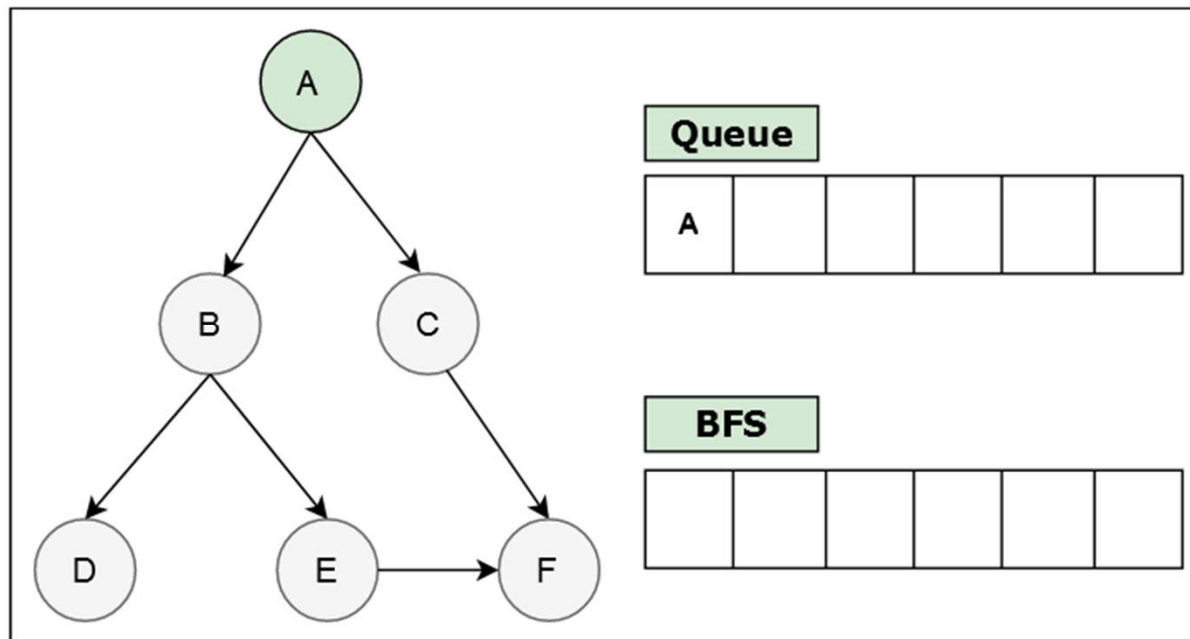


The background of the slide is a complex, abstract pattern composed of numerous triangles in various shades of purple, blue, and black. The triangles are arranged in a way that creates a sense of depth and movement, with some triangles appearing to overlap others. The overall effect is a modern, digital aesthetic.

