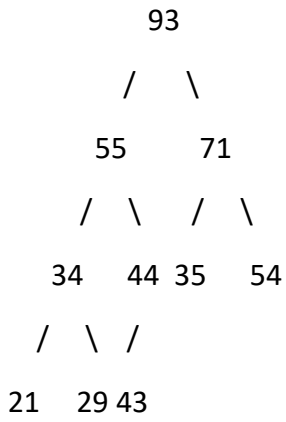
Max Heapify function to Build Max Heap CodeStudio

You are given an integer array with N elements. Your task is to build a max binary heap from the array.

A max-heap is a complete binary tree in which the value of each internal node is greater than or equal to the values of the children of that node.

Example: [34, 43, 54, 21, 44, 35, 71, 55, 29, 93]

Output: [93, 55, 71, 34, 44, 35, 54, 21, 29, 43]



Approach 1: Function to build a max heap iteratively

- **Function Purpose:** To build a max heap from an array using iterative max-heapify.
- **Explanation:**
    - Start from the last non-leaf node and move up to the root.
    - At each step, apply maxHeapifyIterative to adjust the heap.
- **Time Complexity: O(N * log(N)), where N is the number of elements in the array.**
- **Space Complexity: O(1) since it operates in-place.**

Approach 2: Function to build a max heap recursively

- **Function Purpose:** To build a max heap from an array using recursive max-heapify.
- **Explanation:**
    - Start from the last non-leaf node and move up to the root.
    - At each step, apply maxHeapifyRecursive to adjust the heap.
- **Time Complexity**: **O(N * log(N)), where N is the number of elements in the array.**

- **Space Complexity: O(log(N)) due to the call stack.**

Approach 3: Function to build a max heap using STL Priority Queue (Max Heap)

- **Function Purpose:** To build a max heap from an array using a priority queue (STL).
- **Explanation:**
    - Create a max-heap (priority queue) and push all elements into it.
    - Extract elements from the priority queue to obtain the max heap.
- **Time Complexity: O(N * log(N)) to insert elements into the priority queue.**
- **Space Complexity: O(N) due to the priority queue.**

Conclusion:

- Both the iterative and recursive approaches have the same time complexity (O(N * log(N)).
- **The iterative approach is more space-efficient with O(1) space complexity.**