Week 3 Exercise

A. Give a linear-time algorithm in pseudocode that takes as input a directed acyclic graph G (V, E) and two vertices s and t, and returns the number of simple paths from s to t in G. No need to list the simple paths; just count them.

B. Using the algorithm of part **A** on the graph to the right, how many simple paths exist from vertex p to vertex x? List these paths.

C. Is the path between two vertices in a minimum spanning tree necessarily a shortest path between the two vertices in the full graph? Give a proof or a counterexample.

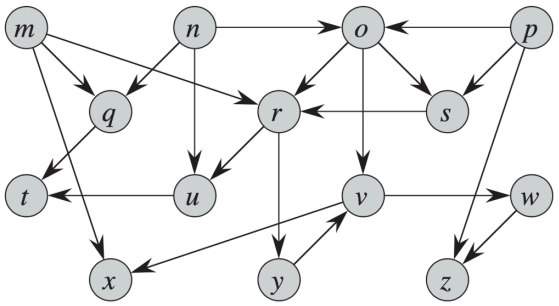
Answers

A. Traditionally, there are only BFS and DFS that run in $O(V + E)$ on a graph. We will attempt to do it on DFS. The idea is to count the number of visit that the destination receives. If we have a dictionary to keep track of the the number of ways that a vertex can reach the destination, and we have the destination's one to be 1 only when it is first visited, We can do the count.

Additionally, the idea is be adapted to run using dynamic programming, but it would require a topological sort before performing dynamic programming, which would require either DFS or BFS.

```
def dfs_to_count_simple_paths(graph, source, destination):
    # define global a dictionary to keep track of
    # the number of path to the destination
    path_count_dict = {v:0 for v in G.V}

    # define a set to keep track of the visited vertices
    visited = set()

    # Define dfs function at vertex
    def dfs(vertex):
        # check for the first time it is visited from the source
        if vertex == destination:
            # if it is the destination, have it as one path
            path_count_dict[vertex] = 1

        # add vertex to visited
        visited.add(vertex)

        # loop through all vertex's neighbor
        for u in graph[v]:
            # check if the neighbor has already been visited
            if u not in visited:
                # call dfs on neighbor
                dfs(u)

            # when a neighbor is already visited
            # add the number of ways it can visit destination
            # to the vertex
            path_count_dict[vertex] += path_count_dict[u]

    # run dfs from the source
    dfs(source)
```

B. On the given graph, how many simple path from p to x




```
Number of path from p to x is 4
The paths are:
p -> o -> r -> y -> v -> x
p -> o -> s -> r -> y -> v -> x
p -> o -> v -> x
p -> s -> r -> y -> v -> x
```

C. The path between two vertices in a minimum spanning tree IS NOT necessarily the shortest path between the two vertices in the full graph.

In this graph, the MST is $A \rightarrow B \rightarrow C \rightarrow D$, with the minimum total weight of 5. However, the shortest path to D is $A \rightarrow D$ with the weight of 4.