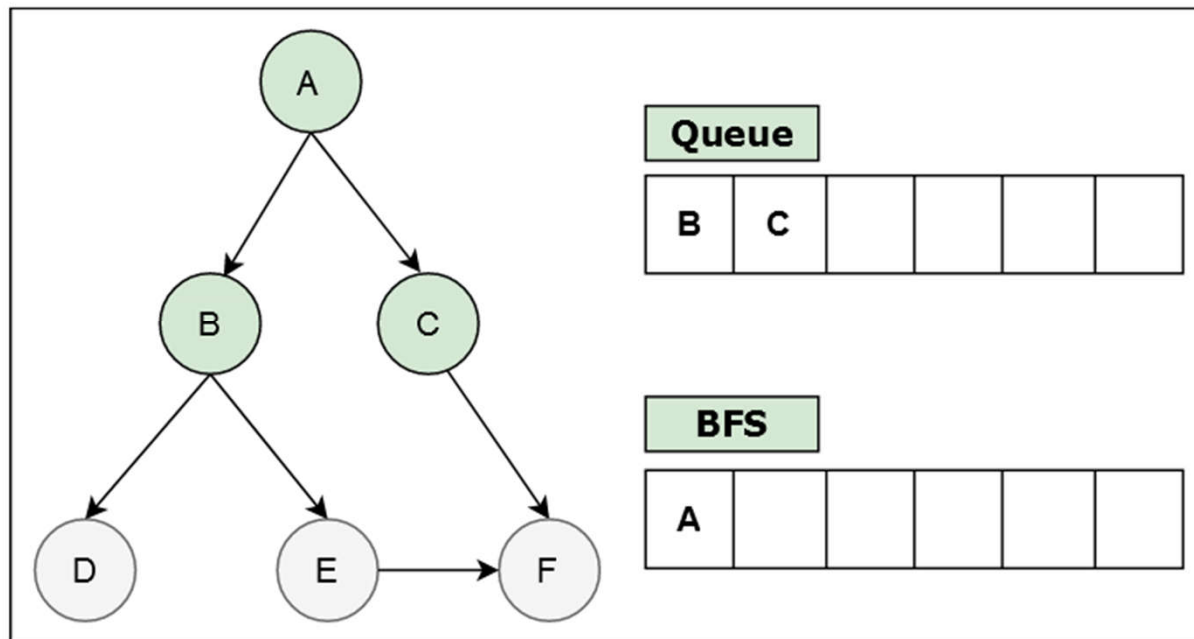
# **AI LAB WEEK 4**

**Dr. Mubashir Ahmad**

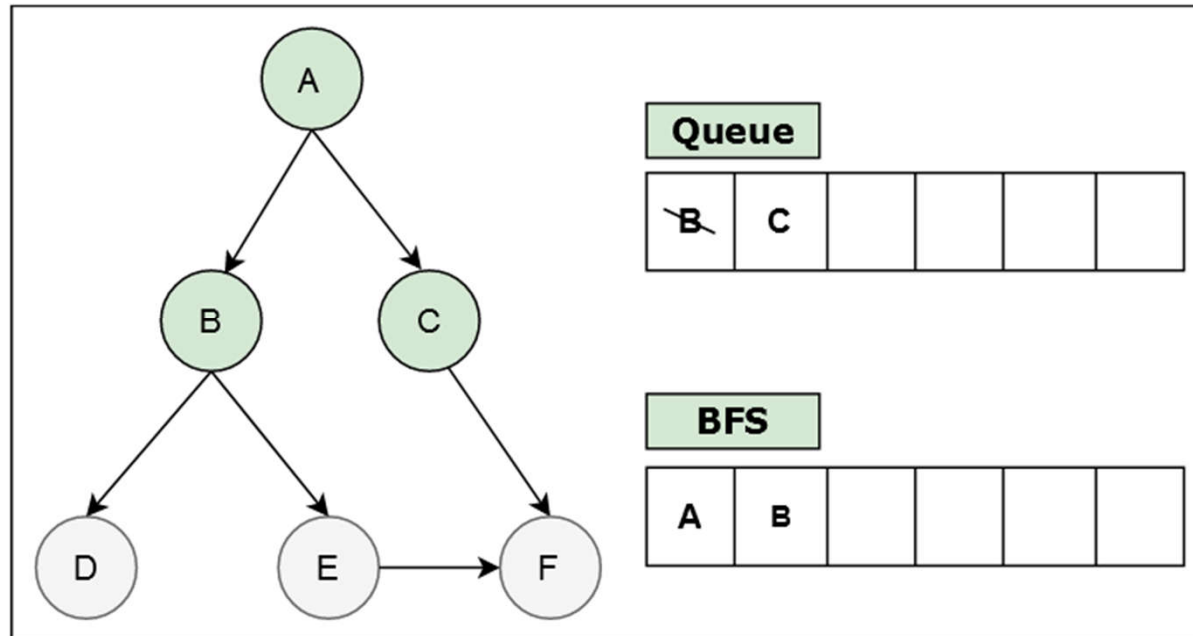
# BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



