

DSA using Python

Heap

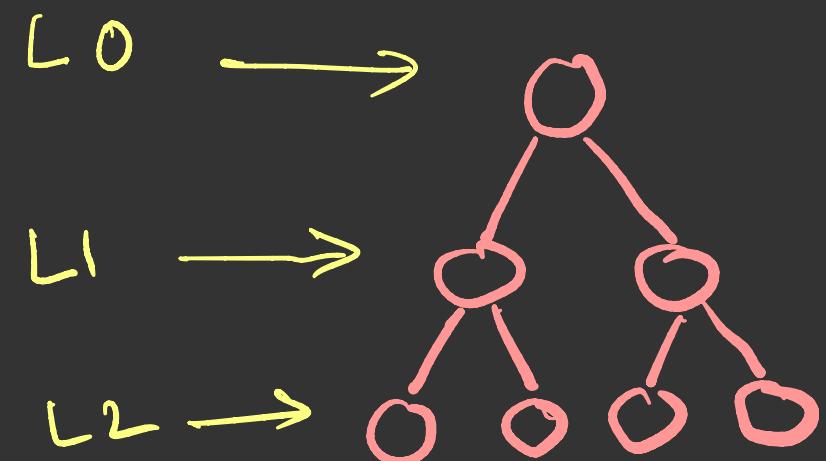


Saurabh Shukla (MySirG)

Agenda

- ① Complete Binary Tree
- ② Almost Complete Binary Tree
- ③ Heap
- ④ Insertion in heap
- ⑤ Deletion in heap
- ⑥ Heap Sort

Complete Binary Tree



All levels must be completely filled.

$$L_0 \rightarrow 1$$

$$L_1 \rightarrow 2$$

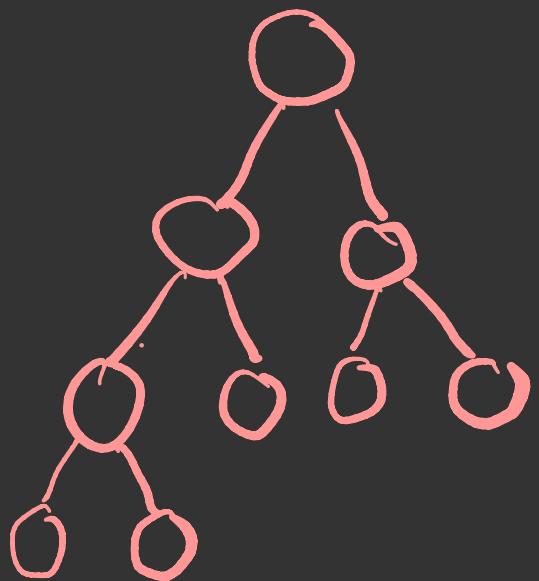
$$L_2 \rightarrow 4$$

$$L_3 \rightarrow 8$$

$$L_4 \rightarrow 16$$

$$L_n = 2^n$$

Almost Complete Binary Tree



All the levels must be completely filled except possibly the last level and all the nodes in the last level must be left aligned

