Non-Linear Data Structures

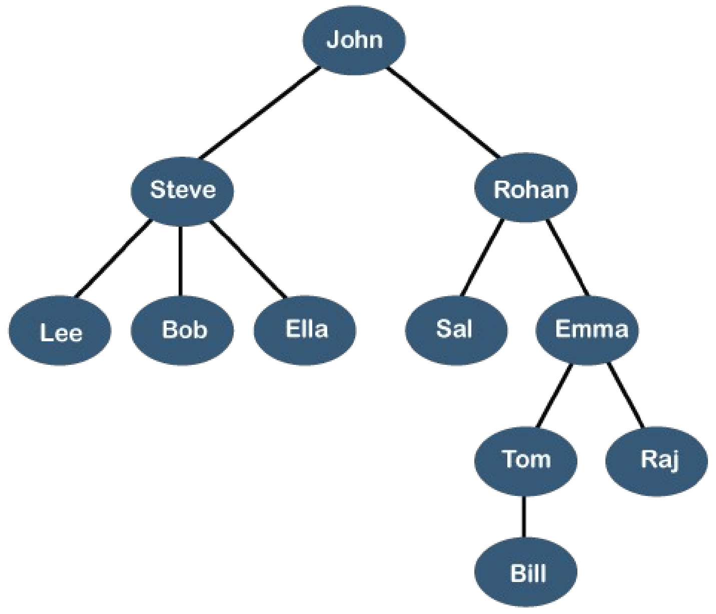
Tushar B. Kute,
http://tusharkute.com

Choosing a Data Structure

- What type of data needs to be stored?: It might be a possibility that a certain data structure can be the best fit for some kind of data.
- Cost of operations: If we want to minimize the cost for the operations for the most frequently performed operations. For example, we have a simple list on which we have to perform the search operation; then, we can create an array in which elements are stored in sorted order to perform the binary search. The binary search works very fast for the simple list as it divides the search space into half.
- Memory usage: Sometimes, we want a data structure that utilizes less memory.

Tree

- A tree is also one of the data structures that represent hierarchical data.
- Suppose we want to show the employees and their positions in the hierarchical form then it can be represented as shown below:



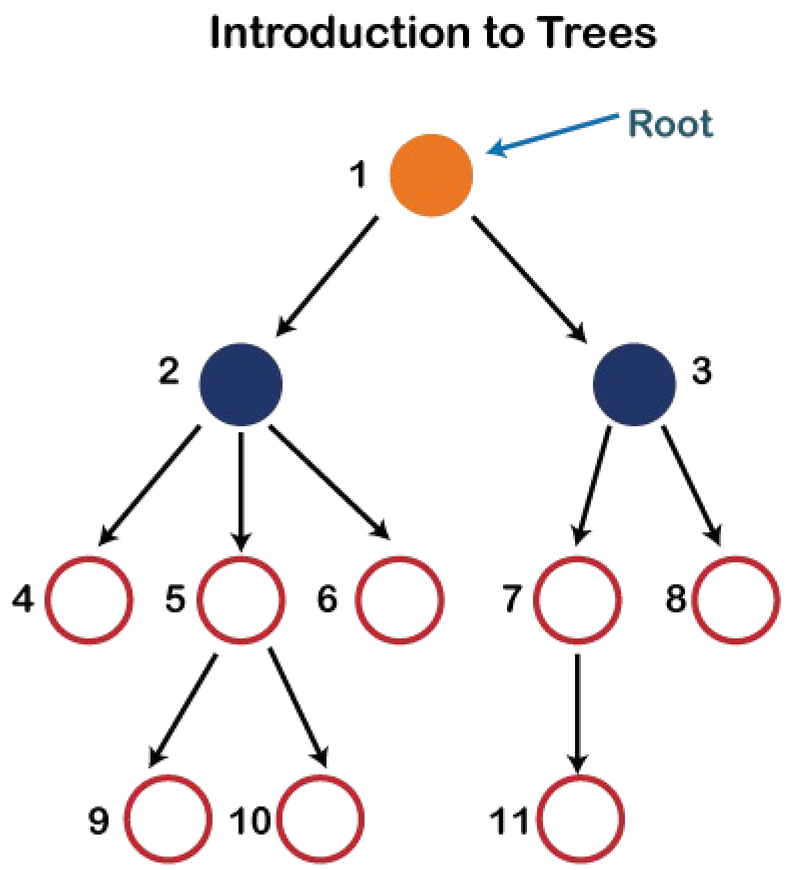
Tree

- The above tree shows the organization hierarchy of some company. In the above structure, john is the CEO of the company, and John has two direct reports named as Steve and Rohan.
- Steve has three direct reports named Lee, Bob, Ella where Steve is a manager. Bob has two direct reports named Sal and Emma. Emma has two direct reports named Tom and Raj.
- Tom has one direct report named Bill. This particular logical structure is known as a Tree. Its structure is similar to the real tree, so it is named a Tree. In this structure, the root is at the top, and its branches are moving in a downward direction.
- Therefore, we can say that the Tree data structure is an efficient way of storing the data in a hierarchical way.