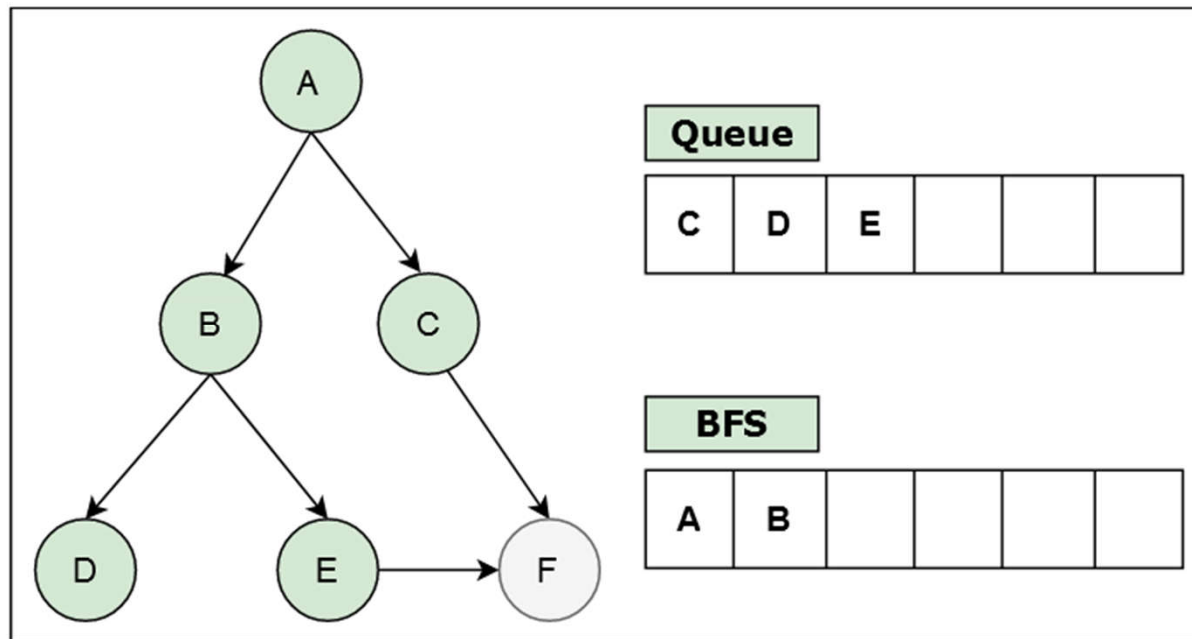
# BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



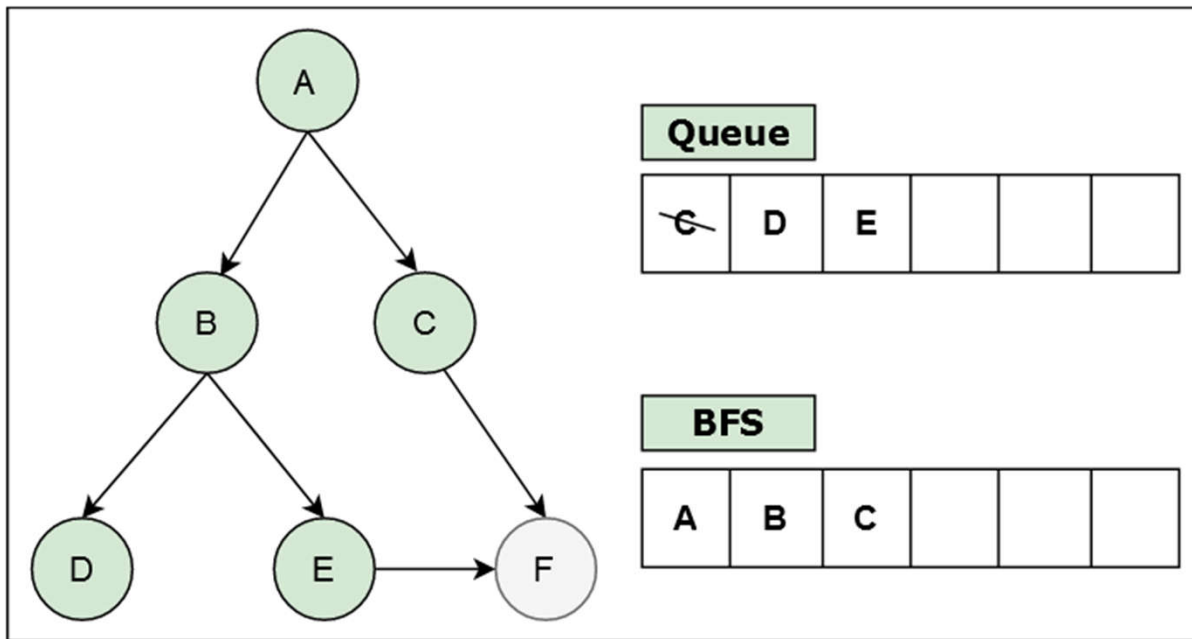
# BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