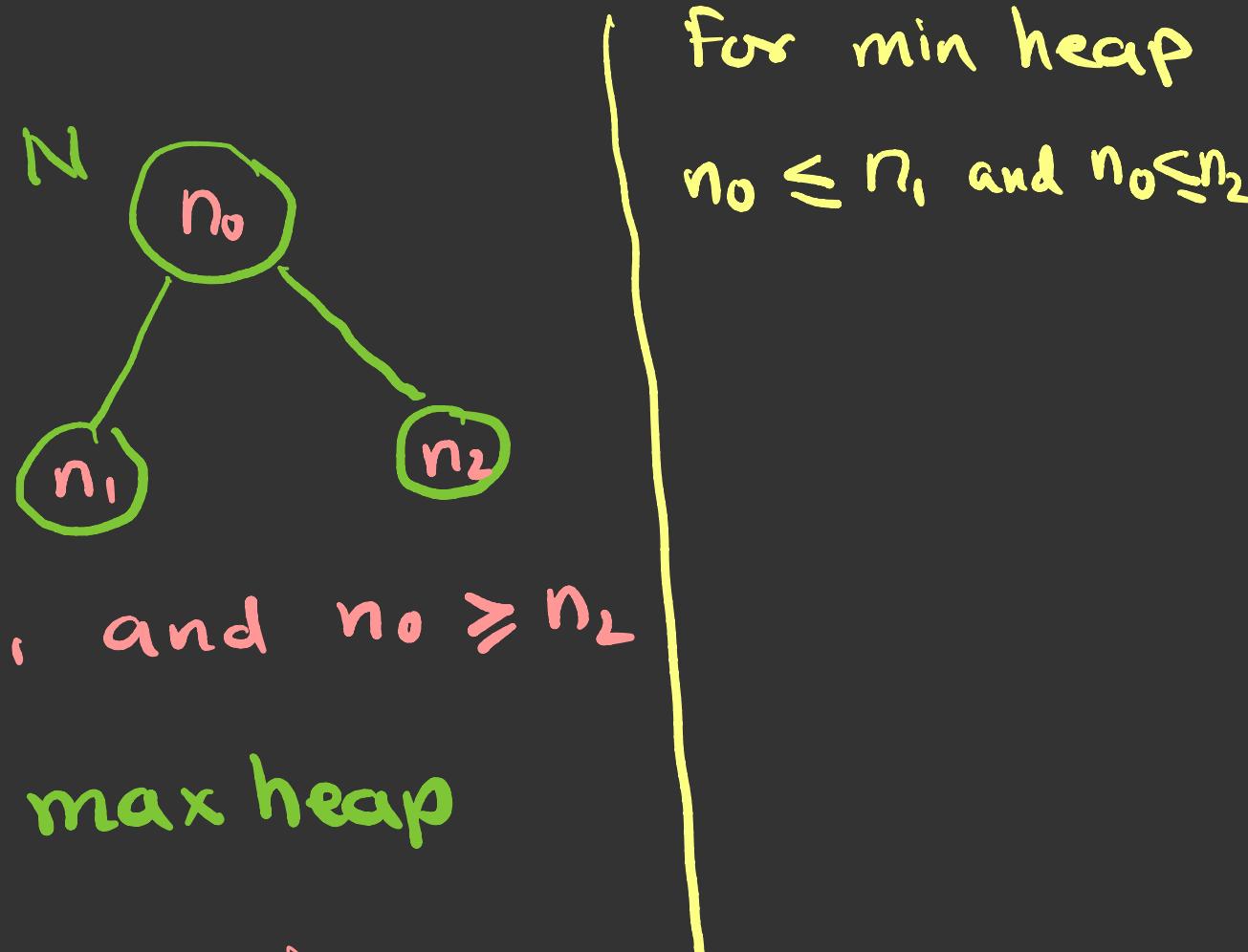
Heap

- Heap is a data structure
- Heap is used in a sorting algorithm known as heap sort
- Heaps are of two types
 - Max Heap (default)
 - Min Heap

