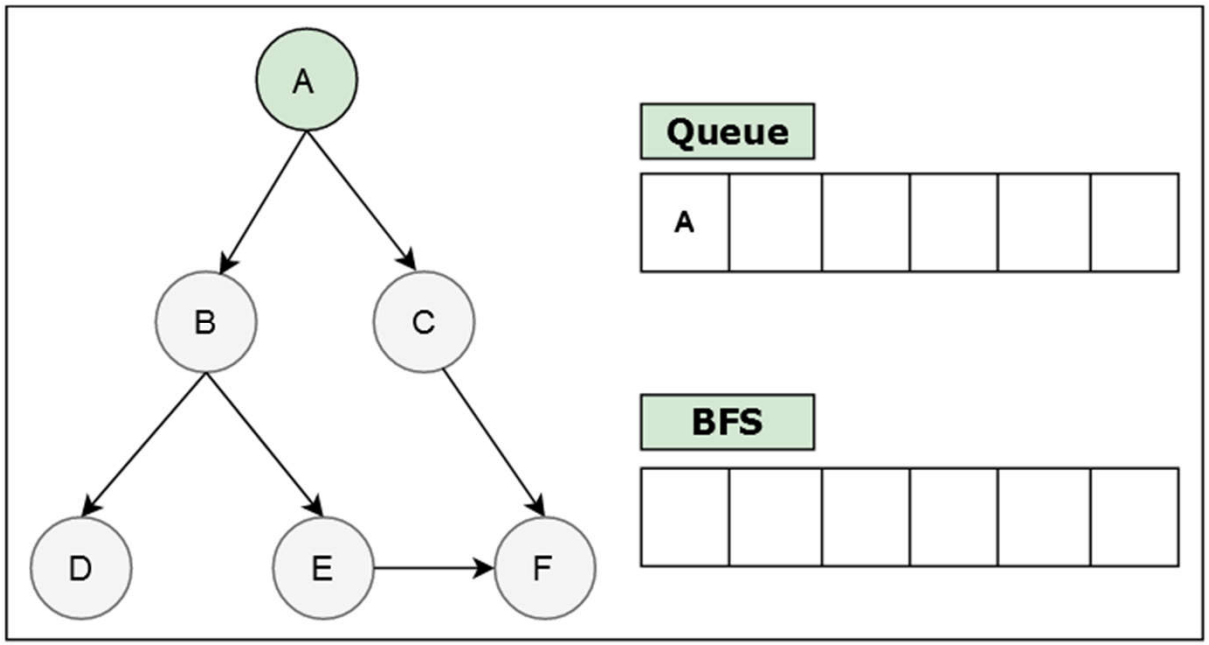
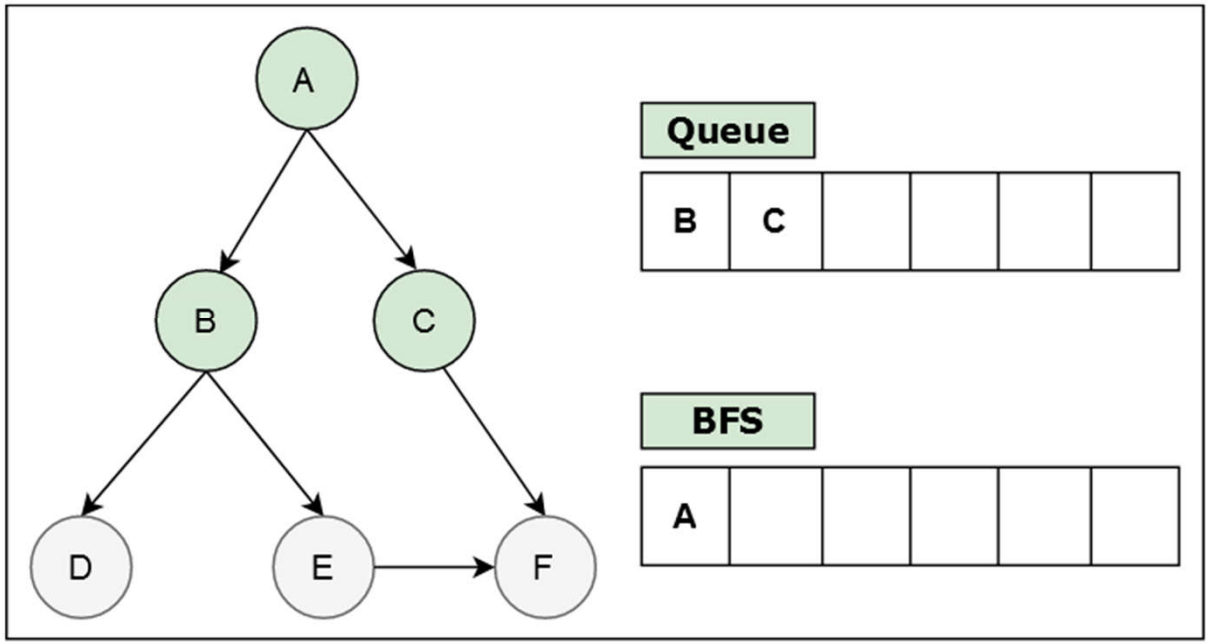
AI LAB WEEK 4

