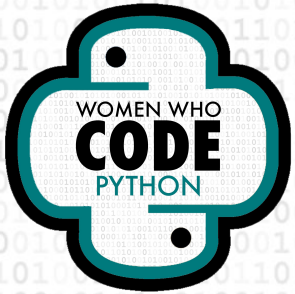
Welcome everyone!

- You can find these slides on GitHub here: https://github.com/WomenWhoCode/WWCodePython
- Please make sure your chat is set to “All panelists and attendees”.
- Some housekeeping rules:
  - Everyone will be muted throughout the webinar, but there will be opportunities for participation!
  - Please share your thoughts on the chat and/or ask questions in the Q&A.
  - The entire team is here today. Please reach out to us with any technical questions!



WELCOME

WOMEN WHO CODE

Women Who Code Python

Intro to Data Structures with Python: Ace the Technical Interview

Session #6: Trees, Binary Trees, Binary Search Trees

WOMEN WHO CODE®

MEET YOUR TEAM

Rishika Singh
Track Lead

Jasmeen Rajpal
Evangelist

WOMEN WHO CODE

OUR MISSION

Inspiring women to excel in technology careers.

WOMEN WHO CODE

OUR VISION

A world where women are representative as technical executives, founders, VCs, board members and software engineers.

WOMEN WHO CODE

OUR TARGET

Engineers with two or more years of experience looking for support and resources to strengthen their influence and levelup in their careers.

WOMEN WHO CODE

CODE OF CONDUCT

WWCode is an inclusive community, dedicated to providing an empowering experience for everyone who participates in or supports our community, regardless of gender, gender identity and expression, sexual orientation, ability, physical appearance, body size, race, ethnicity, age, religion, socioeconomic status, caste, creed, political affiliation, or preferred programming language(s).

Our events are intended to inspire women to excel in technology careers, and anyone who is there for this purpose is welcome. We do not tolerate harassment of members in any form. Our **Code of Conduct** applies to all WWCode events and online communities.

Read the full version and access our incident report form at **womenwhocode.com/codeofconduct**

WOMEN WHO CODE

230,000
Members

70 networks in 20 countries
Members in 97+ countries
10K+ events
$1025 daily Conference tickets
$2M Scholarships
Access to jobs + resources
Infinite connections

WOMEN WHO CODE

OUR MOVEMENT

As the world changes, we can be a connecting force that creates a sense of belonging while the world is being asked to isolate.

WOMEN WHO CODE

Upcoming Events

SAT 29 MAY

🔥 **Introduction to Deep Learning for Edge Devices: Session 8: Mini Project** 🔥 *Featured*

Online | Data Science | 8:00 AM – 9:00 AM PDT (UTC–0700)

Register

THU 10 JUN

CONNECT REIMAGINE Virtual Conference *Featured*

Online | 8:00 AM – 2:00 PM PDT (UTC–0700)

Register

THU 10 JUN

CONNECT REIMAGINE Virtual Conference *Featured*

Online | 4:00 PM – 8:00 PM PDT (UTC–0700)

Register

THU 10 JUN

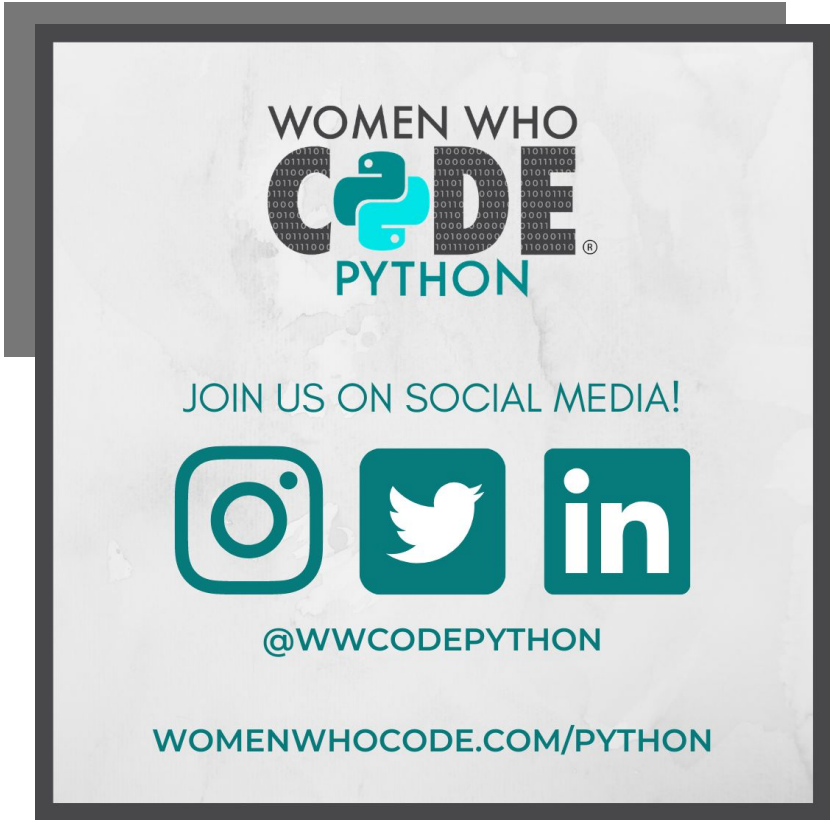
✨ **Intro to Data Structures with Python: Ace the Technical Interview (Session #7: Heaps)** ✨ *Featured*

Online | Python | 5:00 PM – 6:30 PM PDT (UTC–0700)

Register

WOMEN WHO CODE PYTHON

Stay Connected



Shermaine Ang

Incoming Freshman at Imperial EIE | WWCode Python Volunteer

Shivangi

SDE 2 @Grofers India | WWCode Python Volunteer

Today's Agenda

1. What are Trees?
2. Terminology of trees
    a. Examples of trees
3. What is a Binary Tree?
4. What is a Binary Search Tree?
5. BST Operations
    a. Counting Nodes
    b. Searching
    c. Insertion
    d. Traversal
6. Live Coding
7. Resources & Preparation

What are Trees?

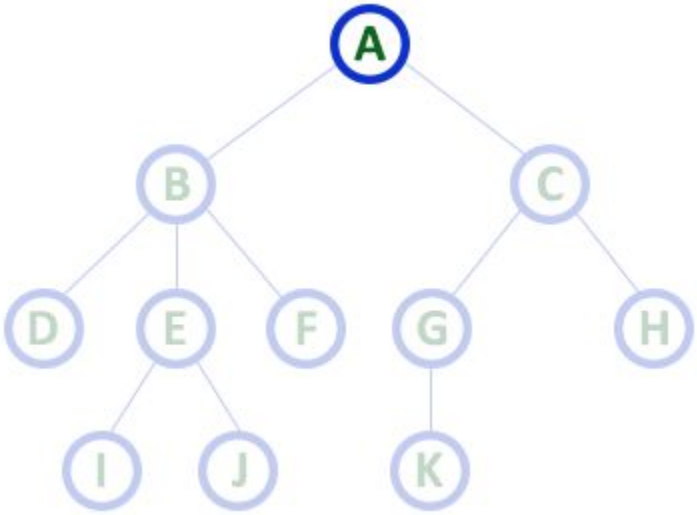
Trees

- A tree is a non-linear data structure which organizes data in hierarchical structure.
- Trees have two main characteristics:
  - Each item can have multiple children.
  - All items, except a privileged item called the **root**, have exactly one parent.

