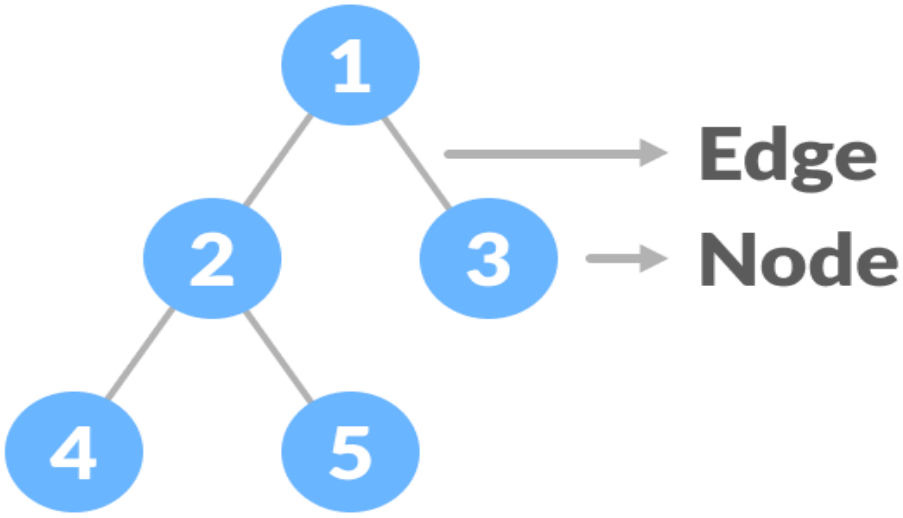
DS-UNIT 2

1. Explain Trees data Structure.

Ans-A tree is non-linear and a hierarchical data structure consisting of a collection of nodes such that each node of the tree stores a value and a list of references to other nodes (the "children"). This data structure is a specialized method to organize and store data in the computer to be used more effectively.



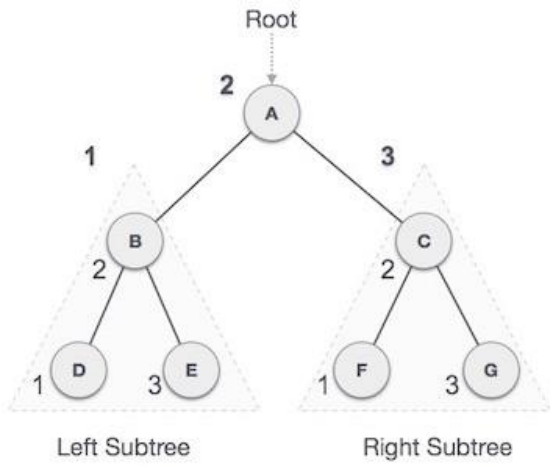
2. Explain operations on Tree.

Ans-The basic operations that can be performed on a binary search tree data structure, are the following –

- **Insert** – Inserts an element in a tree/create a tree.
- **Search** – Searches an element in a tree.
- **Preorder Traversal** – Traverses a tree in a pre-order manner.
- **Inorder Traversal** – Traverses a tree in an in-order manner.
- **Postorder Traversal** – Traverses a tree in a post-order manner.

3. Explain traversing algorithm used in TREE.

Ans-In-order Traversal-In this traversal method, the left subtree is visited first, then the root and later the right sub-tree. We should always remember that every node may represent a subtree itself.



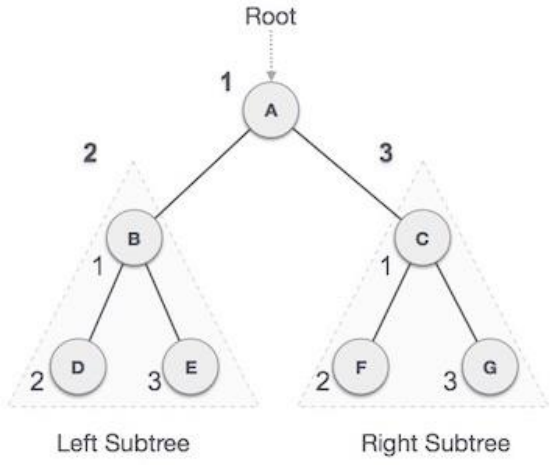
$D \rightarrow B \rightarrow E \rightarrow A \rightarrow F \rightarrow C \rightarrow G$

Step 1 − Recursively traverse left subtree.

Step 2 − Visit root node.

Step 3 − Recursively traverse right subtree.

Pre-order Traversal-In this traversal method, the root node is visited first, then the left subtree and finally the right subtree.



