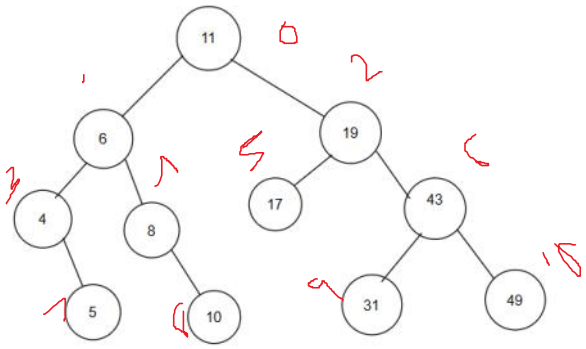
16 January 2026 08:55

Binary Search Tree :

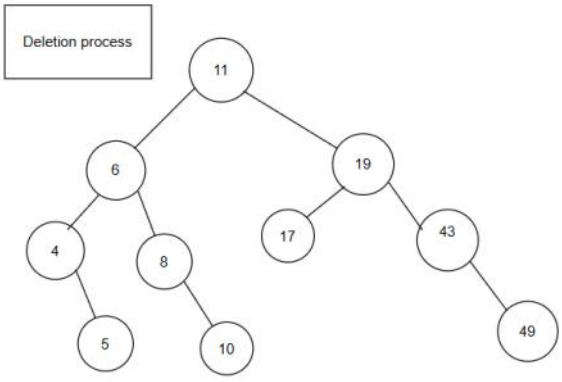
Contains at most 2 nodes
0,1,2

11	6	8	19	4	10	5	17	43	49	31



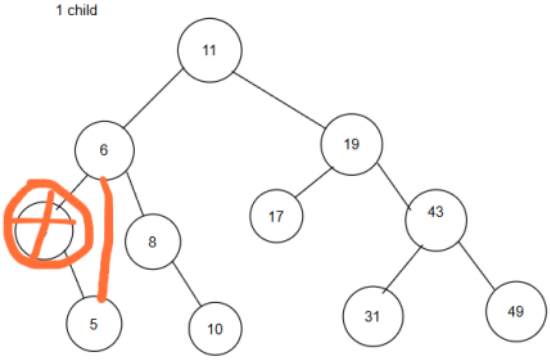
DELETION:

1. 0 child
2. 1 child
3. 2 children --> either one following 2 condition
    a) In-order predecessor
    b) In-order successor

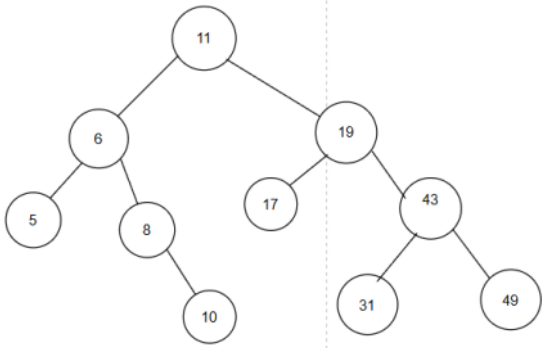


30 deleted but there is no any replacement because its has no children

1 child

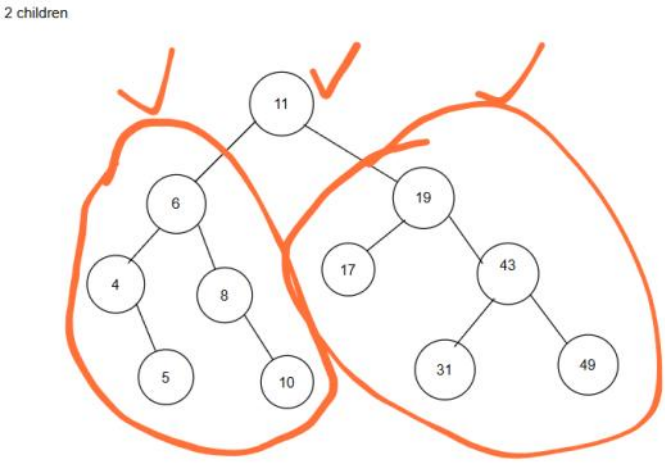


16-01-2026 09:35 - Screen Clipping



4 is deleted and replaced with 6

2 children

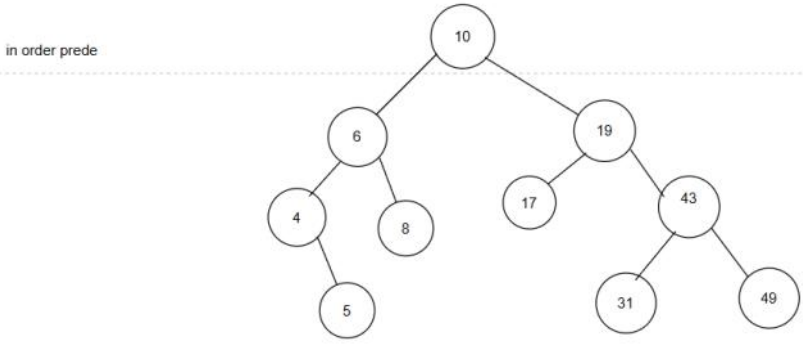


we are deleteing the 11 node

We need to follow these conditions for replacement
either one following 2 condition