Terminology of Trees

Terminology of Trees

- **Node:** An item stored in a tree.
- **Root:** The topmost node in a tree.

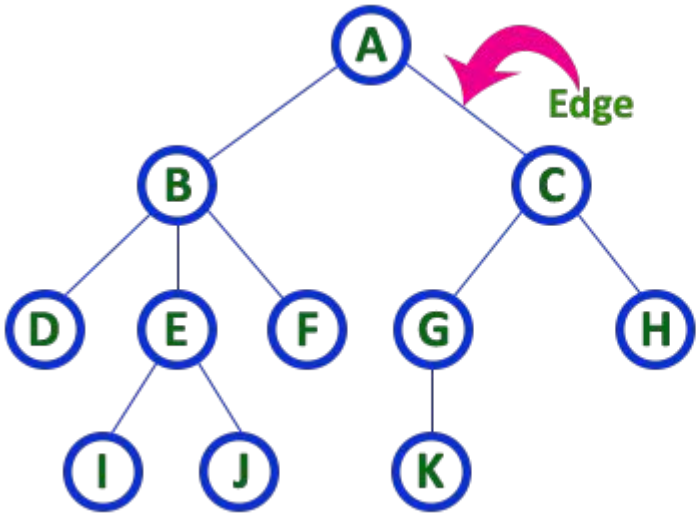


Here ‘A’ is the ‘root’ node

- In any tree the first node is called as ROOT node

Terminology of Trees (contd.)

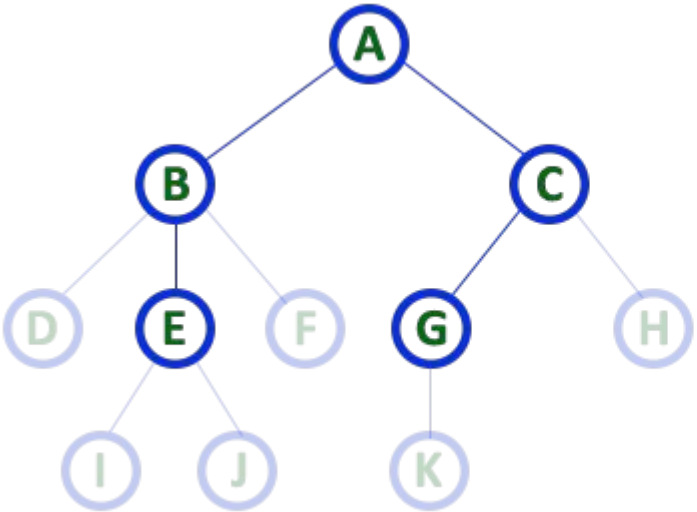
- **Edge:** The connecting link between any two nodes.



- In any tree, 'Edge' is a connecting link between two nodes.

Terminology of Trees (contd.)

- **Parent:** The node which is a predecessor of any node.

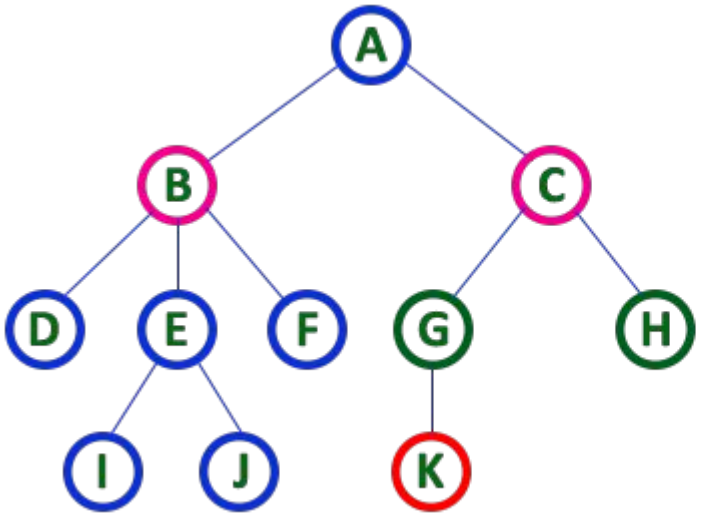


Here A, B, C, E & G are Parent nodes

- In any tree the node which has child / children is called 'Parent'
- A node which is predecessor of any other node is called 'Parent'

Terminology of Trees (contd.)

- **Child:** The node which has a link from its parent node.



Here B & C are Children of A

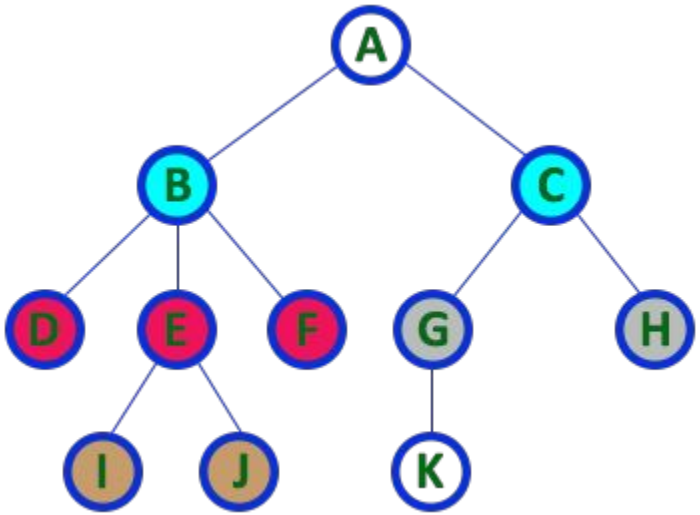
Here G & H are Children of C

Here K is Child of G

- descendant of any node is called as CHILD Node

Terminology of Trees (contd.)

- **Siblings:** Nodes which belong to same parent.

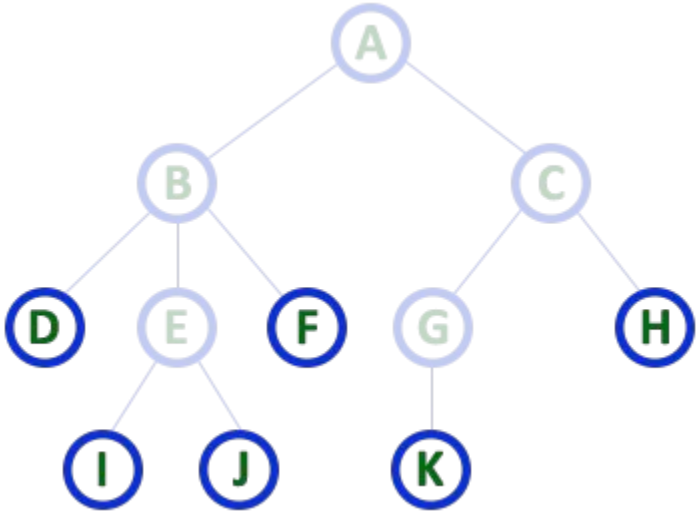


Here B & C are Siblings
Here D E & F are Siblings
Here G & H are Siblings
Here I & J are Siblings

- In any tree the nodes which has same Parent are called 'Siblings'
- The children of a Parent are called 'Siblings'

Terminology of Trees (contd.)

