Problem 3

To find the maximum number of shortest paths, we can use a modified BFS approach since it naturally finds the shortest path distance.

Initialization

We initialize three arrays that loop through each vertex $v \in V$:

- A distance array $dist[v] = \infty$ that stores the shortest distance from the source s.
- A path array $paths[v] = 0$ that stores the number of shortest paths from s.
- A Boolean array $visited[v] =$ false that stores whether or not the vertex has been visited.

Given the source s, we set $dist[s] = 0$ and $paths[s] = 1$. We then enqueue s into a queue Q.

BFS Traversal

While Q is not empty, we dequeue a vertex u. For each neighboring vertex v of u:

1. If v is undiscovered (i.e., $dist[v] = \infty$), we have found a shortest path to v. We set:
$$dist[v] = dist[u] + 1$$
$$paths[v] = paths[u]$$
We then enqueue v into Q.
2. If v is discovered (i.e., $dist[v] \neq \infty$) and $dist[v] = dist[u]+1$, this represents another valid shortest path to v via u. We update:
$$paths[v] = paths[v] + paths[u]$$
3. If $dist[v] < dist[u] + 1$, we do nothing.

Finding the Maximum

Once the queue is empty, we iterate through the *paths* array and return the vertex with the greatest path value.

Time Complexity

All vertices are enqueued and dequeued once, and each edge is only explored once. Therefore, the time complexity is $O(V + E)$.