Tree : Key Points

- A tree data structure is defined as a collection of objects or entities known as nodes that are linked together to represent or simulate hierarchy.
- A tree data structure is a non-linear data structure because it does not store in a sequential manner. It is a hierarchical structure as elements in a Tree are arranged in multiple levels.
- In the Tree data structure, the topmost node is known as a root node. Each node contains some data, and data can be of any type. In the above tree structure, the node contains the name of the employee, so the type of data would be a string.
- Each node contains some data and the link or reference of other nodes that can be called children.

Tree : Key Points



Tree : Key Points

- Root: The root node is the topmost node in the tree hierarchy. In other words, the root node is the one that doesn't have any parent.
- In the above structure, node numbered 1 is the root node of the tree. If a node is directly linked to some other node, it would be called a parent-child relationship.
- Child node: If the node is a descendant of any node, then the node is known as a child node.
- Parent: If the node contains any sub-node, then that node is said to be the parent of that sub-node.

Tree : Key Points

- Sibling: The nodes that have the same parent are known as siblings.
- Leaf Node:- The node of the tree, which doesn't have any child node, is called a leaf node. A leaf node is the bottom-most node of the tree. There can be any number of leaf nodes present in a general tree. Leaf nodes can also be called external nodes.
- Internal nodes: A node has atleast one child node known as an internal

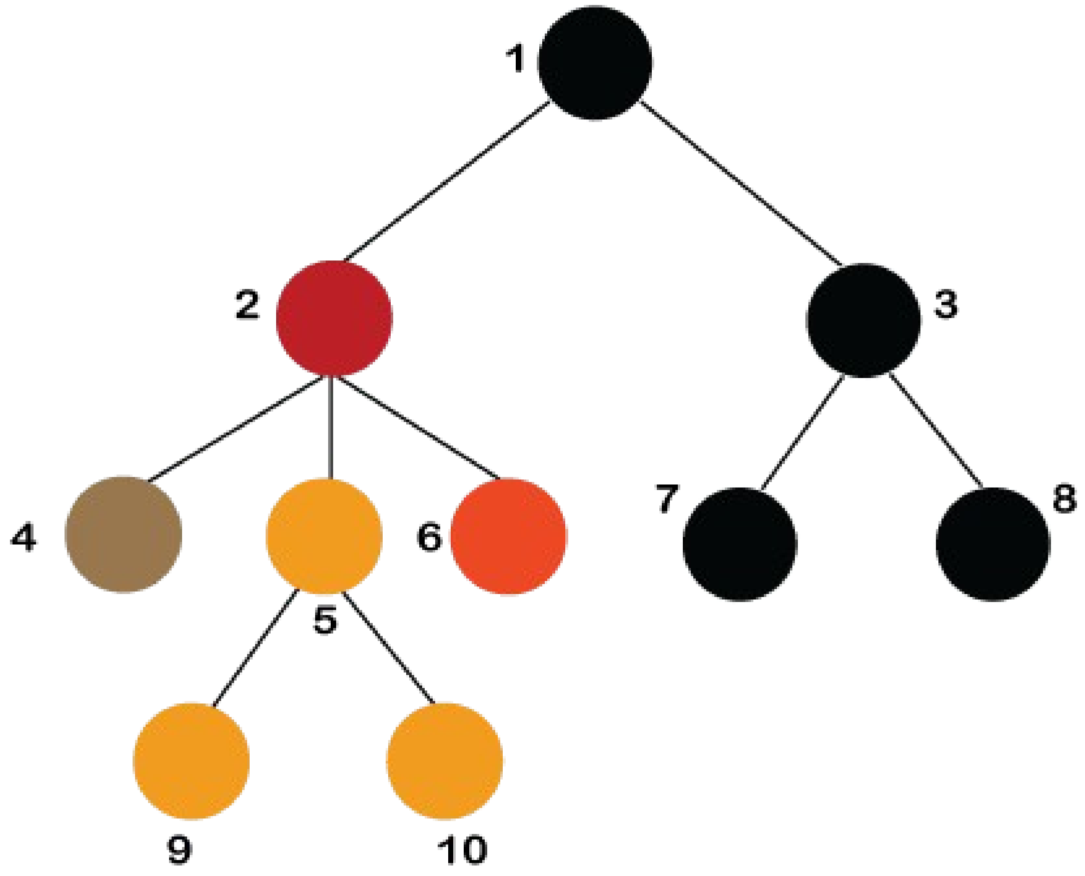
Tree : Key Points

- Ancestor node:- An ancestor of a node is any predecessor node on a path from the root to that node.
- The root node doesn't have any ancestors. In the tree shown in the above image, nodes 1, 2, and 5 are the ancestors of node 10.
- Descendant: The immediate successor of the given node is known as a descendant of a node. In the above figure, 10 is the descendant of node 5.

Tree : Properties

- Recursive data structure:
  - The tree is also known as a recursive data structure. A tree can be defined as recursively because the distinguished node in a tree data structure is known as a root node.
  - The root node of the tree contains a link to all the roots of its subtrees. The left subtree is shown in the yellow color in the below figure, and the right subtree is shown in the red color.
  - The left subtree can be further split into subtrees shown in three different colors. Recursion means reducing something in a self-similar manner.