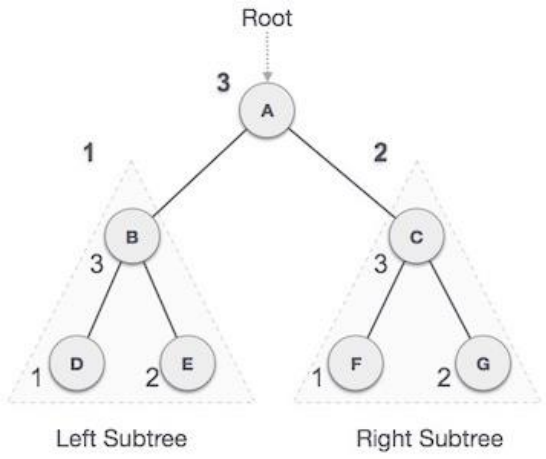
$A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow F \rightarrow G$

Step 1 − Visit root node.

Step 2 − Recursively traverse left subtree.

Step 3 − Recursively traverse right subtree.

Post-order Traversal-In this traversal method, the root node is visited last, hence the name. First we traverse the left subtree, then the right subtree and finally the root node.



D → E → B → F → G → C → A

Step 1 − Recursively traverse left subtree.

Step 2 − Recursively traverse right subtree.

Step 3 − Visit root node.

4. Explain AVL Tree.

Ans-An AVL tree is a type of binary search tree. Named after it's inventors Adelson, Velskii, and Landis, AVL trees have the property of dynamic self-balancing in addition to all the other properties exhibited by binary search trees. A BST is a data structure composed of nodes.

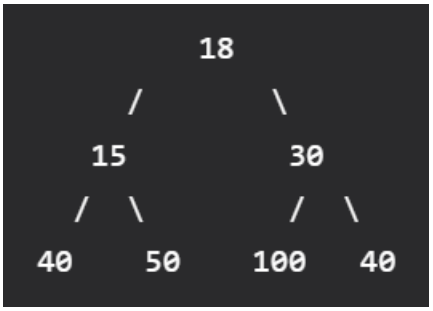
Balance Factor (k) = height (left(k)) - height (right(k))

- If balance factor of any node is 1, it means that the left sub-tree is one level higher than the right sub-tree.
- If balance factor of any node is 0, it means that the left sub-tree and right sub-tree contain equal height.
- If balance factor of any node is -1, it means that the left sub-tree is one level lower than the right sub-tree.

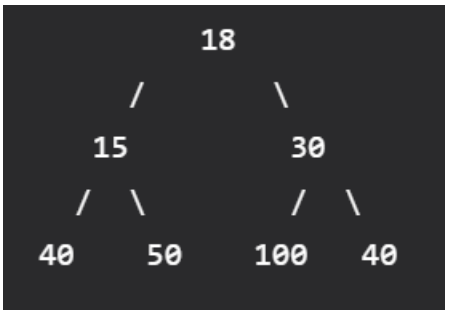
5. Explain Binary Tree.

Ans-A binary tree is a rooted tree that is also an ordered tree (a.k.a. plane tree) in which every node has at most two children. A rooted tree naturally imparts a notion of levels (distance from the root), thus for every node a notion of children may be defined as the nodes connected to it a level below.

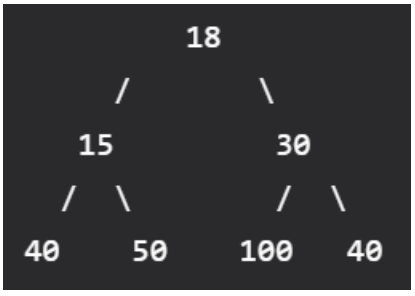
Full Binary Tree: A Binary Tree is a full binary tree if every node has 0 or 2 children. A full Binary tree is a special type of binary tree in which every parent node/internal node has either two or no children.



Complete Binary Tree: A Binary Tree is a Complete Binary Tree if all the levels are completely filled except possibly the last level and the last level has all keys as left as possible.



Perfect Binary Tree: A Binary tree is a Perfect Binary Tree in which all the internal nodes have two children and all leaf nodes are at the same level.



6. Explain AVL Tree.

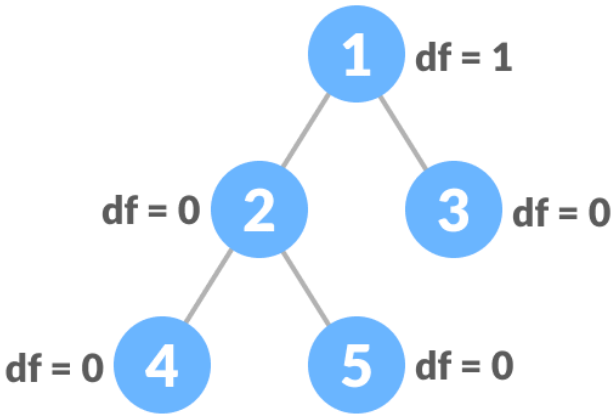
Ans- Same as Ans 4.

