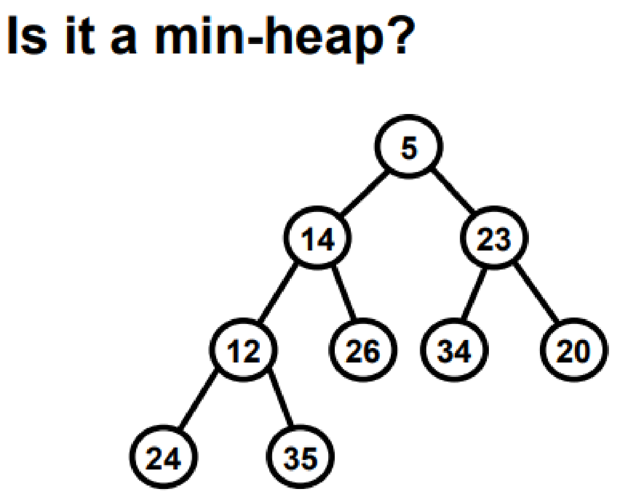
2. Min Heap

A min-heap is a binary tree such that - the data contained in each node is less than (or equal to) the data in that node's children.

Is it a min-heap?



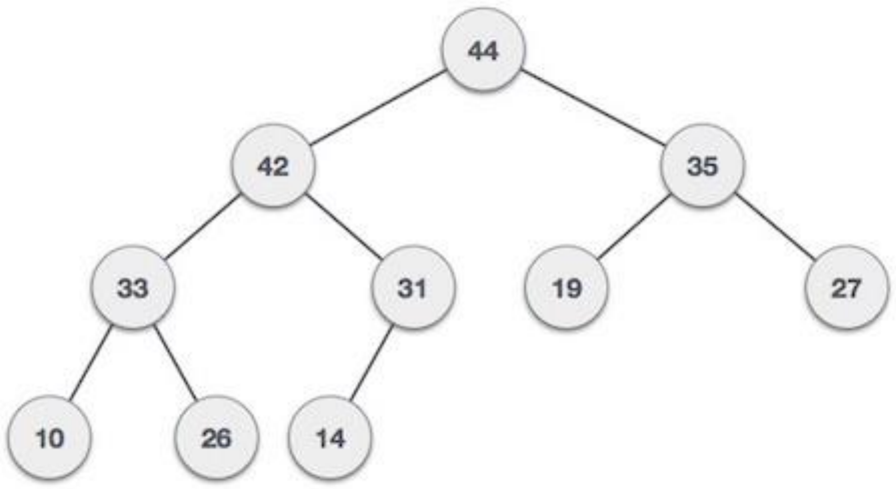
Yes. As All the parent nodes are lesser than their child nodes – it is a Min Heap.



No, this isn't a min – heap as the value 14 as parent is not lesser then its child node 12.

3. Max-Heap

A max-heap is a binary tree such that - the data contained in each node is greater than (or equal to) the data in that node's children.



4. Insert In Min Heap

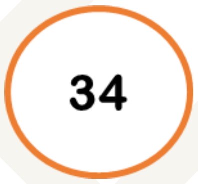
Consider a following example to understand the concept of insertion in Min-Heap.

Q1. Construct a Min-Heap for the following data: 34, 22, 42, 16, 19, 13, 17, 12, 5.

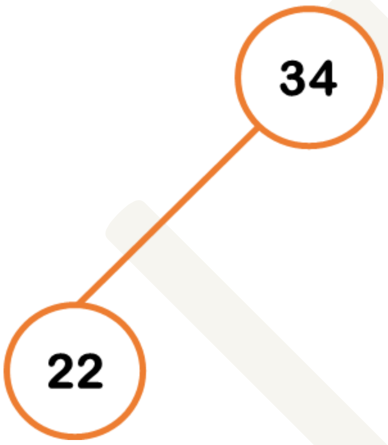
Solution:

To construct a Min-Heap, we need to insert one value at a time and need to verify the Min-Heap Property that (Parent ≤ Child).

Step 1 – 1st value as a root.

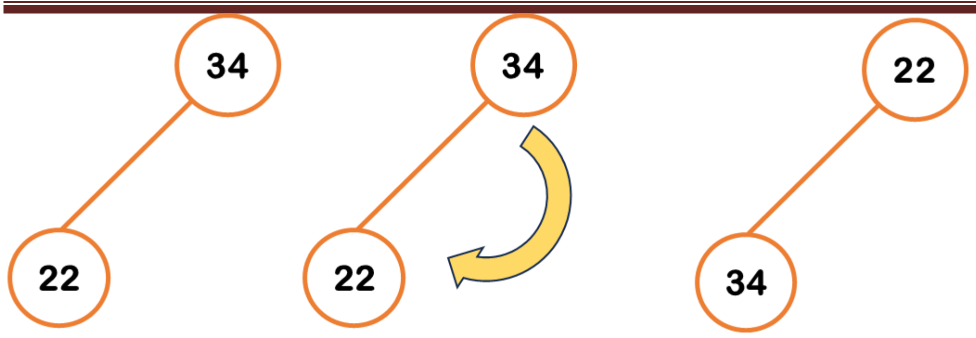


Step 2 – Insert next value. Which will be placed as the left child of 34.

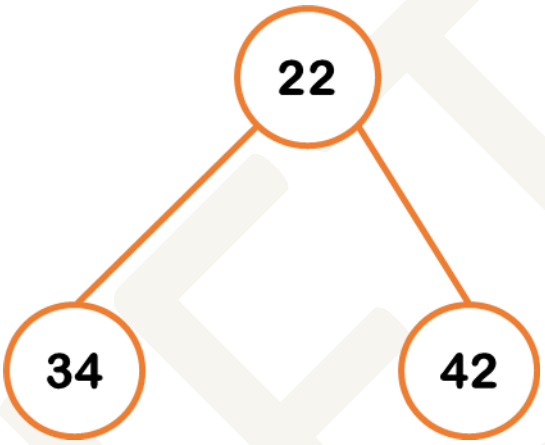


But, here now, checking the Min- Heap property for 34 (Parent) & 22(Child). It doesn't satisfy.

Hence, we need to swap 34 & 22. Now, 22 will be the Parent and 34 will be child.

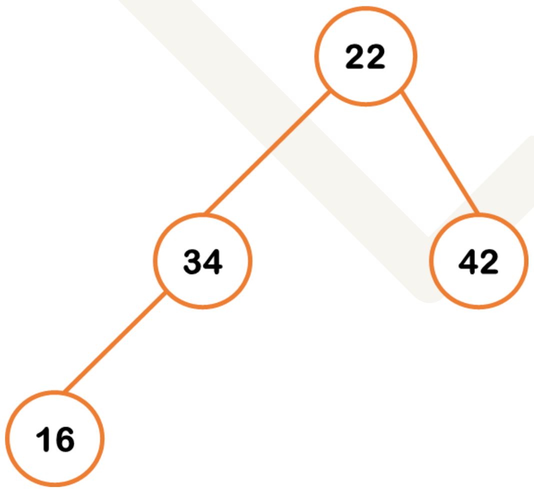


Step 3 – Now, insert value 42. 42 will be placed as the right child of 22.

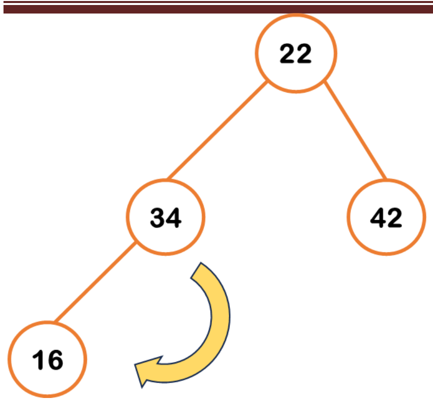


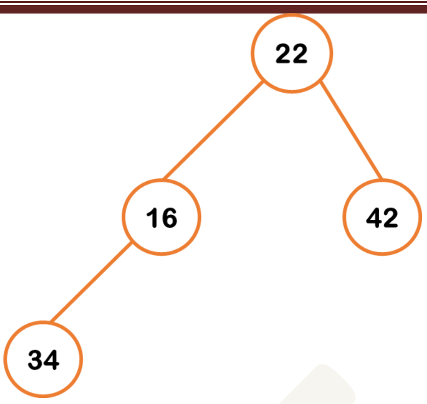
After insertion again checking the parent-child property. Here Parent (22) is lesser than both the children (34 & 42).

Step 4 – Now, inserting value 16. That will be inserted as the position of left child of 34.

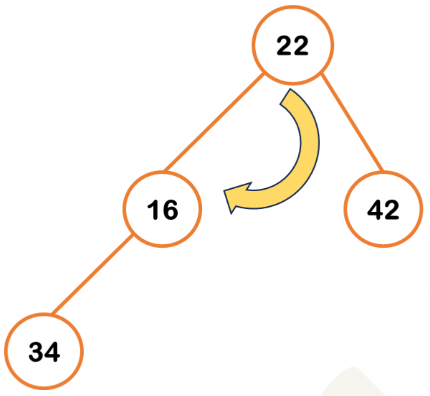


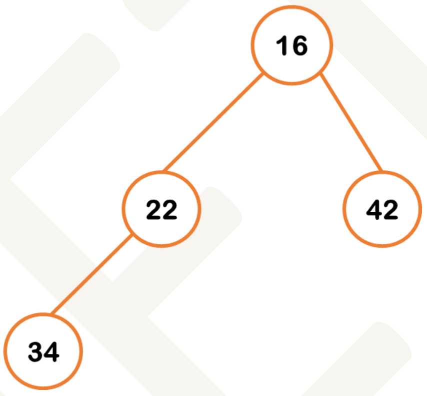
But here as you can see – Parent (34) is not less than its child (16). Hence swapping will occur.



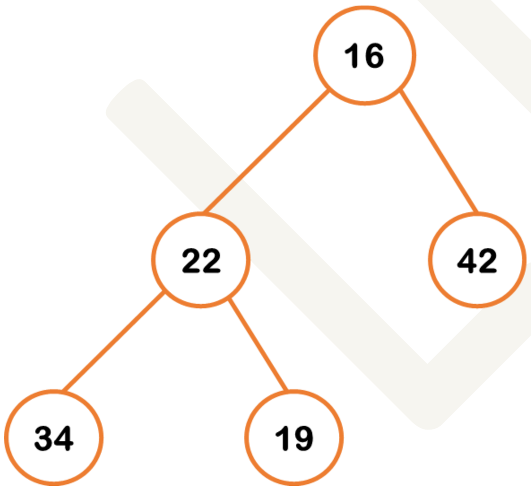


But Now again, the parent (22) is greater than it’s child (16) – which doesn’t satisfy the property of Min-Heap. Hence another swapping would be required.

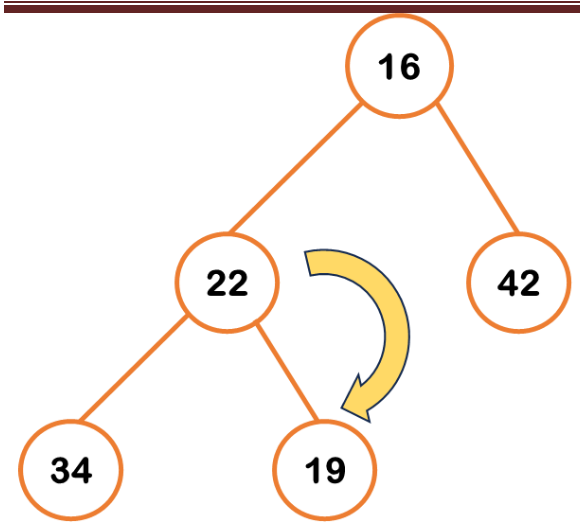


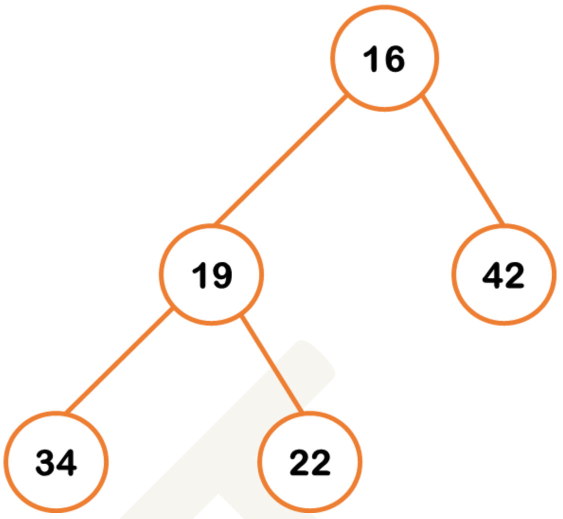


Step 5 – Now, insert 19, which will be placed as the Right child of 22.

