CS208 Assignment 3

Author: 王超逸 Chaoyi WANG

SID: 11811014

Content: Answers of q1 and q3 of chapter 3 of the textbook.

3.1

Q: Consider the directed acyclic graph G in Figure 3.10. How many topological orderings does it have?

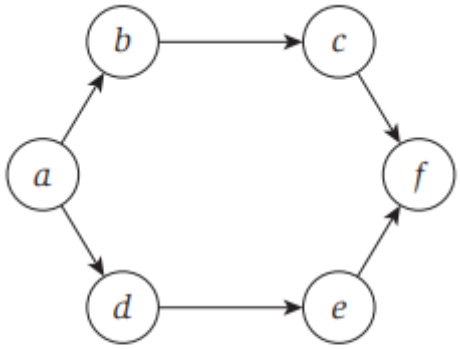

Figure 3.10 How many topological orderings does this graph have?

A: First we try to find the nodes with arrows outwards only, which is node a.

Eliminate a, we got b and d. Eliminate b and d, we got c and e correspondingly. Finally we got f.

Therefore, there are only **six** topological orderings: abcdef, abdecf, abdcef, adbecf, adbcef and adebcf.

3.3

Q: The algorithm described in Section 3.6 for computing a topological ordering of a DAG repeatedly finds a node with no incoming edges and deletes it. This will eventually produce a topological ordering, provided that the input graph really is a DAG. But suppose that we're given an arbitrary graph that may or may not
be a DAG. Extend the topological ordering algorithm so that, given an input directed graph G, it outputs one of two things:

(a) a topological ordering, thus establishing that G is a DAG; or

(b) a cycle in G, thus establishing that G is not a DAG. The running time of your algorithm should be O(m + n) for a directed graph with n nodes and m edges.

A: The original algorithm finds and eliminates a node with only arrows outwards and its arrows in each iteration, until such nodes cannot be found. We can apply some little modification to the algorithm to achieve the goal described in the question:

We allocate a boolean array indicating whether each node has been traversed in advance. In each iteration, we try to find a node with only arrows outwards and eliminate it as the original algorithm does. However, if we cannot find such a node, we obtain a random node then. The difference is that, instead of eliminating the node and its arrows, we keep it and label it as traversed in the boolean array and then go to find its successors in the same way. Since the graph must contain a circle, we must find that, at some moment, a successor has been labeled traversed, and in this way we can output the complete cycle.

In each iteration of this modified algorithm, there is one node eliminated or labeled, so the running time is O(n), which is in the range of O(m+n).