Bharathidasan University

Centre for Differently Abled Persons

Tiruchirappalli - 620024.

- Programme Name : Bachelor of Computer Applications
- Course Code : 20UCA6CC10
- Course Title : Data Structures
- Unit : Unit III
- Compiled by :Dr. M. Prabavathy

  Associate Professor

  Ms. Patric Matharasi

  Guest Faculty

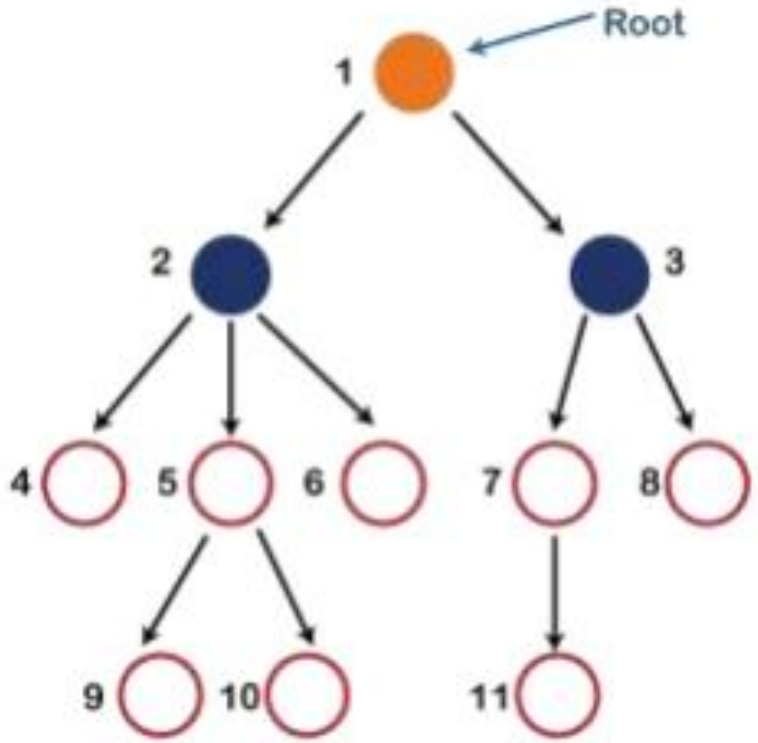
UNIT-III

TREE

Tree

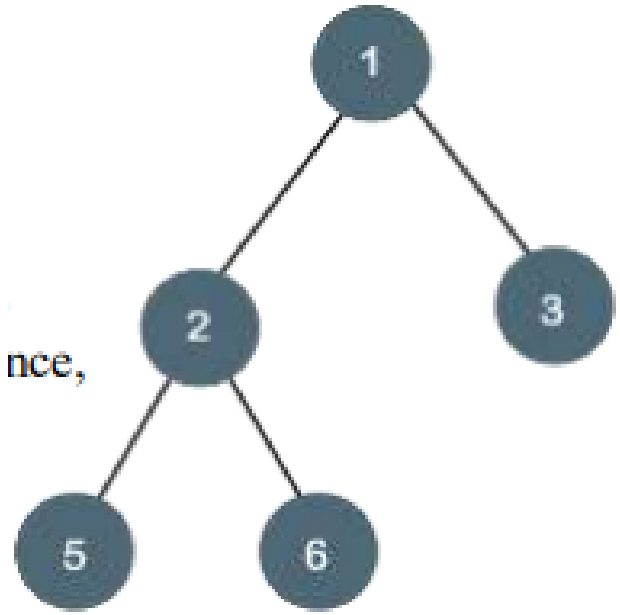
- A tree is also one of the data structures that represent hierarchical data.
- A tree data structure is defined as a collection of objects or entities known as nodes that are linked together to represent or simulate hierarchy.
- A tree data structure is a non-linear data structure because it does not 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.
- Each node contains some data and the link or reference of other nodes that can be called children.