Dr. Mubashir Ahmad

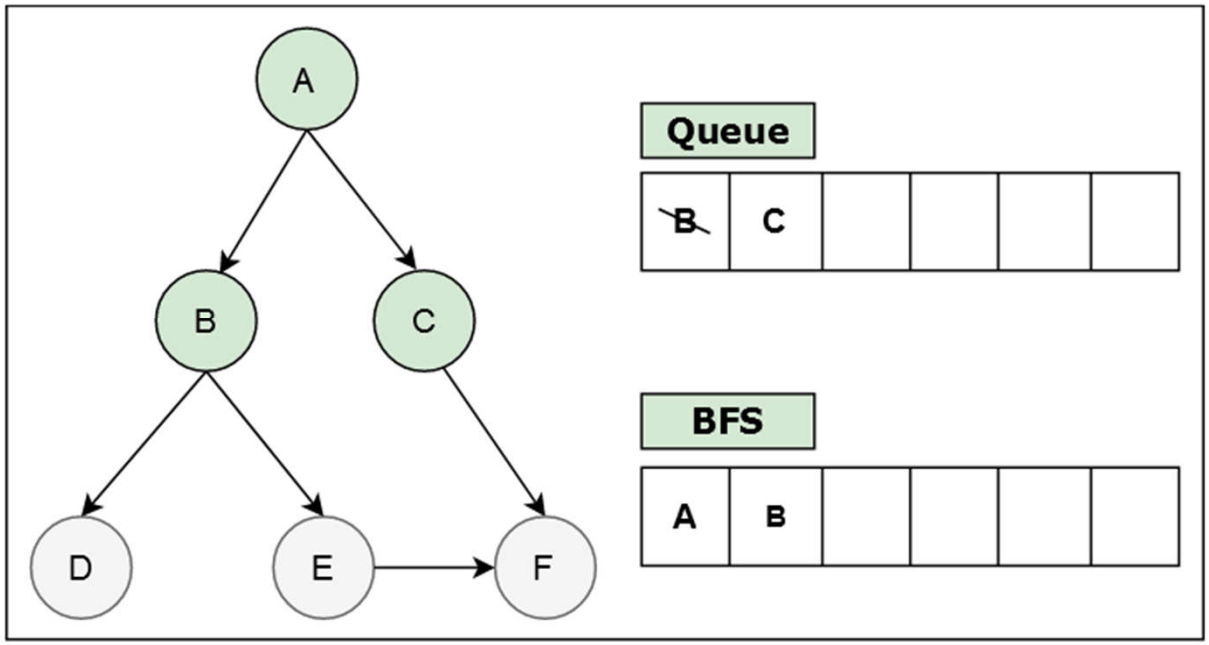
BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



