Assignment 3

To determine whether an undirected graph is a tree, we can use **depth-first search (DFS) or breadth-first search (BFS)** and check for the following conditions:

Conditions for a Graph to be a Tree:

Connectedness: Every node must be reachable from every other node. This implies that the graph is connected.

Acyclic: There should be no cycles in the graph.

N-1 Edges: The number of edges in the graph should be one less than the number of nodes (n-1), ensuring that the graph is connected and acyclic.

Algorithm:

1. Perform a DFS or BFS starting from any node.
2. Check if all nodes are visited during the traversal. If not, the graph is not connected, and it cannot be a tree.
3. Check if there are any back edges (edges leading back to already visited nodes). If any back edge is found, the graph has a cycle, and it cannot be a tree.
4. Count the number of edges visited during the traversal. If the count is not equal to n-1 (where n is the number of nodes), the graph cannot be a tree.

Running Time:

The running time of this algorithm is O(V + E), where V is the number of nodes (vertices) and E is the number of edges in the graph. Both DFS and BFS have linear time complexity for undirected graphs.

This is because every node and every edge is visited once during the traversal, and the adjacency list representation of the graph allows for efficient access to neighbors, the algorithm is efficient and has a linear runtime, making it suitable for checking whether an undirected graph is a tree or not.