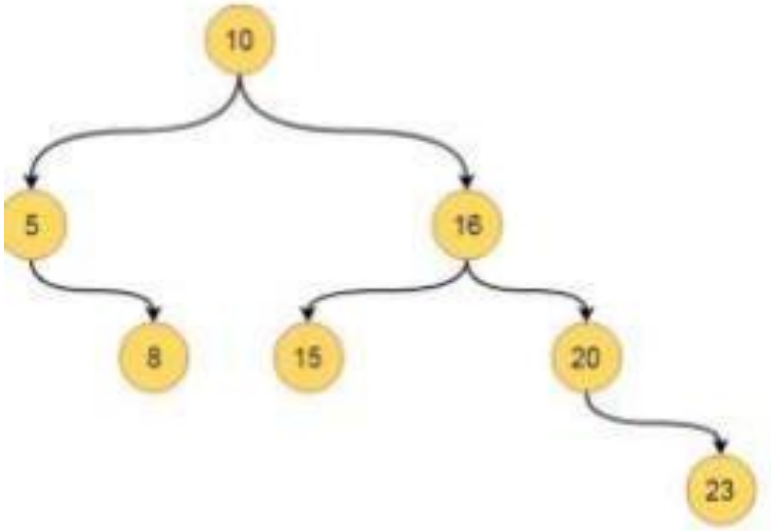


Binary tree Representation

- A binary tree is a non-linear data structure of the tree type.
- Binary name itself suggests that 'two'.
- It has a maximum of two children for every parent node.
- The node at the top of the entire binary tree is called the root node.
- In any binary tree, every node has a left reference, right reference, and data element.



- There are two different methods for representing.
- These are using array and using linked list.
- Suppose we have one tree like this
- The array representation stores the tree data by scanning elements using level order fashion.
- So it stores nodes level by level.
- If some element is missing, it left blank spaces for it.
- The representation of the above tree is like below:



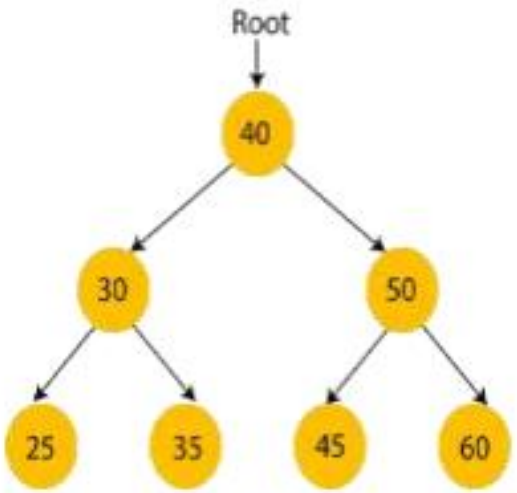
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	5	16	-	8	15	20	-	-	-	-	-	-	-	23

- The index 1 is holding the root, it has two children 5 and 16, they are placed at location 2 and 3.
- Some children are missing, so their place is left as blank.
- In this representation we can easily get the position of two children of one node by using this formula:
- child1 = 2 * parent
- child2 = (2 * parent) + 1
- This approach is good, and easily we can find the index of parent and child, but it is not memory efficient.
- It will occupy many spaces that have no use.
- This representation is good for complete binary tree or full binary tree.

