DSA through Java

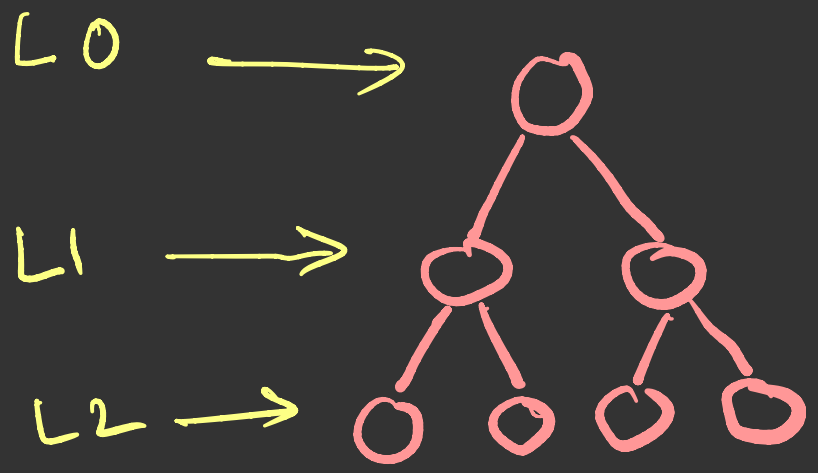
Heap

Saurabh Shukla (MySirG)

Agenda

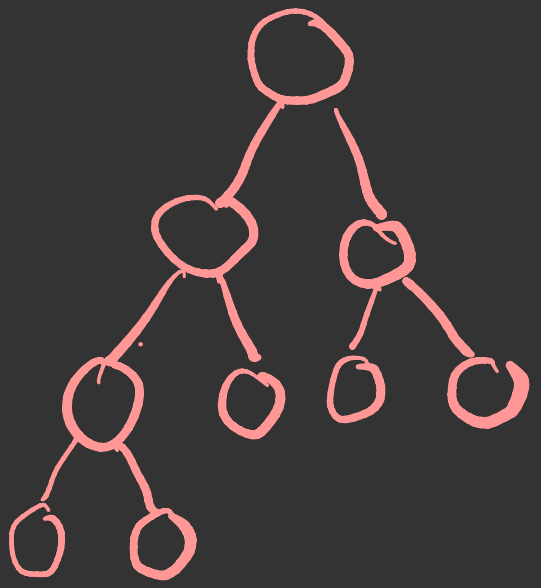
1. Complete Binary Tree
2. Almost Complete Binary Tree
3. Heap
4. Insertion in heap
5. Deletion in heap

Complete Binary Tree



All levels must be completely filled.

Almost Complete Binary Tree



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

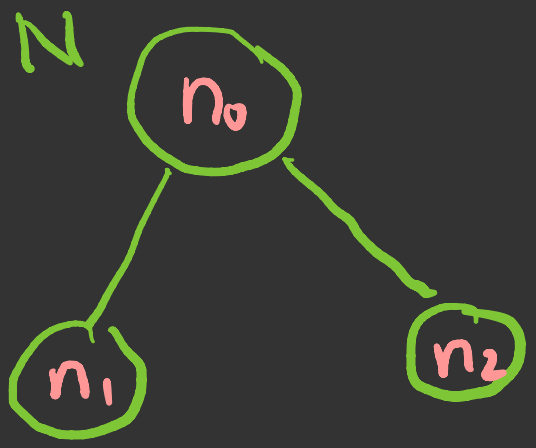
Heap

- Heap is a data structure
- Heap is used in a sorting algorithm known as heap sort
- Heap 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.



