Unit - II
Heaps

COURSE CODE: 21AI33

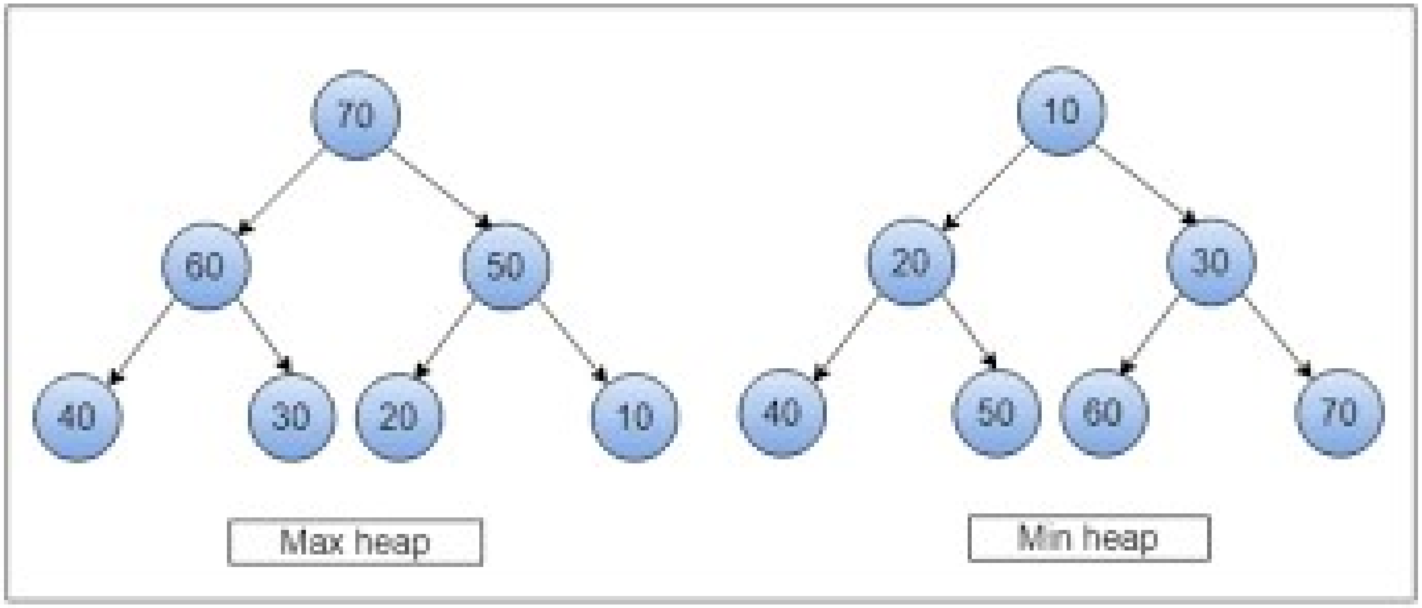
Priority Queues

- In a priority queue, the element with highest (or lowest) priority is deleted from the queue, while elements with arbitrary priority are inserted.
- A data structure that supports these operations is called a max(min) priority queue.
- A priority queue can be implemented by a simple, unordered linked list.
- Insertions can be performed in $O(1)$ time. However, a deletion requires a search for the element with the largest priority followed by its removal.
- The search requires time linear in the length of the linked list.
- A max heap is used, both of these operations can be performed in $O(\log n)$ time.

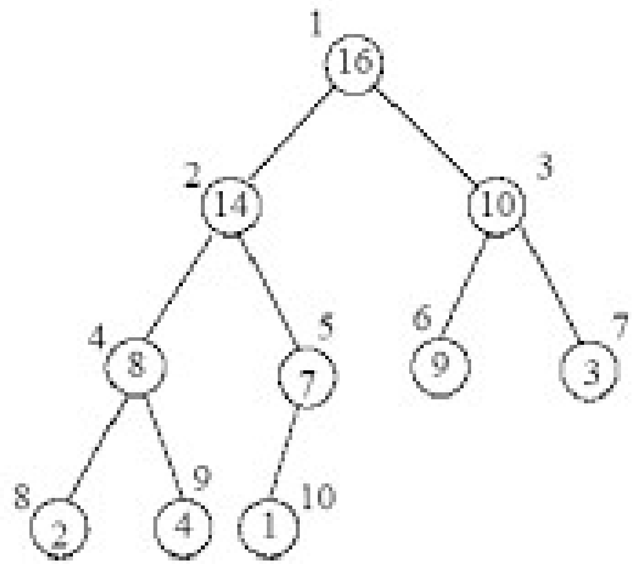
Definitions

- **Min-Heap:** The key present at the root node is smaller than or equal to keys of all the nodes present in the children nodes.
  - And this same rule is recursively followed by all the subtrees of the binary tree.
- **Max-Heap:** In this data structure, the key which is present at the root node is greater than or equal to the keys of all the children nodes of the tree.
  - The same property is recursively applicable for all the subtrees of the tree. The maximum key is present at the root of the tree for a Max-Heap.