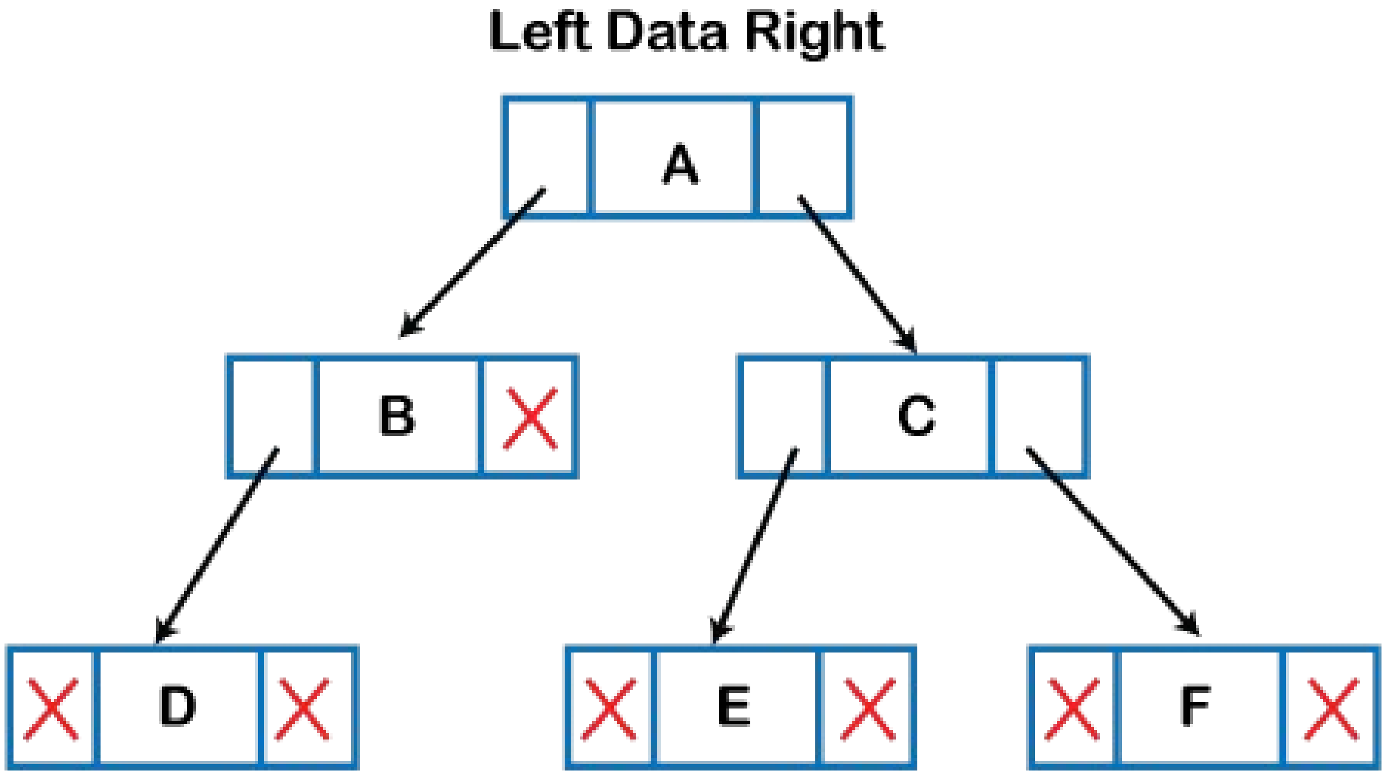
Tree : Properties



Tree : Properties

- Number of edges: If there are n nodes, then there would n-1 edges. Each arrow in the structure represents the link or path. Each node, except the root node, will have atleast one incoming link known as an edge. There would be one link for the parent-child relationship.
- Depth of node x: The depth of node x can be defined as the length of the path from the root to the node x. One edge contributes one-unit length in the path. So, the depth of node x can also be defined as the number of edges between the root node and the node x. The root node has 0 depth.
- Height of node x: The height of node x can be defined as the longest path from the node x to the leaf node.

Tree: Implementation



Tree: Implementation

- struct node
  ```
  {
      int data;
      struct node *left;
      struct node *right;
  }
  ```
- The above structure can only be defined for the binary trees because the binary tree can have utmost two children, and generic trees can have more than two children.
- The structure of the node for generic trees would be different as compared to the binary tree.

Tree: Application

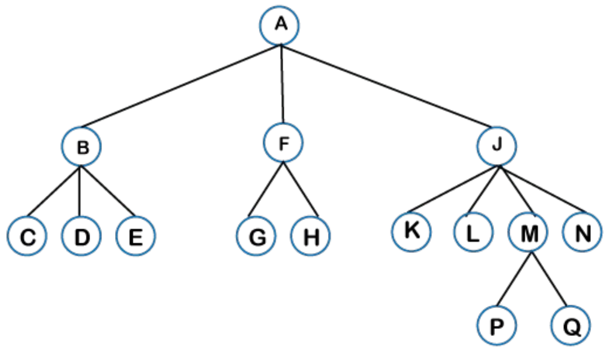
- Storing naturally hierarchical data: Trees are used to store the data in the hierarchical structure. For example, the file system. The file system stored on the disc drive, the file and folder are in the form of the naturally hierarchical data and stored in the form of trees.
- Organize data: It is used to organize data for efficient insertion, deletion and searching. For example, a binary tree has a logN time for searching an element.
- Trie: It is a special kind of tree that is used to store the dictionary. It is a fast and efficient way for dynamic spell checking.