a) In-order predecessor
b) In-order successor

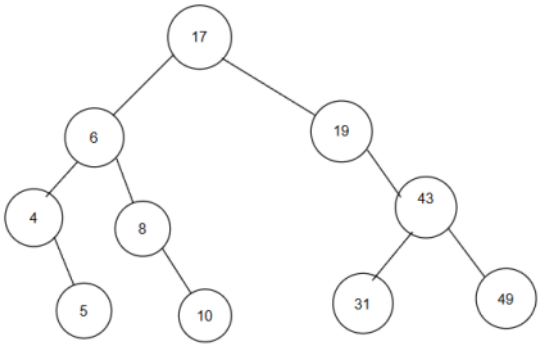
In-order predecessor --> take left sub tree and find out the largest element in the left subtree and replace it with root element



Here 10 is largest element we are replcaed with 11

In-order successor --> take the right side sub tree and find smallest element and replace with root node

in order success



Here 17 is the smallest element and we are replced it with 11

11	6	8	19	4	10	5	17	43	49	31

4	5	6	8	10	11	17	19	31	43	49

Algorithm :

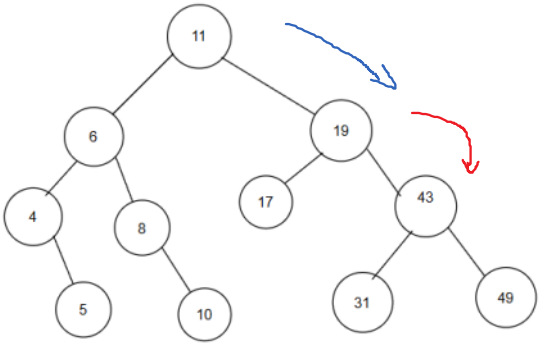
1. Start
2. Start at the root node
3. If tree Is empty -> create new node as root
4. If the value to insert is **less than** current node --> move to **left subtree**
5. If the value insert is **greater than** current node --> move to **Right subtree**
6. Stop

Pseudo code :

```
Function insert (root,key):
    If root == NULL:
    Root = create_node(key)
    Return root

    If key <root.value:

    Root.left=insert(root.left,key)

    Else if key>root.value :
        Root.right=insert(root.right,key)
    Return root
```

Searching Algorithm :

1. Start
2. If root is NULL --> element is not found
3. If key == root ---> element found
4. If key < root --> search left subtree
5. If key >root --> search right subtree
6. Continue until found / or subtree becomes null



Seaching element is 43:

43===11
Not found go to next step
43<11
43>11

43==19
43<19
43>19
43==43
Element found at 6

Pseudo code :

```
Function  search (root, key)
    If root == NULL
    Return  "not found"

    If key == root
        Retrun element found
    Else if key < root.value :
        Return search(root,left,key)
    Else :
        Return search(root.right, key)
```

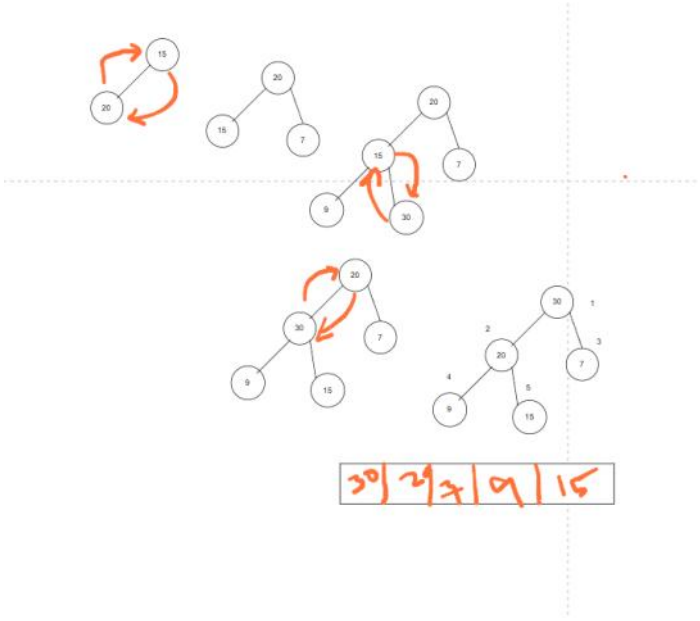
Heap sort :

