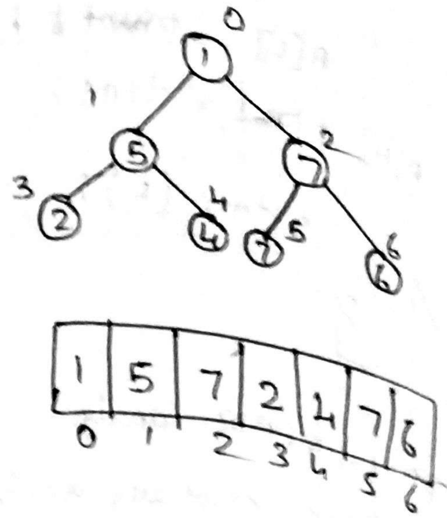
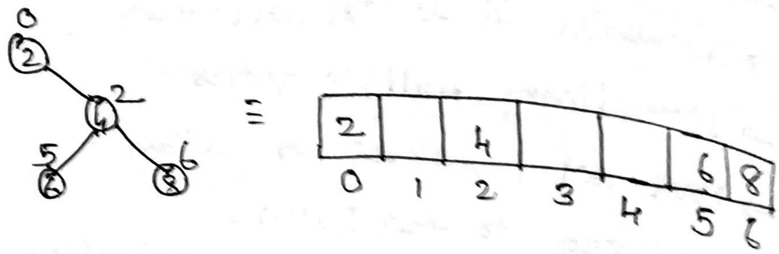
TREES

→ Tree is stored in an array based implementation as described pictorially



→ If any node is childless, the corresponding positions in the array is left blank.

→ For Eg:-



→ Space usage of tree implemented using an array is $O(N)$ in best case and $O(2^n-1)$ in worst case.

→ Running time of traversing the tree implemented using an array is $O(\log_2 N)$ in the best case and $O(N)$ in the worst case.

→ N - size of array.

n - number of elements.

$n \in [\log_2 N, N]$

→ Array index for left child = 2 * Index of Tree's height

Array index for right child = 2 * Index of Tree's height +1

→ If every node of a tree has either 0 or 2 children, it is known as proper tree.

- Nodes without children are known as external nodes.
- Nodes with children are known as internal nodes.
- height is counted from 0.

$$\boxed{\begin{array}{l} n \in [h+1, 2^{h+1}-1] \\ n_I \in [h, 2^h-1] \\ n_E \in [1, 2^h] \\ h \in [\log(n+1)-1, n-1] \end{array}}$$

— Binary Tree

- If every node of a tree has 0, 1 or 2 children, it is known as binary tree.

$$\boxed{\begin{array}{l} n \in [2h+1, 2^{h+1}-1] \\ n_I = [h, 2^h-1] \\ n_E \in [n_E, 2^h] \\ h \in [\log(n+1)-1, (n-1)/2] \end{array}}$$

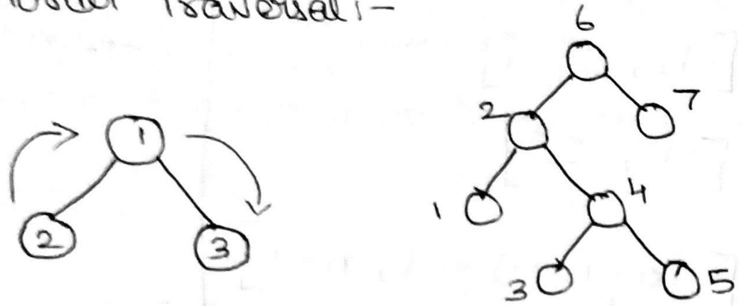
– Proper Binary Tree

- There are three types of traversals in trees namely Inorder traversal, Preorder Traversal, Postorder traversal.
- In Inorder traversal, the left child will be traversed first, followed by the parent and then the right child.

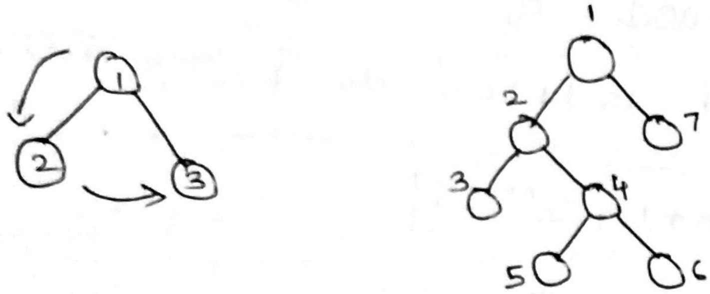
→ In Preorder traversal parent is traversed first followed by left child and right child.

→ In Post order traversal the left child is traversed first followed by right child and parent.