Tree: Application

- Heap: It is also a tree data structure implemented using arrays. It is used to implement priority queues.
- B-Tree and B+Tree: B-Tree and B+Tree are the tree data structures used to implement indexing in databases.
- Routing table: The tree data structure is also used to store the data in routing tables in the routers.

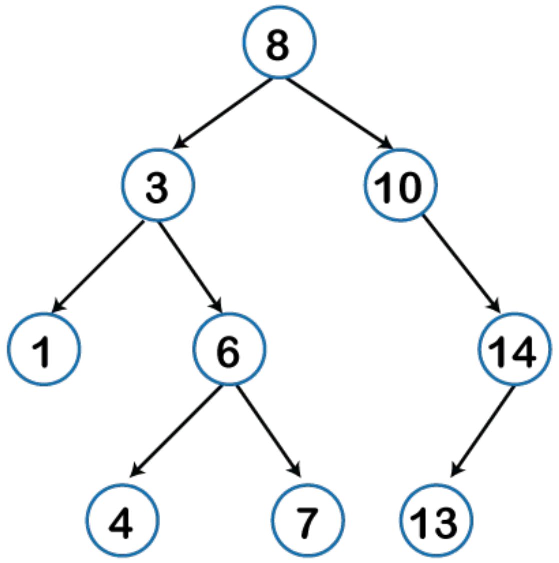
Type: General Tree

- The general tree is one of the types of tree data structure. In the general tree, a node can have either 0 or maximum n number of nodes.
- There is no restriction imposed on the degree of the node (the number of nodes that a node can contain). The topmost node in a general tree is known as a root node. The children of the parent node are known as subtrees.



Type: Binary Tree

- Here, binary name itself suggests two numbers, i.e., 0 and 1. In a binary tree, each node in a tree can have utmost two child nodes. Here, utmost means whether the node has 0 nodes, 1 node or 2 nodes.



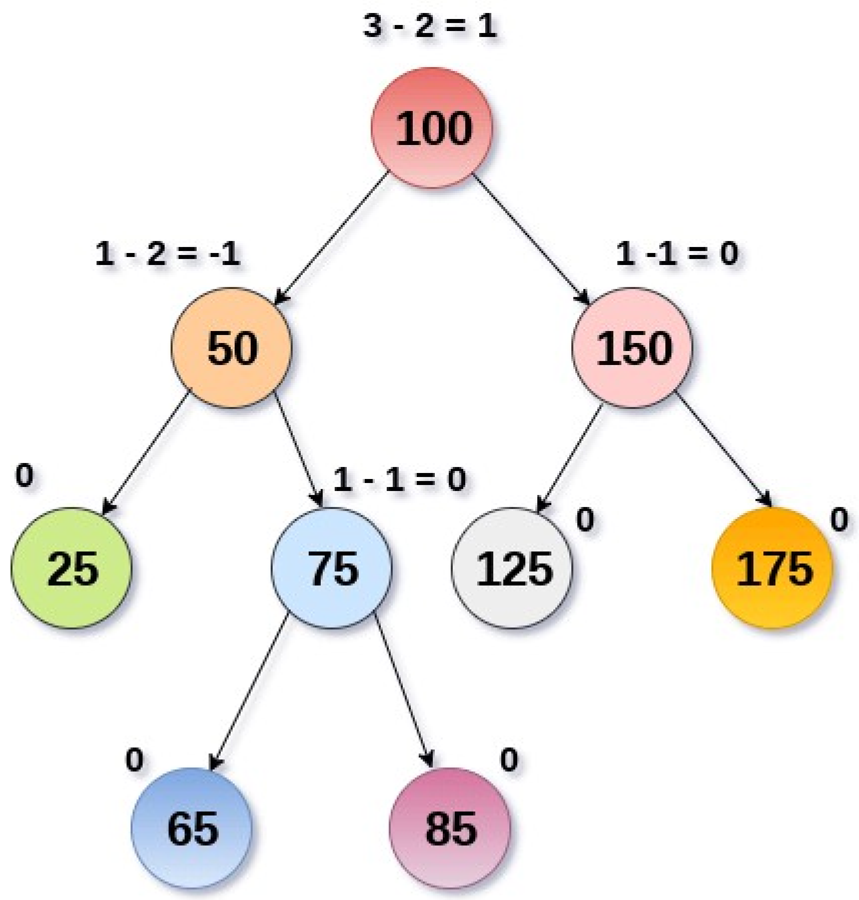
Type: Binary Search Tree

- Binary search tree is a non-linear data structure in which one node is connected to n number of nodes. It is a node-based data structure.
- A node can be represented in a binary search tree with three fields, i.e., data part, left-child, and right-child. A node can be connected to the utmost two child nodes in a binary search tree, so the node contains two pointers (left child and right child pointer).
- Every node in the left subtree must contain a value less than the value of the root node, and the value of each node in the right subtree must be bigger than the value of the root node.

