PROBLEM 1: BICOLORING

Problem Link:

https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&problem=94

Problem Statement:

Given an undirected graph with n vertices (labeled 0 to n−1) and l edges, determine whether the graph is bicolorable.
A graph is considered bicolorable if each vertex can be assigned one of two colors so that no edge connects two vertices of the same color.

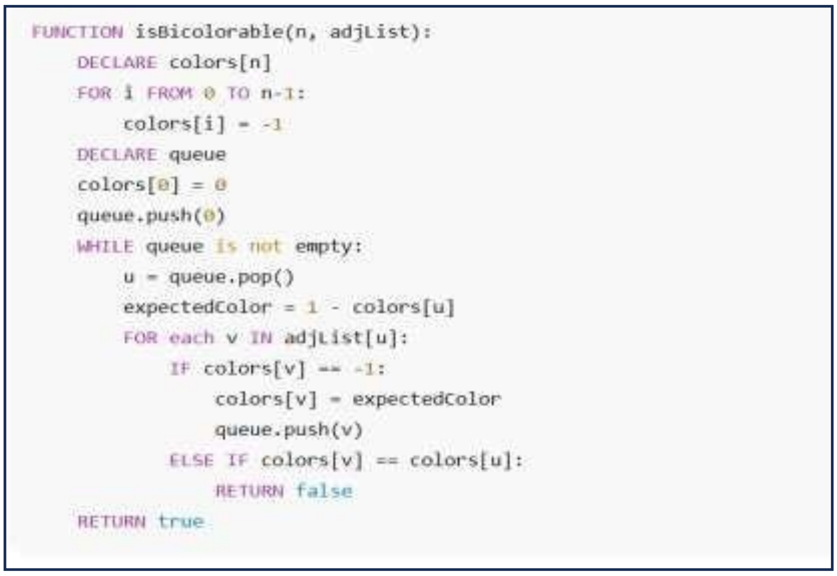
Input consists of multiple test cases:

- Each test case begins with an integer n (number of vertices).
- If n = 0, input ends.
- The next integer is l (number of edges).
- Then l pairs of integers follow, each indicating an undirected edge between two vertices.

For each test case, output:

- "**BICOLORABLE**." if the graph can be properly colored with two colors.
- "**NOT BICOLORABLE**." otherwise.

Pseudocode



```
FUNCTION isBicolorable(n, adjList):
    DECLARE colors[n]
    FOR i FROM 0 TO n-1:
        colors[i] = -1
    DECLARE queue
    colors[0] = 0
    queue.push(0)
    WHILE queue is not empty:
        u = queue.pop()
        expectedColor = 1 - colors[u]
        FOR each v IN adjList[u]:
            IF colors[v] == -1:
                colors[v] = expectedColor
                queue.push(v)
            ELSE IF colors[v] == colors[u]:
                RETURN false
    RETURN true
```

## Hints for Solving the Problem

Hint 1 — Use a Graph Coloring Strategy

Assign two possible colors, such as 0 and 1.
If two adjacent vertices ever receive the same color, the graph is not bicolorable.

Hint 2 — Perform BFS or DFS

Start from any vertex (commonly vertex 0).
Traverse the graph.
Assign alternating colors as neighbors are visited.

Hint 3 — Keep a Color Array

Use an array initialized with a value such as -1 to represent uncolored vertices.
When visiting a vertex:

- Assign the opposite color to each uncolored neighbor.
- Check for conflicts with already colored neighbors.

Hint 4 — Stop on First Conflict

As soon as an edge with equal colors on both ends is detected, conclude that the graph is not bicolorable.

Hint 5 — Consider Disconnected Graphs (General Case)

If the graph contains multiple components, run BFS/DFS on each uncolored vertex until all components are processed.

Implementation Link:

https://github.com/TasnimaSultana/Algo_Lab_Final/blob/main/BFS/Bicoloring/bicoloring.cpp