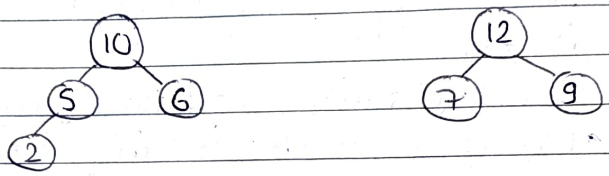
Date _____ Page _____

Day - 189

Heap - 5

* **Merge two binary Max Heap:**

⇉ | 10 | 5 | 6 | 2 | (n) and | 12 | 7 | 9 | (m)

⇉ 

⇉ In the first approach, we can create a priority queue & insert all the elements of both arrays.

⇉ Then pop all the elements and store in a vector & return it.

$T.C \rightarrow (n+m)\log(n+m)$

⇉ In the second approach, we can combine two arrays then make heap from this array.

⇉ T.C. → $O(n+m)$.

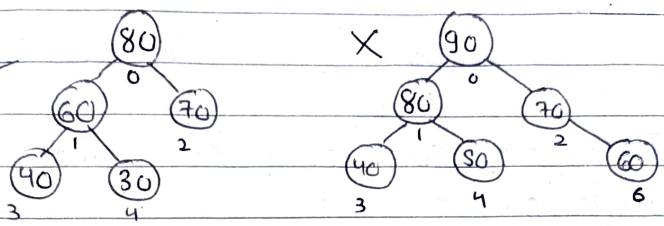
* **Is Binary Tree Heap:**

Date _____ Page _____

⇉ Properties for Max Heap :

⇉ CBT.

⇉ Every parent >= its child.



⇉ First, we will give index to the nodes (for better understanding).

⇉ Now, we will create a array from the nodes. if we get any space left in b/w array then that tree is not CBT.

⇉ But this approach is inefficient.

⇉ For better approach, we will count the nodes.

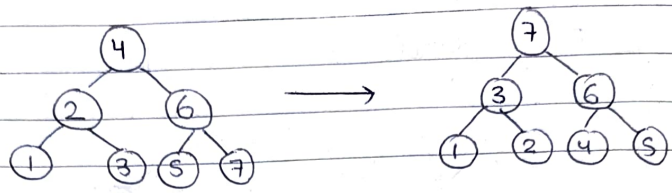
⇉ Then check if any index of node is equal to or more than the no. of nodes then the tree is not CBT. because if a tree contain n nodes then the indexing is b/w (0-(n-1)).

⇉ Then we will check maxheap property by traversing all the nodes.

T.C. → O(N) (N → no. of nodes).

Date _____ Page _____

* BST to Max Heap:



⇒ There are some conditions that we have to follow —

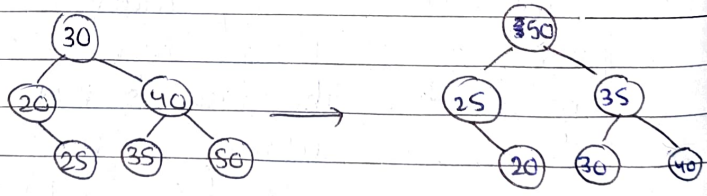
→ parent > children

→ left subtree < right subtree

⇒ First, we will find the ~~the~~ inorder traversal of the BST.

⇒ Then we will start filling the tree in post order traversal → LRN.

⇒ There will be no change in the structure of the BST.



inorder → 20, 25, 30, 35, 40, 50