Type: AVL Tree

- It is one of the types of the binary tree, or we can say that it is a variant of the binary search tree. AVL tree satisfies the property of the binary tree as well as of the binary search tree.
- It is a self-balancing binary search tree that was invented by Adelson Velsky Lindas. Here, self-balancing means that balancing the heights of left subtree and right subtree.
- This balancing is measured in terms of the balancing factor.

Type: AVL Tree



AVL Tree

Type: AVL Tree

- We can consider a tree as an AVL tree if the tree obeys the binary search tree as well as a balancing factor.
- The balancing factor can be defined as the difference between the height of the left subtree and the height of the right subtree.
- The balancing factor's value must be either 0, -1, or 1; therefore, each node in the AVL tree should have the value of the balancing factor either as 0, -1, or 1.

Tree Traversal

- The term 'tree traversal' means traversing or visiting each node of a tree.
- There is a single way to traverse the linear data structure such as linked list, queue, and stack. Whereas, there are multiple ways to traverse a tree that are listed as follows -
  - Preorder traversal
  - Inorder traversal
  - Postorder traversal

Preorder Traversal

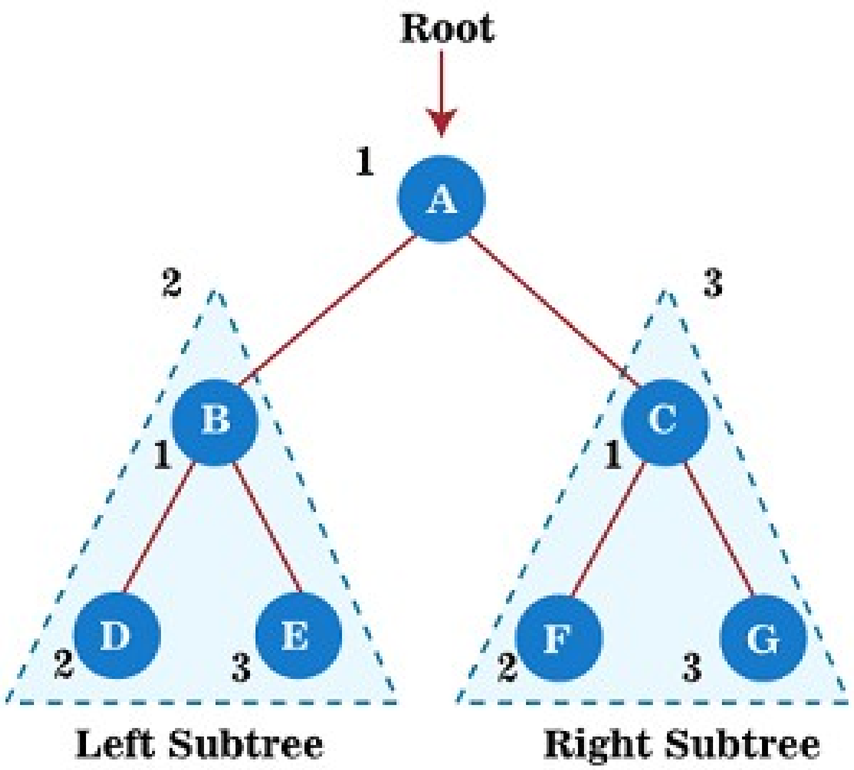
- This technique follows the 'root left right' policy. It means that, first root node is visited after that the left subtree is traversed recursively, and finally, right subtree is recursively traversed.
- As the root node is traversed before (or pre) the left and right subtree, it is called preorder traversal.
- So, in a preorder traversal, each node is visited before both of its subtrees.

Preorder Traversal

- The applications of preorder traversal include -
  - It is used to create a copy of the tree.
  - It can also be used to get the prefix expression of an expression tree.
- Algorithm

```
Until all nodes of the tree are not visited
Step 1 - Visit the root node
Step 2 - Traverse the left subtree recursively.
Step 3 - Traverse the right subtree recursively.
```