7. What is balanced binary tree? Explain it.

Ans-A balanced binary tree, also referred to as a height-balanced binary tree, is defined as a binary tree in which the height of the left and right subtree of any node differ by not more than 1.

Following are the conditions for a height-balanced binary tree:

1. Difference between the left and the right subtree for any node is not more than one.
2. The left subtree is balanced.
3. The right subtree is balanced.



8. Explain threaded binary tree.

Ans-A binary tree is made threaded by making all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists).

There are two types of threaded binary trees.

- Single Threaded: Where a NULL right pointers is made to point to the inorder successor (if successor exists)
- Double Threaded: Where both left and right NULL pointers are made to point to inorder predecessor and inorder successor respectively. The predecessor threads are useful for reverse inorder traversal and postorder traversal.

Types of Threaded Binary Tree

There are two types of threaded Binary Tree:

- One-way threaded Binary Tree
- Two-way threaded Binary Tree

9. What is Heap? Explain it.

Ans-A heap is a specialized tree-based data structure that satisfied the heap property: if B is a child node of A, then key(A) ≥ key(B).

10. Explain heappush() and heappop() functions.

Ans-heappush(heap, ele): This function is used to insert the element mentioned in its arguments into a heap. The order is adjusted, so that heap structure is maintained.

heappop(heap): This function is used to remove and return the smallest element from the heap. The order is adjusted, so that heap structure is maintained.

11. Explain how to create heap.

Ans-Heap is a special case of balanced binary tree data structure where the root-node key is compared with its children and arranged accordingly. If α has child node β then −

$$key(\alpha) \geq key(\beta)$$

Step 1 − Create a new node at the end of heap.

Step 2 − Assign new value to the node.

Step 3 − Compare the value of this child node with its parent.

Step 4 − If value of parent is less than child, then swap them.

Step 5 − Repeat step 3 & 4 until Heap property holds.

12. Explain how to create Tree data structure.

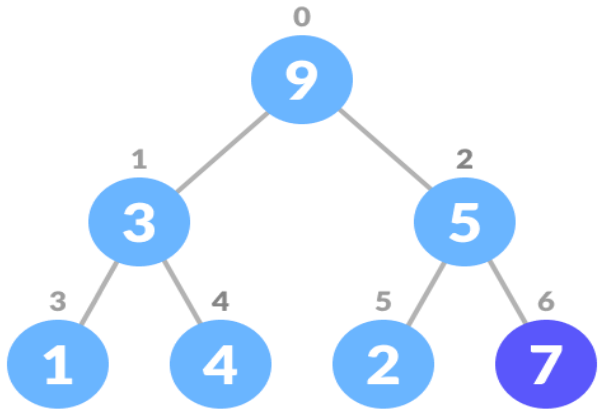
Ans-A tree is non-linear and a hierarchical data structure consisting of a collection of nodes such that each node of the tree stores a value and a list of references to other nodes (the “children”).

13. Explain how to perform operations with priority queue.

Ans-Priority Queue Operations

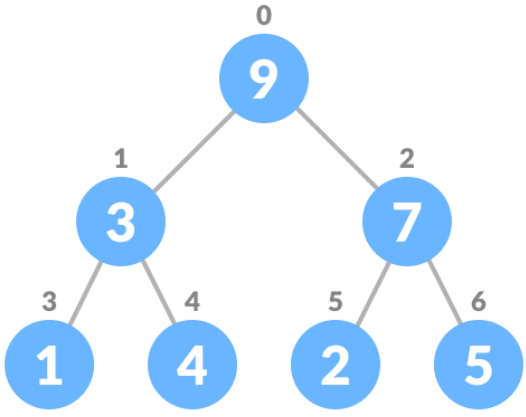
Basic operations of a priority queue are inserting, removing, and peeking elements.

1. Inserting an Element into the Priority Queue



Insert an element at the end of the queue

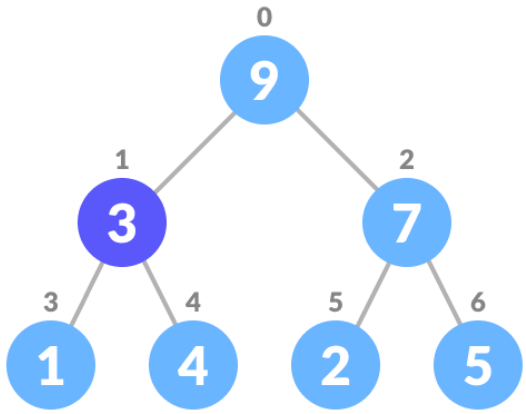
Heapify the tree.