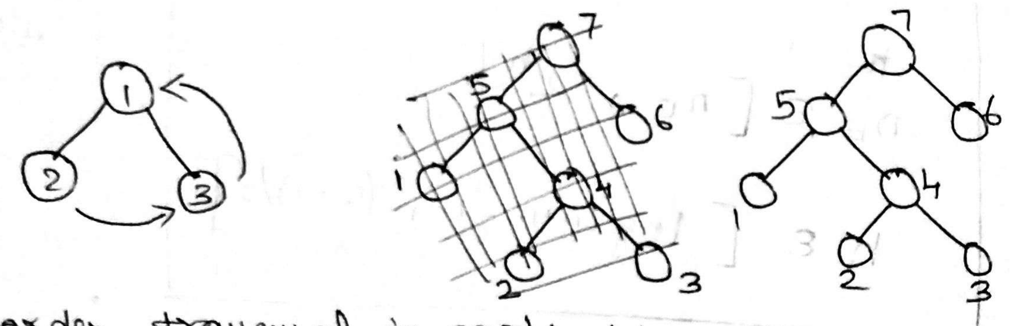
Inorder Traversal :-



Preorder Traversal :-



Postorder Traversal :-



→ Inorder traversal is applicable only to binary trees

→ Inorder Expression :- A * B + D   A*B+D

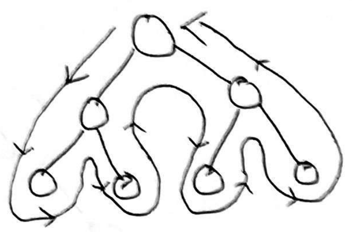
→ Preorder / Prefix Expression :- + * ABD   +* ABD

→ Postorder / Postfix Expression :- AB * D +   AB * D +

BREADTH – FIRST TRAVERSAL
BINARY SEARCH TREE

- Tree is a graph with n nodes and n-1 edges.
- Dynamic tree is a special type of binary search trees where edges are added and removed dynamically.
- Euler tour is a type of traversal on the tree.

Euler tour :-



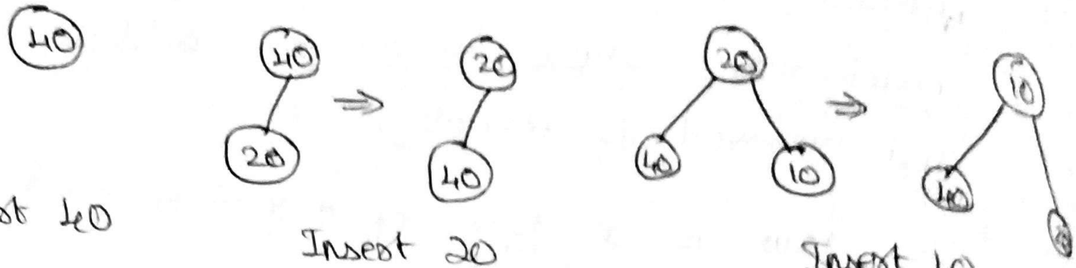
- (A,B) trees are the type of trees where each node has n children where $n \in [A,B]$.

BINARY SEARCH TREES

- If one node is deleted its child gets promoted
- Heap is a complete tree.
- Bubblesup and bubblesdown operations are done to insert / remove elements to / from a heap respectively.
- Top-down construction complexity of tree $= O(n \log n)$.
- Bottom-up construction complexity of tree $= O(n)$
- When the queue is static (ie. no new records are added during the process) bottom up construction can be used.

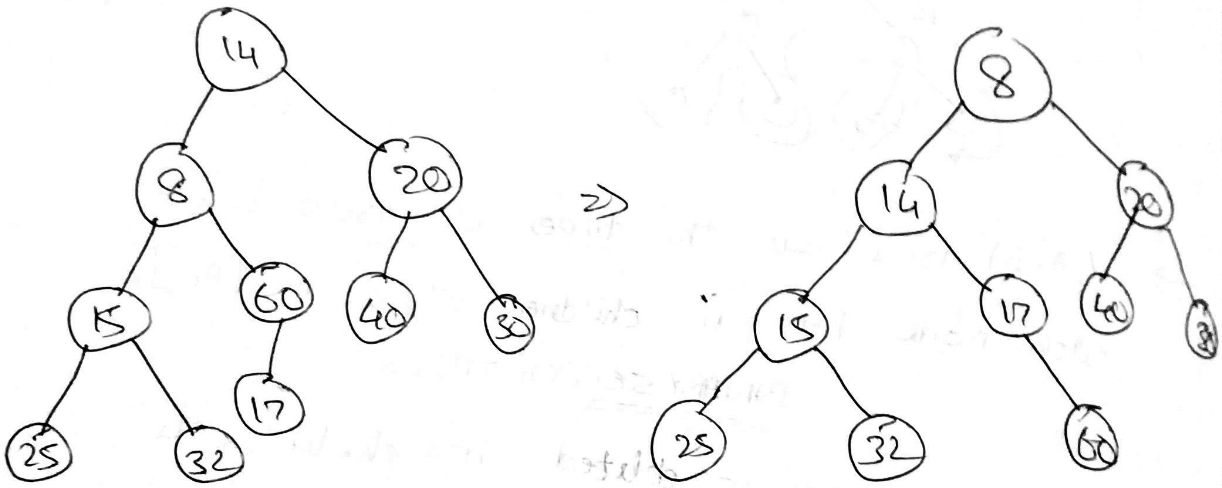
→ When the queue is dynamic top-down construction of tree is used.