Preorder Traversal



A → B → D → E → C → F → G

Postorder Traversal

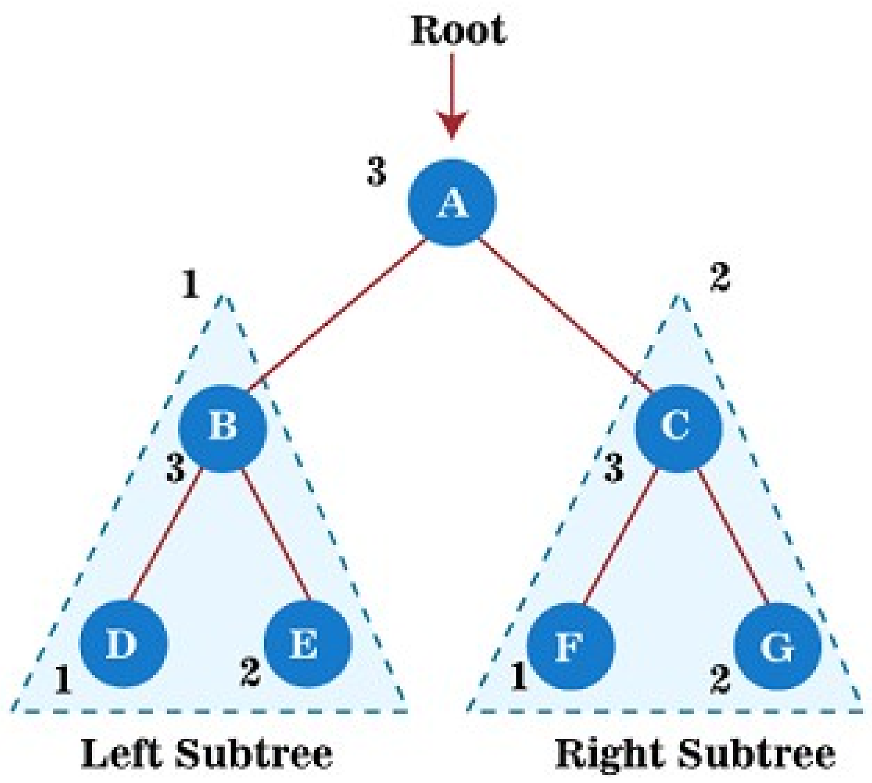
- This technique follows the 'left-right root' policy. It means that the first left subtree of the root node is traversed, after that recursively traverses the right subtree, and finally, the root node is traversed.
- As the root node is traversed after (or post) the left and right subtree, it is called postorder traversal.
- So, in a postorder traversal, each node is visited after both of its subtrees.

Postorder Traversal

- The applications of postorder traversal include -
  - It is used to delete the tree.
  - It can also be used to get the postfix expression of an expression tree.
- Algorithm

```
Until all nodes of the tree are not visited
Step 1 - Traverse the left subtree recursively.
Step 2 - Traverse the right subtree recursively.
Step 3 - Visit the root node.
```

Postorder Traversal



D → E → B → F → G → C → A

Inorder Traversal

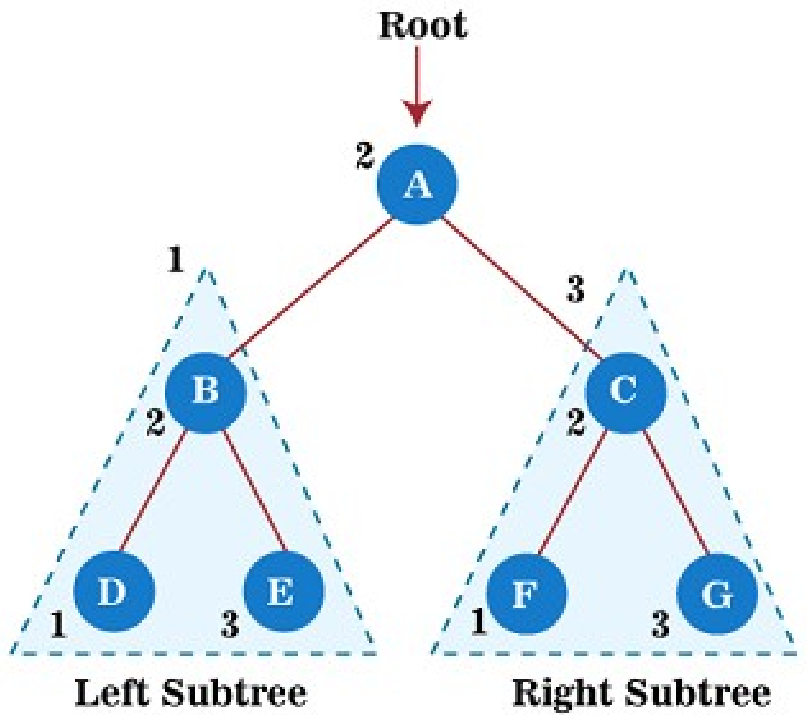
- This technique follows the 'left root right' policy. It means that first left subtree is visited after that root node is traversed, and finally, the right subtree is traversed.
- As the root node is traversed between the left and right subtree, it is named inorder traversal.
- So, in the inorder traversal, each node is visited in between of its subtrees.

Inorder Traversal

- The applications of Inorder traversal includes -
  - It is used to get the BST nodes in increasing order.
  - It can also be used to get the prefix expression of an expression tree.
- Algorithm

```
Until all nodes of the tree are not visited
Step 1 - Traverse the left subtree recursively.
Step 2 - Visit the root node.
Step 3 - Traverse the right subtree recursively.
```

Inorder Traversal



D → B → E → A → F → C → G

Complexities of Tree Traversal

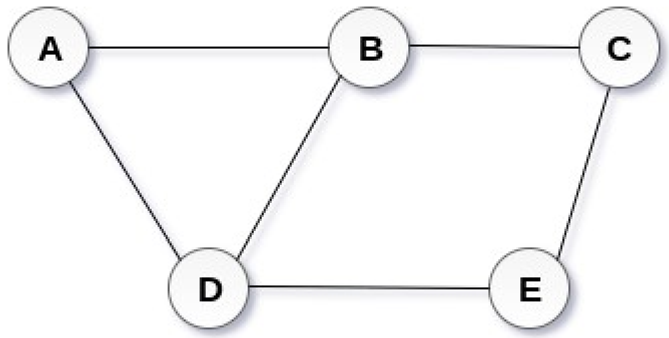
- The time complexity of tree traversal techniques discussed above is O(n), where 'n' is the size of binary tree.
- Whereas the space complexity of tree traversal techniques discussed above is O(1) if we do not consider the stack size for function calls.
- Otherwise, the space complexity of these techniques is O(h), where 'h' is the tree's height.