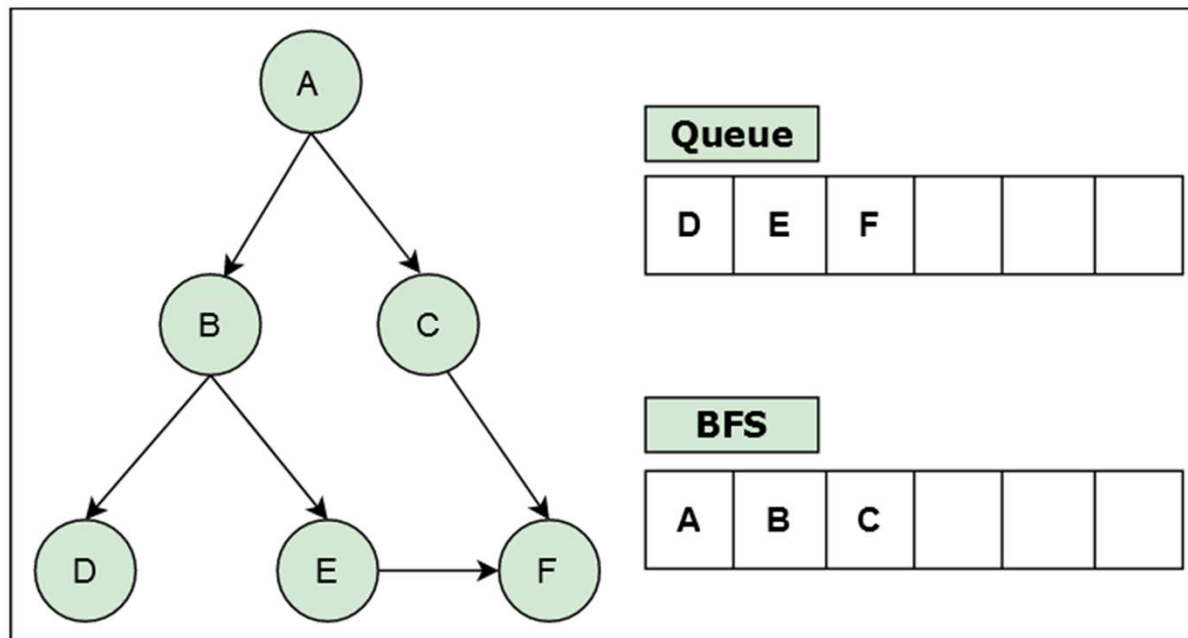
# BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