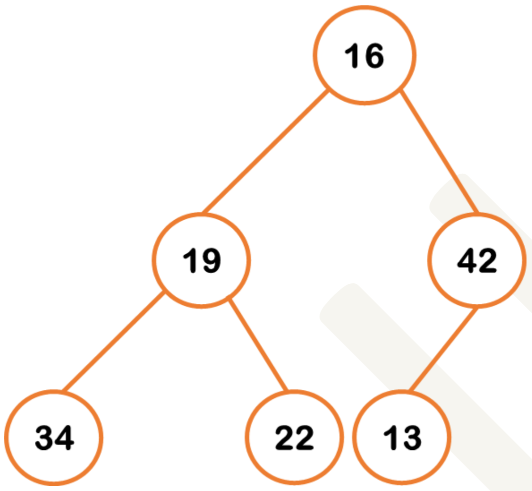


At this point parent-child property is not satisfied for parent (22) & child (19).

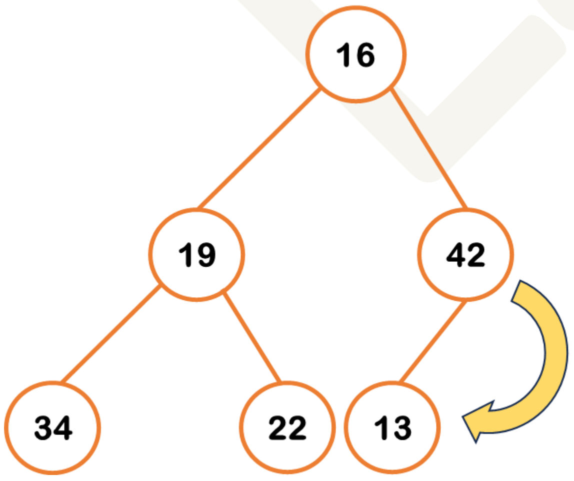


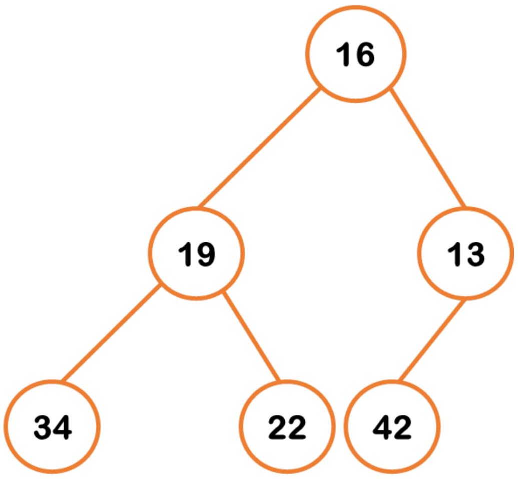


Step 6 – Now, Insert 13.

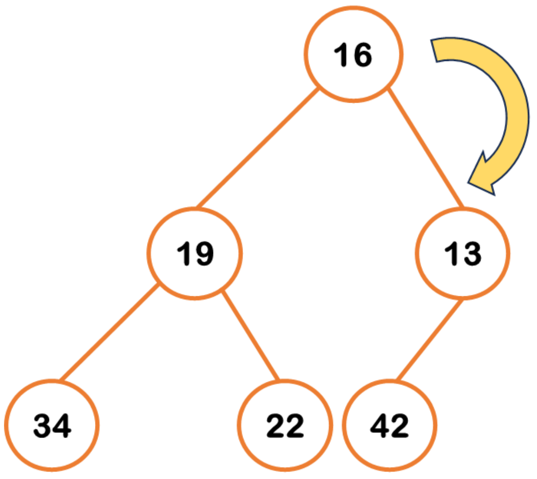


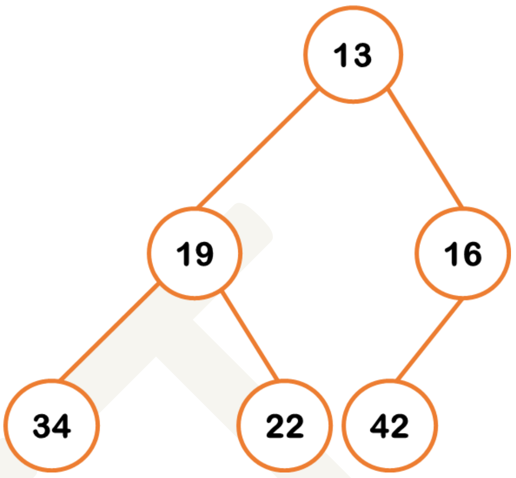
Now, again the property is not getting satisfied for parent (42) & child (13). Hence swapping will happen.



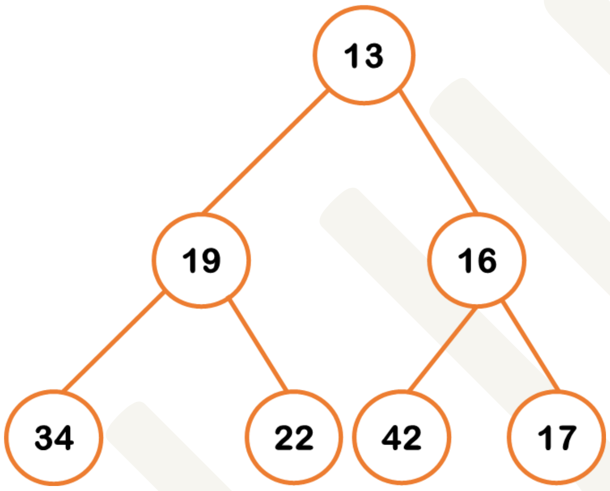


Now, again Parent (16) is greater than the child (13), hence one more swapping is required.



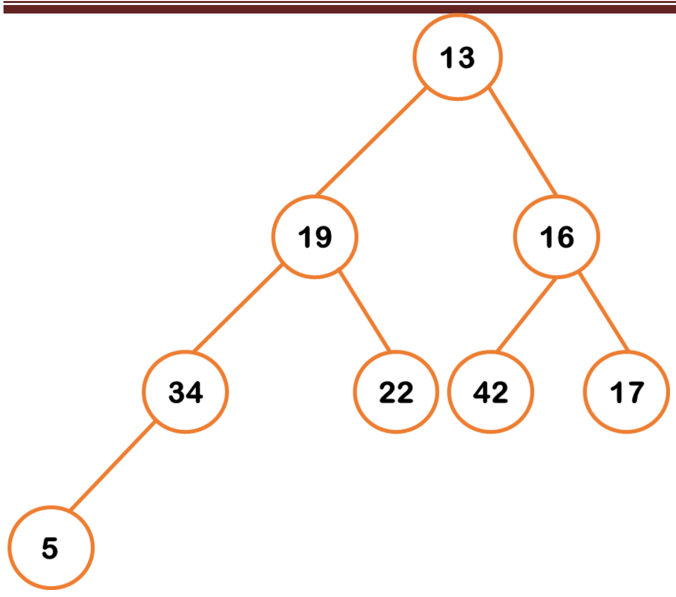


Step 7 – Now, Insert 17.

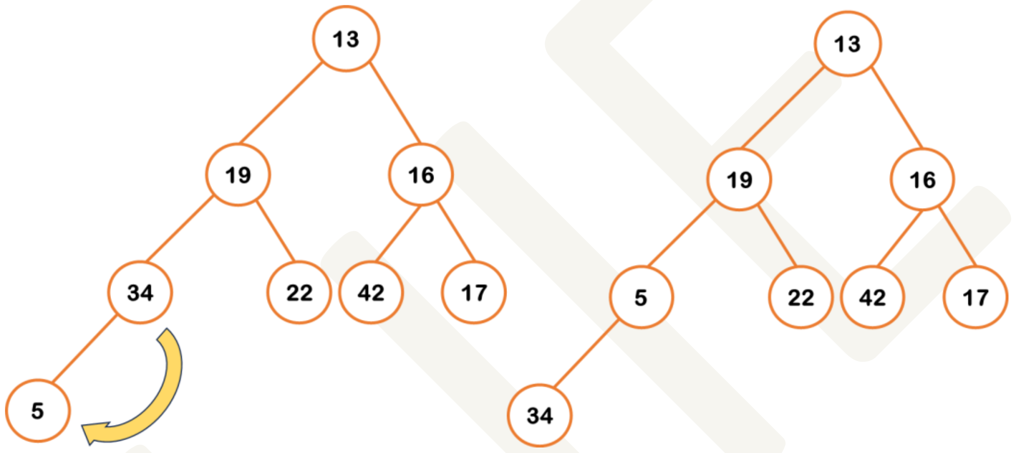


At this point all levels parent-child property is getting satisfied.

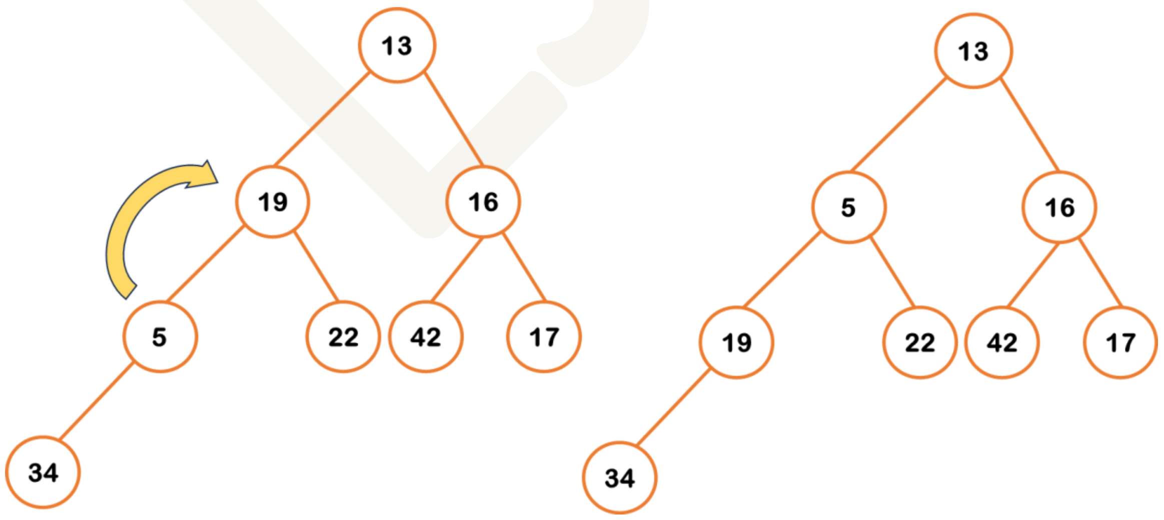
Step 8 – Now, Insert 5.