Graph

- A graph can be defined as group of vertices and edges that are used to connect these vertices.
- A graph can be seen as a cyclic tree, where the vertices (Nodes) maintain any complex relationship among them instead of having parent child relationship.

Graph

- A graph G can be defined as an ordered set G(V, E) where V(G) represents the set of vertices and E(G) represents the set of edges which are used to connect these vertices.
- A Graph G(V, E) with 5 vertices (A, B, C, D, E) and six edges ((A,B), (B,C), (C,E), (E,D), (D,B), (D,A)) is shown in the following figure.



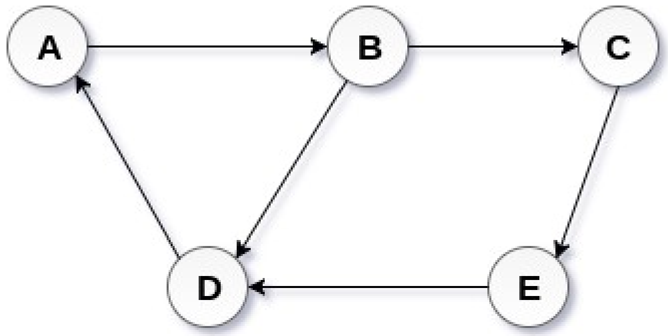
Undirected Graph

Directed and Undirected Graph

- A graph can be directed or undirected. However, in an undirected graph, edges are not associated with the directions with them.
- An undirected graph is shown in the above figure since its edges are not attached with any of the directions.
- If an edge exists between vertex A and B then the vertices can be traversed from B to A as well as A to B.

Directed and Undirected Graph

- In a directed graph, edges form an ordered pair. Edges represent a specific path from some vertex A to another vertex B. Node A is called initial node while node B is called terminal node.
- A directed graph is shown in the following figure



Directed Graph

Graph Terminologies

- Path
  - A path can be defined as the sequence of nodes that are followed in order to reach some terminal node V from the initial node U.
- Closed Path
  - A path will be called as closed path if the initial node is same as terminal node. A path will be closed path if $V_0=V_N$.

Graph Terminologies

- Simple Path
  - If all the nodes of the graph are distinct with an exception $V_0=V_N$, then such path P is called as closed simple path.
- Cycle
  - A cycle can be defined as the path which has no repeated edges or vertices except the first and last vertices.

Graph Terminologies

- Connected Graph
  - A connected graph is the one in which some path exists between every two vertices (u, v) in V. There are no isolated nodes in connected graph.
- Complete Graph
  - A complete graph is the one in which every node is connected with all other nodes. A complete graph contain n(n-1)/2 edges where n is the number of nodes in the graph.

Graph Terminologies

- Weighted Graph
  - In a weighted graph, each edge is assigned with some data such as length or weight. The weight of an edge e can be given as w(e) which must be a positive (+) value indicating the cost of traversing the edge.
- Digraph
  - A digraph is a directed graph in which each edge of the graph is associated with some direction and the traversing can be done only in the specified direction.

Graph Terminologies

- Loop
  - An edge that is associated with the similar end points can be called as Loop.
- Adjacent Nodes
  - If two nodes u and v are connected via an edge e, then the nodes u and v are called as neighbours or adjacent nodes.
- Degree of the Node
  - A degree of a node is the number of edges that are connected with that node. A node with degree 0 is called as isolated node.

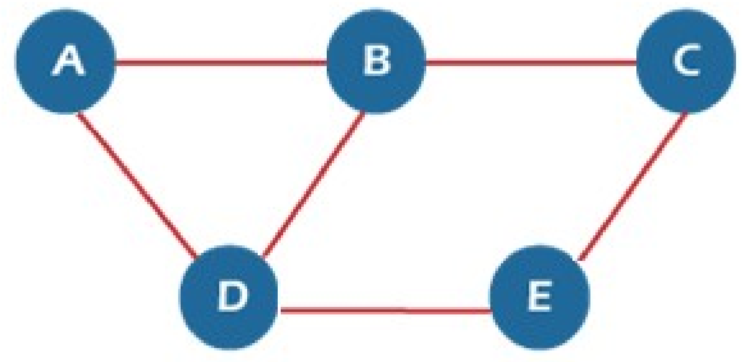
Graph Representation

- By Graph representation, we simply mean the technique to be used to store some graph into the computer's memory.
- A graph is a data structure that consist a sets of vertices (called nodes) and edges. There are two ways to store Graphs into the computer's memory:
  - Sequential representation (or, Adjacency matrix representation)
  - Linked list representation (or, Adjacency list representation)

Graph Representation

- In sequential representation, there is a use of an adjacency matrix to represent the mapping between vertices and edges of the graph.
- We can use an adjacency matrix to represent the undirected graph, directed graph, weighted directed graph, and weighted undirected graph.
- If adj[i][j] = w, it means that there is an edge exists from vertex i to vertex j with weight w.