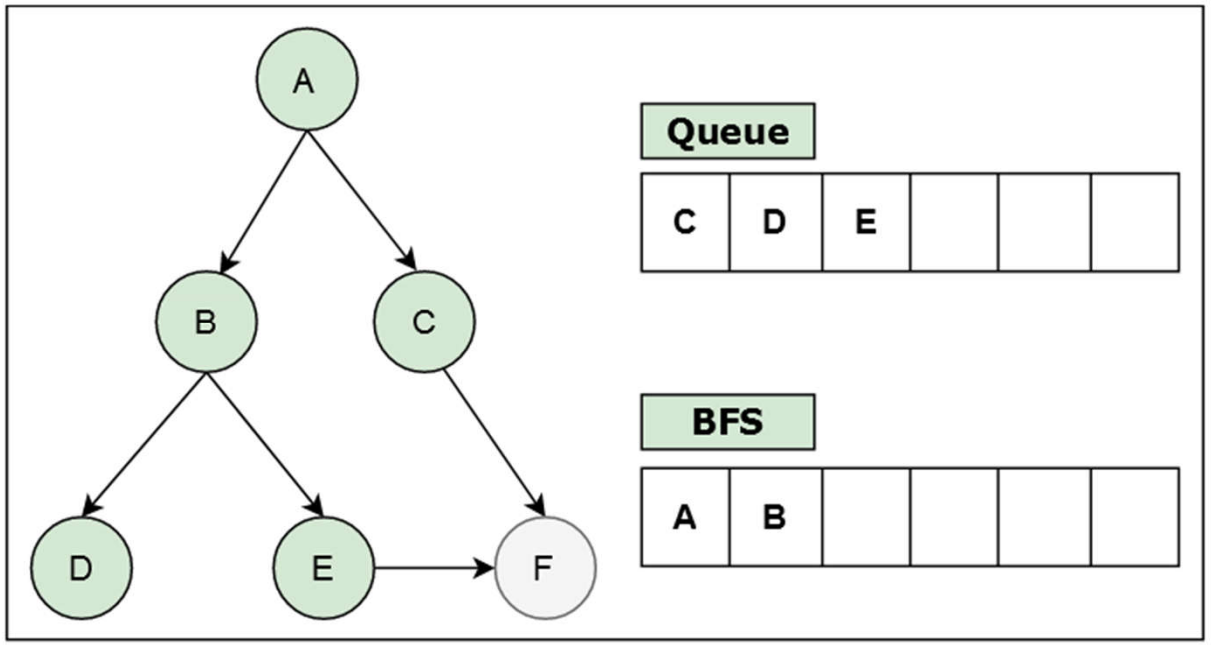
BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



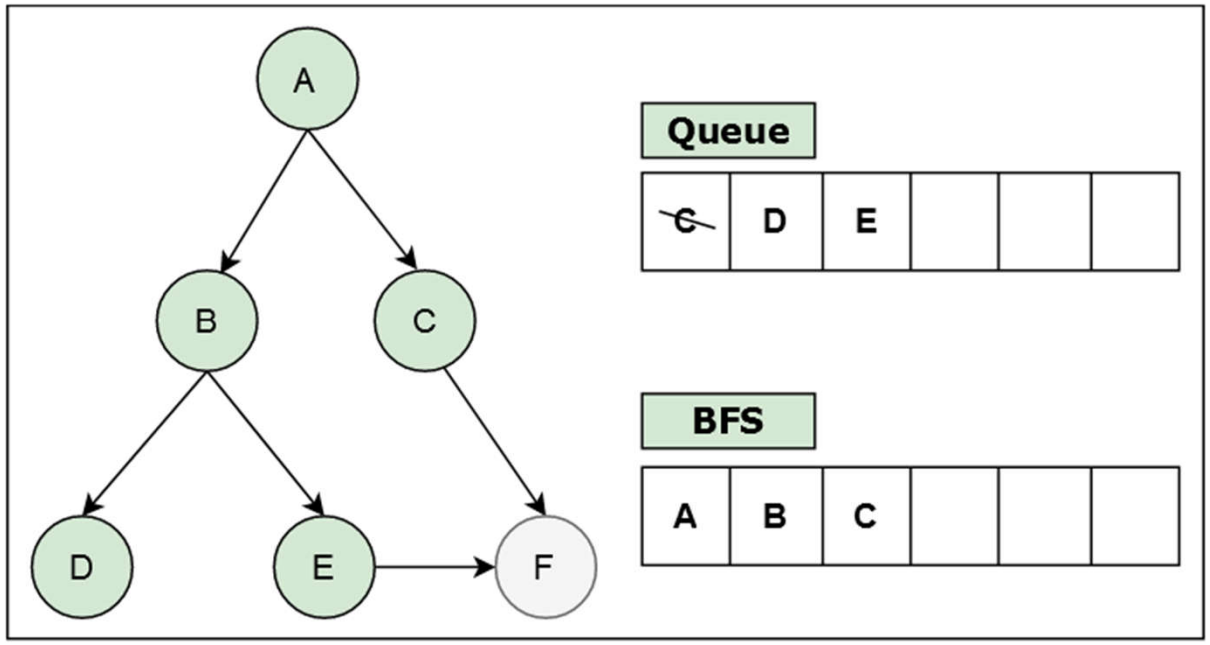
BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