. Max heap method and hepify method

.Max heap method :keep always root biggest element
root node is always greater than child node

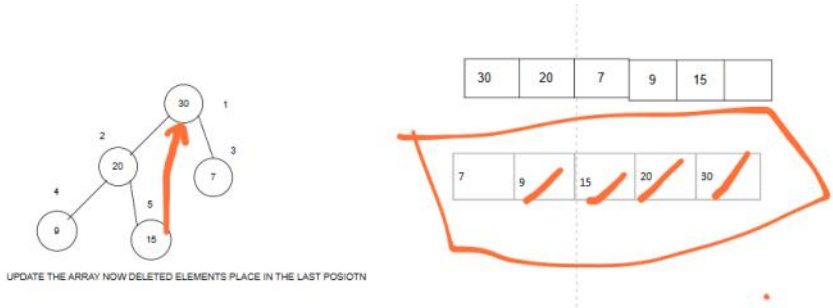
Whenever you constructing the tree you should follow complete binary tree rule
"
A **complete binary tree** is a binary tree in which every level, except possibly the last one, is completely filled, and all nodes in the last level are as far left as possible"

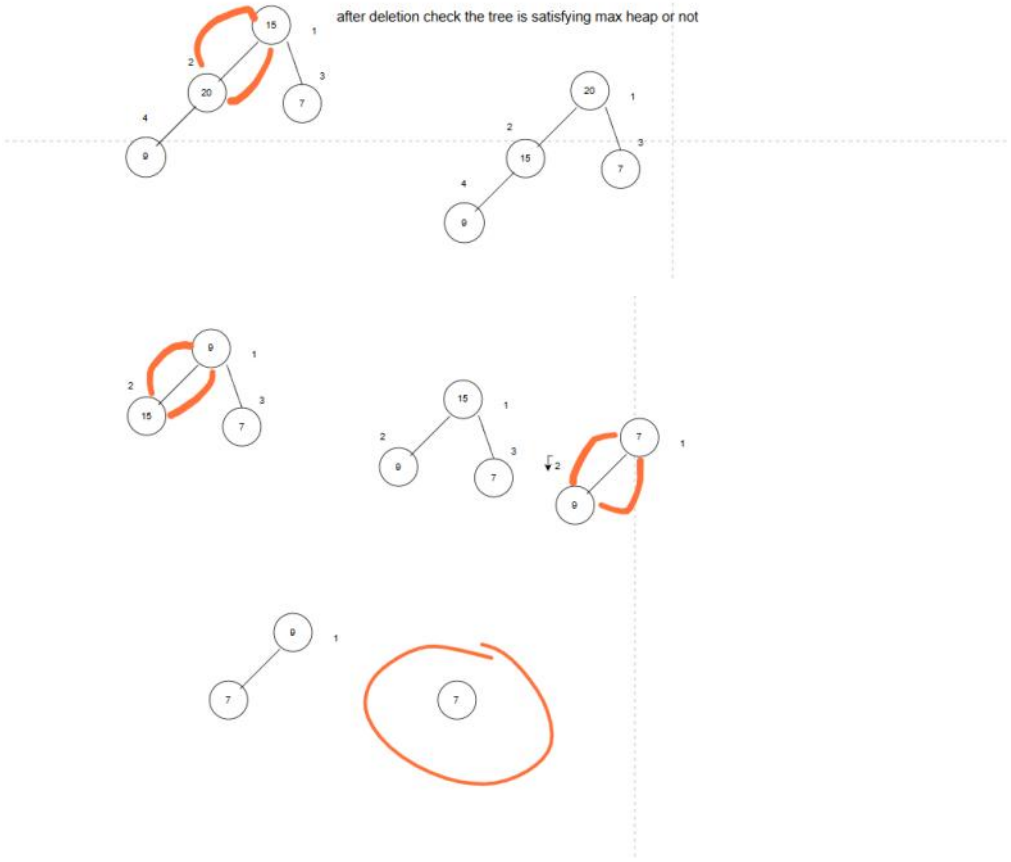
15	20	7	9	30



To make the list in order , we have to delete. The process of deleting is each element from the root and replace the root with last element in the last level

30	20	7	9	15





We have to check each level if u replace wheather it satisfying max heap or not till reach to root element

Time complexity O(logn)
Because each stage we have to search and insert
For deletion take homany time O(nlogn)
So we can acchieve O(n) when we can go with heapify method