

**BFS**

**Code:**

from collections import deque

def bfs(graph, start):

    visited = set()

    queue = deque([start])

    bfs\_order = []

    while queue:

        node = queue.popleft()

        if node not in visited:

            visited.add(node)

            bfs\_order.append(node)

            for neighbor in graph[node]:

                if neighbor not in visited:

                    queue.append(neighbor)

    return bfs\_order

graph = {

    'A': ['B', 'C'],

    'B': ['A', 'D', 'E'],

    'C': ['A', 'F'],

    'D': ['B'],

    'E': ['B', 'F'],

    'F': ['C', 'E']

}

start\_node = 'A'

bfs\_result = bfs(graph, start\_node)

print("BFS Traversal:", bfs\_result)

**output:**

****

**DFS**

**Code:**

def dfs(graph, start, visited=None):

    if visited is None:

        visited = set()

    visited.add(start)

    for neighbor in graph[start]:

        if neighbor not in visited:

            dfs(graph, neighbor, visited)

    return visited

graph = {

    'A': ['B', 'C'],

    'B': ['A', 'D', 'E'],

    'C': ['A', 'F'],

    'D': ['B'],

    'E': ['B', 'F'],

    'F': ['C', 'E']

}

start\_node = 'A'

dfs\_result = dfs(graph, start\_node)

print("DFS Traversal:", dfs\_result)

**output:**

****