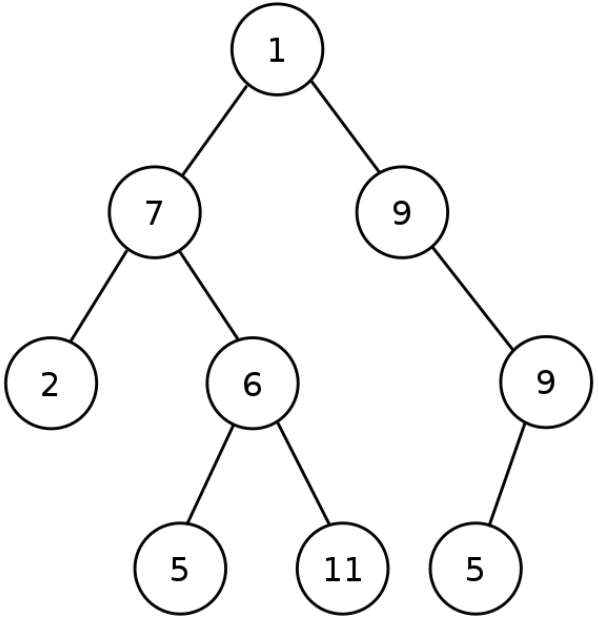
Trees

Trees

- **Tree:** abstract data type that represents a hierarchical structure with a set of connected nodes
- **Nodes** are connected by **edges**
- There's a **hierarchical relationship** between the nodes
- **Root:** Top-most node
- A given node can have **children** and **parents**
- **Depth/Level (of a node):** the count of edges on the path from the root node to a node
- **Height (of a tree):** number of edges from root to furthest child

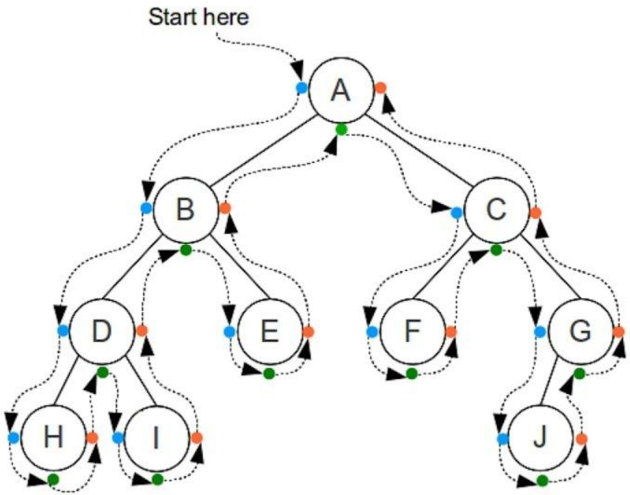


Binary Search Trees (BST)

- **Binary tree:** A tree where each node has 2 children (**left** and **right)**
- **Binary Search Tree** is defined by:
    - All values in **left** subtree are **less** than root
    - All values in **right** subtree are **greater** than root
    - **Both children** of root are **BSTs** as well
- Duplicates generally do not appear in binary trees
- Can be used for binary search with an <u>average</u> O(log n) search, insertion, and deletion operations (compared to O(n) for an array)
- There are balanced variations that you should be familiar with as well that have <u>guaranteed</u> O(log n) operations.

Binary Search Trees (BST)

- **Preorder Traversal:** In a preorder traversal, the tree is traversed according to the following order: **Root-Left-Right**. This means:
  - The **root node** of the subtree is visited first.
  - Then the **left subtree** is traversed.
  - Finally, the **right subtree** is traversed.
- **Inorder Traversal:** In an inorder traversal, the tree is traversed according to the following order: **Left-Root-Right.** This means:
  - The **left subtree** is traversed first.
  - Then the **root node** for that subtree is visited.
  - Finally, the **right subtree** is traversed.

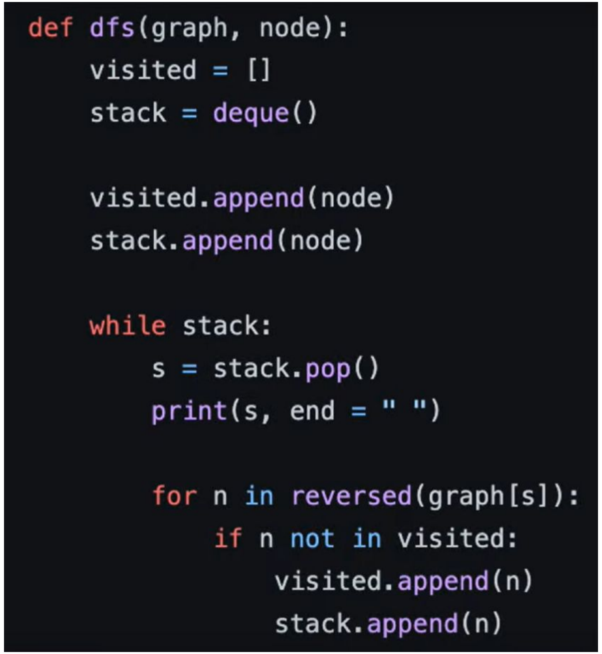


Pre-Order	ABDHIECFGJ
In-Order	HDIBEAFCJG
Post-Order	HIDEBFJGCA

Depth First Search (DFS)

Depth First Search (DFS)

- DFS is an algorithm to visit every node in a tree, it operates with a stack to visit every node.
- Operates horizontally when queuing by using a FILO structure (First In, Last Out)



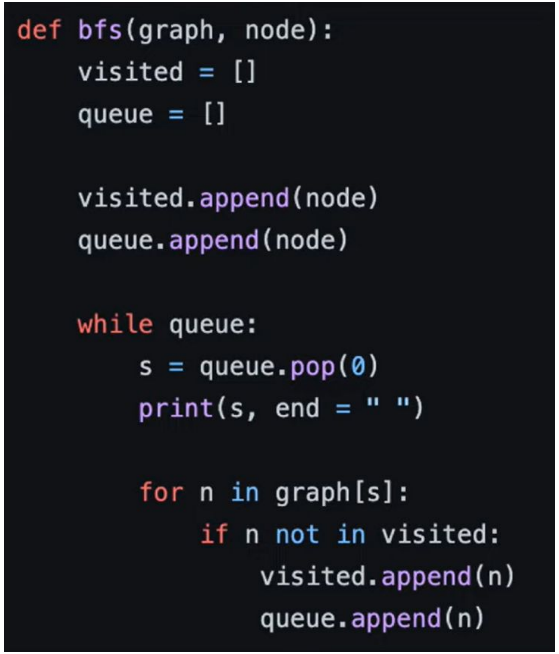
```python
def dfs(graph, node):
    visited = []
    stack = deque()

    visited.append(node)
    stack.append(node)

    while stack:
        s = stack.pop()
        print(s, end = " ")

        for n in reversed(graph[s]):
            if n not in visited:
                visited.append(n)
                stack.append(n)
```

DFS Example, More info

Breadth First Search (BFS)

Breadth First Search (BFS)

- BFS is another algorithm to visit every node in a tree, it operates with a queue to visit every node.
- Operates horizontally when queuing by using
- a FIFO structure (First In, First Out)



```python
def bfs(graph, node):
    visited = []
    queue = []

    visited.append(node)
    queue.append(node)

    while queue:
        s = queue.pop(0)
        print(s, end = " ")

        for n in graph[s]:
            if n not in visited:
                visited.append(n)
                queue.append(n)
```

[BFS Example](), [More info]()