Max Heap

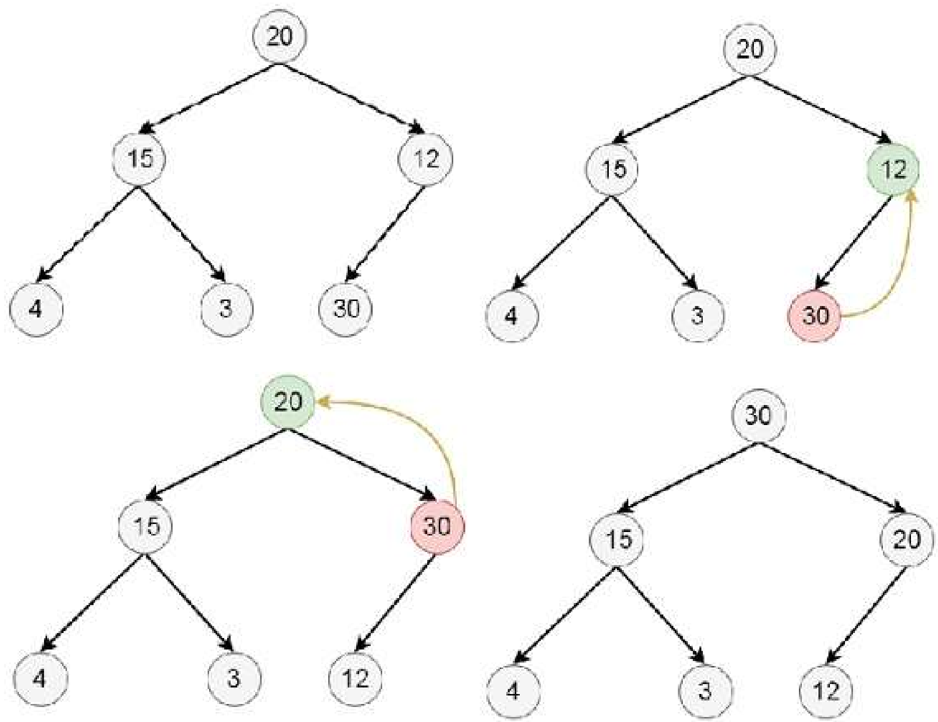


Array Representation of Heaps

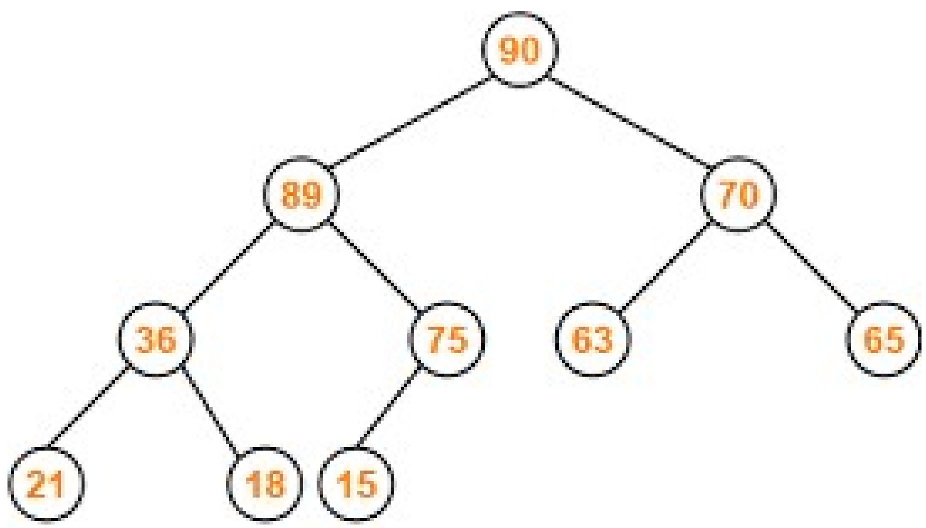


1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

Max Heap - Insertion



Insert 95

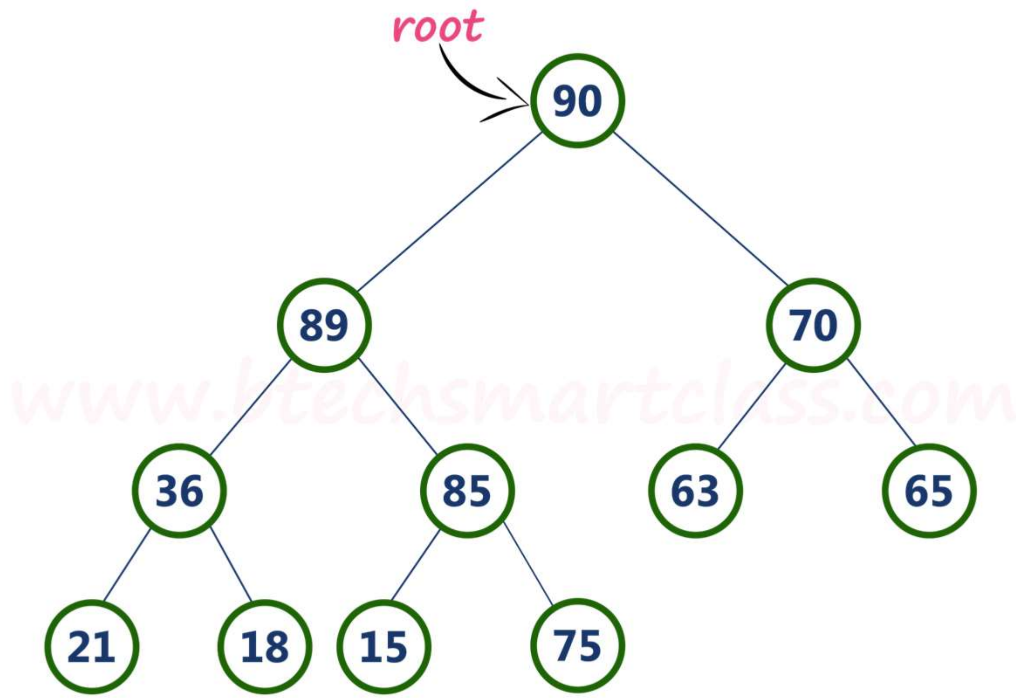


Max Heap Example

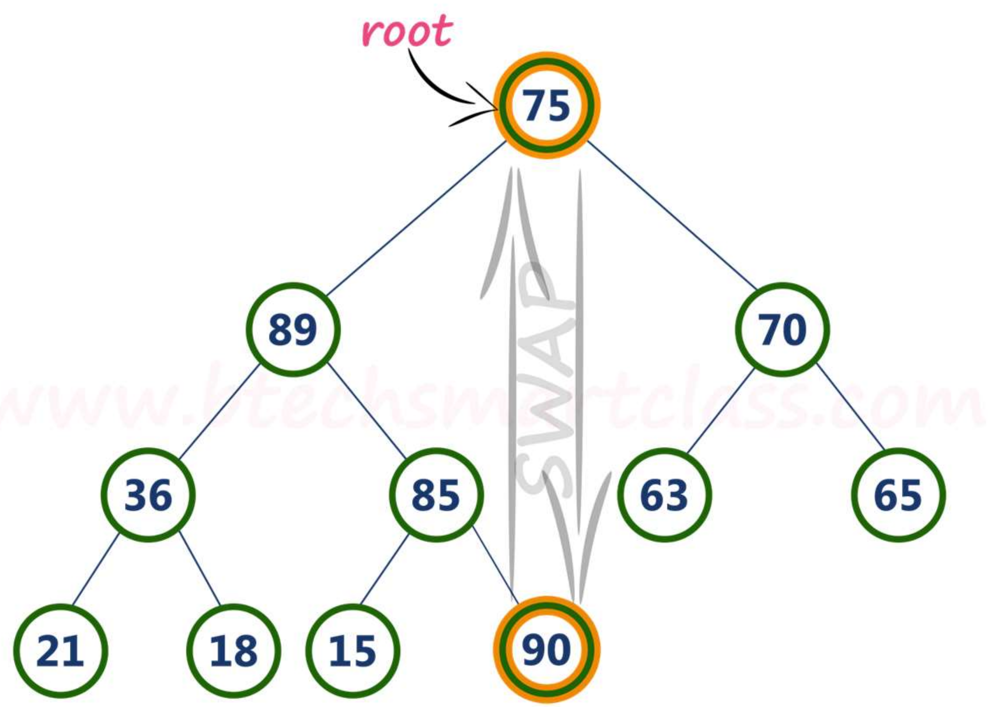
Max Heap Insertion Logic

- Step 1 - Insert the **newNode** as **last leaf** from left to right.
- Step 2 - Compare **newNode value** with its **Parent node**.
- Step 3 - If **newNode value is greater** than its parent, then **swap** both of them.
- Step 4 - Repeat step 2 and step 3 until newNode value is less than its parent node (or) newNode reaches to root.