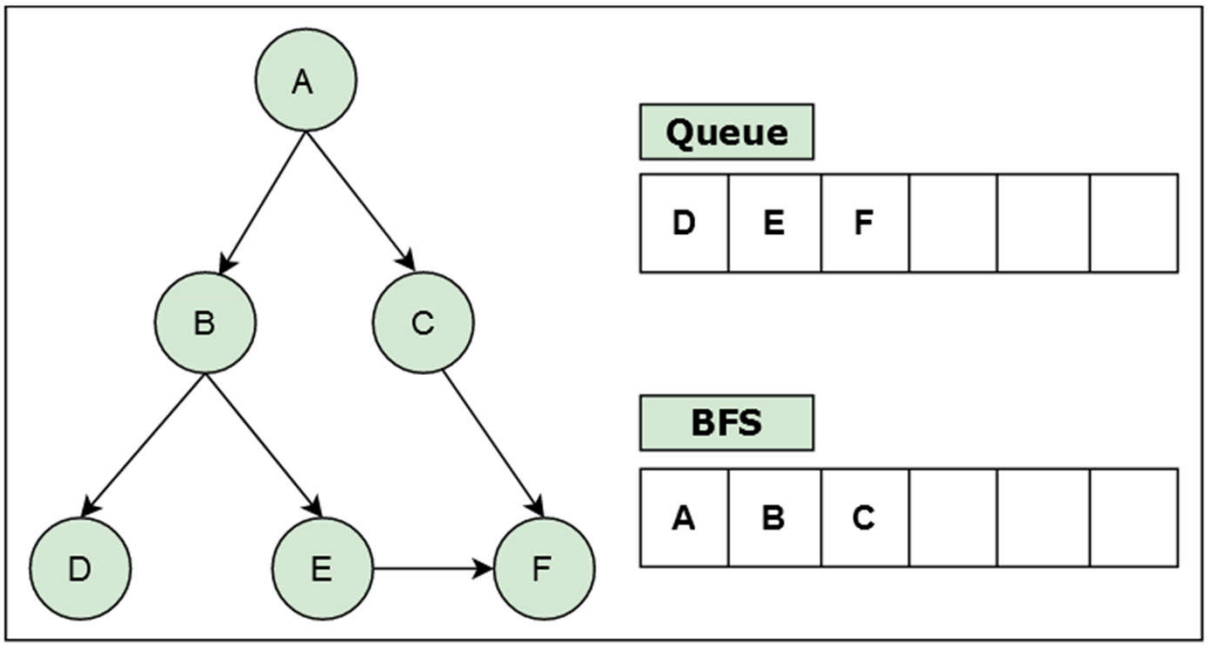
BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



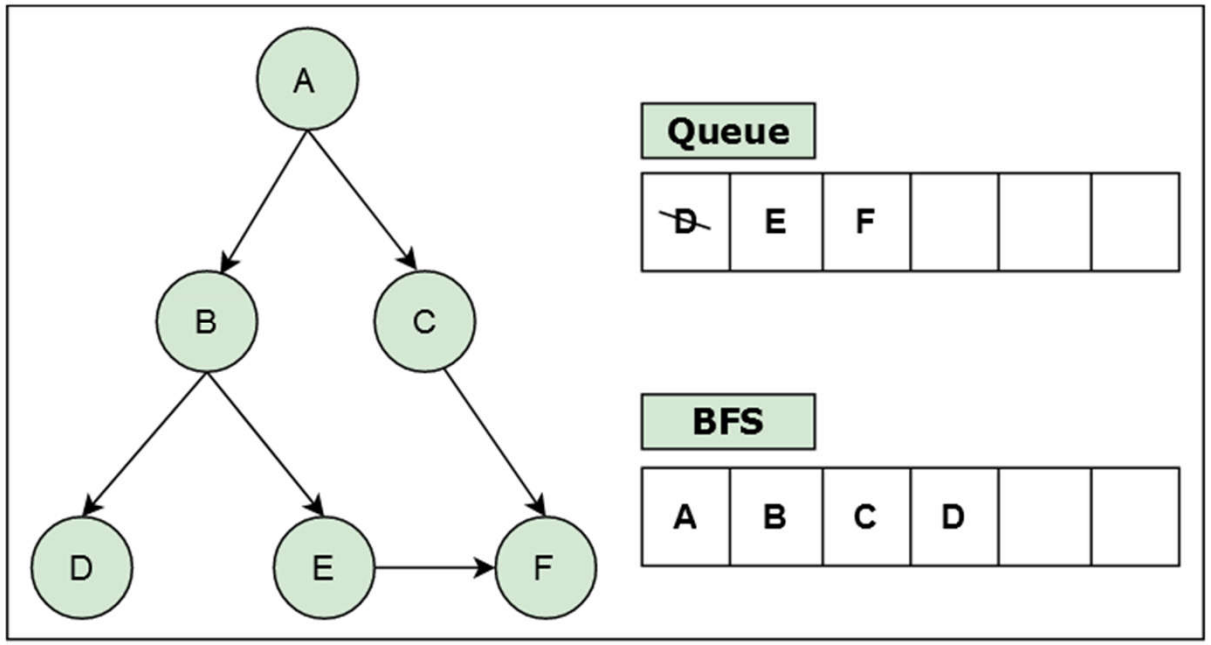
BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