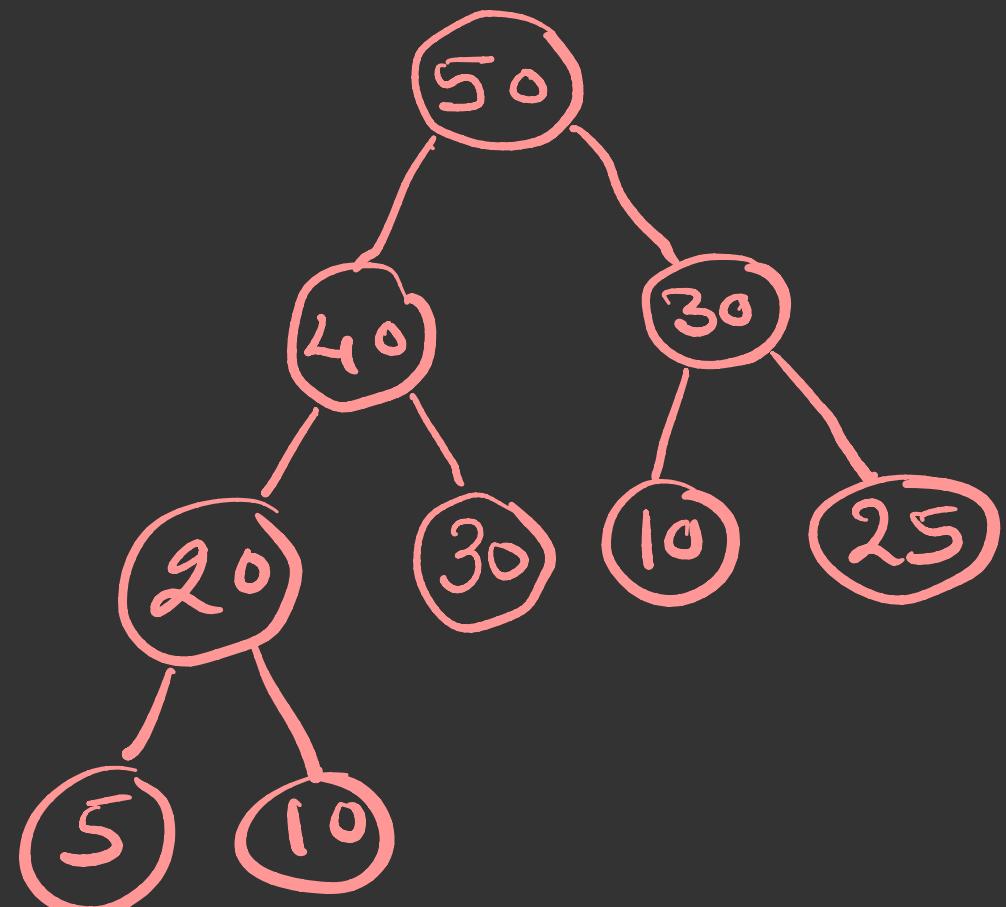
Heap Properties

The value at node N is greater than or equal to value at each children of node N

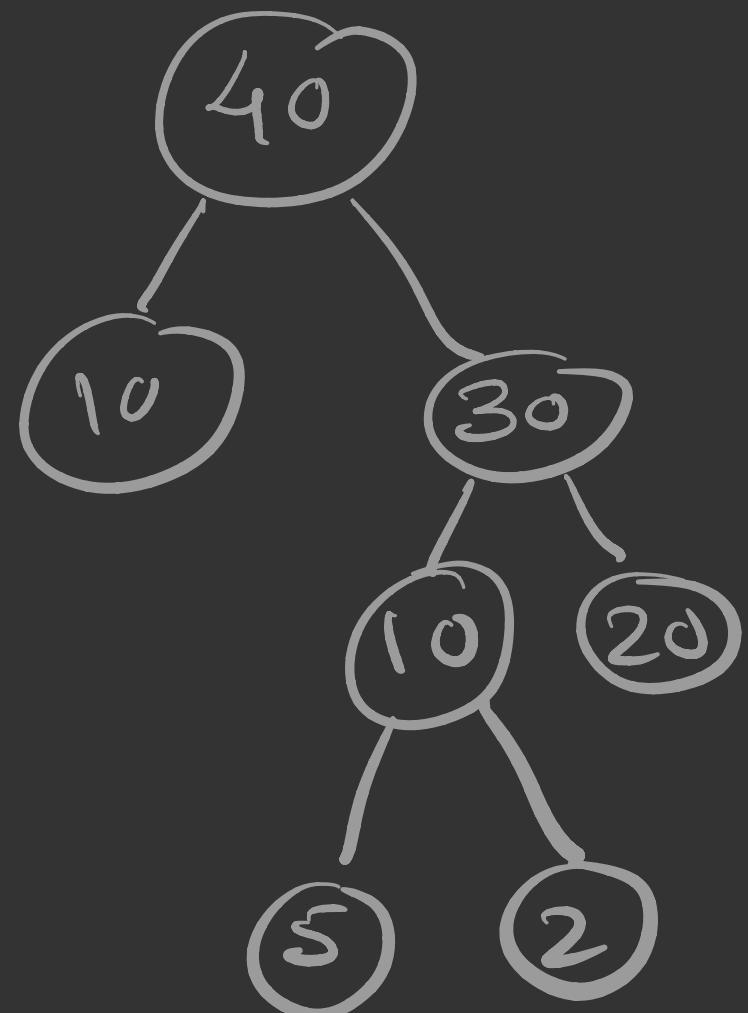
Heap must be an almost complete binary tree.