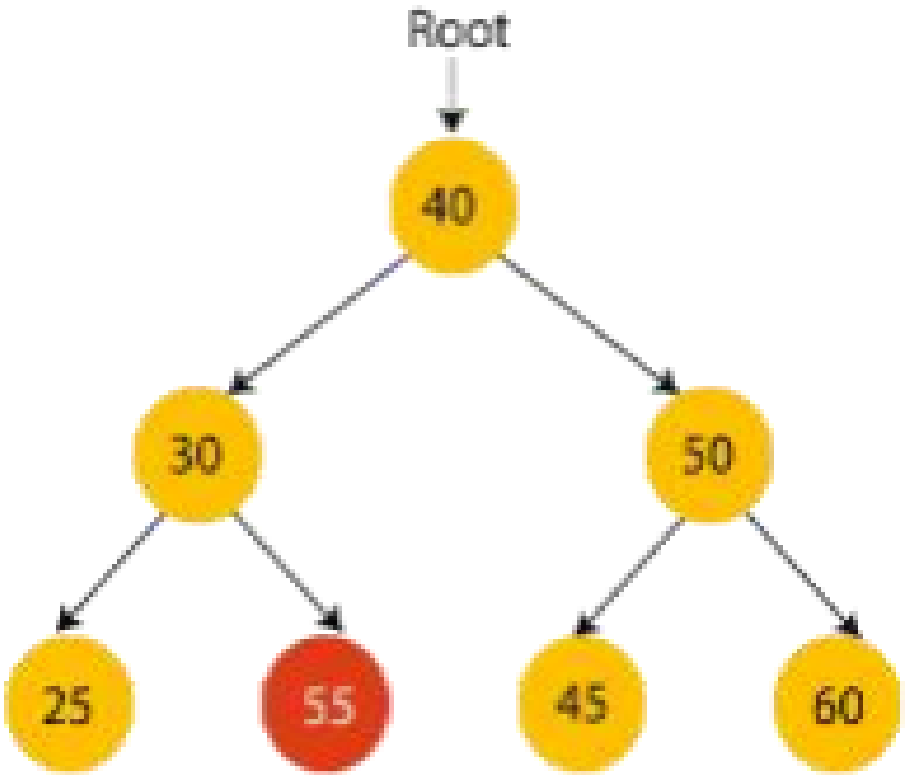
Binary Search tree

- A binary search tree follows some order to arrange the elements.
- In a Binary search tree, the value of left node must be smaller than the parent node, and the value of right node must be greater than the parent node.
- This rule is applied recursively to the left and right sub trees of the root.
- Let's understand the concept of Binary search tree with an example.

- In the above figure, we can observe that the root node is 40, and all the nodes of the left
- subtree are smaller than the root node, and all the nodes of the right subtree are greater than the root node.
- Similarly, we can see the left child of root node is greater than its left child and smaller than its right child.
- So, it also satisfies the property of binary search tree.



- Therefore, we can say that the tree in the above image is a binary search tree.
- Suppose if we change the value of node 35 to 55 in the above tree, check whether the tree will be binary search tree or not.



Advantages of Binary search tree

- Searching an element in the Binary search tree is easy as we always have a hint that which
- Subtree has the desired element.
- As compared to array and linked lists, insertion and deletion operations are faster in BST.

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

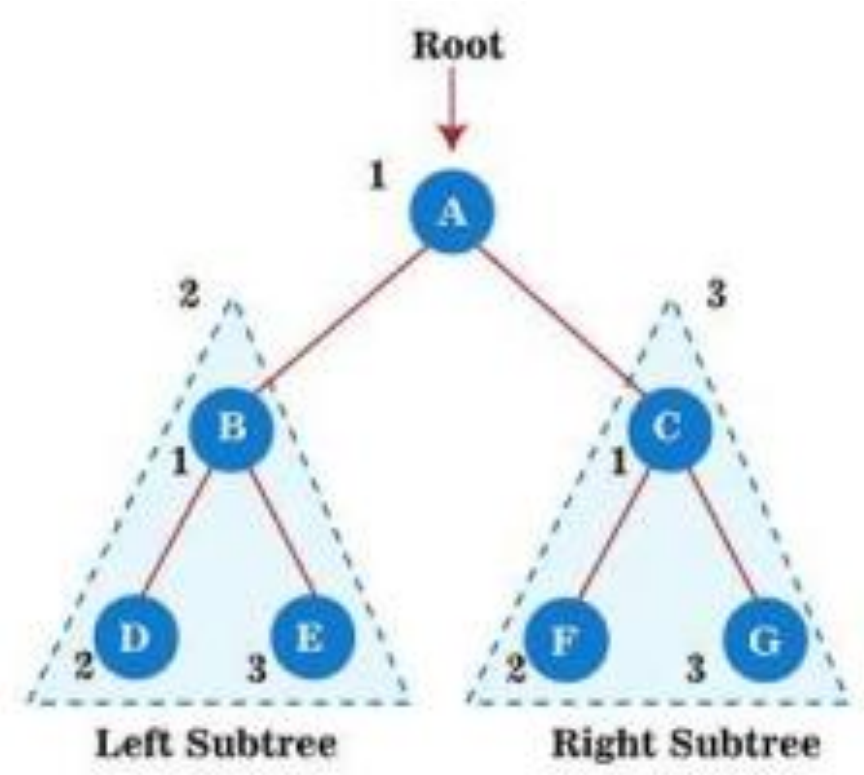
- This technique follows the 'root left right' policy.
- It means that, first root node is visited after that the left subtree is traversed recursively, and finally, right subtree is recursively traversed.
- As the root node is traversed before (or pre) the left and right subtree, it is called preorder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