Heapify after insertion

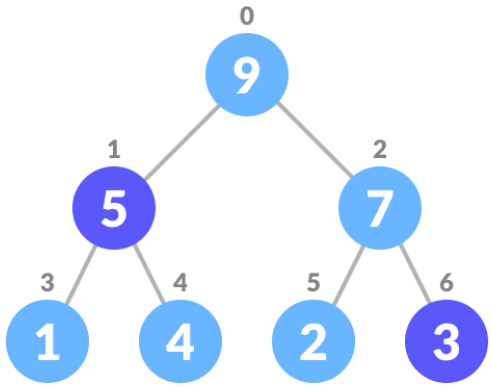
2. Deleting an Element from the Priority Queue

1. Select the element to be deleted.



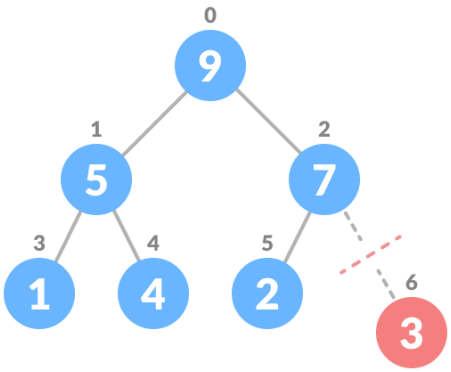
Select the element to be deleted

2. Swap it with the last element.



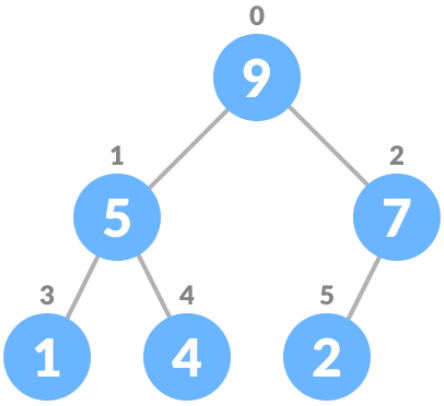
Swap with the last leaf node element

3. Remove the last element.



Remove the last element leaf

4. Heapify the tree.



Heapify the priority queue

14. What is priority queue ? explain it.

Ans-Priority Queue is an abstract data type that is similar to a queue, and every element has some priority value associated with it. The priority of the elements in a priority queue determines the order in which elements are served (i.e., the order in which they are removed).

Properties of Priority Queue

- Every item has a priority associated with it.
- An element with high priority is dequeued before an element with low priority.
- If two elements have the same priority, they are served according to their order in the queue.

15. Explain enqueue and dequeue operations.

Ans-Enqueue Operation

- check if the queue is full
- for the first element, set the value of (FRONT0 to 0
- increase the (REAR) index by 1
- add the new element in the position pointed to by (REAR)

Dequeue Operation

- check if the queue is empty
- return the value pointed by (FRONT)
- increase the (FRONT) index by 1
- for the last element, reset the values of (FRONT) and (REAR) to -1

16. Explain doubly linked list.

Ans-Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List. Following are the important terms to understand the concept of doubly linked list.

- **Link** – Each link of a linked list can store a data called an element.
- **Next** – Each link of a linked list contains a link to the next link called Next.
- **Prev** – Each link of a linked list contains a link to the previous link called Prev.

Doubly Linked List Representation

- Doubly Linked List contains a link element called first and last.
- Each link carries a data field(s) and two link fields called next and prev.
- Each link is linked with its next link using its next link.
- Each link is linked with its previous link using its previous link.
- The last link carries a link as null to mark the end of the list.

17. List and explain advantages and disadvantages of doubly linked list.

Ans-Advantages Of DLL:

- Reversing the doubly linked list is very easy.
- It can allocate or reallocate memory easily during its execution.
- As with a singly linked list, it is the easiest data structure to implement.
- The traversal of this doubly linked list is bidirectional which is not possible in a singly linked list.

Disadvantages Of DLL:

- It uses extra memory when compared to the array and singly linked list.
- Since elements in memory are stored randomly, therefore the elements are accessed sequentially no direct access is allowed.

18. Explain how to calculate balance factor in AVL tree with example.

Ans-Balance Factor

Balance factor of a node in an AVL tree is the difference between the height of the left subtree and that of the right subtree of that node.

Balance Factor = (Height of Left Subtree - Height of Right Subtree) or (Height of Right Subtree - Height of Left Subtree)

The self balancing property of an avl tree is maintained by the balance factor. The value of balance factor should always be -1, 0 or +1.