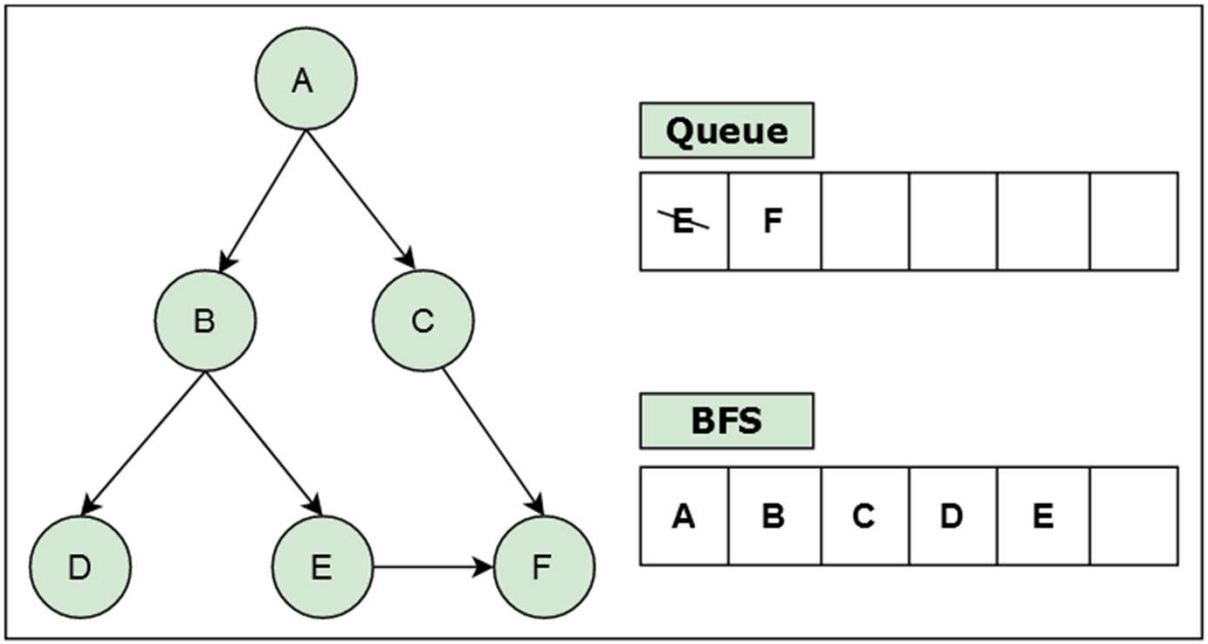
BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