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.

Example:

- The output of the preorder traversal of the above tree is:
- 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.

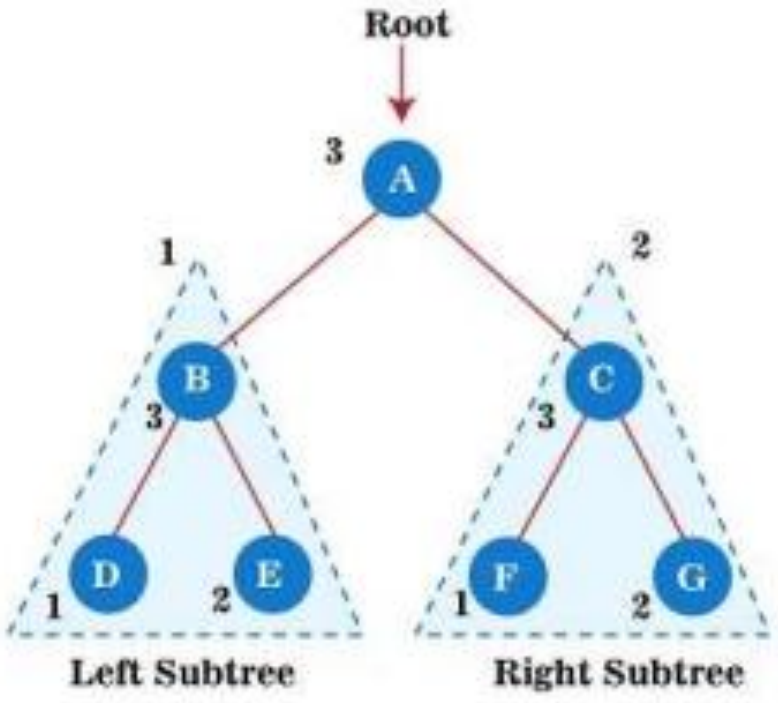
Algorithm

- 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.

Example:

The output of the postorder traversal of the tree is -

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.

Algorithm

Until all nodes of the tree are not visited

Step 1 - Traverse the left subtree recursively.

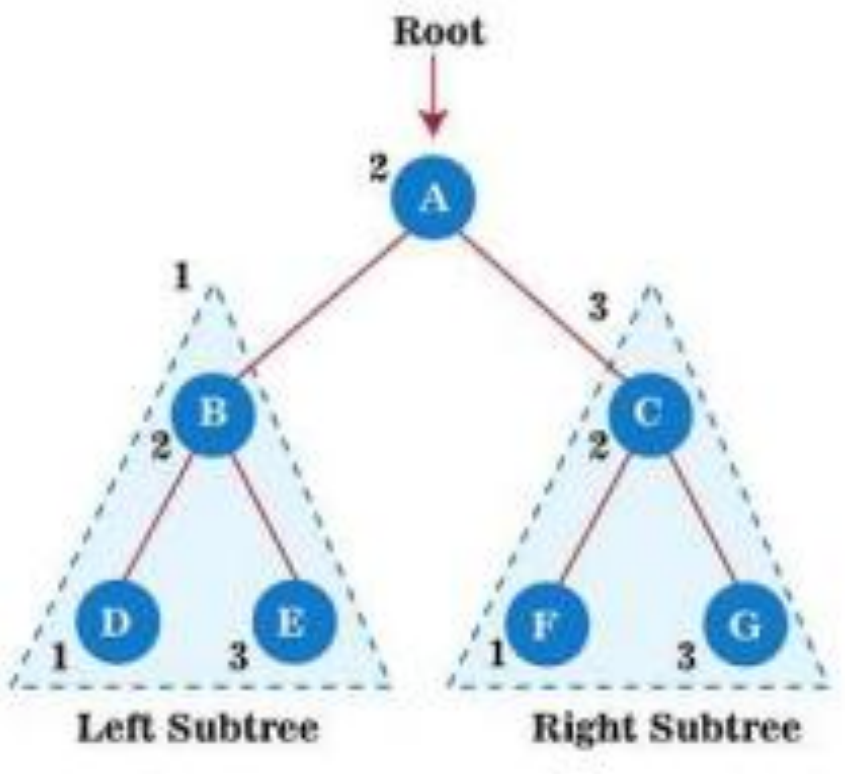
Step 2 - Visit the root node.

