**Level 2:**

**Problem Statement:**

To find the number of shortest paths available in a directed graph. If the given directed graph contains a negative or a zero cycle then there is no simple shortest path to a particular vertex from the source and so finding the number of simple shortest paths is impossible. In these cases such cycle needs to be returned.

**Approach:**

Run Bellmanford’s shortest path algorithm on the given graph to find the shortest distance to that particular vertex from source. This algorithm also helps in finding the negative cycle. In case of negative cycle, walk-back on a directed graph through its parent is done to detect such cycle.

If the Bellmanford’s algorithm gets executed successfully without a negative cycle, then the number of shortest path to a particular vertex is found using the distance field and the reverse-adjacency list of edges. All the edges that are a part of shortest paths are marked valid and a subgraph is considered only with vertices that are reachable and the vertices marked valid. [The edges contributing to positive cycle will not be part of the valid set of edges as they are not part of shortest paths.]

Next step is to check for zero cycle in the graph. This is done by topological ordering. The topological ordering shows if there is cycle.

To detect a cycle, DFS is done on the subgraph considering only valid edges. If a particular vertex is pushed inside the recursion stack inspite of it being in stack already, then the cycle exists. Then pop the vertices of recursion stack to form the cycle.

If there is no zero cycle, then number of shortest paths to a particular vertex from source is calculated based on the topological ordering of vertices.

**Conclusion:**

The assigned task helps us to realize how can we make us of existing algorithms to solve a problem that has direct solution to it and gave good insights on using proper data structures to complete the required task.