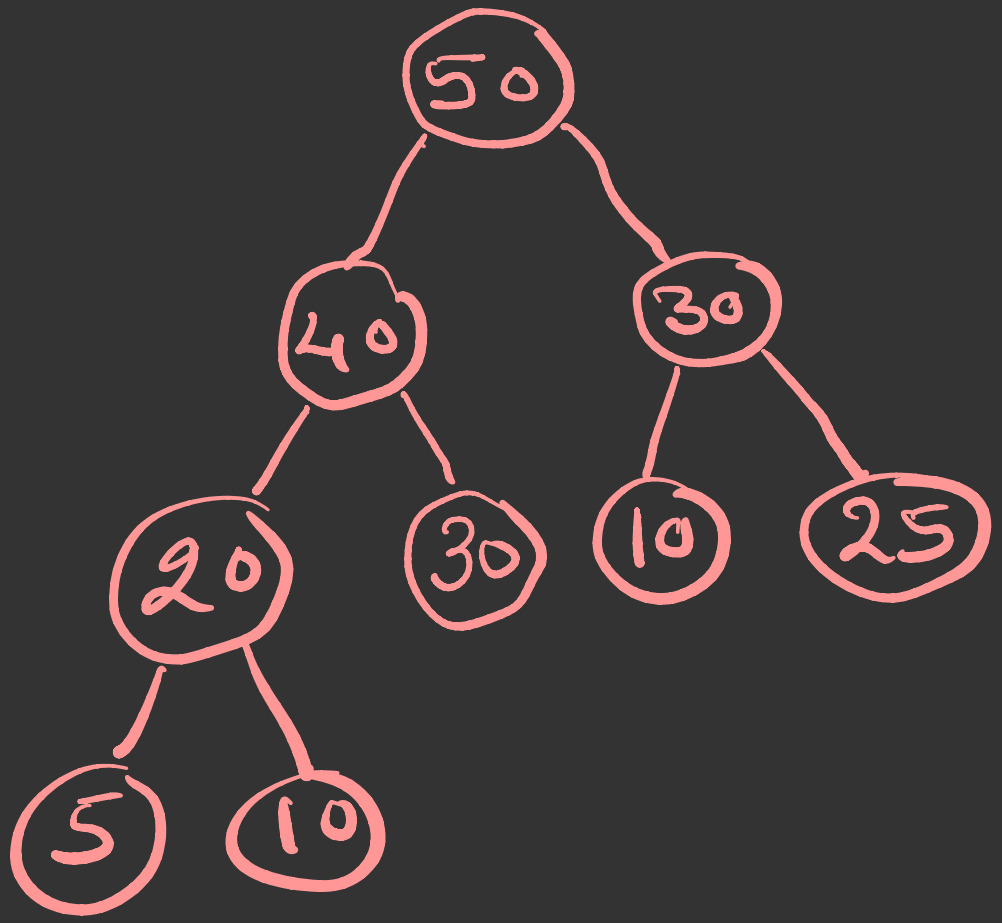
$n_0 \geq n_1$ and $n_0 \geq n_2$

This heap is max heap

For min heap

$n_0 \leq n_1$ and $n_0 \leq n_2$

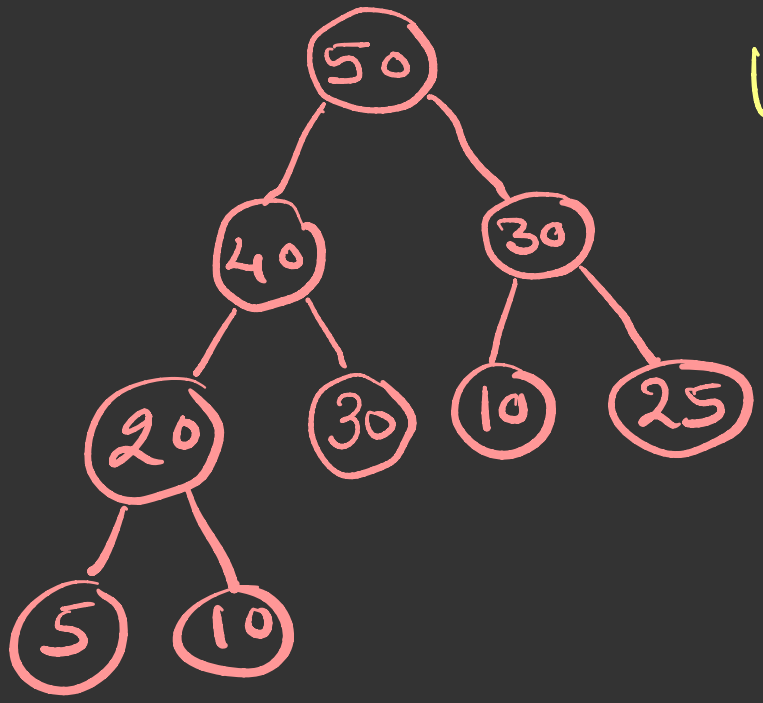
Example



Max - heap

Representation of Heap

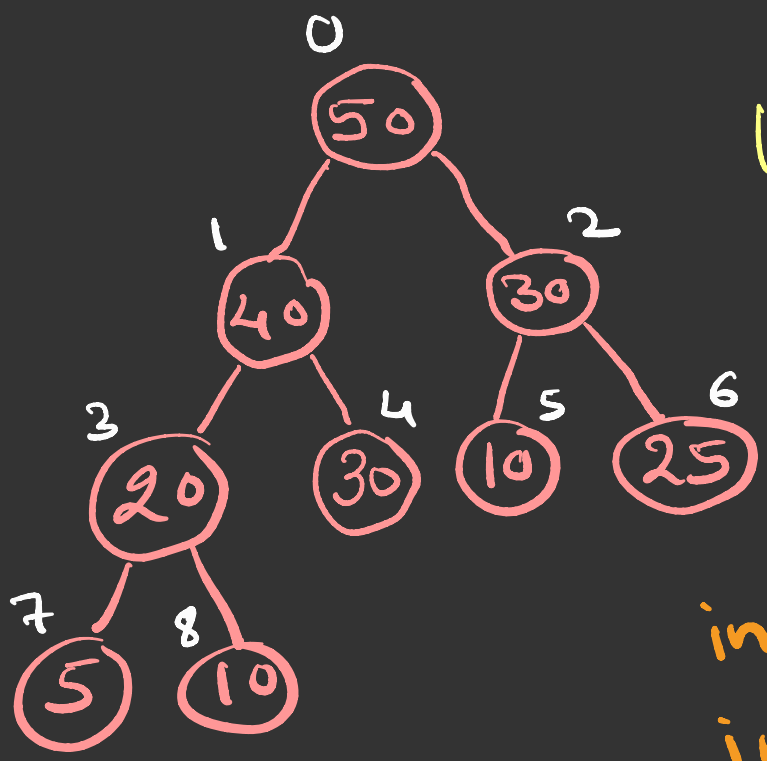
Unless otherwise stated, heap is maintained in memory by a linear array.



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

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