- **Leaf:** The node which does not have a child.

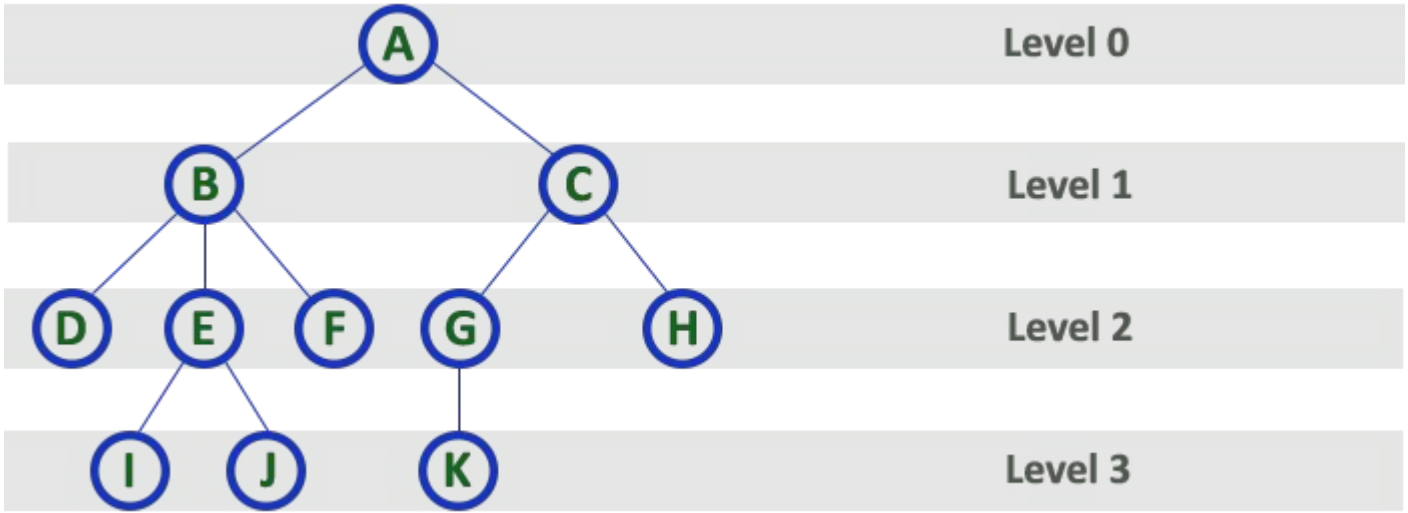


Here D, I, J, F, K & H are Leaf nodes

- In any tree the node which does not have children is called 'Leaf'
- A node without successors is called a 'leaf' node

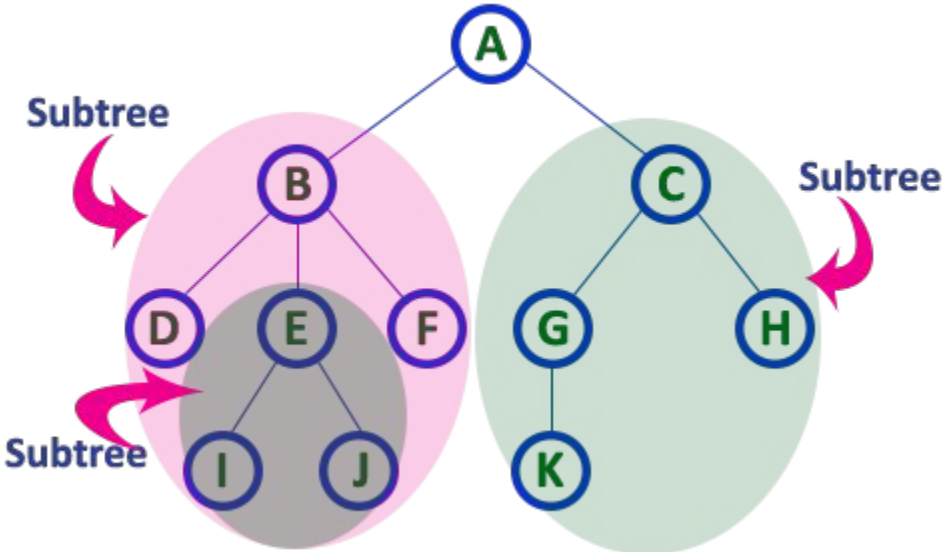
Terminology of Trees (contd.)

- **Level:** Each step from top to bottom, starting with Level 0.



Terminology of Trees (contd.)

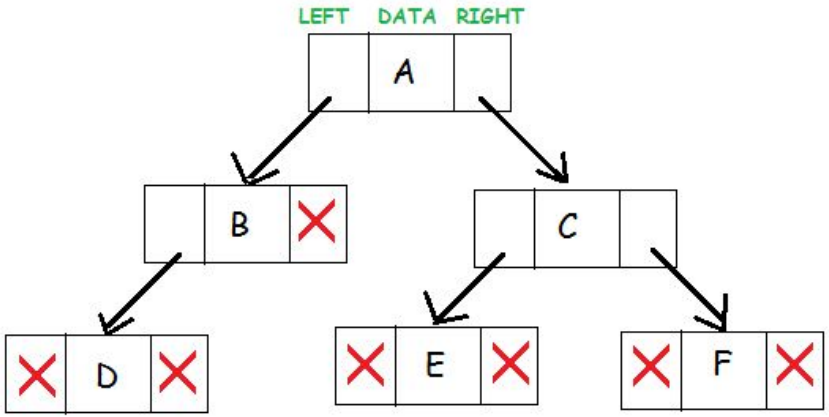
- **Subtree:** Every child node will form a subtree on its parent node.



What is a Binary Tree?

Binary Trees

- In a binary tree, each node has **at most** two children:
  - Left child
  - Right child
- Components of a Binary Tree:
  - Data element
  - Left pointer
  - Right pointer