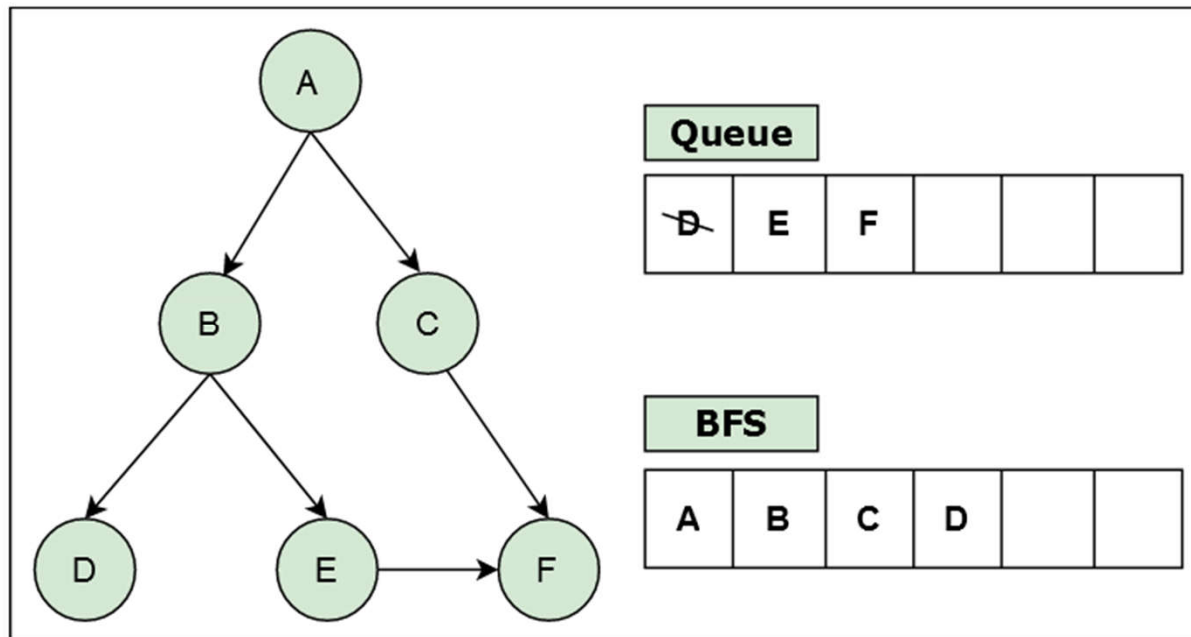
# BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



