ASSIGNMENT #3

To determine whether an undirected graph is a tree, there are two main properties:

1. **Connectivity**: The graph must be connected. This means there should be a path between every pair of vertices.

2. **Acyclicity**: The graph must not contain any cycles.

A graph that satisfies both conditions is a tree. We can check for these properties:

Methodology:

1. **Check for Acyclicity**: Use Depth-First Search (DFS) or Breadth-First Search (BFS) to ensure the graph doesn't contain a cycle. If you visit a node that has already been visited and is not the direct parent of the current node, a cycle exists.

2. **Check for Connectivity**: After the DFS or BFS, check if all vertices were visited. If any vertex was not visited, the graph is not connected.

Implementation:

1. Start DFS or BFS from any vertex.

2. Mark visited vertices.

3. If you encounter an already visited vertex (that is not the direct parent), a cycle exists.

4. After completing DFS or BFS, check if all vertices were visited. If not, the graph is not connected.

Running Time:

- **DFS/BFS**: O(V + E), where V is the number of vertices and E is the number of edges. This accounts for visiting every vertex and traversing every edge once.

- **Checking for unvisited vertices**: O(V), but this can be incorporated into the DFS/BFS step.

Overall, the running time to determine if an undirected graph is a tree is O(V + E).