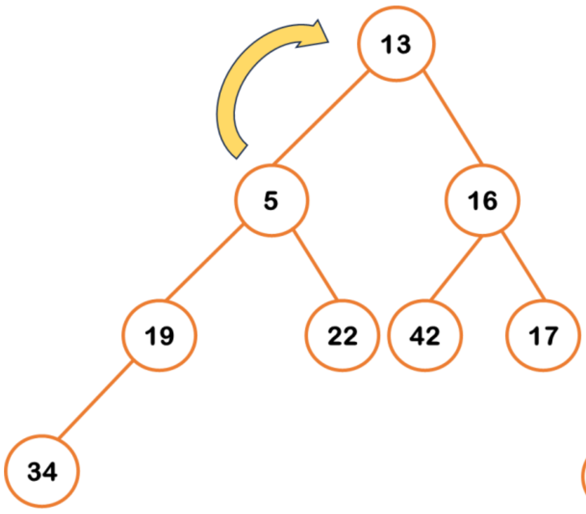
As the property is not getting satisfied, Swapping is required.

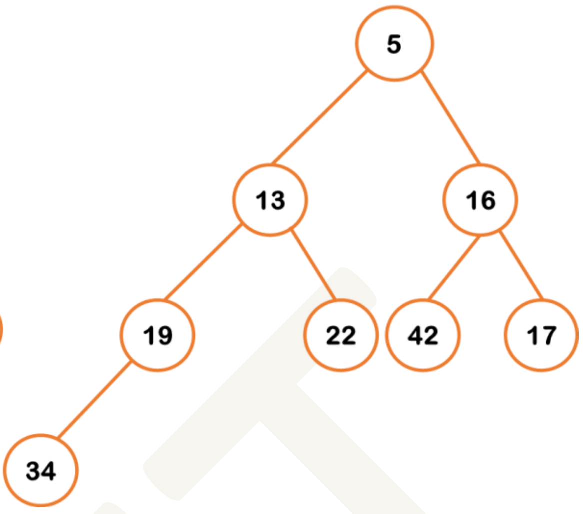


Now, another swap is required.



Now, another swap is required.





This is our final Min-Heap tree after all the insertion.

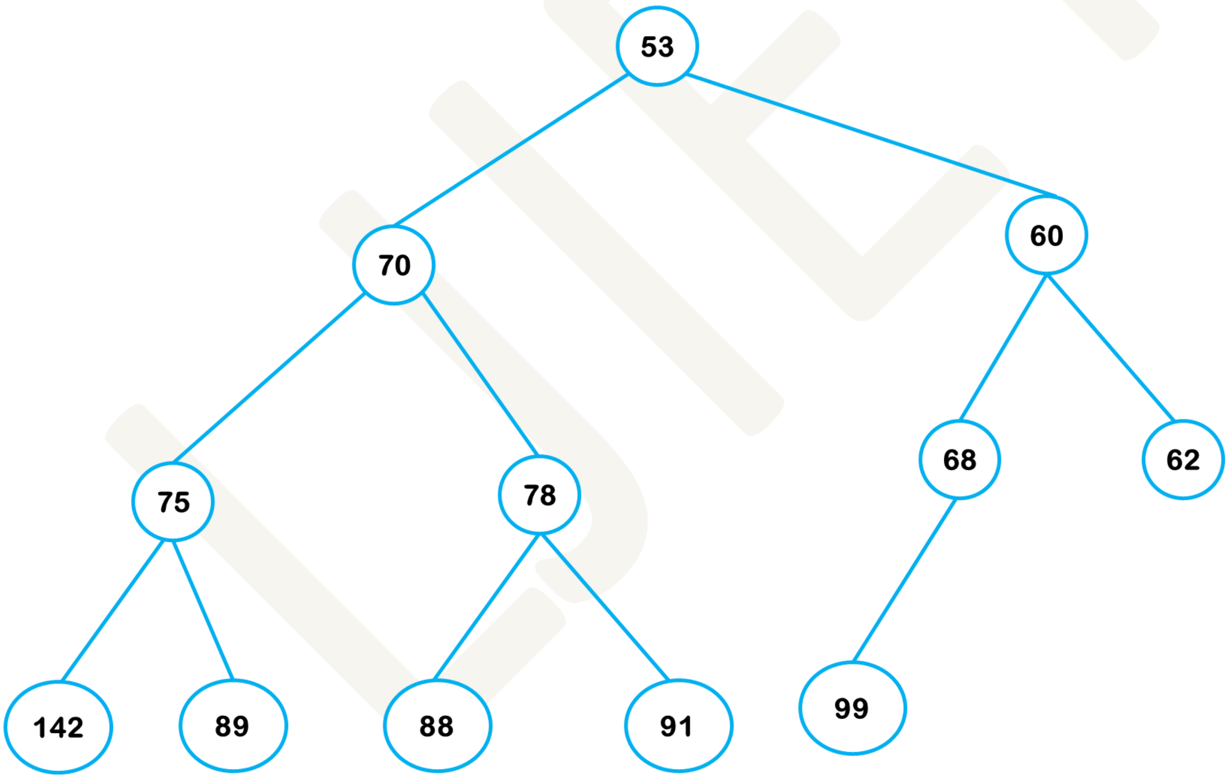
5. Delete In Min Heap

IN ANY HEAP TREE – THE DELETION OPERATION HAPPENS ONLY ON THE ROOT NODE. HENCE, THE ROOT NODE GETS DELETED IN THE TREE AND THE LAST ELEMENT OF THE TREE BECOMES THE NEW ROOT OF THE HEAP TREE.

ONCE THE LAST ELEMENT OF THE HEAP TREE BECOMES THE NEW ROOT AFTER DELETION, WE NEED TO CHECK THE PARENT-CHILD PROPERTY AGAIN.

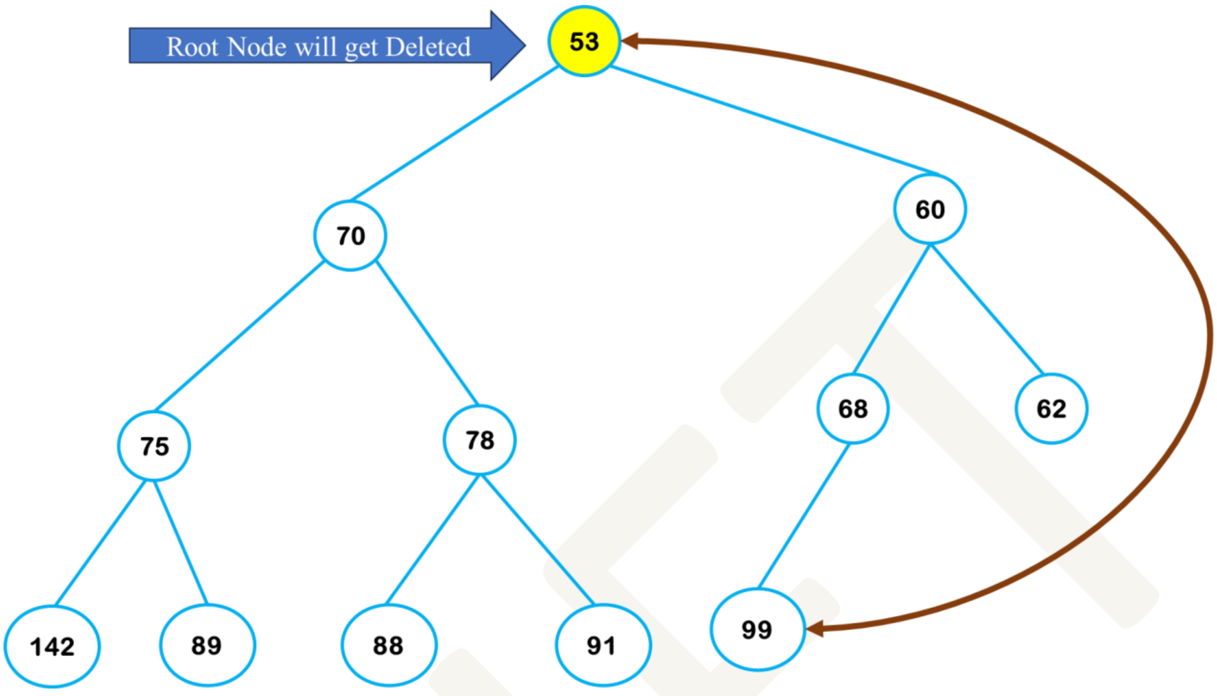
To understand the fundamental of deletion – let us see an example.

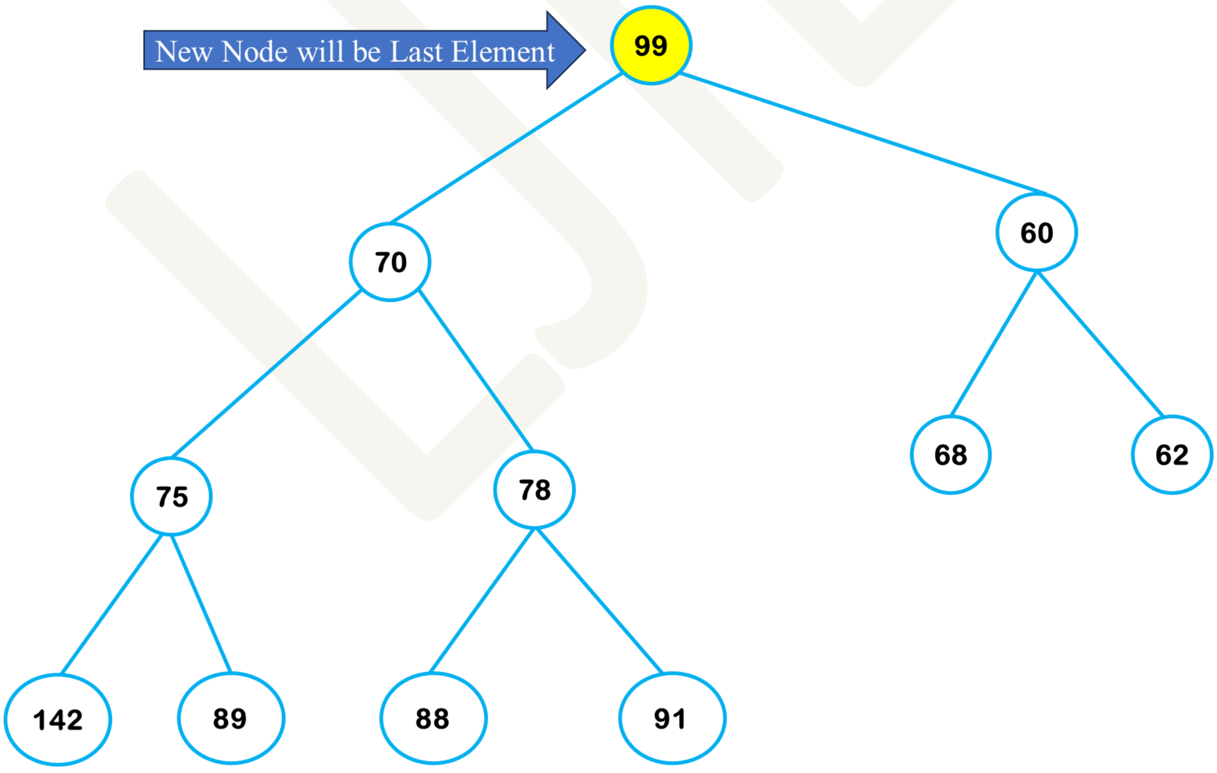
Q. Consider the following Min-heap tree and perform the delete operation twice.