Example



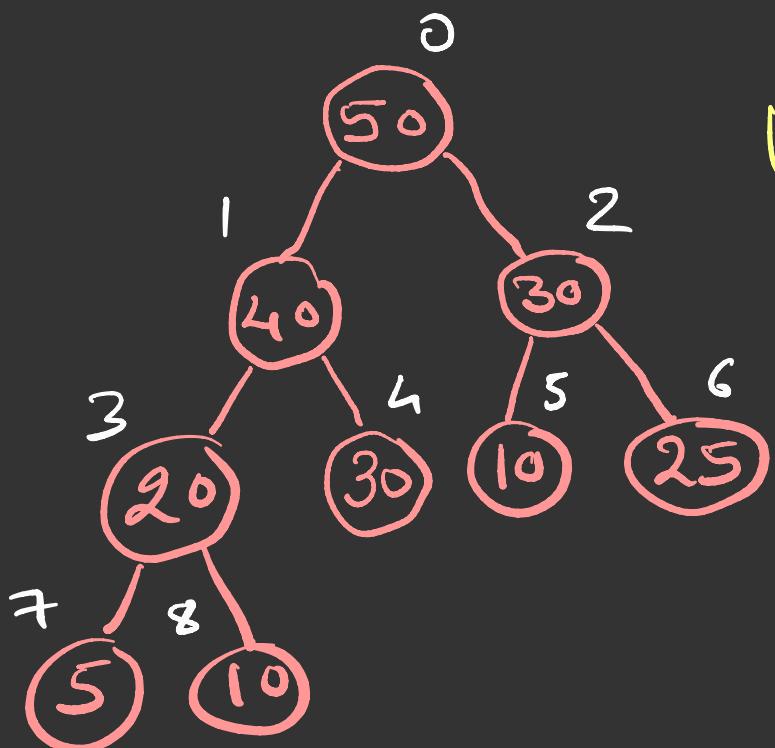
Max - heap



Not a heap

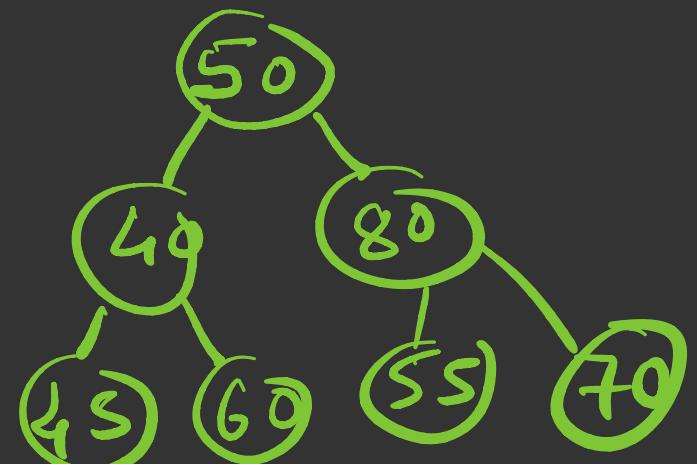
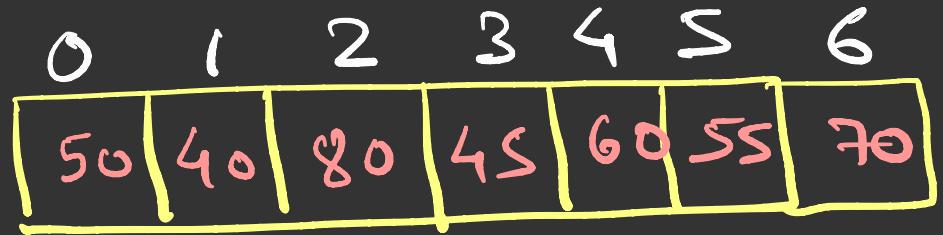
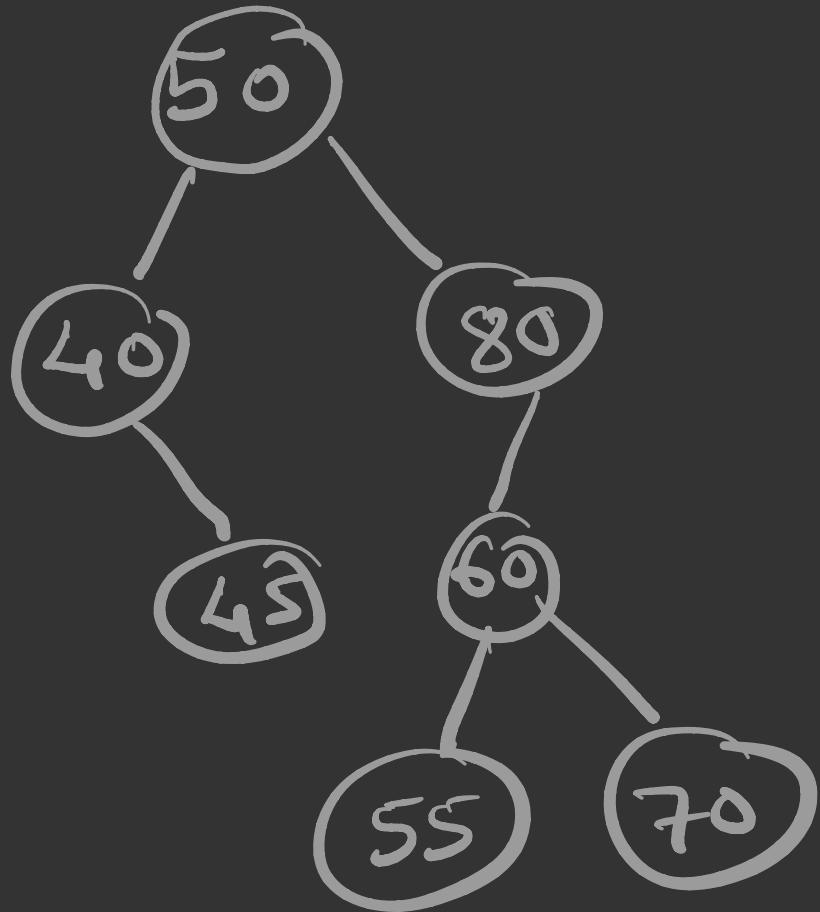
Representation of Heap