# BREADTH FIRST SEARCH PYTHON IMPLEMENTATION

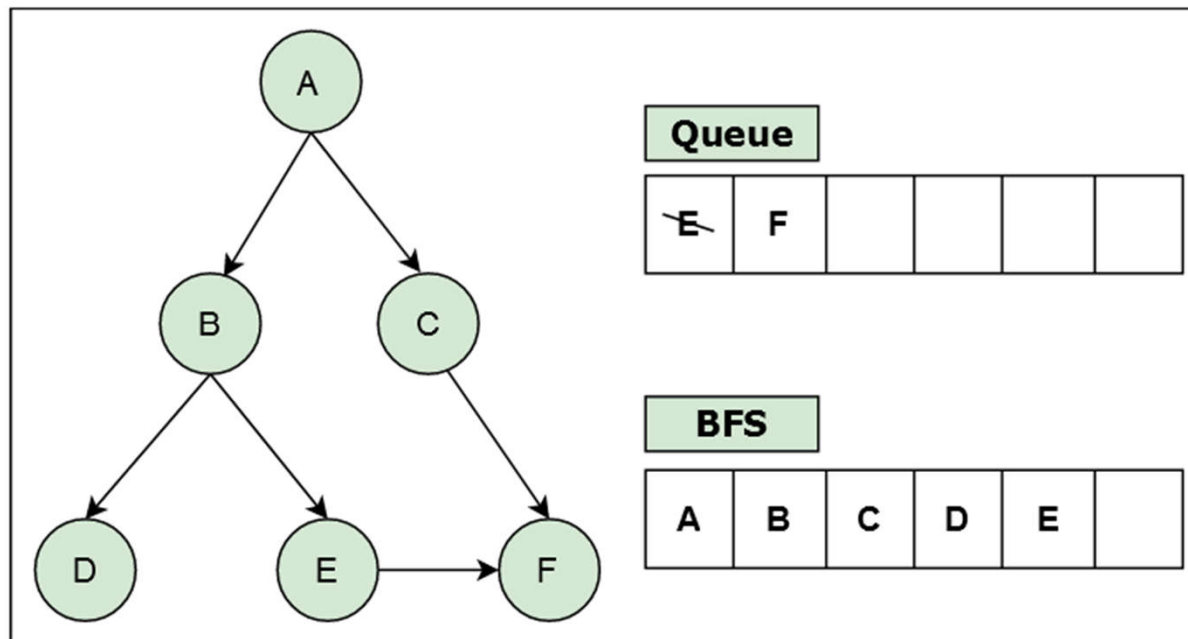


# BREADTH FIRST SEARCH PYTHON IMPLEMENTATION

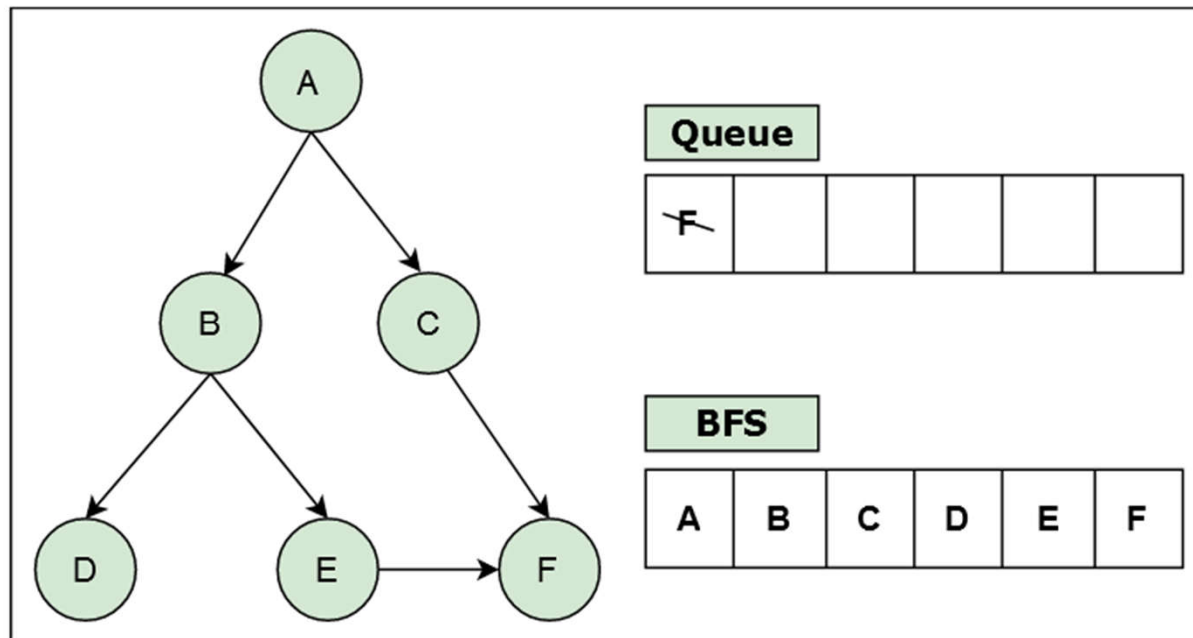




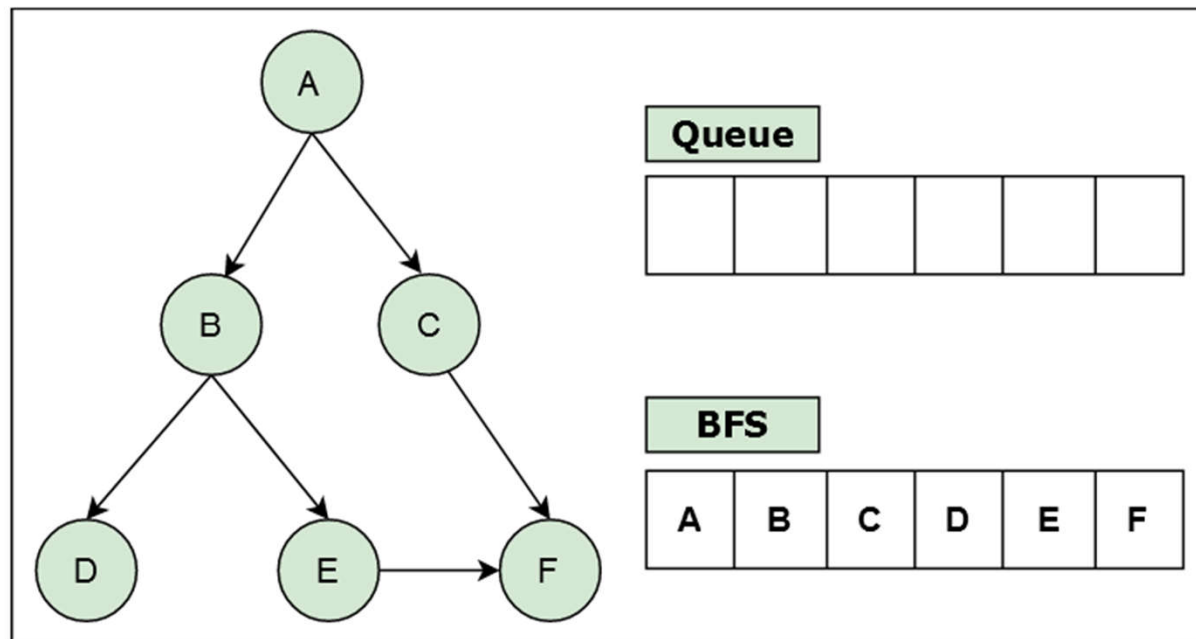
# BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



# BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



# BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



```

graph = {
    'A' : ['B', 'C'],
    'B' : ['D', 'E'],
    'C' : ['F'],
    'D' : [],
    'E' : [],
    'F' : []
}

visited = [] # List to keep track of visited nodes.
queue = []    #Initialize a queue

def bfs(visited, graph, node):
    visited.append(node)
    queue.append(node)

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

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

# Driver Code
bfs(visited, graph, 'A')

```

```

graph = {
    'A' : ['B', 'C'],
    'B' : ['D', 'E'],
    'C' : ['F'],
    'D' : [],
    'E' : ['G'],
    'F' : ['H', 'I']
}

visited = [] # List to keep track of visited nodes.
queue = []    #Initialize a queue

def bfs(visited, graph, node, goal):
    visited.append(node)
    queue.append(node)

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

        if s == goal:
            print("\nGoal state reached:", goal)
            return

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

# Driver Code
bfs(visited, graph, 'A', 'G')

```

# 8-Puzzle Problem without Heuristic

→ Blind Search (Uninformed)

