```
from collections import deque
def bfs(graph, start):
    visited = set()
    queue = deque([start])
    result = []
    while queue:
        vertex = queue.popleft()
        if vertex not in visited:
            visited.add(vertex)
            result.append(vertex)
            # Add all unvisited neighbors
            queue.extend([v for v in graph[vertex] if v not in
visited])
    return result
def dfs(graph, start, visited=None, result=None):
    if visited is None:
        visited = set()
    if result is None:
        result = []
    visited.add(start)
    result.append(start)
    for neighbor in graph[start]:
```

```
            if neighbor not in visited:
                dfs(graph, neighbor, visited, result)
        return result
# ----- Main Program -----
if __name__ == "__main__":
    graph = {}
    n = int(input("Enter number of vertices: "))
    e = int(input("Enter number of edges: "))
    # Initialize adjacency list
    for i in range(1, n + 1):
        graph[i] = []
    print("\nEnter edges (u v):")
    for _ in range(e):
        u, v = map(int, input().split())
        graph[u].append(v)
        graph[v].append(u)  # For undirected graph
    start = int(input("\nEnter starting vertex: "))
    print("\nAdjacency List:")
    for node, neighbors in graph.items():
        print(f"{node} -> {neighbors}")
    print("\nBFS Traversal:", bfs(graph, start))
    print("DFS Traversal:", dfs(graph, start))
```