Step 3 - Traverse the right subtree recursively.

Example:

The output of the inorder traversal of the above tree is -

D → B → E → A → F → C → G



Threaded Binary Tree

- A threaded binary tree is a type of binary tree data structure where the empty left and

right child pointers in a binary tree are replaced with threads that link nodes directly to

their in-order predecessor or successor, thereby providing a way to traverse the tree without using recursion or a stack.

- Threaded binary trees can be useful when space is a concern, as they can eliminate the need for a stack during traversal.

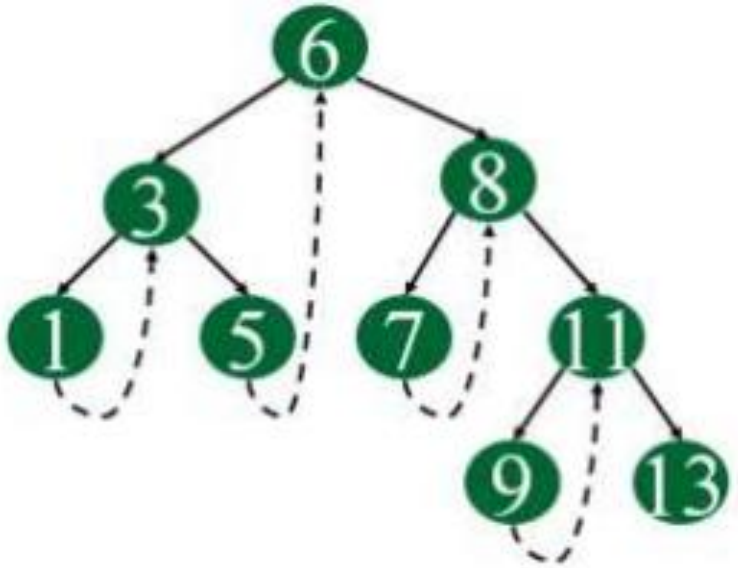
- However, they can be more complex to implement than standard binary trees.
- There are two types of threaded binary trees.

  - **Single Threaded:**

- Where a NULL right pointers is made to point to the inorder successor (if successor exists)