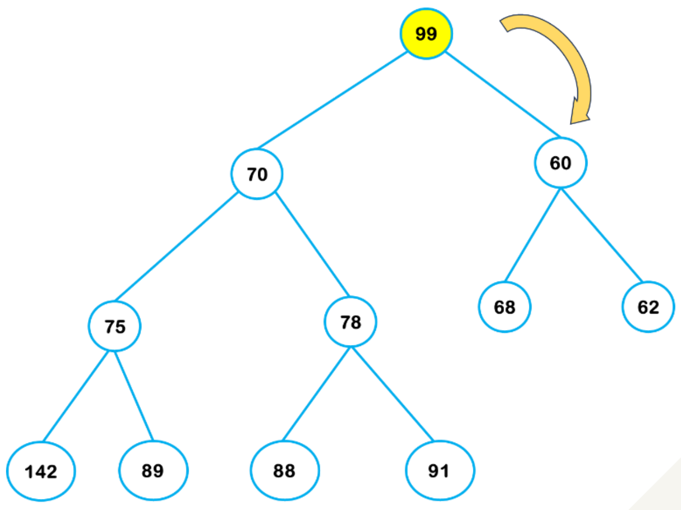
Solution:

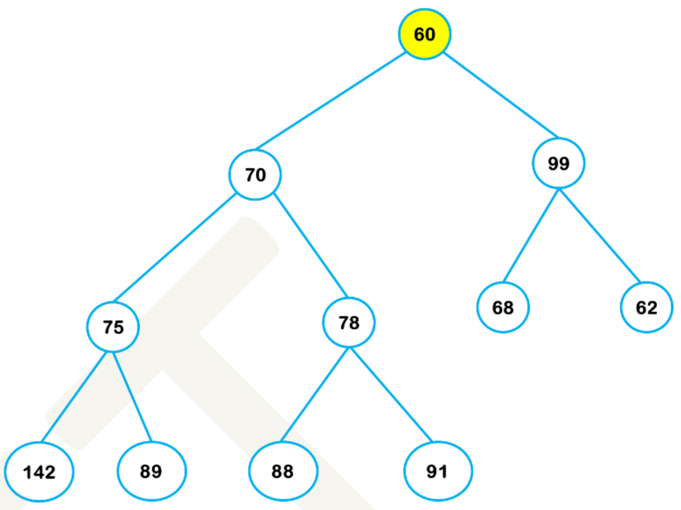
Step 1: Delete root for 1st delete operation.

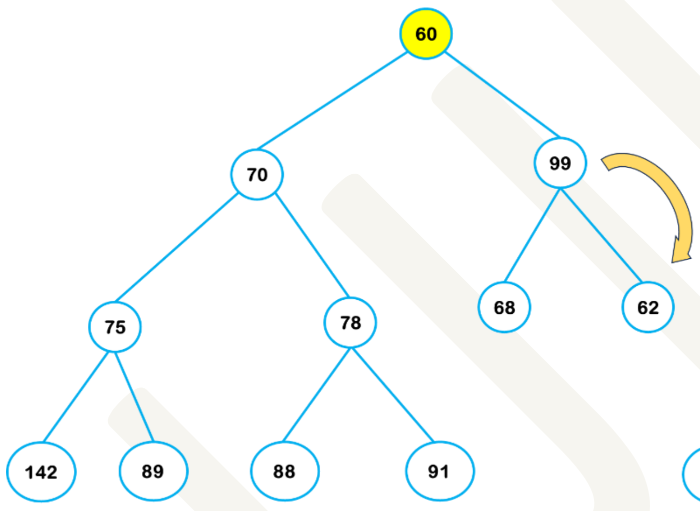


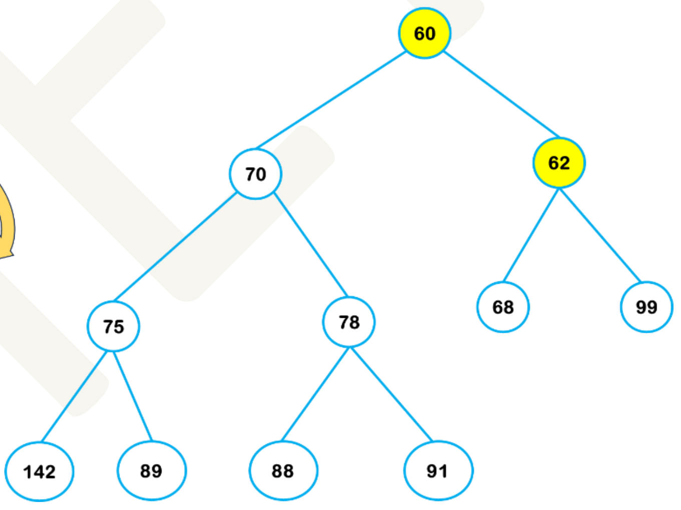


Now, the delete operation has been performed but we need to check the parent-child property at each node.