Unless otherwise stated, heap is maintained in memory by a linear array (list)

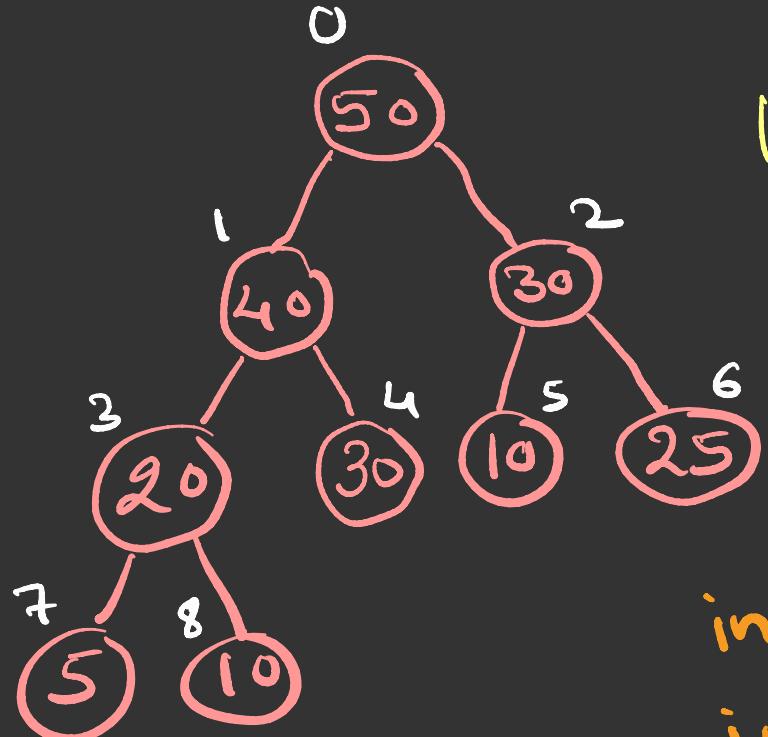


H

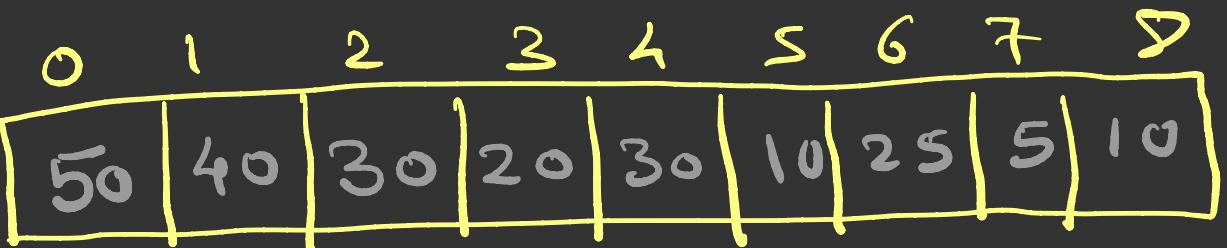
0	1	2	3	4	5	6	7	8
50	40	30	20	30	10	25	5	10



How to find parent or child node?