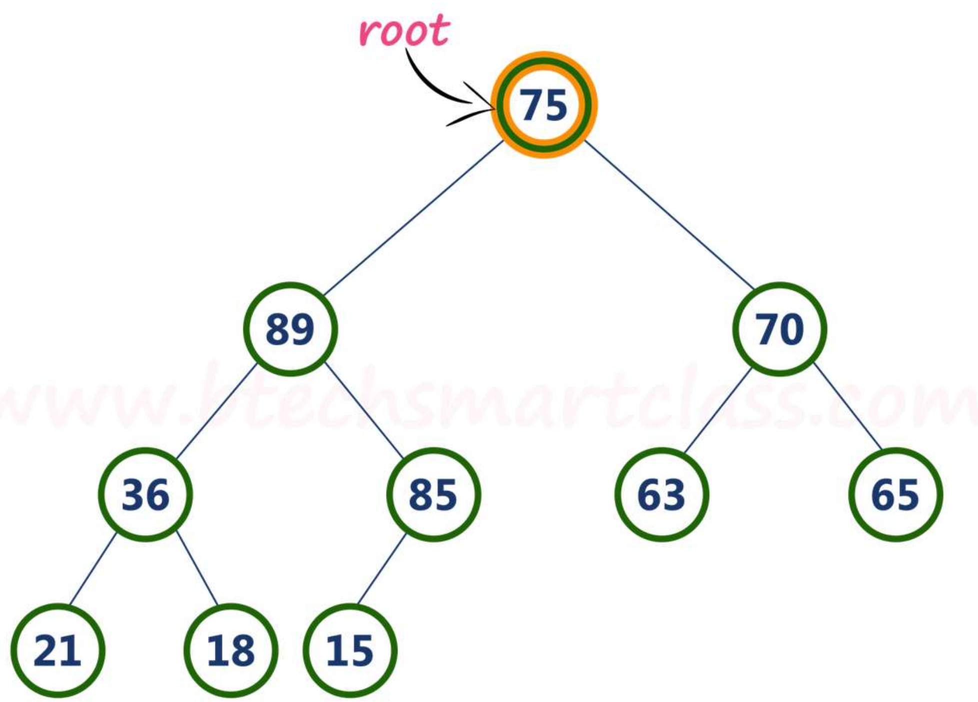
Max Heap - Deletion



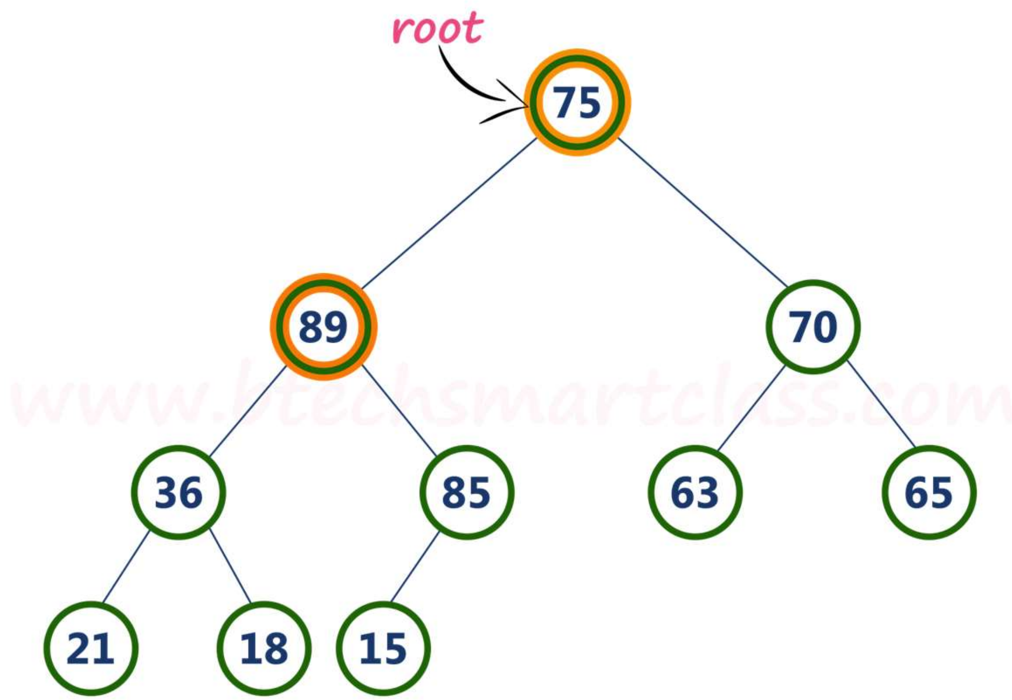
Max Heap – Deletion – Step 1



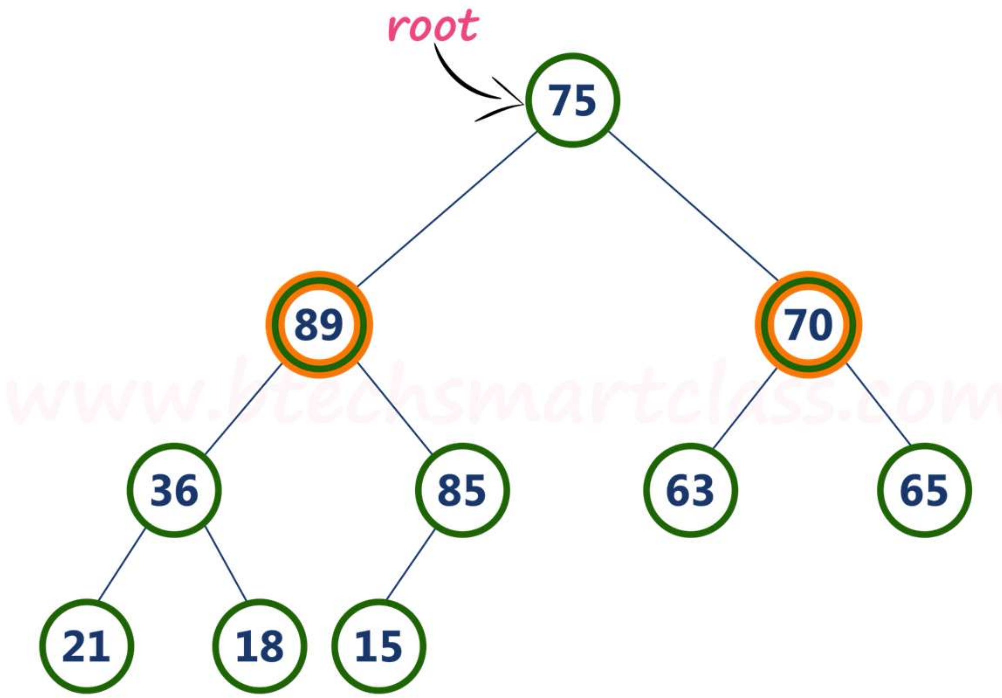
Max Heap – Deletion – Step 2



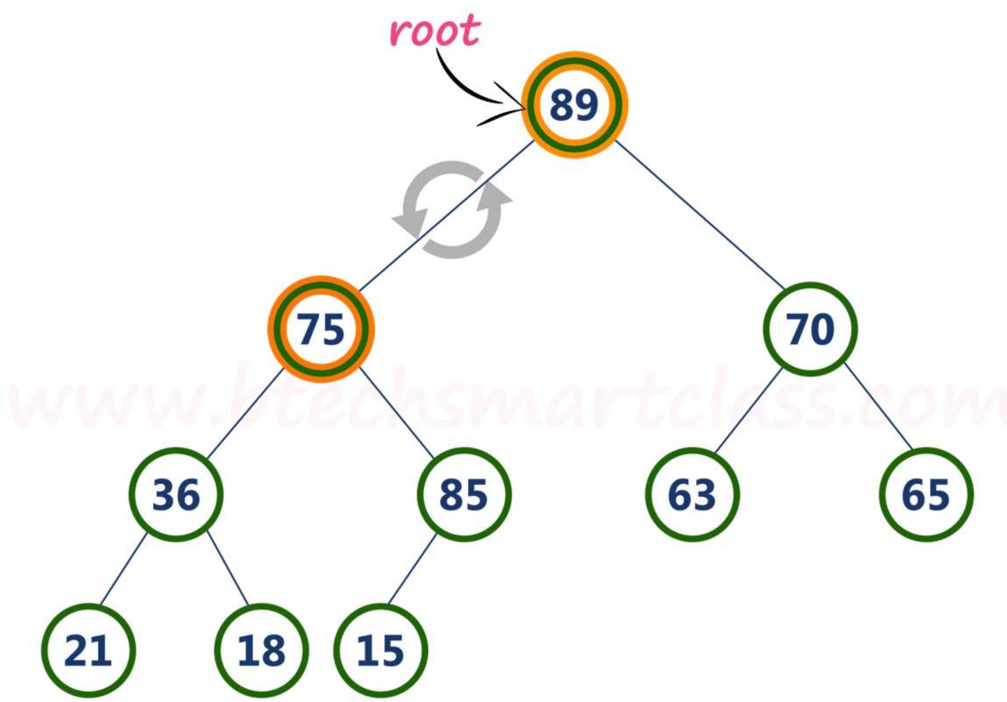
Max Heap – Deletion – Step 3



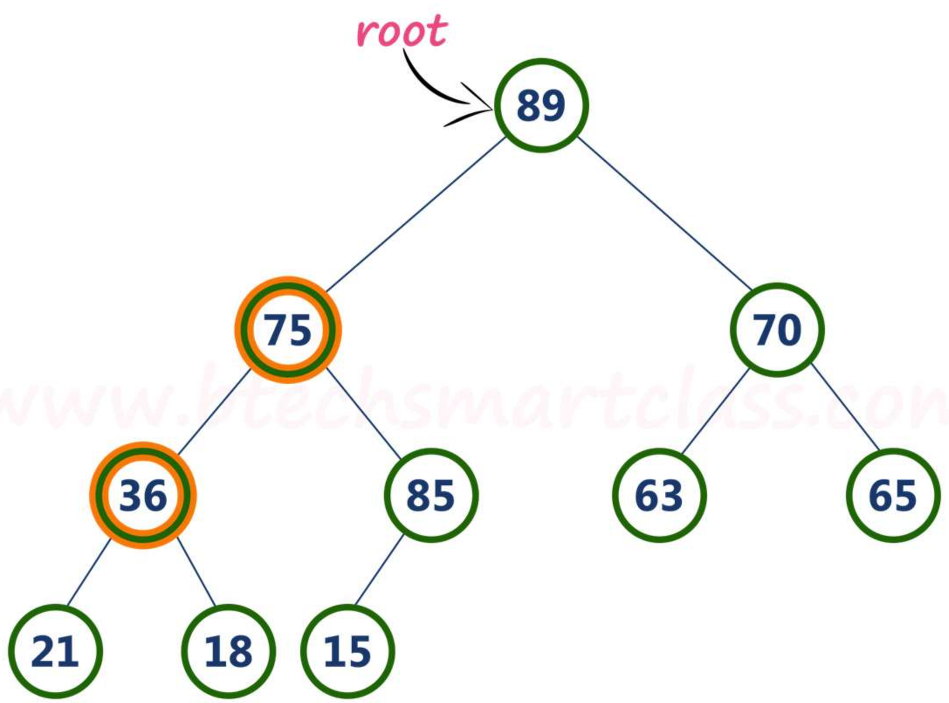