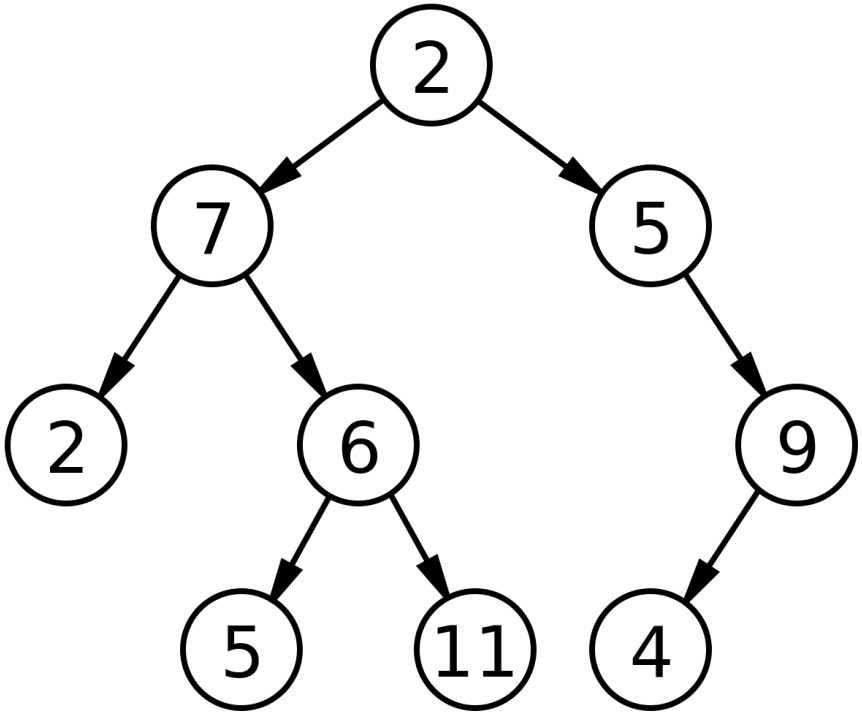


Examples of Binary Trees

- Full Binary Tree
- Complete Binary Tree
- Perfect Binary Tree

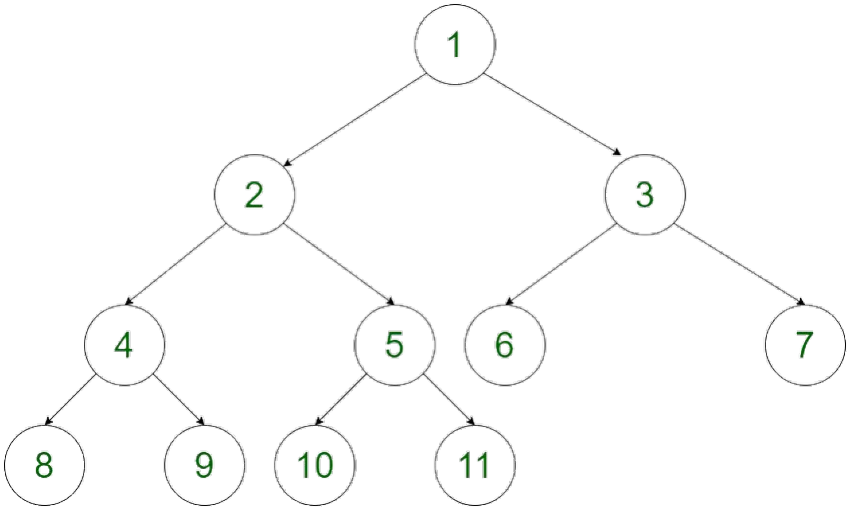
Full Binary Tree

- A binary tree that has either zero children or two children



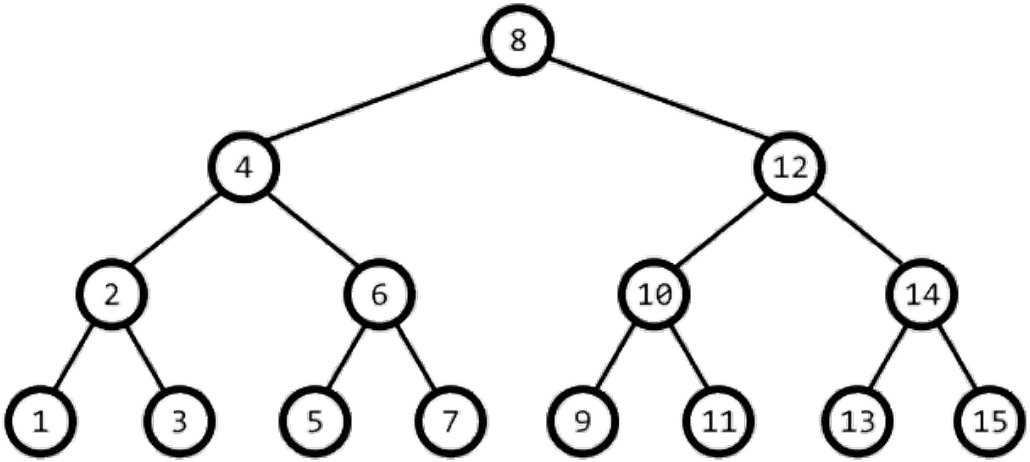
Complete Binary Tree

- All the tree levels are filled entirely with nodes, except the lowest level of the tree
- In the lowest level of the binary tree, every node should possibly reside on the left side

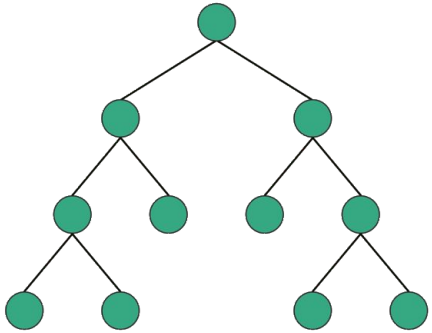


Perfect Binary Tree

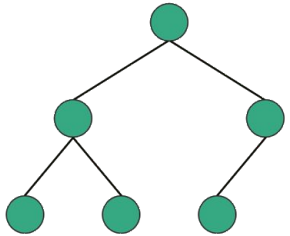
- All the internal nodes have strictly two children, and every external or leaf node is at the same level or same depth within a tree



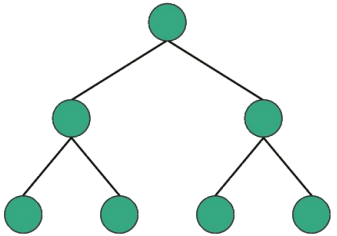
Examples of Binary Trees



Full



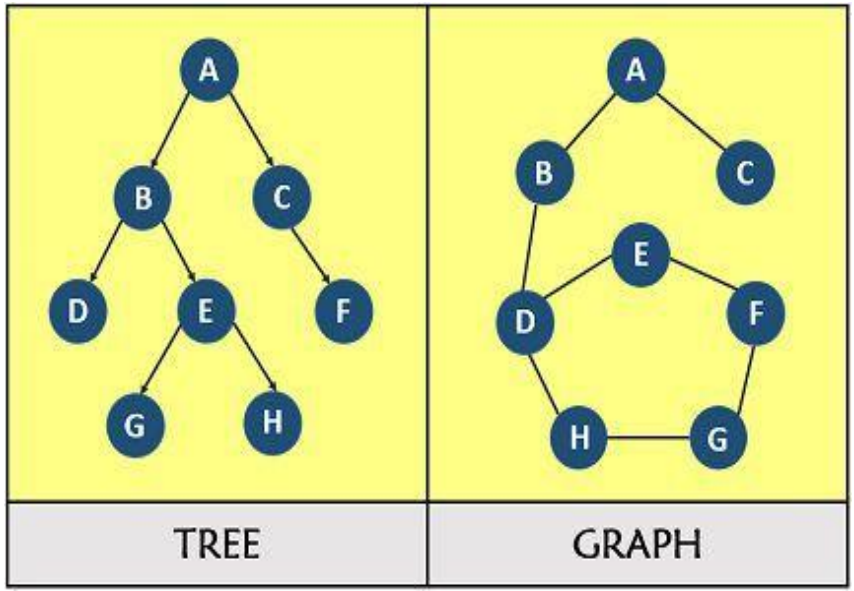
Complete



Perfect

Binary Trees VS Graphs

Graphs	Binary Trees
Does not have a root node	Has exactly one root node
Can have loops	No loops are permitted
Each node can have many number of edges	Every node can have at most two child nodes