H



How to find index of child nodes?

index is index of node N

index of left child = $2 * \text{index} + 1$

index of right child = $2 * \text{index} + 2$

How to find index of parent node?

index of node N = index

index of Parent node of N = $\frac{\text{index} - 1}{2}$

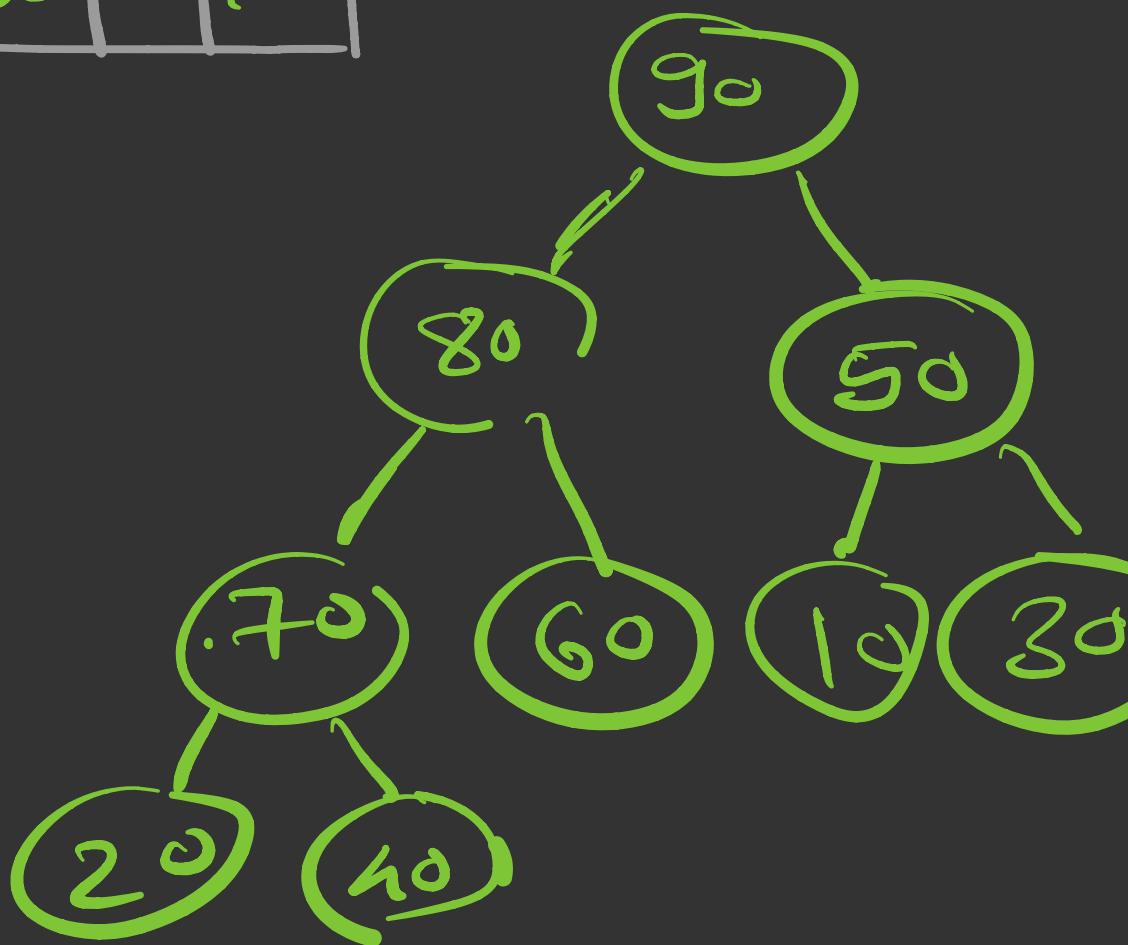
Insertion

0	1	2	3	4	5	6	7	8
40	70	10	90	60	30	50	20	80

H

