CS5800: Algorithms Spring 2018
Assignment 6.5

Saptaparna Das

April 2, 2018

Algorithm:

For this problem, we are given a set categories denoting the order in which they should appear, if possible.
Let categories be an list of pairs where first element in the each pair should come before second.
For example income questions should appear before deduction questions if possible. The modified algorithm to incorporate this list, is as follows:

The graph is formed by maintaining array of linkedlist data structure where when a new edge is added (i.e. the pair of dependent questions), first question is added to the array and the dependent one is added to its corresponding linkedlist.

Let result be a list to hold the final sequence of vertices.
Let indegree be an array to store indegree of a vertex.
Let queue be a double ended queue to store the nodes to be processed. This data structure will support insertion deletion from both ends. This helps in processing all nodes as well as gives priority to specific nodes when necesary.
Basically, the idea is to check for the categories list whenever we are adding a new node to queue. If the node appears as second element in an pair, we add the node to back of the queue, so that its processed after the nodes which also has indegree zero but has no dependency.

```
Step 1: For every vertex v, do the following:
        Step a: Get all it's adjacent nodes.
        Step b: For all such nodes, increase corresponding count
                   in indegree array by 1.

Step 2: Check all vertex. If a vertex has indegree 0,
               Step 2.1: If that vertex doesn't appear as a
                            second element in any pair of categories,
                            add it to front of queue.
               Step 2.2: Else add it to back of queue.

Step 3: Initialized processedNodes to 0.

Step 4: Until q is not empty, do the following:
        Step 4.a: Remove element from front of the queue.
        Step 4.b: Add to result list.
        Step 4.c: For all it's adjacent nodes, reduce the corresponding
                  indegree by 1. If the indegree becomes zero for any
                  node,
                  Step 4.c.1: If that vertex doesn't appear as a
                              second element in any pair of categories,
                              add it to front of queue.
                  Step 4.c.2: Else add it to back of queue.
        Step d: Increament processedNodes by 1.

Step 5: If processedNodes doesn't match total number of vertices
         in the graph, then print topological order is not possible

Step 6: Print nodes from result list.
```

analysis: To calculate the indegree we need to visit every vertex and for each vertex we will traverse the edges. since it is represented as adjacency list, the time taken should be $O(|V| + |E|)$ Space complexity also should be $O(|V| + |E|)$ to store the graph and additional queue of size V. so total space complexity $O(|V| + |E|)$

If the ordering of the categories doesn't matter, i.e. if we just have to group the subprocess if possible, then in the Step 2 and Step 4.c, we can check for category (Vertex) to add to the queue, instead of checking categories list. That is, if the adjacent node is of same category as parent node, it is inserted to front of queue else it is inserted to the back of the queue.