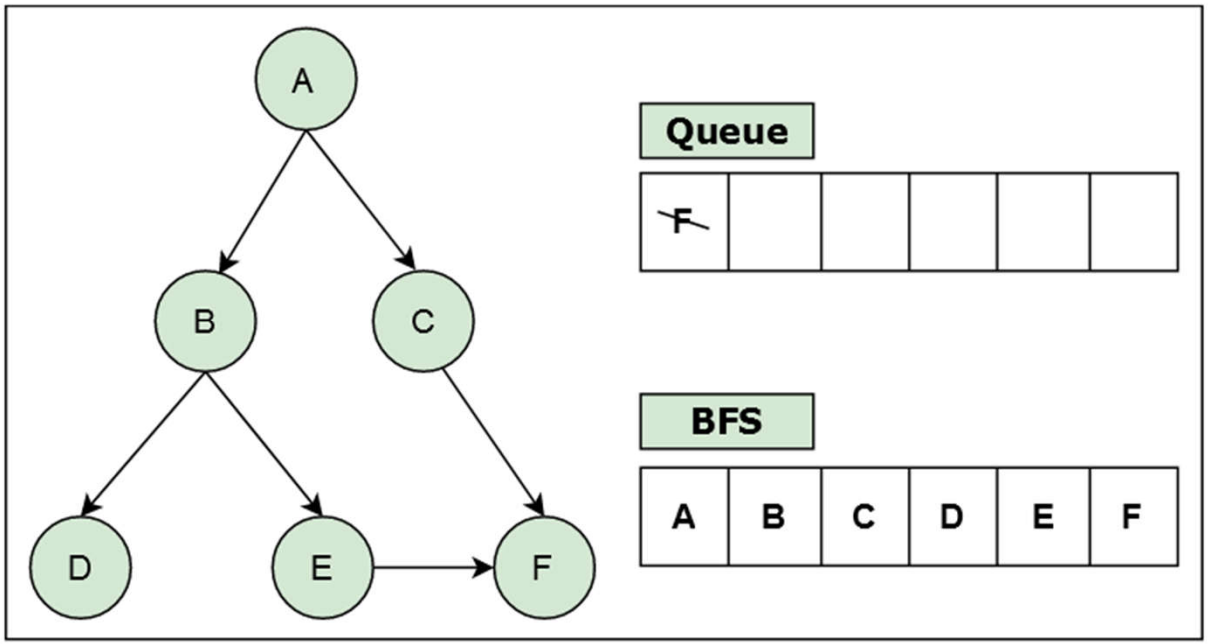
BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



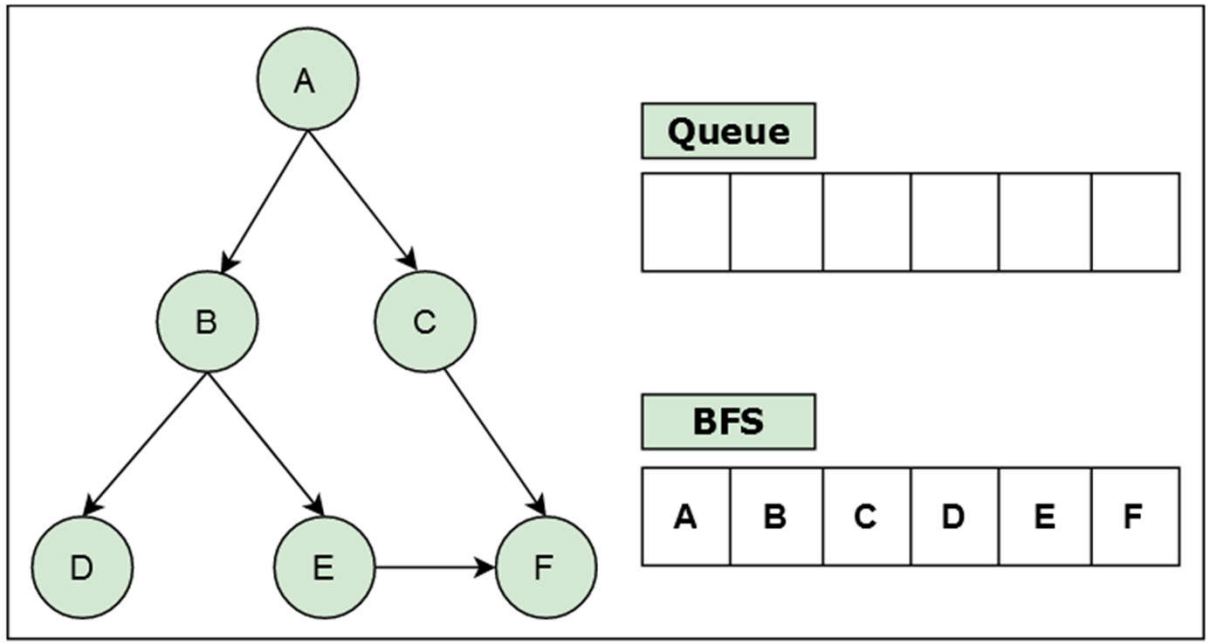
BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