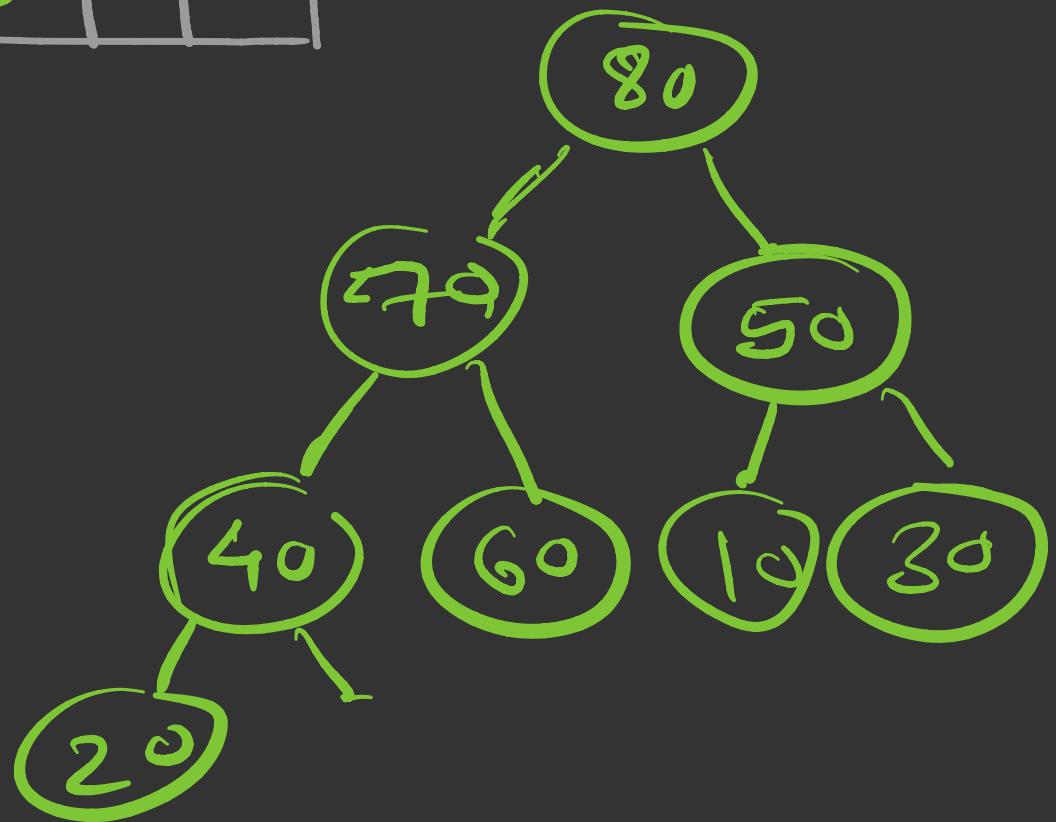
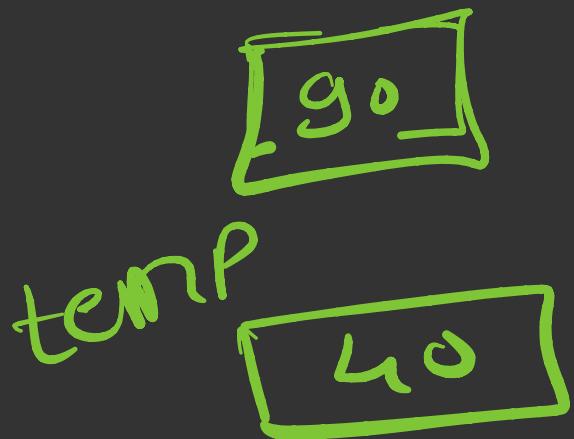
0	1	2	3	4	5	6	7	8
90	80	50	70	60	10	30	20	40

[]



Deletion

H	0	1	2	3	4	5	6	7	8
	80	70	50	40	60	10	30	20	



Heap Sort

Delete values from the heap(max-heap) and store them in an array from right to left. As a result, at the end of deleting all the elements of heap, array becomes sorted

Heap can be used to implement priority queue