→ BFS

→  $O(b^d)$  branch depth

→ 4 moves (UP, down, left, Right)

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline & 4 & 6 \\ \hline 7 & 5 & 8 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline 7 & 8 & \\ \hline \end{array}$$


G

$O(3^{20})$

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & & 6 \\ \hline 7 & 5 & 8 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline & 2 & 3 \\ \hline 1 & 4 & 6 \\ \hline 7 & 5 & 8 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 7 & 4 & 6 \\ \hline & 5 & 8 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline 7 & & 8 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline & 4 & 6 \\ \hline 7 & 5 & 8 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 6 & \\ \hline 7 & 5 & 8 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline 1 & & 3 \\ \hline 4 & 2 & 6 \\ \hline 7 & 5 & 8 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 3 & 6 \\ \hline 7 & 8 & \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline 5 & 7 & 8 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline \cdot & x & \cdot \\ \hline x & 3 & x \\ \hline \cdot & x & \cdot \\ \hline \end{array}$$

$$\frac{4 \times 2 + 4 \times 3 + 4}{9} = \frac{24}{9} = \sqrt{2.67} = 3$$

# TASKS

- #1. Using DFS find the goal state (Python Implementation).**
- #2. Implementation of DFS to traverse each node.**
- #3. Implementation of Sliding puzzle using DFS.**