This code implements Tarjan's algorithm for finding bridges in an undirected graph. A bridge is an edge whose removal would disconnect the graph into two separate components. The algorithm should identify all such bridges in the given graph. The resulting set of bridges should represent all the critical connections in the graph structure. The algorithm should work as follows:

**How it works:**

1. Initialize the algorithm with a graph represented by vertices and edges.
2. Perform a depth-first search (DFS) traversal of the graph. During the DFS:
    1. Assign each vertex an index (discovery time) and a low-link value. The low-link value represents the lowest index of any vertex reachable from this vertex, including itself.
    2. Update low-link values based on the traversal: Initially set to the vertex's own index. When updating, updated to be the minimum of its own value, its children's low-link values, and the indices of back edge destinations.
    3. Identify bridges when a vertex's low-link value is greater than its parent's index, which indicates no back edge exists connecting the subtree to an ancestor.
3. Add identified bridges to the result list.
4. Repeat the process for any unvisited vertices to handle disconnected graphs.
5. Return the list of all bridges found in the graph.

Currently, the implementation of this feature contains a logic bug that causes the algorithm to deviate from its intended functionality.

**Algorithm's expected outcomes:**

For the proposed input graph in the codebase, the expected output (Bridge) is:

- C-D

**Algorithm's actual outcomes:**

For the proposed input graph in the codebase, the expected output (Bridge) is:

- No bridges found