Top-down Construction



Insert 40 Insert 20 Insert 10

Bottom up construction :



→ We are using top-down or bottom-up to construct heap doing n removals to get sorted order.

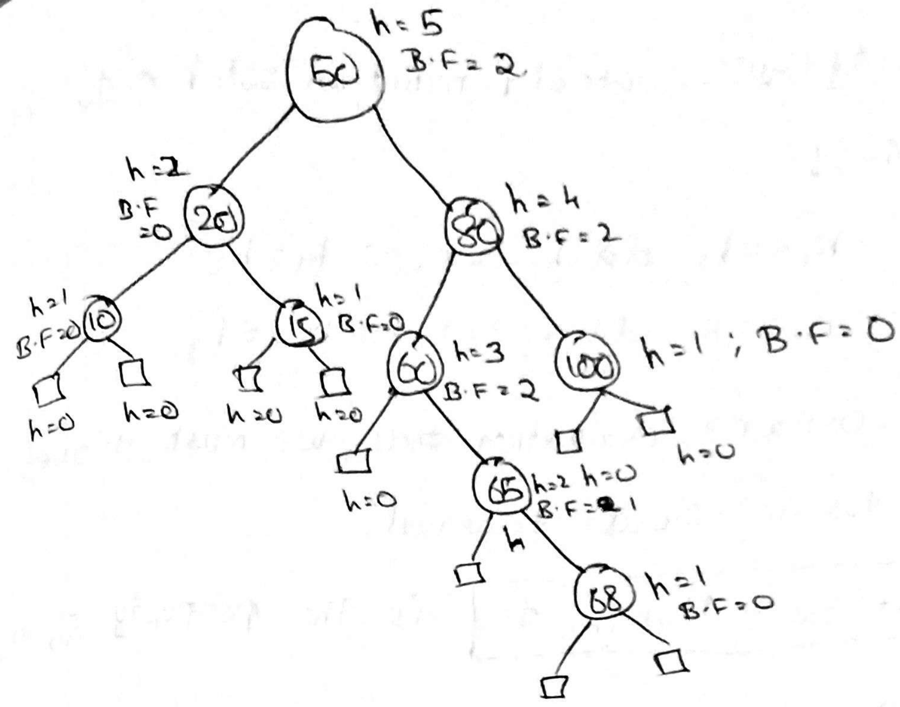
AVL Trees

→ AVL Trees perform optimized binary searches.

→ They are balanced.

→ If the tree is unbalanced, we should restructure the tree.

→ When the difference of h_l and h_r is minimum it is known as height balanced BST.

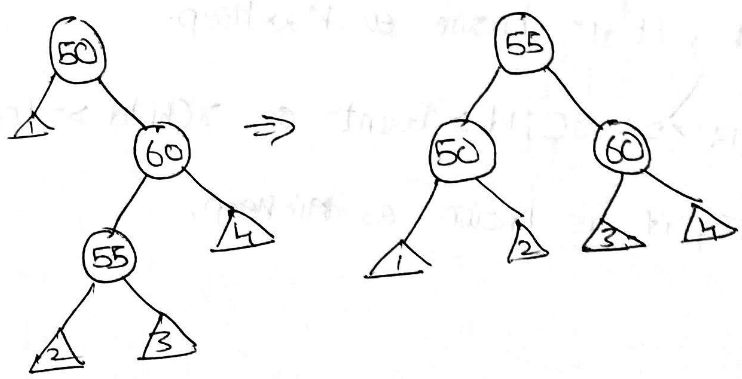


→ Height of parent = max (Height of leftchild, Height of right child) +1.

→ Balancing Factor (B.F) = |Height of leftchild - Height of rightchild|. (ie. It can also be negative).

→ We must restructure the AVL tree such that the balancing factor is 0 or 1.

Eg:-



→ Priority queues should maintain total order relation property.

Eg:- $k_1 <= k_2$ && $k_2 <= k_1 \Rightarrow k_1 = k_2$

$k_1 <= k_2$ && $k_2 <= k_3 \Rightarrow k_1 <= k_3$

→ To evaluate evaluation trees, we must traverse the tree in inorder traversal.

→ $\boxed{N_{external} = N_{internal} + 1}$ is the property of proper trees.

→ Comparator ADT is a comparator method for comparing abstract data types.

→ If lChild < Parent < rChild or rChild < Parent < lChild, it is a Binary search tree.

→ Binary search Trees (BST) are AVL trees if Balancing Factor (B.F) is 1.

→ If lchild <= rChild < Parent or rChild <= lchild < Parent, it is known as MaxHeap.

→ If lChild >= rChild > Parent or rChild >= lChild > Parent, it is known as MinHeap.