How to find index of child nodes?

index is index of node N

index of left child = 2 * index + 1

index of right child = 2 * index + 2

How to find index of parent node?

index of node N = index

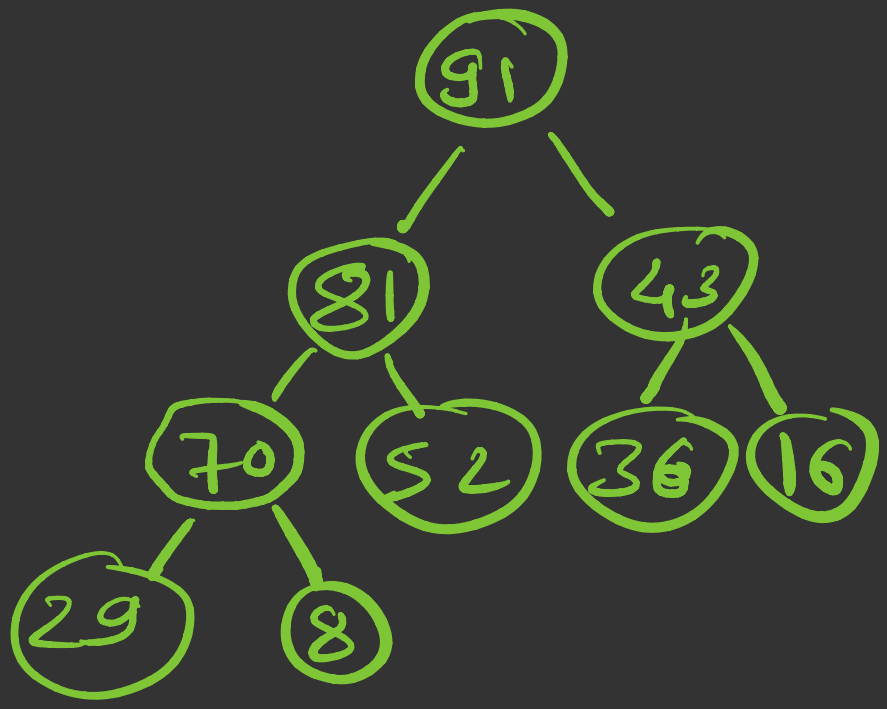
index of Parent node of N = $\frac{(index - 1)}{2}$

Insertion

0	1	2	3	4	5	6	7	8
70	81	36	29	52	43	16	91	8

	0	1	2	3	4	5	6	7	8
H	91	81	43	70	52	36	16	29	8

size = 9



Deletion

	0	1	2	3	4	5	6	7	8
H	70	52	43	29	8	36	16	1	

Size = 8 8 7

91 81