Date _____ Page _____

Day - 182

Heap

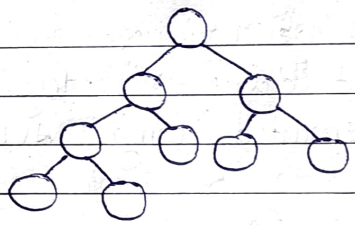
* **Heap:**

⇉ It is a complete Binary Tree.

⇉ All levels are completely filled except the last level.

⇉ At last level, nodes should be filled from left side.

Ex:

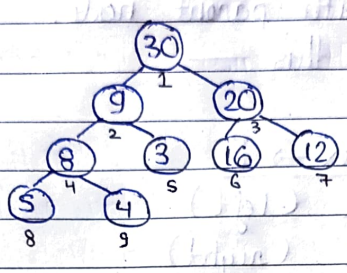


⇉ Heap are of two types —

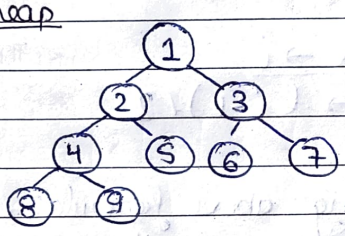
Max heap	Min heap
⇉ CBT.	⇉ CBT.
⇉ Parent node should be greater than or equal to child node.	⇉ Parent node should be less than or equal to child node.

Date _____ Page _____

Ex: Max heap



Min heap



⇉ Build Max-Heap:

⇉ First, we have to find the level. (O(n)).

⇉ Then insert the node.

⇉ After that, we will check if that node is following the properties or not.

⇉ If not then we have to make the tree according to properties.

⇉ So, implementing this by using previous methods will not very efficient.

⇉ So, we will use array for implementation.

0	1	2	3	4	5	6	7	8
30	9	20	8	3	16	12	5	4

Date _____ Page _____

⇉ For maintaing properties, we will require to check with parent node.

⇉ Now, for this —

Parent - index → i
↳ 2*i+1 (left)
↳ 2*i+2 (right)

Child - index → i
↳ Parent → (i-1)/2

⇉ So, by using above formulas, we find the correct position of the upcoming node.

⇉ Time complexity of creating a max-heap will be O(nlogn).

⇉ Deletion in Max Heap:

⇉ In deletion, we delete only the top node.

⇉ First, we delete the top node.

⇉ After, we have to find a node that can take their place & no properties ~~ee~~ conflicts.

⇉ For that, we will take the last node & change it with top node. then delete the top node.

Date _____ Page _____

⇉ Now, we have to find the correct position of top node.

⇉ So, top node will compare itself with its children.

⇉ And if any children is greater than ~~their parent~~ that node, we will swap them.

⇉ After that again check.

⇉ This whole process is called Heapify.

⇉ Means finding the correct position of any node.

⇉ Time Complexity for deletion is O(log n).