```
from collections import deque

def bfs(graph, start_vertex):
    visited = set()
    queue = deque([start_vertex])

    while queue:
        current_vertex = queue.popleft()
        if current_vertex not in visited:
            print(current_vertex, end=' ')
            visited.add(current_vertex)


            queue.extend(neighbor for neighbor in graph[current_vertex] if neighbor not in visited)


graph = {
    0: [1, 3],
    1: [0, 2, 3],
    2: [1, 4, 5],
    3: [0, 1, 4],
    4: [2, 3, 5],
    5: [2, 4],
}

start_vertex = 0
print("BFS traversal starting from vertex", start_vertex, ":")
bfs(graph, start_vertex)
```

```
BFS traversal starting from vertex 0 :
0 1 3 2 4 5
```

```
def dfs(graph, start):
    visited = set()
    stack = [start]

    while stack:
        current_node = stack.pop()

        if current_node not in visited:
            print(current_node, end=' ')
            visited.add(current_node)

            # Push neighboring nodes onto the stack in reverse order to maintain desired order
            stack.extend(neighbor for neighbor in reversed(graph[current_node]) if neighbor not in visited)

# Example graph represented as an adjacency list
graph = {
    'A': ['B', 'S'],
    'B': ['A'],
    'C': ['D', 'E', 'F','S'],
    'D': ['C'],
    'E': ['H', 'C'],
    'F': ['C','G'],
    'G': ['S','H','F'],
    'H': ['G','E'],
    'S': ['A','C','G'],
}

start_node = 'A'
print("DFS traversal starting from node", start_node)
dfs(graph, start_node)
```

```
DFS traversal starting from node A
A B S C D E H G F
```