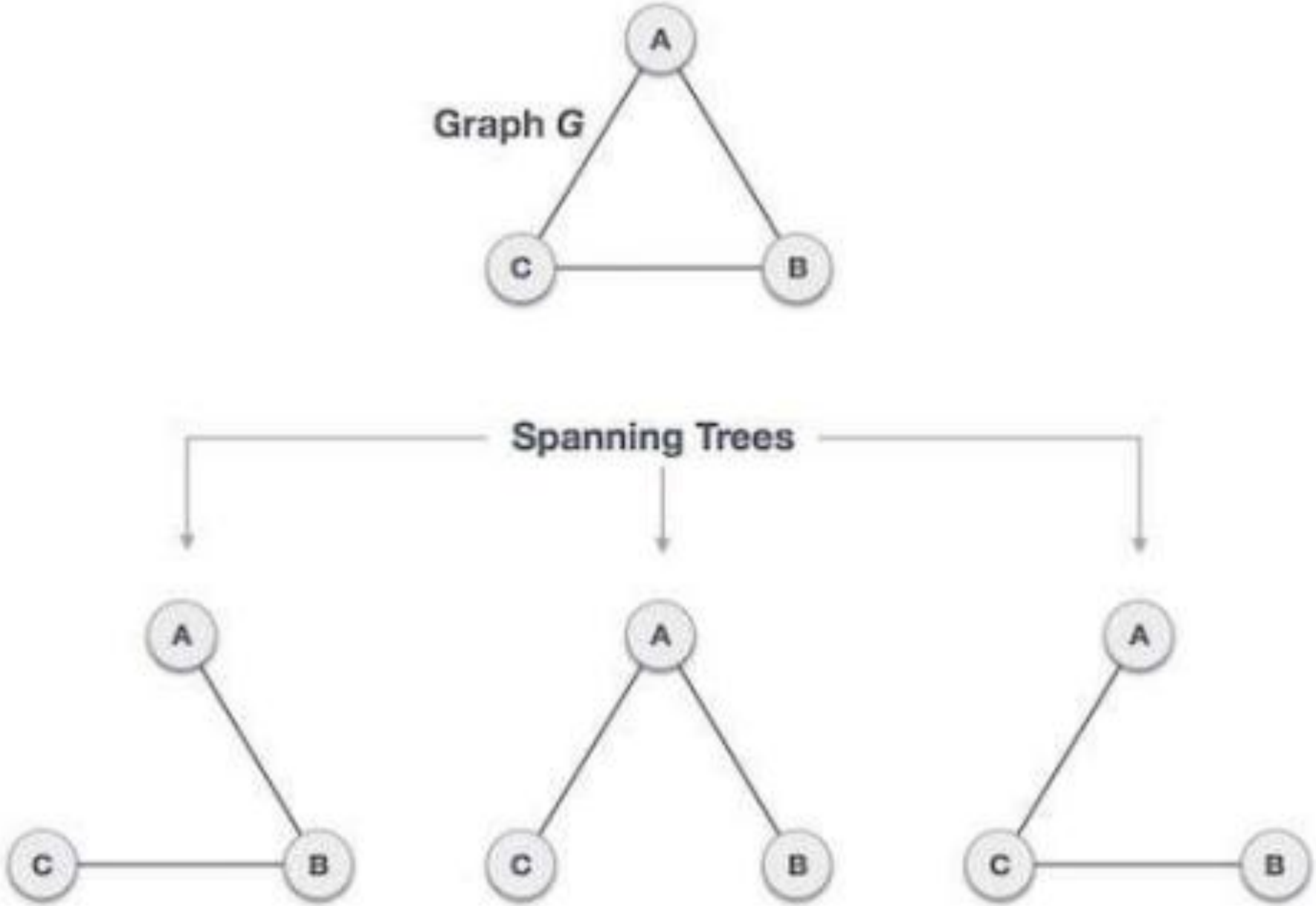
Double Threaded:

- 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.
- The above diagram shows an example of Single Threaded Binary Tree.
- The dotted lines represent threads.



Spanning Trees

- A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges.
- Hence, a spanning tree does not have cycles and it cannot be disconnected.
- By this definition, we can draw a conclusion that every connected and undirected Graph
- G has at least one spanning tree.
- A disconnected graph does not have any spanning tree, as it cannot be spanned to all its vertices.


Graph G
A
C
B
Spanning Trees
A
C
B
A
C
B
A
C
B

Properties of Spanning Tree

- We now understand that one graph can have more than one spanning tree.
- Following are a few properties of the spanning tree connected to graph G:
- A connected graph G can have more than one spanning tree.
- All possible spanning trees of graph G, have the same number of edges and vertices.
- The spanning tree does not have any cycle (loops).
- Removing one edge from the spanning tree will make the graph disconnected,
- i.e. the spanning tree is minimally connected.
- Adding one edge to the spanning tree will create a circuit or loop, i.e. the
- spanning tree is maximally acyclic.

Application of Spanning Tree

- Spanning tree is basically used to find a minimum path to connect all nodes in a graph.
- Common application of spanning trees are:
- Civil Network Planning
- Computer Network Routing Protocol
- Cluster Analysis

THANK YOU