Daily Coding Problem #182

Problem

This problem was asked by Facebook.

A graph is minimally-connected if it is connected and there is no edge that can be removed while still leaving the graph connected. For example, any binary tree is minimally-connected.

Given an undirected graph, check if the graph is minimally-connected. You can choose to represent the graph as either an adjacency matrix or adjacency list.

Solution

Suppose we have a graph that is not minimally-connected. That means that there exists an edge (u, v) in the graph that, if we remove it, would still have a path from u to v. This means that there must be a cycle in the graph along u and v. Conversely, if there is no cycle, then the graph is minimally-connected. We can run DFS on the graph to detect a cycle, and return false if there is one.

However, an easier way to look at this is to simply notice that each vertex must have exactly one edge coming from it, which means that there must be `n - 1` edges in the graph. So we can also simply count up the number of edges in the graph and check if it's equal to `n - 1`.

```python
def minimally_connected(graph):
    n = len(graph)
    num_edges = 0
    for v, edges in graph.items():
        num_edges += len(edges)
    num_edges //= 2
    return n - 1 == num_edges
```

This takes O(V + E) time.