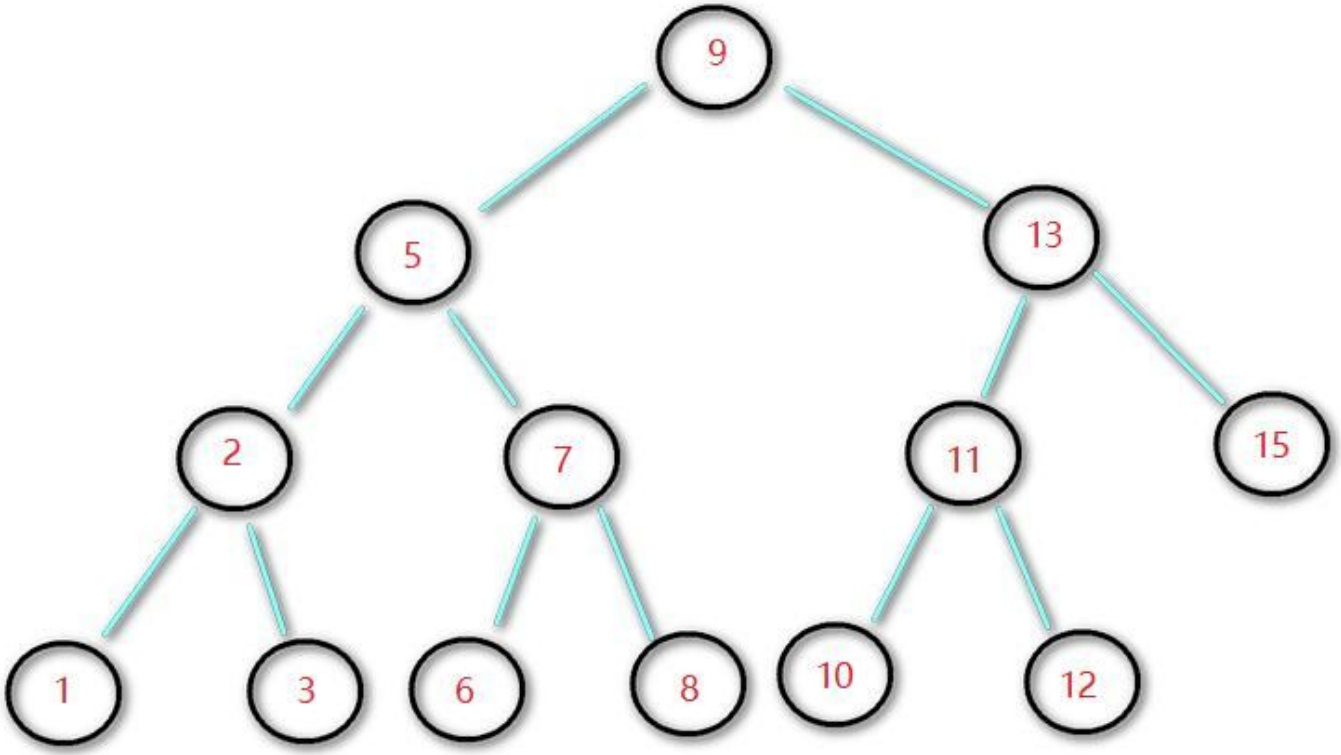


What is a Binary Search Tree (BST)?

Binary Search Trees (BSTs)

- BSTs are sorted Binary Trees
  - The value of all nodes on the left subtree is less than or equal to the value of its root node.
  - The value of all nodes on the right subtree is greater than or equal to the value of its root node.
  - The left and right subtrees are also binary sort trees respectively.

Binary Search Trees (BSTs) (contd.)



Pros and Cons of Binary Search Trees

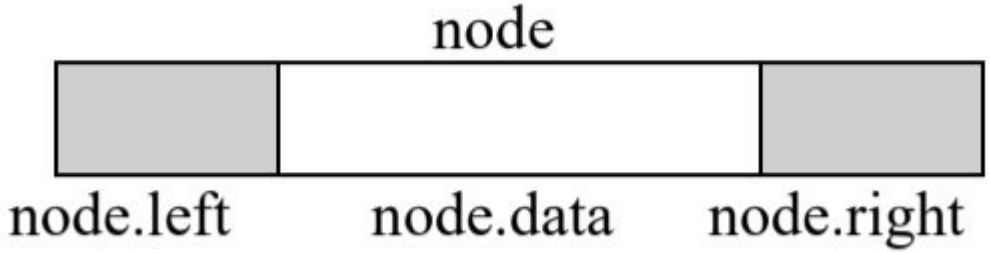
+ Quick search
+ Quick insertion
+ Quick deletion

- Implementation might be difficult

BST Operations

Node Creation

- To create a BST, we first have to create the root node.




```
Class TreeNode:

      def __init__(self, data):
            self.left = None
            self.data = data
            self.right = None
```

Counting Number of Nodes

- Determine the number of nodes in the tree by counting the number of nodes in the left subtree and the right subtree.
- This can be done using recursion.

```
# Gets the size of the tree, i.e. count the nodes in the tree

def CountNodes(self, tree):
      if tree == None:
            return 0
      else:
            return self.CountNodes(tree.left) +
                  self.CountNodes(tree.right)+ 1
```

Searching

- As the BST is sorted, if the item is less than the current data, it will continue searching in the **left** subtree.

