Num Paths

This algorithm goes through BFS and gives every vertex 3 associated values: discovered (boolean), depth (int), and pathsTo (int). Depth is the minimum number of edges from the start vertex, and pathsTo is the amount of shortest paths to get to the vertex from the start.
A vertex is marked as discovered when it is found for the first time, it is then assigned a depth value equal to the vertex whose list it was found in first (parent) +1. It's pathsTo is instantiated as equal to its parent's pathsTo value. **However** when the same vertex is found in another vertex's neighbor list, it will increment its pathsTo value by the amount of the discovering vertex **only if** the discovering vertex's depth is equal to its parent's depth.

Pseudocode

```
findMinPaths(graph, start, end):
    queue ={start}
    depthBeforeEnd = INT_MAX
    currentDepth = 0;

    while( currentDepth <= depthBeforeEnd ):
        parent = queue.pop()
        for every v neighboring parent:
            if (!v.seen):
                queue.push(v)
                v.seen = true
                v.depth = parent.depth + 1
                v.pathsTo = parent.pathsTo

                if (v==end):
                    depthBeforeEnd = parent.depth

            else if (v.depth == parent.depth+1):
                v.pathsTo += parent.pathsTo

    return end.pathsTo
```

Proof of Correctness

The amount of possible shortest paths to a given vertex is equal to the amount of shortest paths to all of its neighbors. Because the breadth first approach guarantees to find the shortest path to every vertex it encounters along the way, and will traverse all vertices its attached to this algorithm is guaranteed to find the amount of shortest paths.

Running time estimate

BFS with adjacency list: O(m+n)