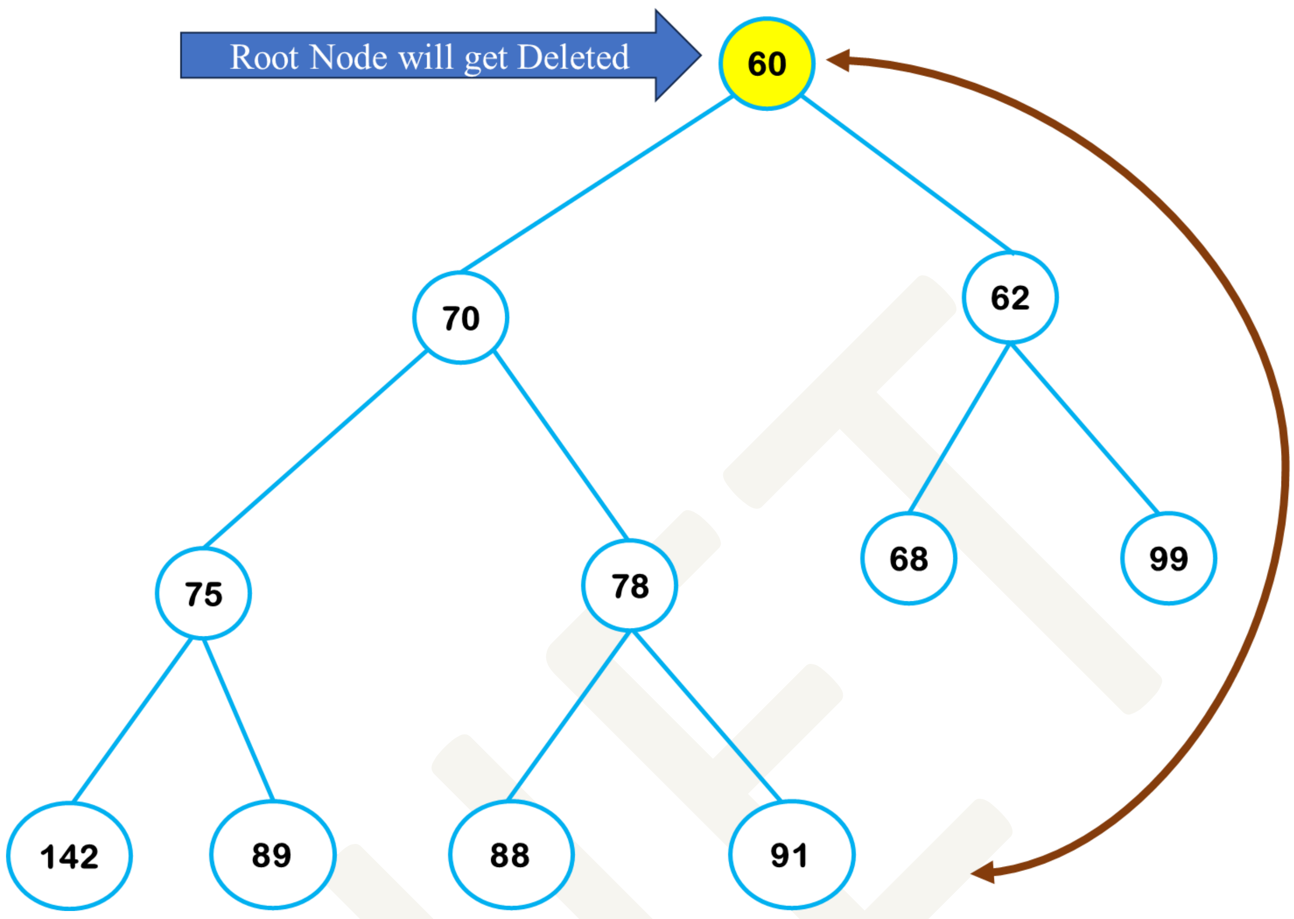


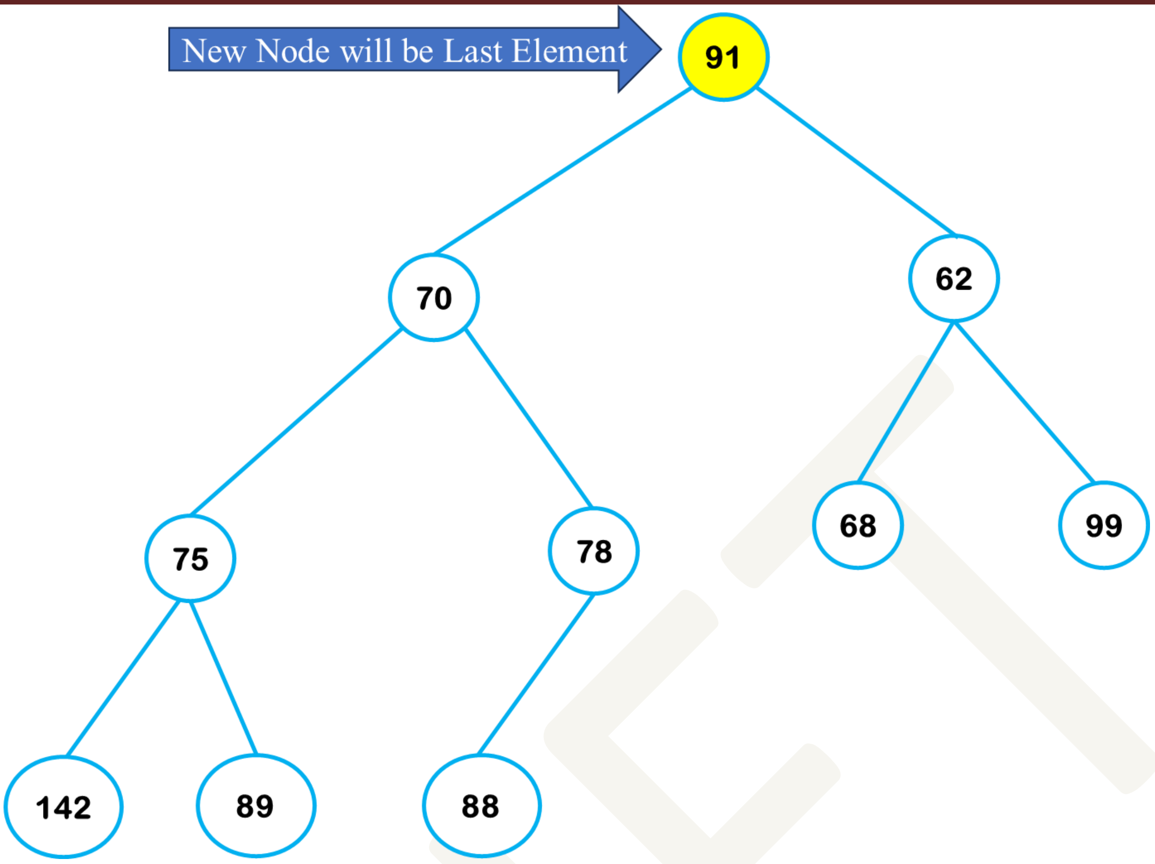




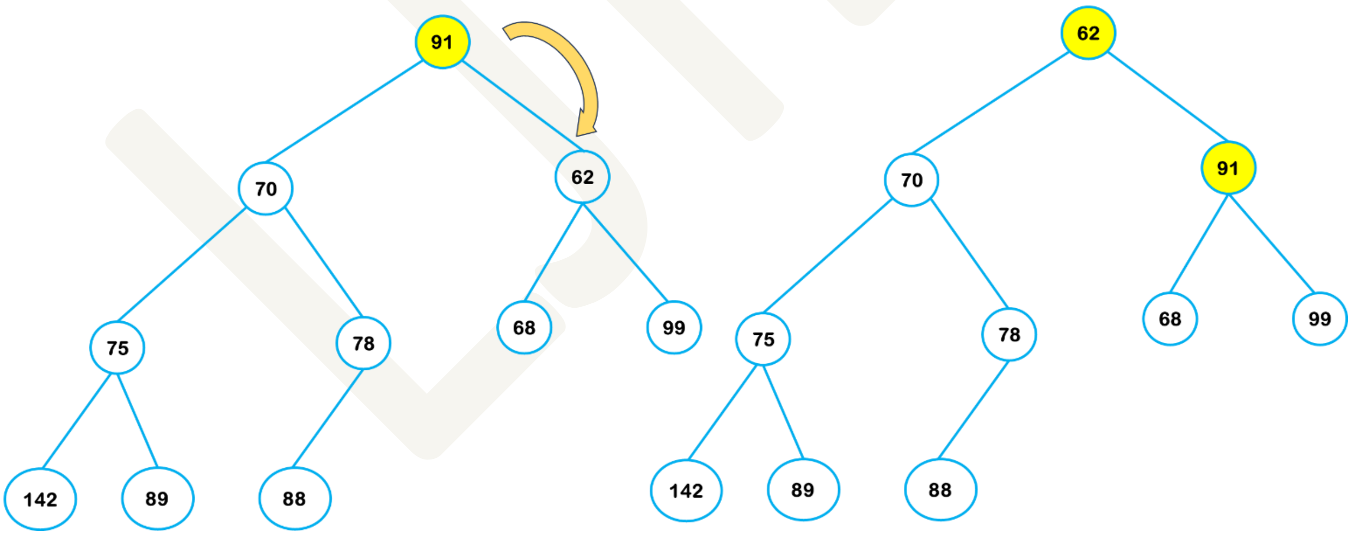


Step 2 – Another delete operation.

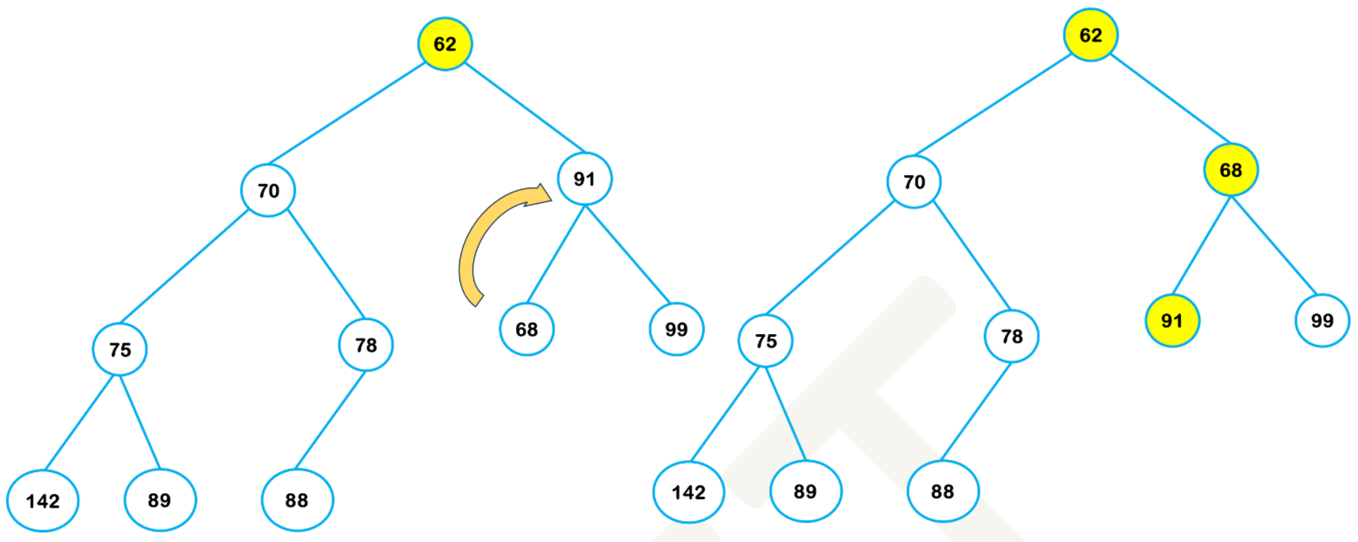




Now, after deletion we need to make sure that at each node parent-child property gets satisfied, if not then swapping shall be done.



Another swapping is required.



6. Insert In Max Heap

Consider a following example to understand the concept of insertion in Max-Heap.

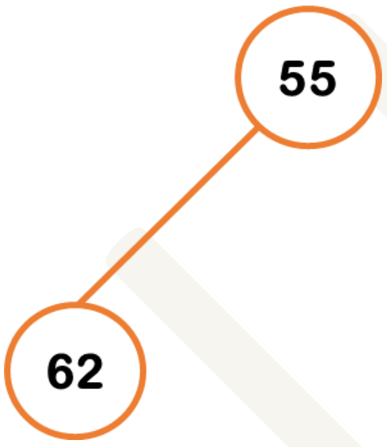
Q1. Construct a Max-Heap for the following data: 55, 62, 42, 35, 77, 85, 91, 12.

Solution:

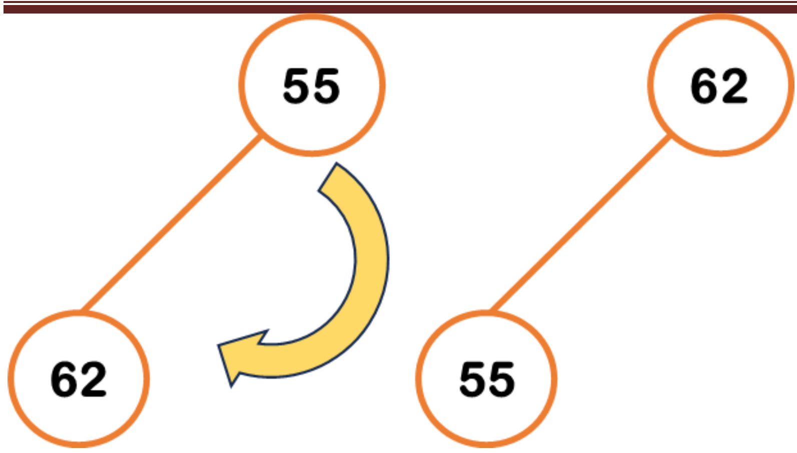
To construct a Max-Heap, we need to insert one value at a time and need to verify the Max-Heap Property that (Parent ≥ Child).

Step 1 – 1st value as a root.

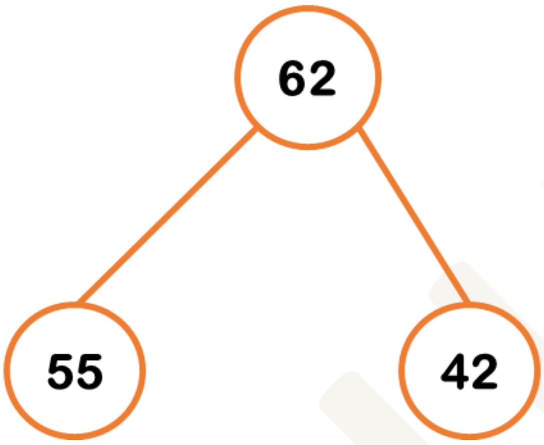
Step 2 – Insert next value 62. The value 62 will be placed as the left child of 55.



But as you can see the parent (55) is not greater than the child (62). Therefore, we need to perform swapping.

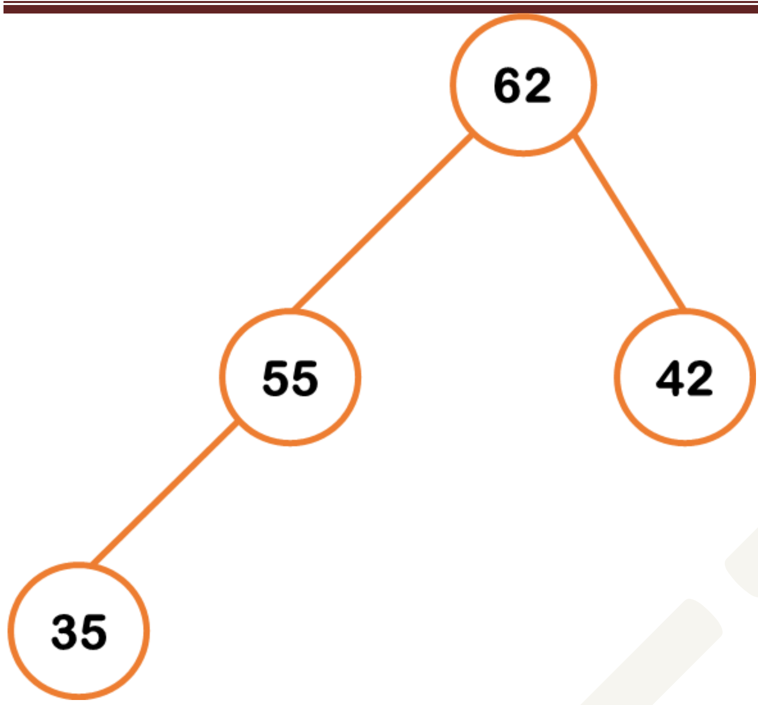


Step 3 – Insert 42. The value 42 will be placed as the right child of 62.



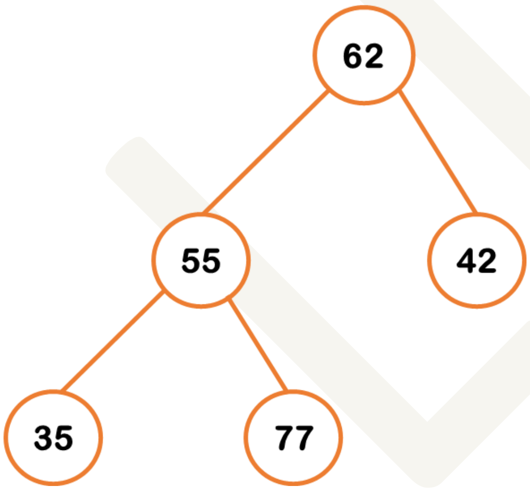
Here, the parent (62) is greater than its children (55 & 42). Hence no further swapping is required and we can add new value.

Step 4 - Insert 35. The value 35 will be placed as the left child of 55.

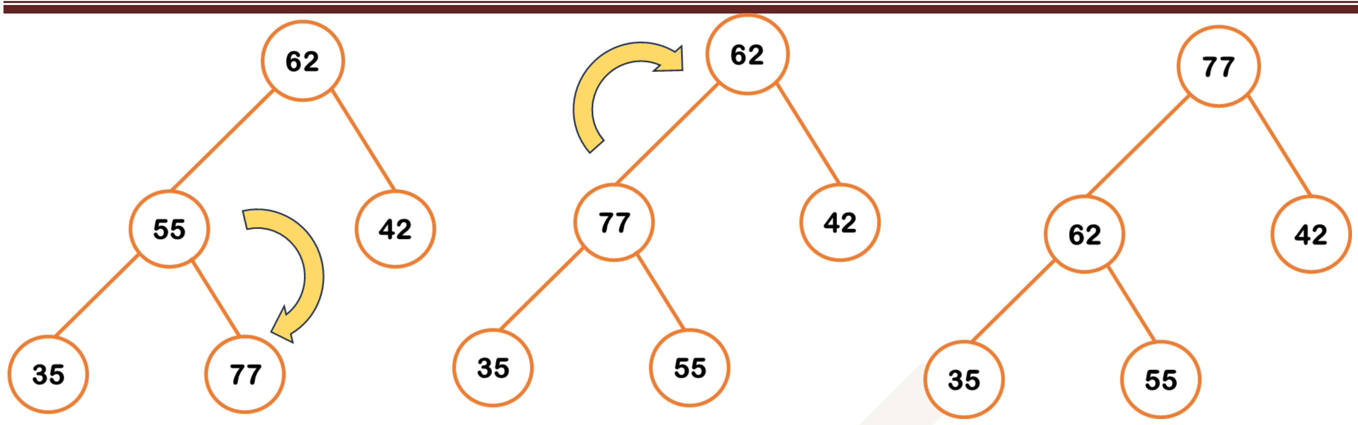


Here, the parent (62) is greater than its children (55 & 42). Also, the next parent (55) is also greater than its child (35). Hence no further swapping is required and we can add new value.