Graph Representation



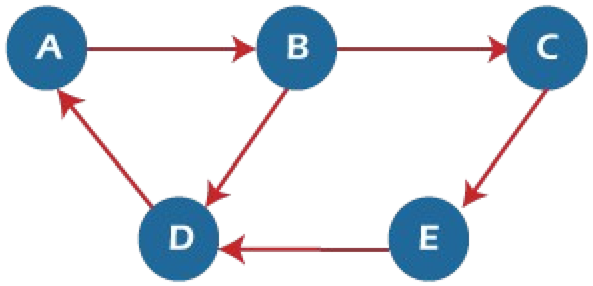
Undirected Graph

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 0 & 1 & 0 \\ B & 1 & 0 & 1 & 1 & 0 \\ C & 0 & 1 & 0 & 0 & 1 \\ D & 1 & 1 & 0 & 0 & 1 \\ E & 0 & 0 & 1 & 1 & 0 \end{array}$$

Adjacency Matrix

Graph Representation

- Adjacency matrix for a directed graph
- In a directed graph, edges represent a specific path from one vertex to another vertex. Suppose a path exists from vertex A to another vertex B; it means that node A is the initial node, while node B is the terminal node.
- Consider the below-directed graph and try to construct the adjacency matrix of it.



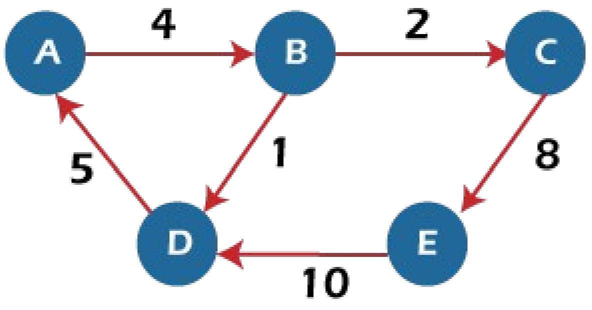
Directed Graph

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 1 & 0 \\ C & 0 & 0 & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 1 & 0 \end{array}$$

Adjacency Matrix

Graph Representation

- Adjacency matrix for a weighted directed graph
- It is similar to an adjacency matrix representation of a directed graph except that instead of using the '1' for the existence of a path, here we have to use the weight associated with the edge. The weights on the graph edges will be represented as the entries of the adjacency matrix. We can understand it with the help of an example.



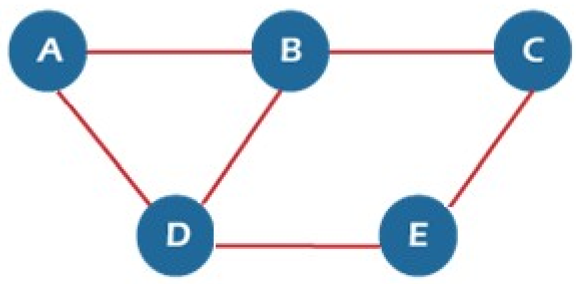
weighted Directed Graph

$$\begin{array}{c|ccccc}
 & A & B & C & D & E \\
\hline
A & 0 & 4 & 0 & 0 & 0 \\
B & 0 & 0 & 2 & 1 & 0 \\
C & 0 & 0 & 0 & 0 & 8 \\
D & 5 & 0 & 0 & 0 & 0 \\
E & 0 & 0 & 0 & 10 & 0
\end{array}$$

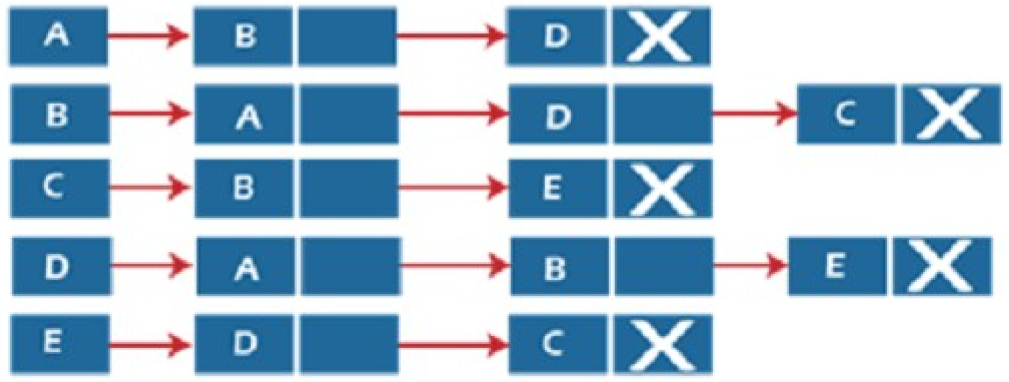
Adjacency Matrix

Graph Representation

- An adjacency list is used in the linked representation to store the Graph in the computer's memory. It is efficient in terms of storage as we only have to store the values for edges.



Undirected Graph



Adjacency List

Thank you

@mitu_skillologies @mITuSkillologies @mitu_group @mitu-skillologies @MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com