Problem – 9,10 : Bipartite Check using :

9. DFS
10. BFS

```
from collections import deque


def is_bipartite_bfs(graph, start_node):
    queue = deque([start_node])
    color = {start_node: 1}  # Colors: 1 and -1 represent the two partitions.


    while queue:
        current_node = queue.popleft()
        for neighbor in graph[current_node]:
            if neighbor not in color:
                color[neighbor] = -color[current_node]
                queue.append(neighbor)
            elif color[neighbor] == color[current_node]:
                return False


    return True


def is_bipartite_dfs(graph, start_node):
    stack = [(start_node, 1)]
    color = {}


    while stack:
        current_node, current_color = stack.pop()
        if current_node in color:
            if color[current_node] != current_color:
                return False
            continue
        color[current_node] = current_color
```

```
        next_color = -current_color
        stack.extend((neighbor, next_color) for neighbor in graph[current_node])

    return True

def is_bipartite(graph):
    start_node = next(iter(graph))
    return is_bipartite_bfs(graph, start_node) and is_bipartite_dfs(graph, start_node)



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

print(is_bipartite(graph))
```