Ex : 8

Title: Design, Develop and Implement a Program in C to create N router connections and display the routers by removal of which will disconnect the network completely.

Problem Description: create a data structure to represent the routers and its connections between the routers and display the router numbers on removal which it disconnects the network completely.

Method: choose the best data structure to represent the routers and a function to find the articulation point.

Theory Reference: Module 5

Explanation:

- **Articulation Point**: A vertex that, when removed, disconnects the graph.
- **DFS**: A traversal technique used to explore nodes and edges of a graph. The `dfnlow` function performs the main DFS operation while tracking discovery and low values.
- **dfn[u]**: Discovery time of vertex `u`.
- **low[u]**: The lowest discovery time reachable from `u`.
- **parent[u]**: The parent vertex of `u` in the DFS tree.
- **Adjacency Matrix**: The graph is represented as an adjacency matrix `adj`, where `adj[u][v]` is `1` if there's an edge between routers `u` and `v`.

Following Initializations are done:

- Arrays are initialized to keep track of:
  - Discovery times (`dfn`)
  - Low values (`low`)
  - Parent vertices (`parent`)
  - Visited vertices (`visited`)
  - Articulation points (`articulation`)

Algorithm:

Step 1: main Function - To initialize the graph and collect user input for the number of vertices and edges. Update the adjacency matrix to mark the edge between u and v as present (set adj[u][v] and adj[v][u] to 1).

Step 2: findArticulationPoints Function - To find articulation points in the graph using Depth-First Search (DFS).

1. For i=0 to V−1:
    - Initialize `parent[i]` to -1 (indicating no parent).
    - Initialize `visited[i]` to 0 (indicating not visited).
    - Initialize `articulation[i]` to 0 (indicating not an articulation point).
2. For i=0 to V−1:
    - If `visited[i]` is 0, call `dfnlow(i, adj, V)` to perform DFS from vertex i.
3. Print the articulation points:
    - For i=0 to V−1:
        - If `articulation[i]` is 1, print "Router i".

Step 3. dfnlow Function - To perform DFS and update dfn and low values, identifying articulation points.

1. Initialize `children` to 0 (to count child vertices of the current vertex u).
2. Mark `u` as visited: set `visited[u]` to 1.
3. Set `dfn[u]` and `low[u]` to `time` (incremented by 1).
4. For each vertex v from 0 to V−1:
    - If there is an edge from u to v (`adj[u][v] == 1`):
        - If v is not visited:
            1. Increment `children` by 1.
            2. Set `parent[v]` to u.
            3. Recursively call `dfnlow(v, adj, V)`.
            4. Update `low[u]`: set `low[u]` to the minimum of `low[u]` and `low[v]`.
            5. Check if u is an articulation point:
                - If u is the root of the DFS tree (i.e., `parent[u] == -1`) and has two or more children, mark u as an articulation point.
                - If u is not the root and `low[v] >= dfn[u]`, mark u as an articulation point.
        - Else if v is visited and v is not the parent of u:
            - Update `low[u]`: set `low[u]` to the minimum of `low[u]` and `dfn[v]`.