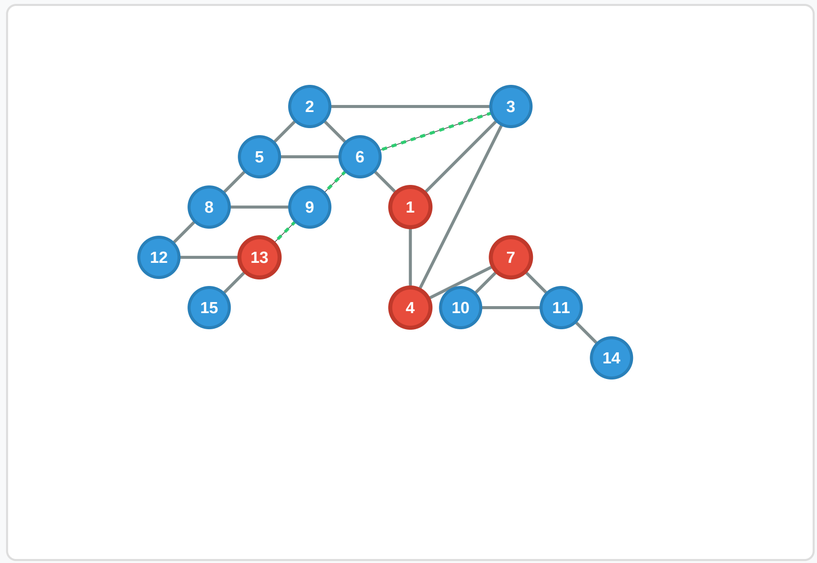
Advanced Graph Theory Problem

Network Vulnerability Analysis

Scenario: You are a network security analyst tasked with analyzing the vulnerability of a computer network. The network is represented as an undirected graph where nodes represent servers and edges represent direct communication links.



Regular Server | Critical Server (Articulation Point) | Regular Connection | Critical Connection (Bridge) | Back Edge (DFS)

📋 Problem Requirements

Given the network topology above, you must perform a comprehensive vulnerability analysis:

1. **DFS Analysis:** Perform DFS traversal starting from node 1 in lexicographic order and compute discovery times, finish times, and low values for all nodes.
2. **Critical Point Identification:** Identify all articulation points (critical servers whose failure would disconnect the network).
3. **Bridge Detection:** Find all bridges (critical connections whose failure would increase the number of connected components).
4. **Strongly Connected Components:** Although this is an undirected graph, identify and analyze the 2-edge-connected components.
5. **Network Resilience Score:** Calculate a resilience score based on the formula:
   `Score = (Total Nodes − Articulation Points) / Total Nodes × (Total Edges − Bridges) / Total Edges × 100`
6. **Attack Simulation:** Determine the minimum number of nodes that need to be compromised to disconnect the network completely.
7. **Recovery Planning:** Suggest additional edges that could be added to eliminate all articulation points and bridges.

🔍 Algorithm Steps

Step 1: Implement Tarjan's algorithm for finding articulation points and bridges during DFS traversal.

Step 2: For each node v, maintain: discovery[v], finish[v], low[v], and parent[v].

Step 3: A node u is an articulation point if:

- u is root of DFS tree and has more than one child, OR
- u is not root and has a child v where low[v] ≥ discovery[u]

Step 4: An edge (u,v) is a bridge if low[v] > discovery[u] (where v is a child of u in DFS tree).

Step 5: Use Union-Find or additional DFS to identify 2-edge-connected components.

Step 6: Calculate network metrics and generate recommendations.

📊 Solution Template

Node	Discovery Time	Finish Time	Low Value	Parent	Is Articulation Point?
1	_	_	_	NIL	_
2	_	_	_	_	_
3	_	_	_	_	_
4	_	_	_	_	_
5	_	_	_	_	_
6	_	_	_	_	_
7	_	_	_	_	_
8	_	_	_	_	_
9	_	_	_	_	_
10	_	_	_	_	_
11	_	_	_	_	_
12	_	_	_	_	_
13	_	_	_	_	_
14	_	_	_	_	_
15	_	_	_	_	_

Bridges Found: _____

Articulation Points: _____

2-Edge-Connected Components: _____

Network Resilience Score: _____

Minimum Nodes to Disconnect Network: _____

⚡ Complexity Analysis

Time Complexity: O(V + E) where V is the number of vertices and E is the number of edges.

Space Complexity: O(V) for the DFS stack and auxiliary arrays.

Why this complexity? Each vertex and edge is visited exactly once during the DFS traversal, and all additional operations (checking articulation points, bridges) are done in constant time per vertex/edge.

💡 Hints & Advanced Considerations

- **Root Articulation Point:** The root of the DFS tree is an articulation point if and only if it has more than one child in the DFS tree.
- **Bridge vs Back Edge:** An edge (u,v) is a bridge if there's no back edge crossing it. Use low values to detect this.
- **Network Hardening:** To eliminate articulation points, add edges to create alternative paths. To eliminate bridges, ensure every edge is part of a cycle.
- **Real-world Application:** This analysis is crucial for network reliability, finding single points of failure, and planning redundant connections.
- **Extension:** Consider edge weights representing bandwidth or reliability for more realistic network analysis.