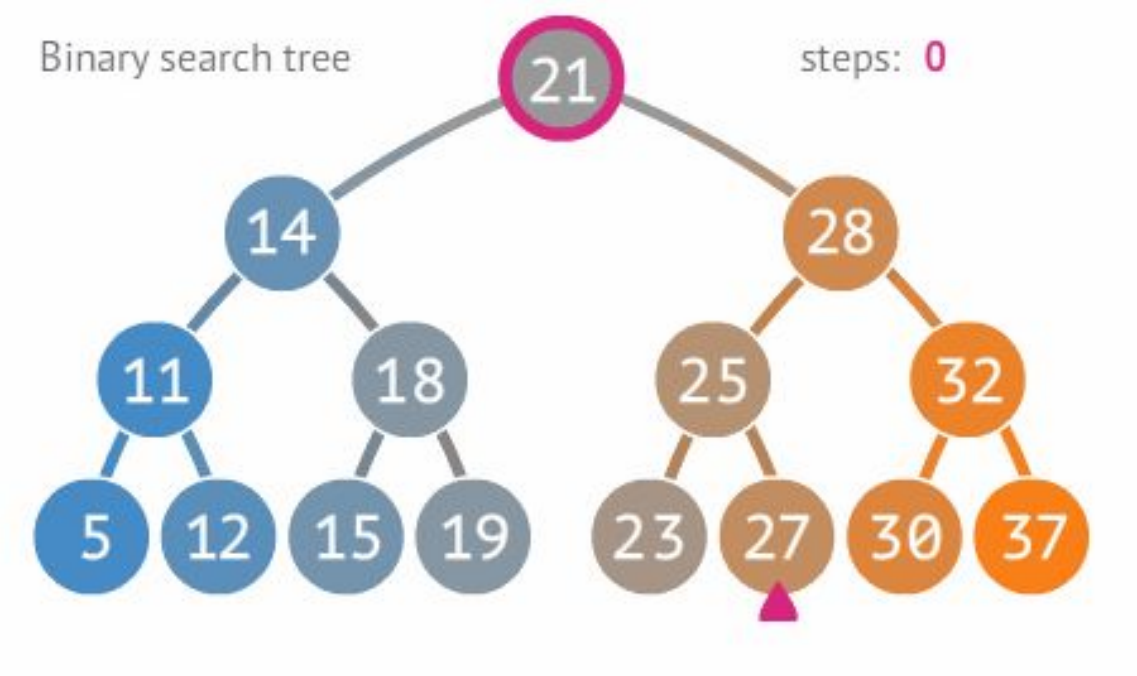
```
# Search the tree for item

def Search(self, tree, item):
    if tree == None:
        return False
    elif item < tree.data :
        self.Search(tree.left, item)
    elif item > tree.data :
        self.Search(tree.right, item)
    else:       # item = tree.data
        return True
```

Time complexity is O(h), where h is height of BST

Searching

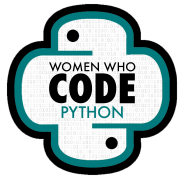
Searching for 27



Insertion

- To ensure that the BST is still sorted, item will be inserted at:
  - The root node, if the tree is already empty.
  - A node in the current node's left subtree, if new item is less than item in current node.
  - A node in the current node's right subtree, if new item is greater than or equal to item in current node.

Time complexity is O(h), where h is height of BST



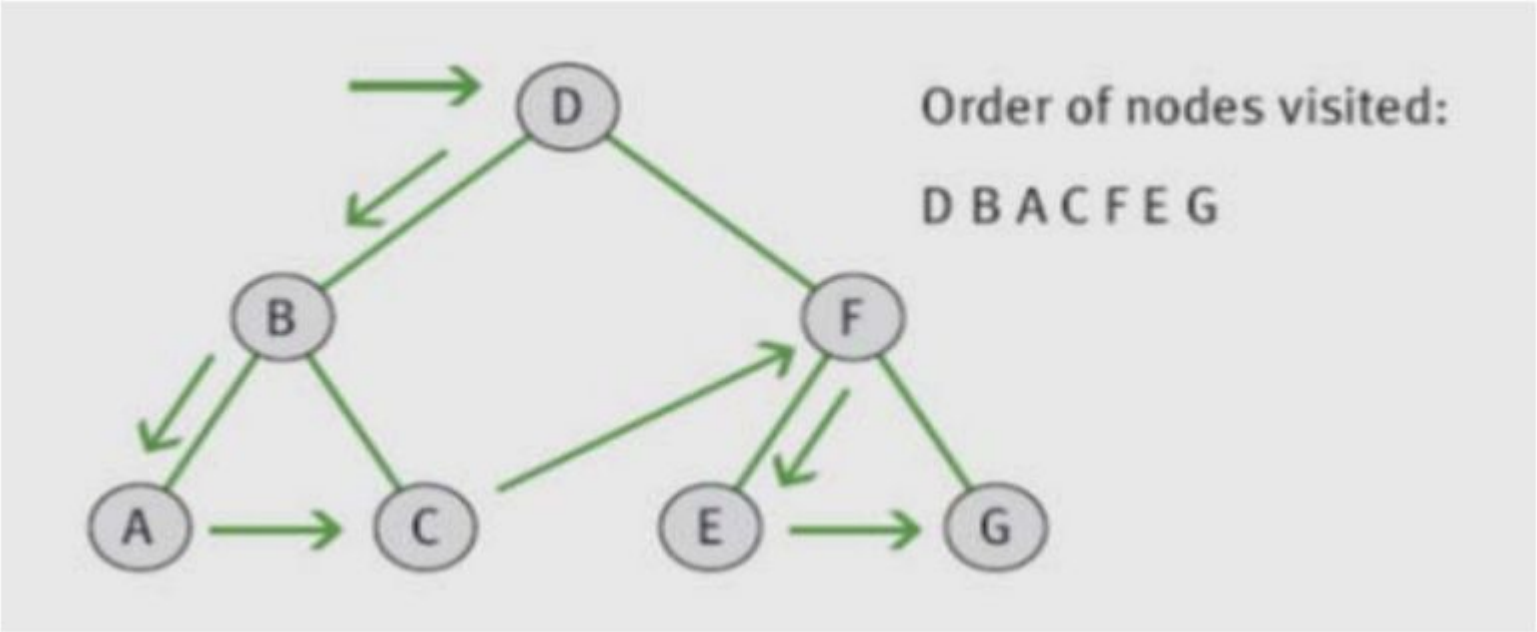
Insertion

Traversal

- The process of visiting each node in a tree data structure, exactly once.
- Types of traversal:
  - Preorder Traversal
  - Inorder Traversal
  - Postorder Traversal

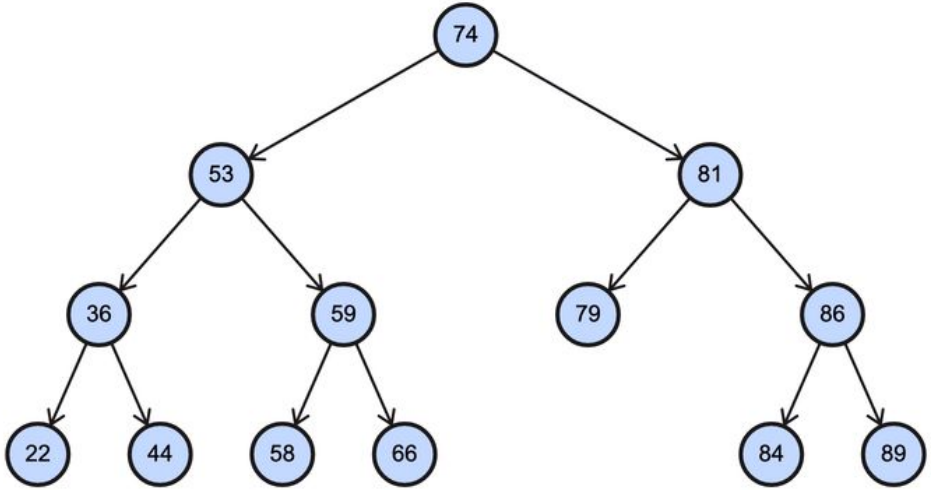
Preorder Traversal

- Visits root node, and then traverses left subtree and right subtree in similar way