Max Heap – Deletion – Step 4



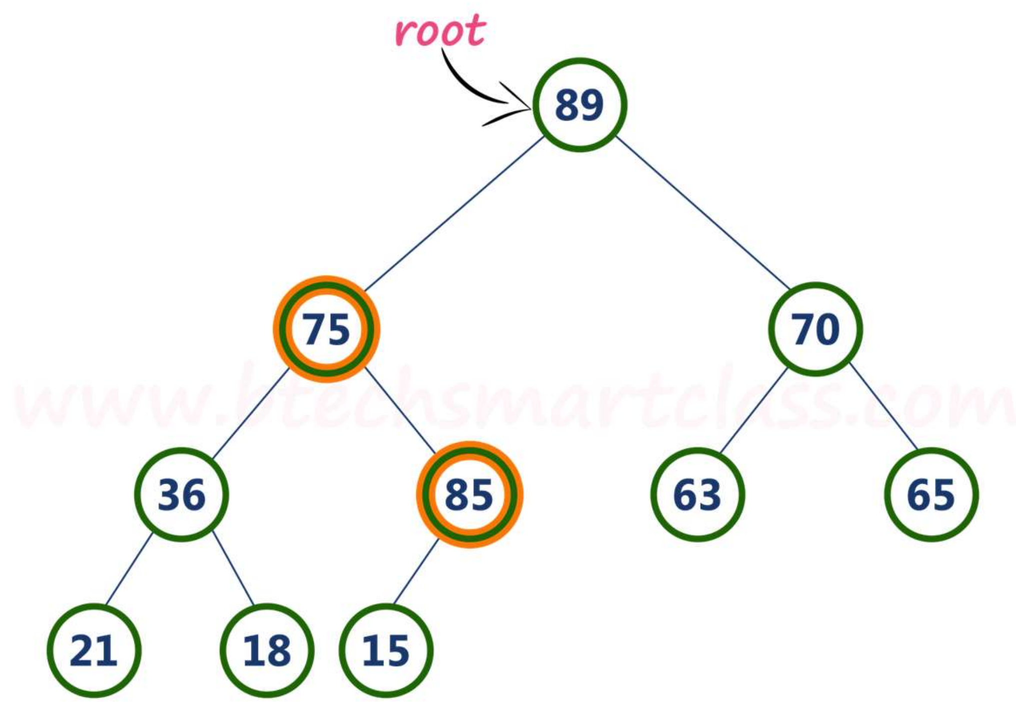
Max Heap – Deletion – Step 5



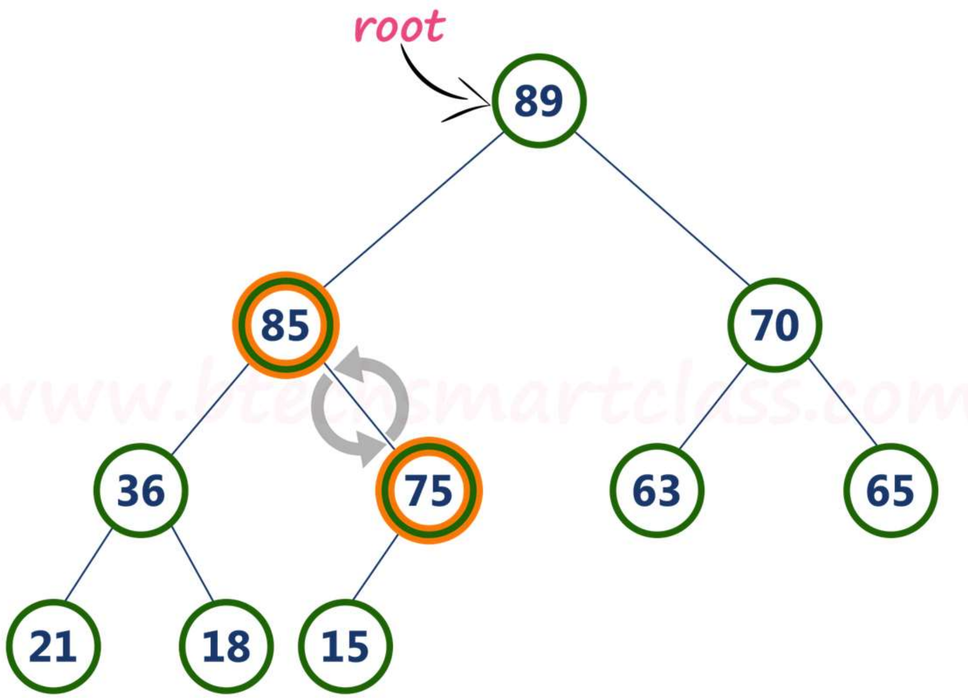
Max Heap – Deletion – Step 6



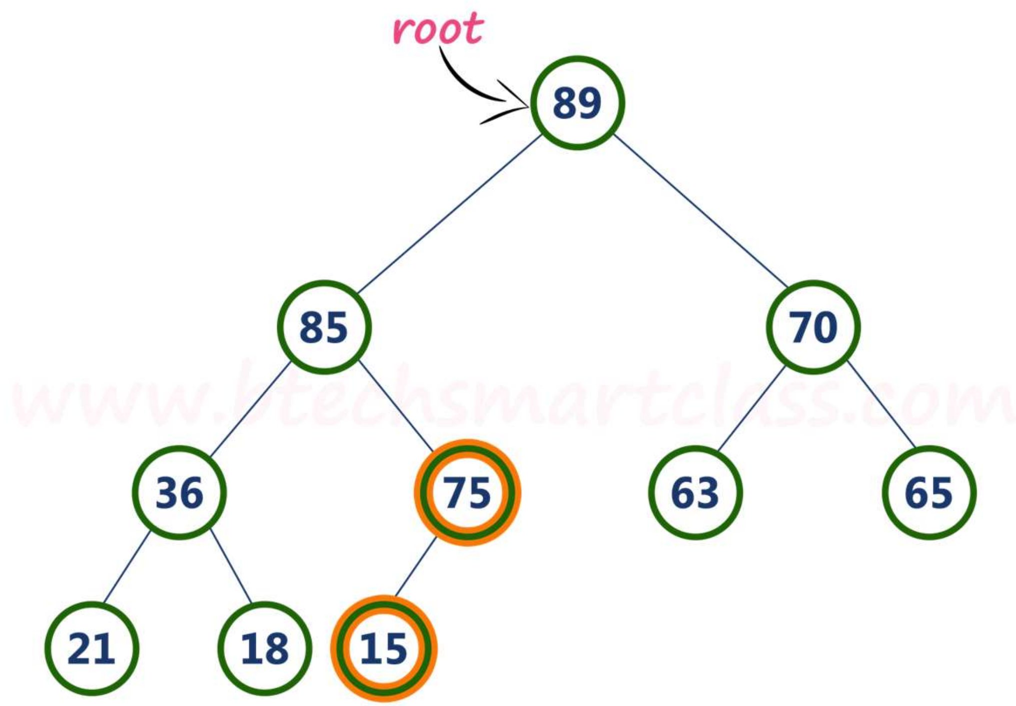
Max Heap – Deletion – Step 7



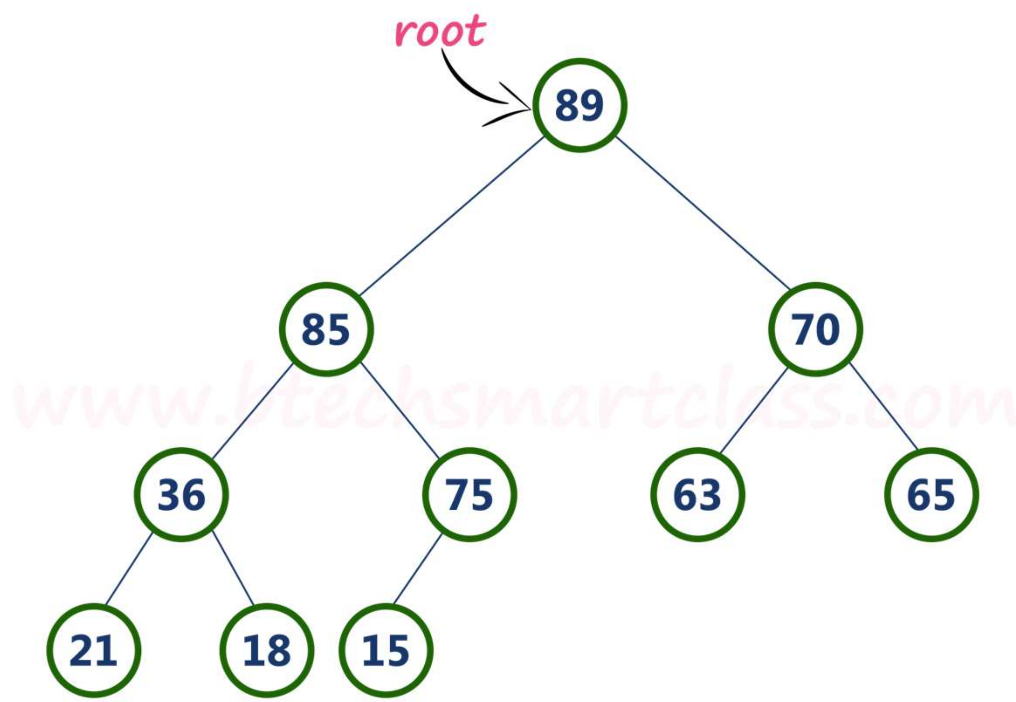
Max Heap – Deletion – Step 8



Max Heap – Deletion – Step 9



Max Heap – Deletion – Step 10



Max Heap Deletion Logic

- Step 1 - **Swap** the **root** node with **last** node in max heap
- Step 2 - **Delete** last node.
- Step 3 - Now, compare **root value** with its **left child value**.
- Step 4 - If **root value is smaller** than its left child, then compare **left child** with its **right sibling**. Else goto **Step 6**
- Step 5 - If **left child value is larger** than its **right sibling**, then **swap root** with **left child** otherwise **swap root** with its **right child**.
- Step 6 - If **root value is larger** than its left child, then compare **root value** with its **right child** value.
- Step 7 - If **root value is smaller** than its **right child**, then **swap root** with **right child** otherwise **stop the process**.
- Step 8 - Repeat the same until root node fixes at its exact position.