Step 5 - Insert 77. The value 77 will be placed as the right child of 55.

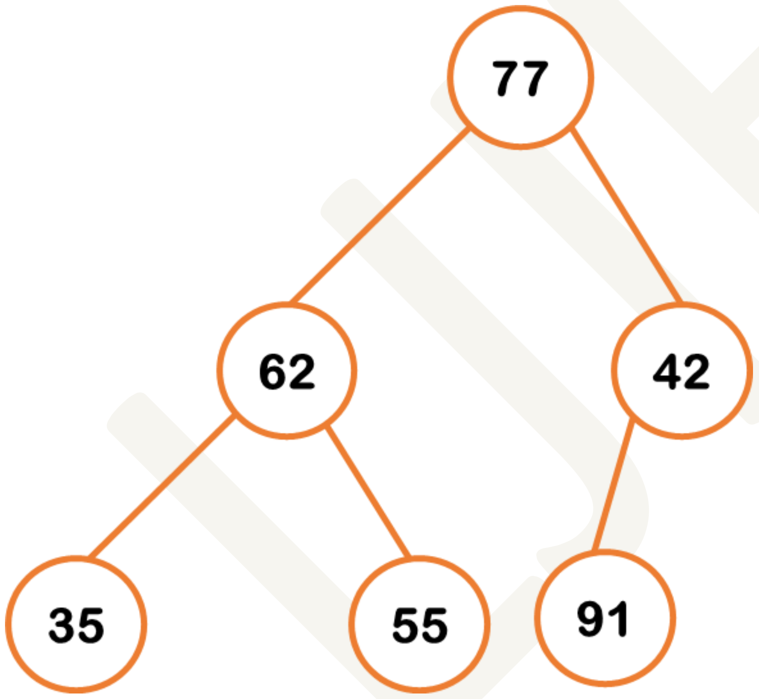


But as you can see parent 55 is not greater than its child 77. Hence swapping is required.

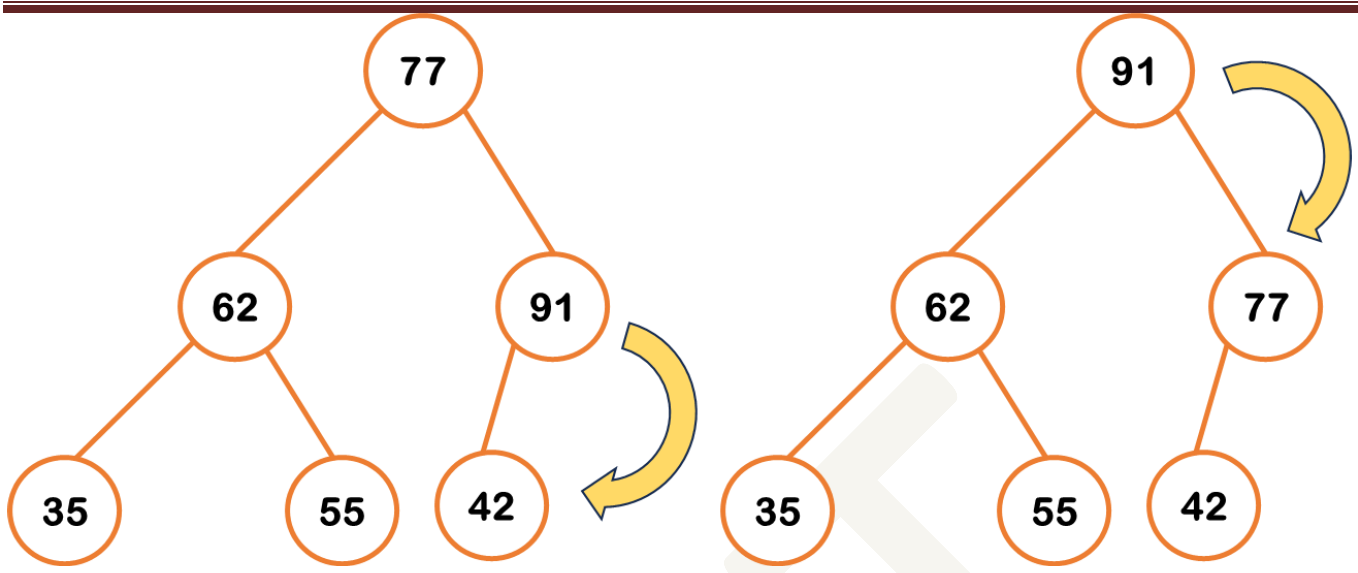


Now, again parent 62 would not be greater than its child 77. Hence 2nd swapping is required.

Step 6 – Insert 91. The value 91 will be placed as left child of 42.

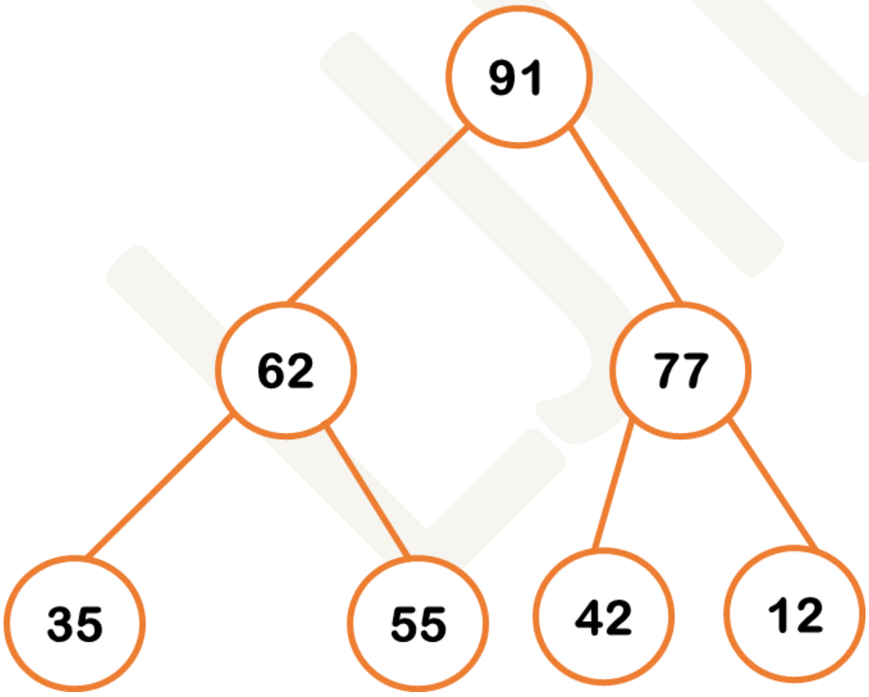


Now, 42 is parent which is not greater than its child 91. Hence swapping will be required.



Now, another swapping would be required as Parent (77) is lesser than the child 91.

Step 7 – Now, Insert 12.



After inserting 12, the property of Max Heap is satisfied at all the levels of tree. Hence, this is the final tree.

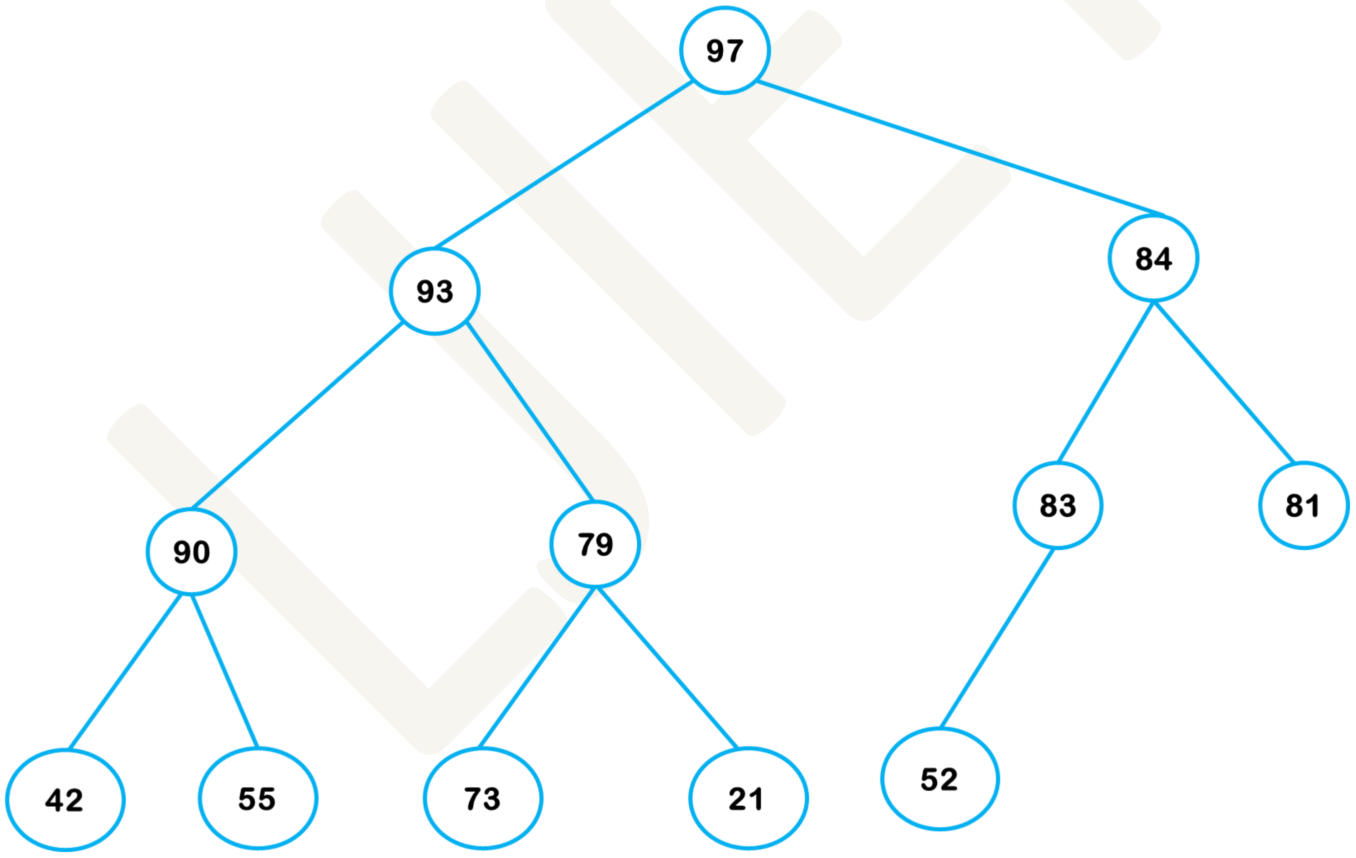
7. Delete In Max Heap

IN ANY HEAP TREE – THE DELETION OPERATION HAPPENS ONLY ON THE ROOT NODE. HENCE, THE ROOT NODE GETS DELETED IN THE TREE AND THE LAST ELEMENT OF THE TREE BECOMES THE NEW ROOT OF THE HEAP TREE.

ONCE THE LAST ELEMENT OF THE HEAP TREE BECOMES THE NEW ROOT AFTER DELETION, WE NEED TO CHECK THE PARENT-CHILD PROPERTY AGAIN.