Preorder Traversal

- Root
- Left
- Right

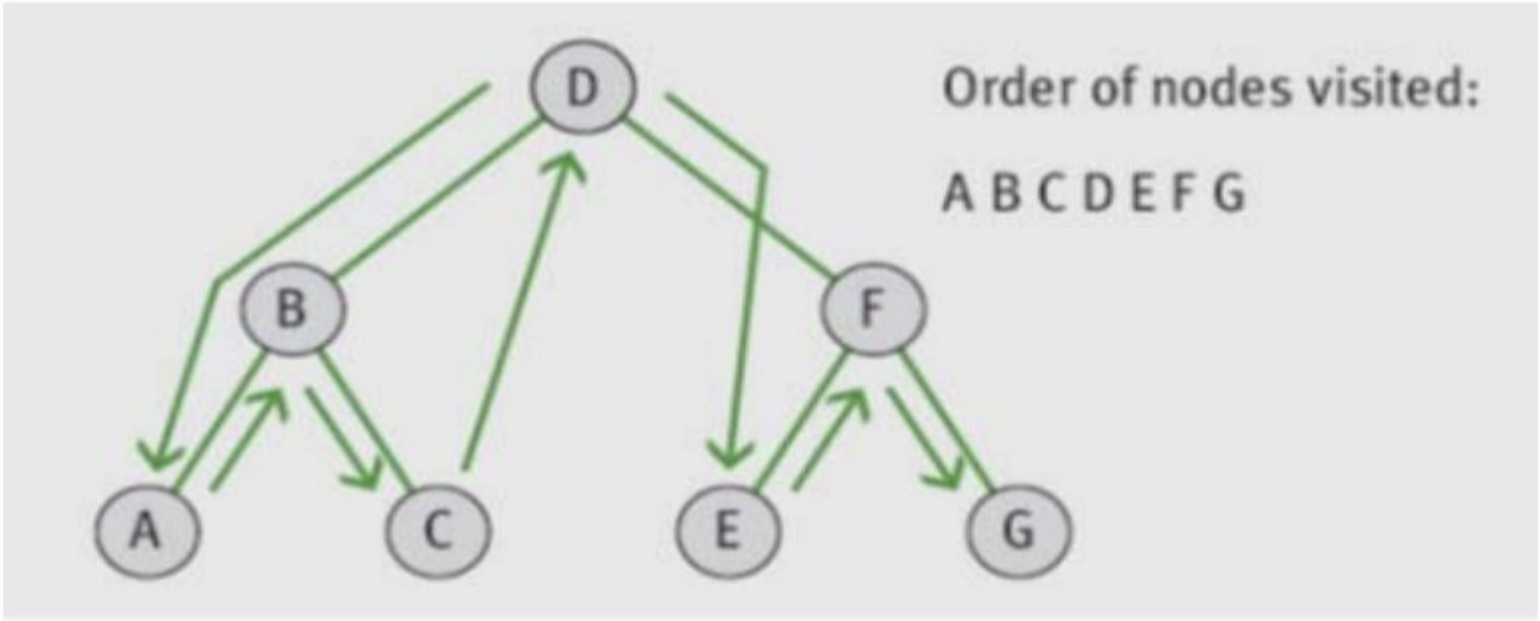


1	2	3	4	5	6	7	8	9	10	11	12	13
74	53	36	22	44	59	58	66	81	79	86	84	89

WOMEN WHO CODE PYTHON

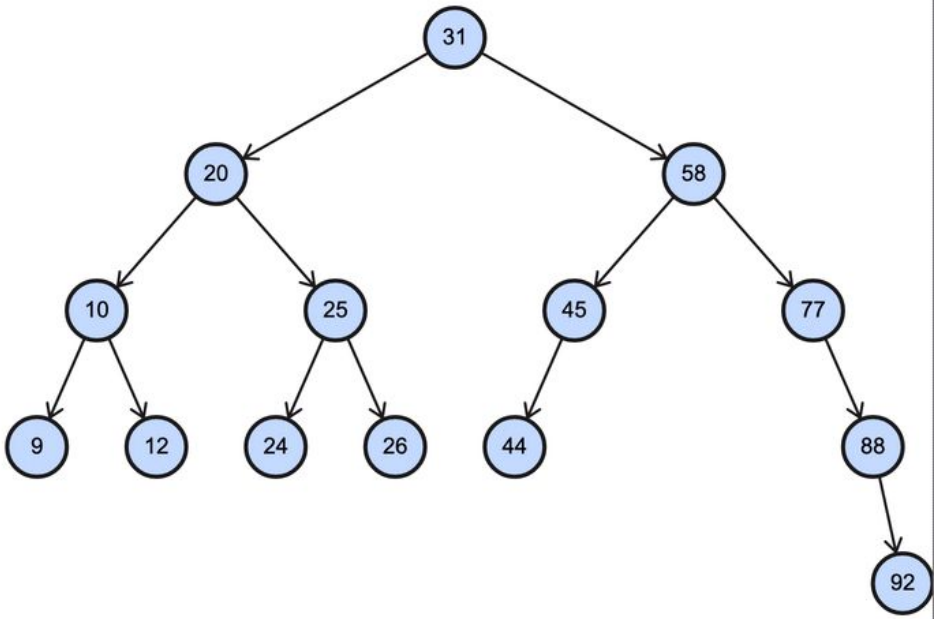
Inorder Traversal

- Traverses left subtree, visits root node, and traverses right subtree (sorted order)