BREADTH FIRST SEARCH PYTHON IMPLEMENTATION



```python
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')
```

```python
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')
```

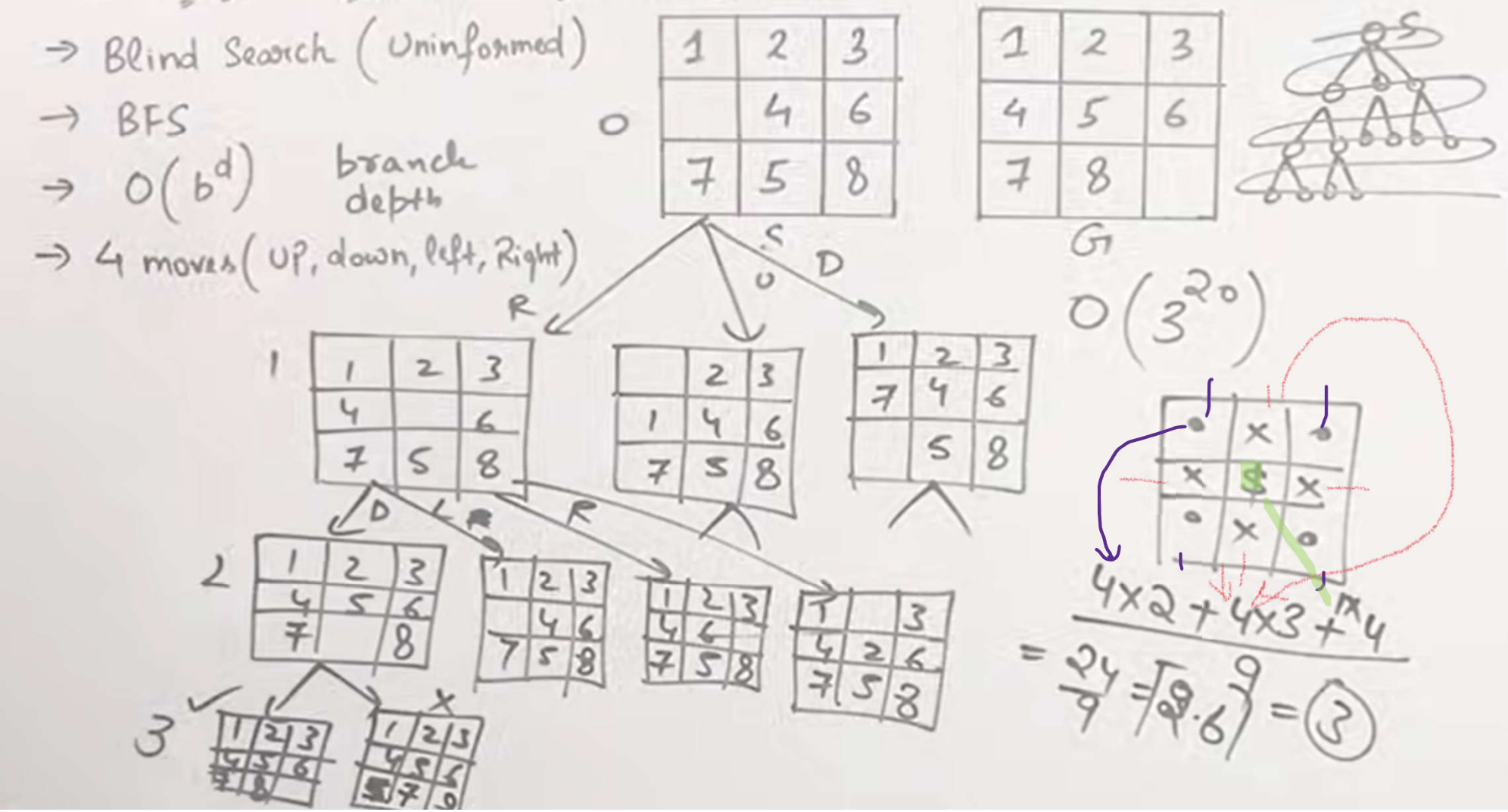
8-Puzzle Problem without Heuristic

→ Blind Search (Uninformed)

→ BFS

→ $O(b^d)$ b = branch, d = depth

→ 4 moves (Up, down, left, Right)



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.