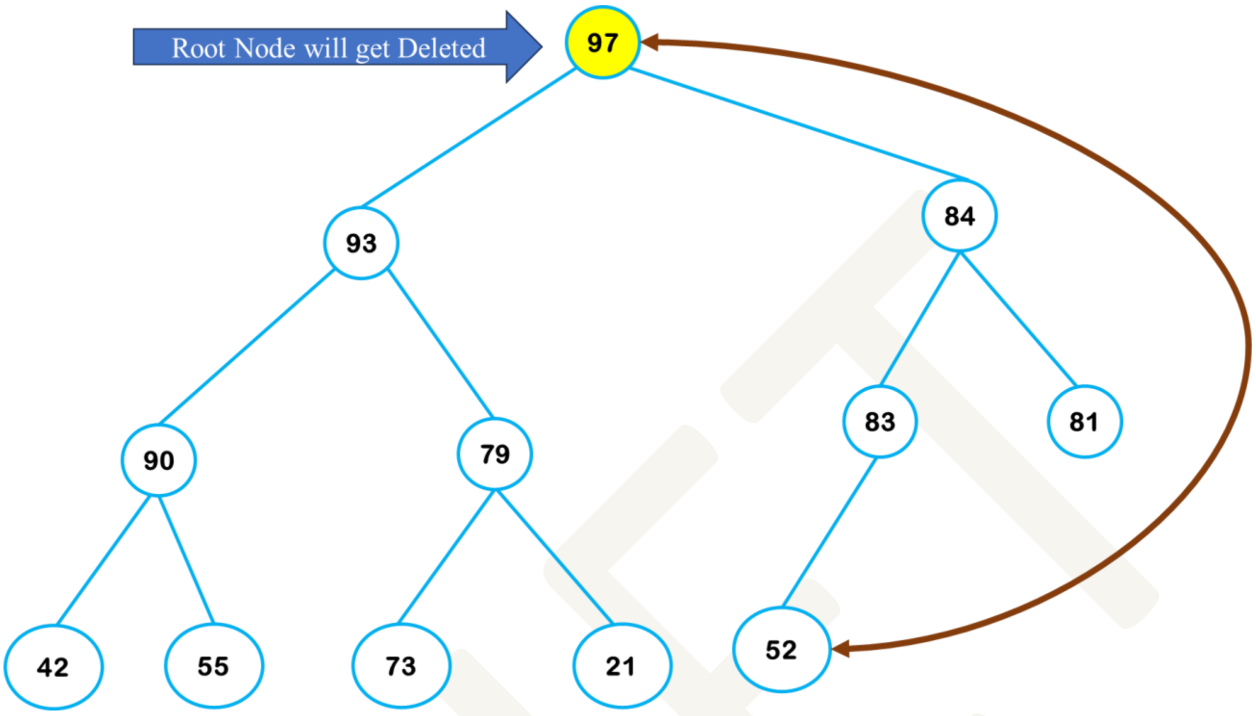
To understand the fundamental of deletion – let us see an example.

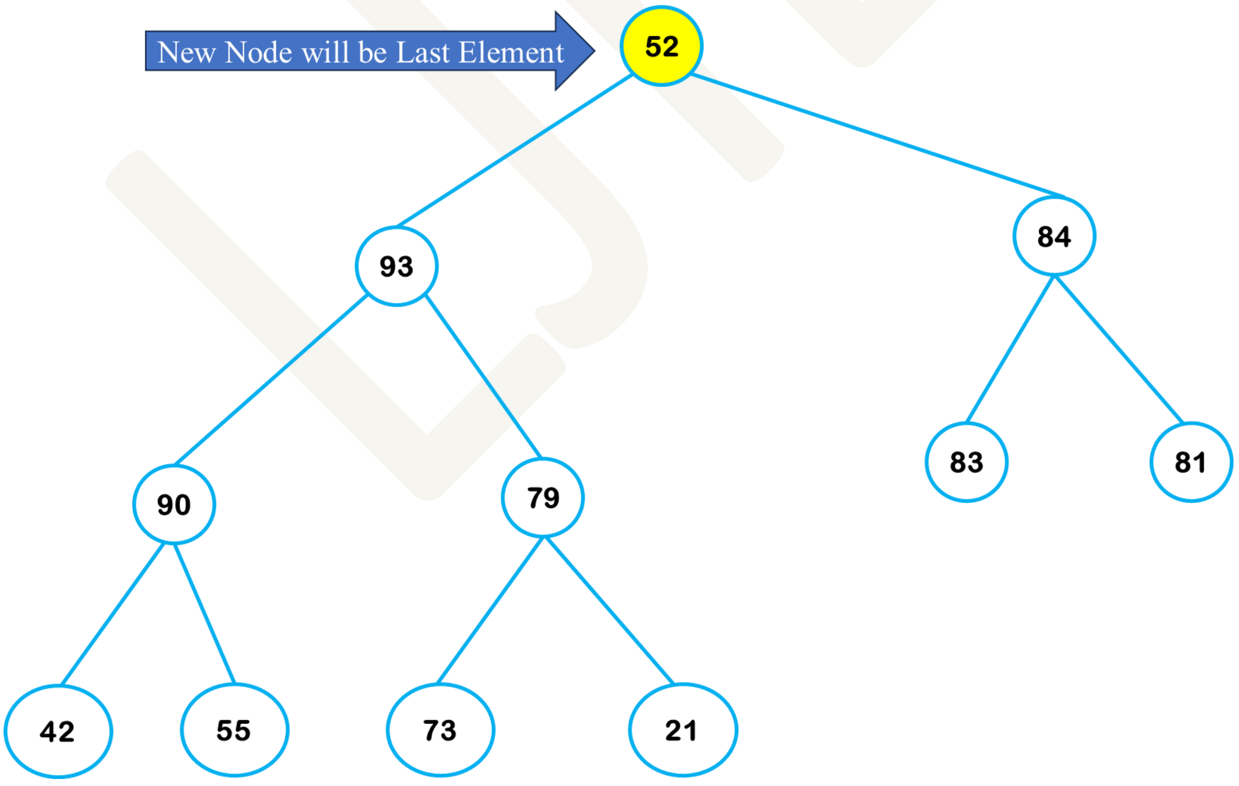
Q. Consider the following Max-heap tree and perform the delete operation twice.



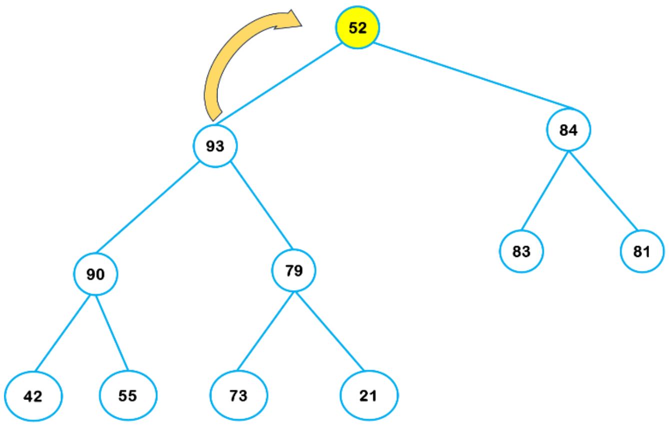
Solution:

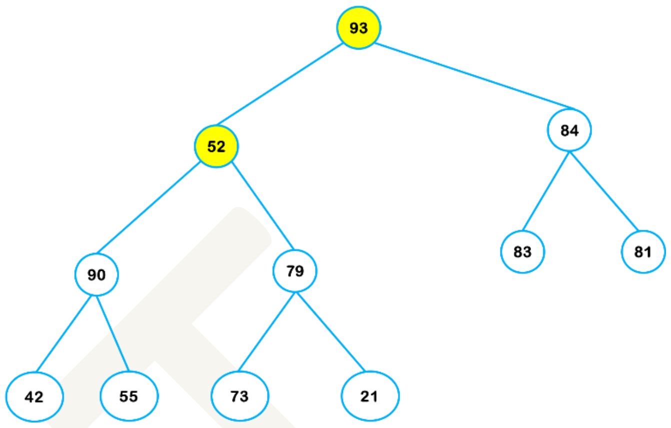
Step 1: Delete Root Node for the 1st delete operation.



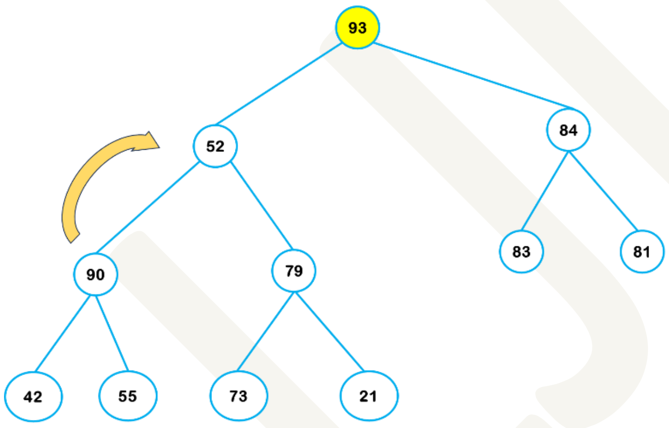


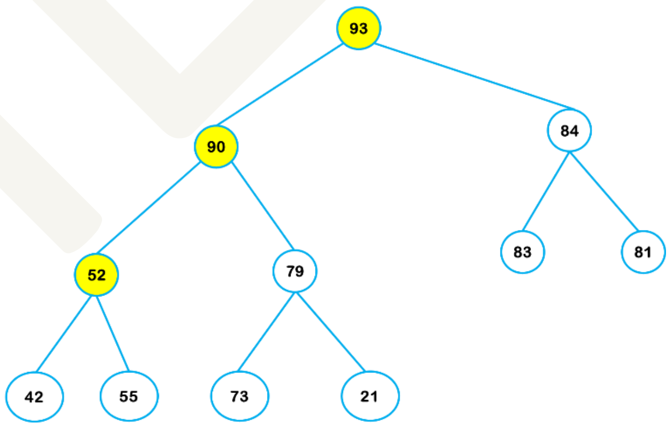
Now, after deletion we need to make sure that at each node parent-child property for Max Heap gets satisfied, if not then swapping shall be done.



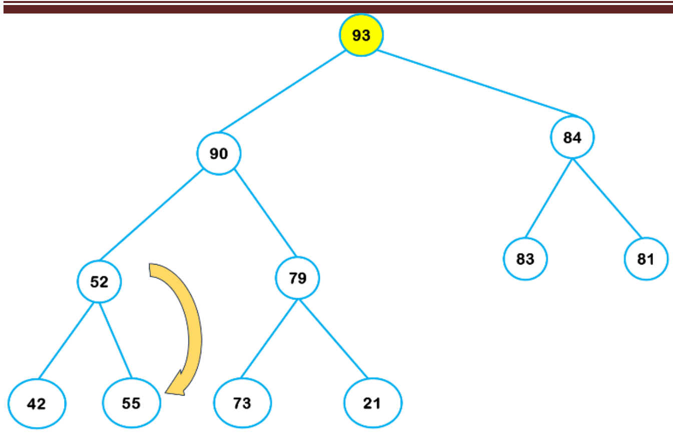


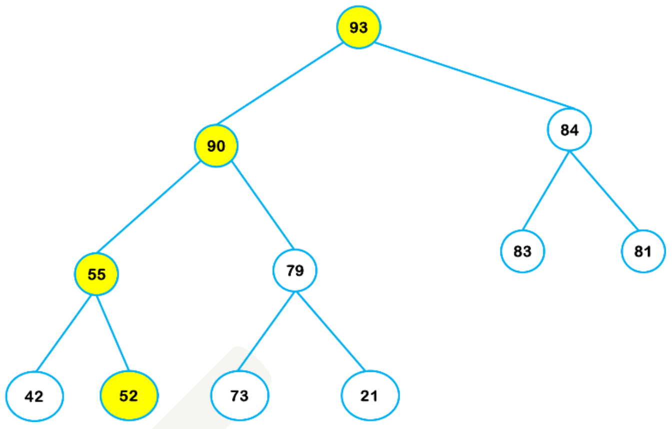
Now, again 52 as parent is not satisfying Max-Heap property of Parent > child. Hence 2nd swapping is required.