Inorder Traversal

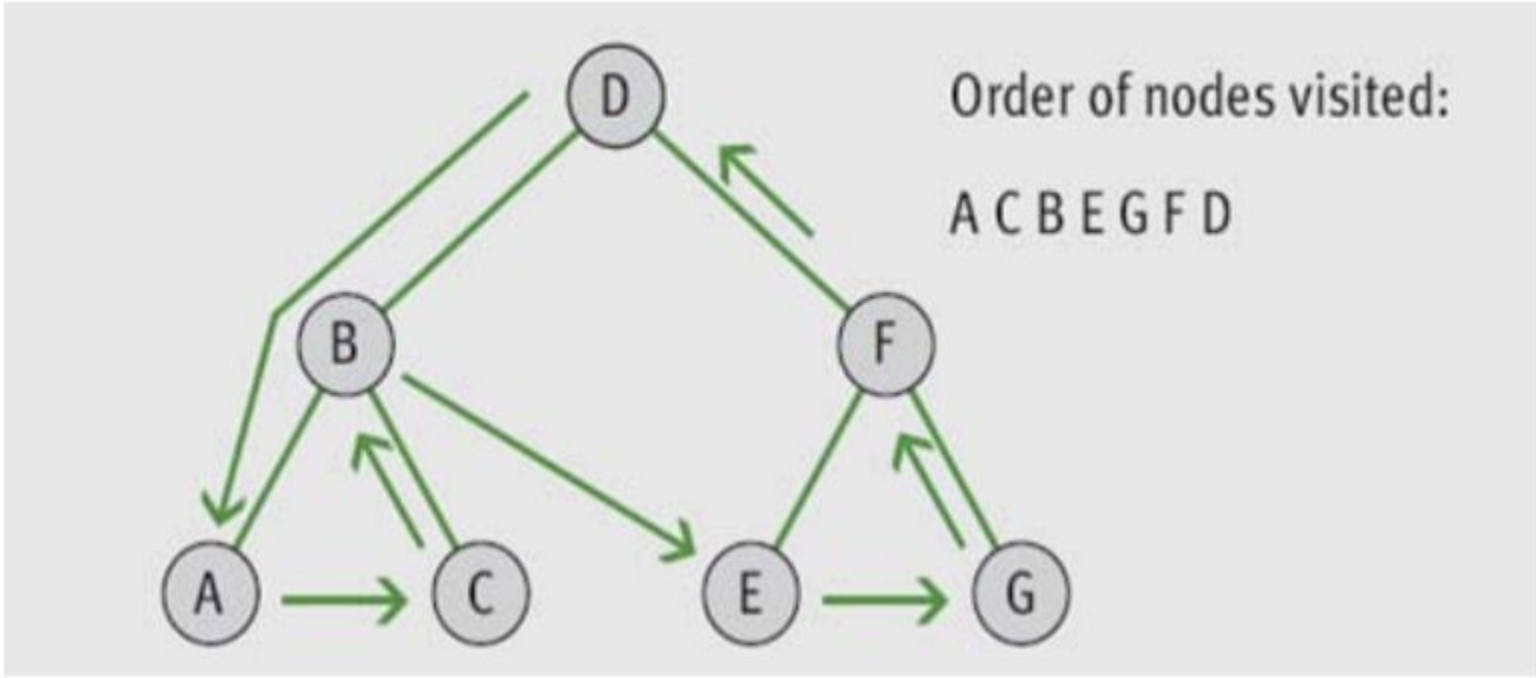
- Left
- Root
- Right



1	2	3	4	5	6	7	8	9	10	11	12	13	14
9	10	12	20	24	25	26	31	44	45	58	77	88	92

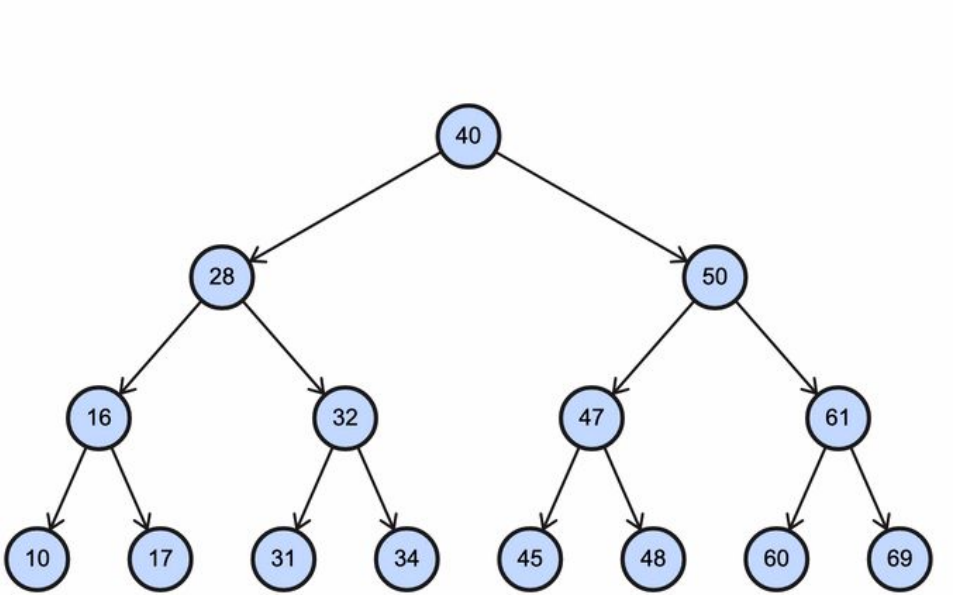
Postorder Traversal

- Traverses left subtree, traverses right subtree, and visits root node



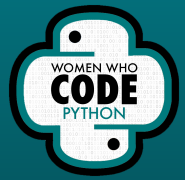
Postorder Traversal

- Left
- Right
- Root



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	17	16	31	34	32	28	45	48	47	60	69	61	50	40

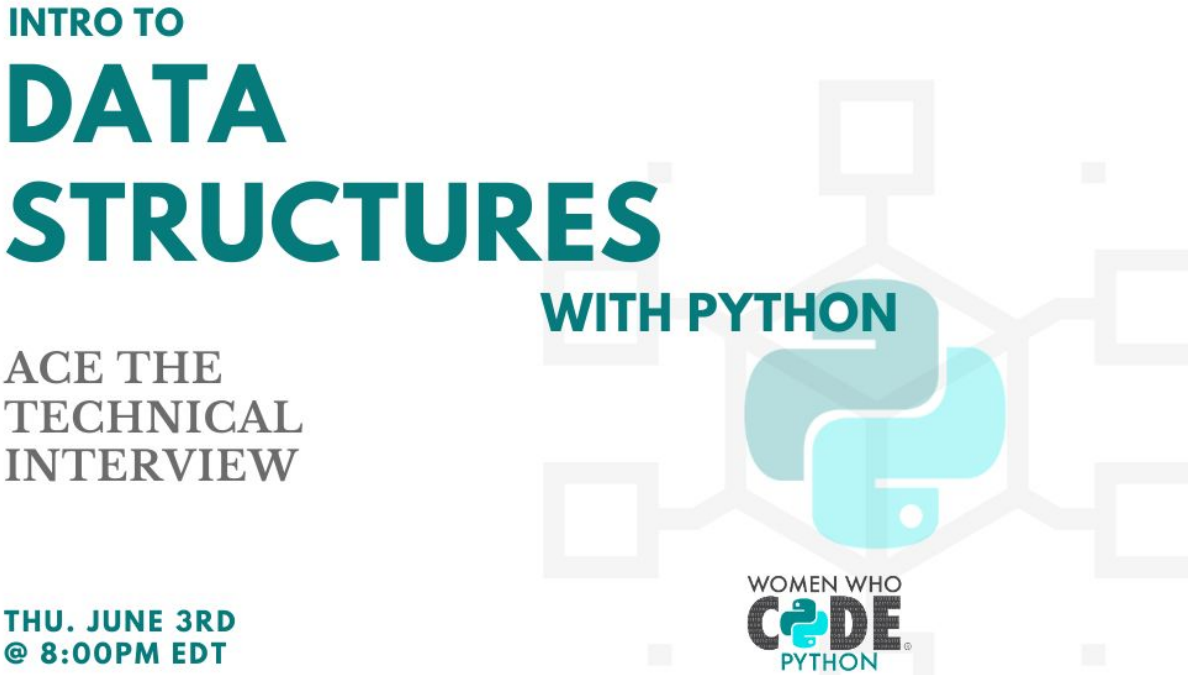
Q&A Time!



Time for Live Coding!

- https://colab.research.google.com/drive/1GF27b0uBFKbNcZe90DyFo38HXTkPM_DR?usp=sharing

Next Session!



WOMEN WHO
CODE
PYTHON

Questions?

Join our Slack channel:
#intro-data-structures-stdy-grp

Thank You!