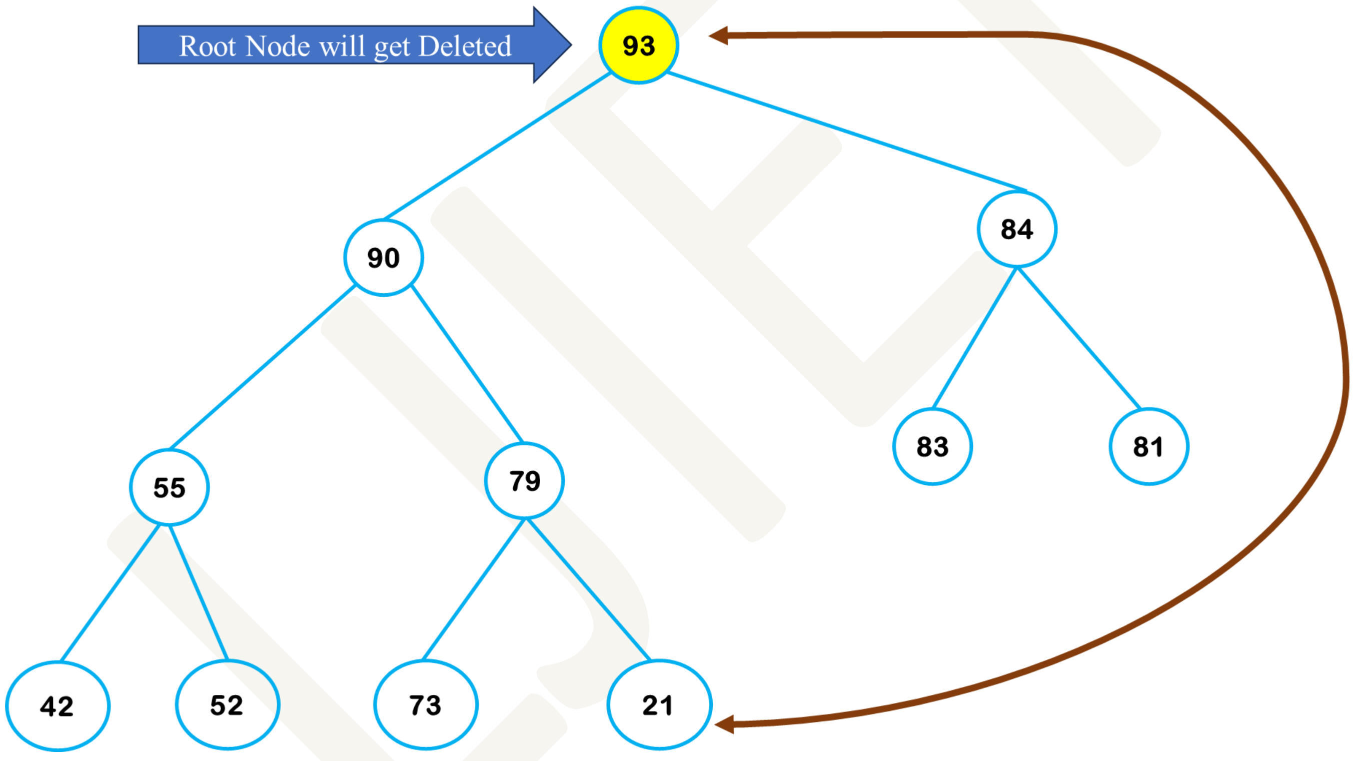


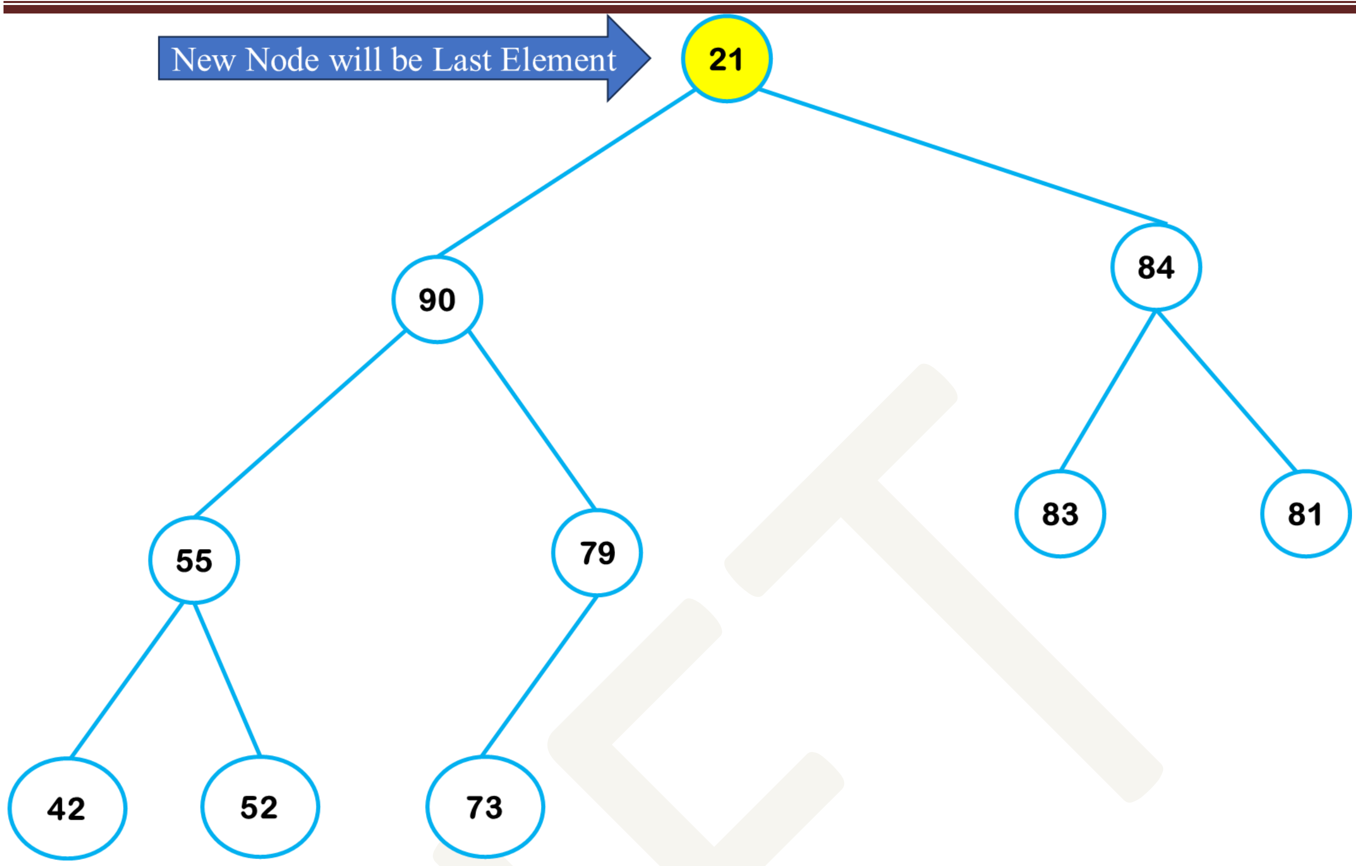
Another swapping is required for 52 & 55.





Step 2 – Now, we will perform 2nd time delete operation.



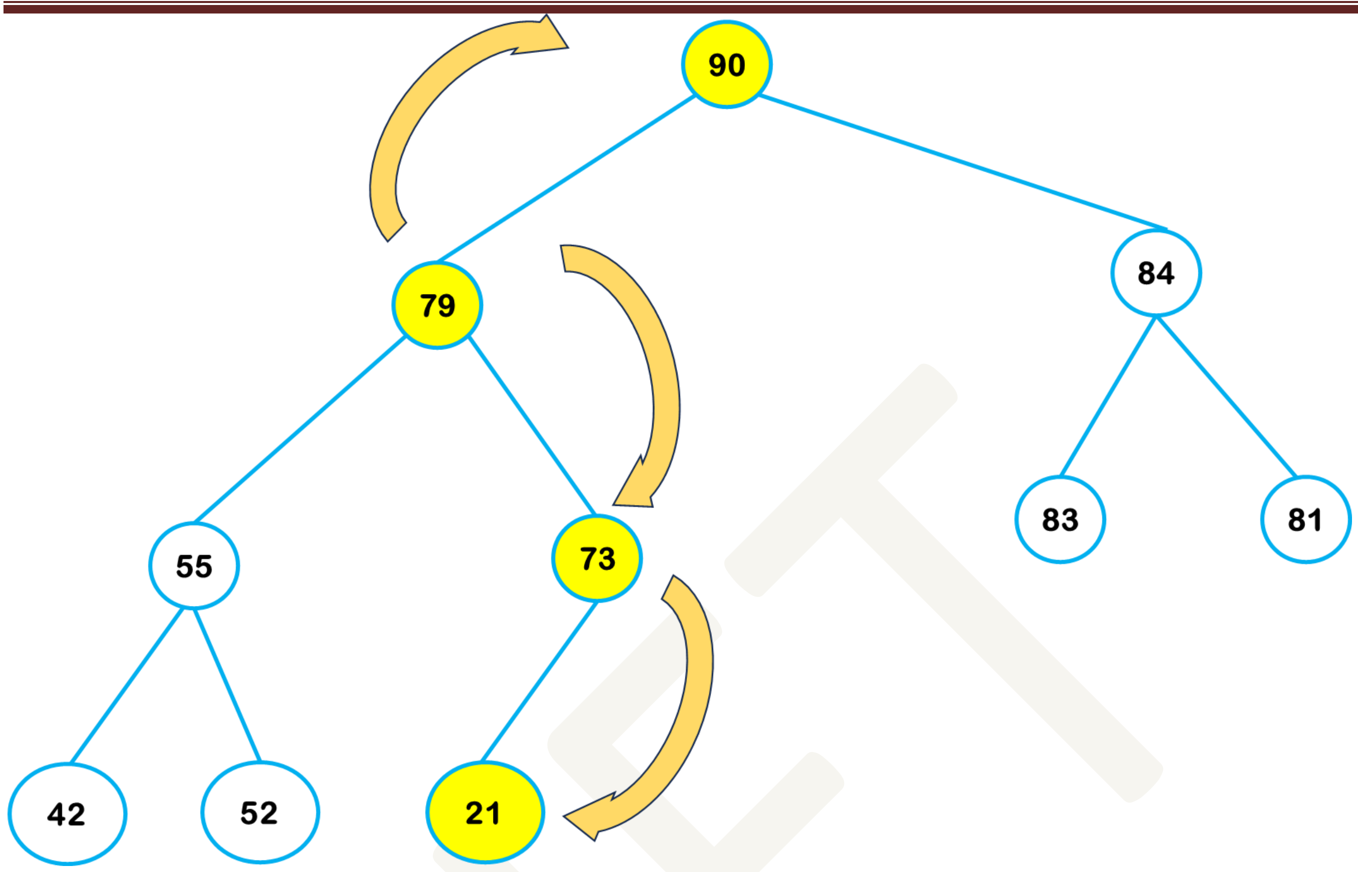


Now, again to satisfy the Max-Heap property, we need to perform swapping.

Swapping 1 – Parent 21 with Child 91.

Swapping 2 – Parent 21 with Child 79.

Swapping 3 – Parent 21 with Child 73.


90
79
84
55
73
83
81
42
52
21