Assignment3

Question1

Consider the directed acyclic graph G in Figure 3.10. How many topolog□ical orderings does it have?

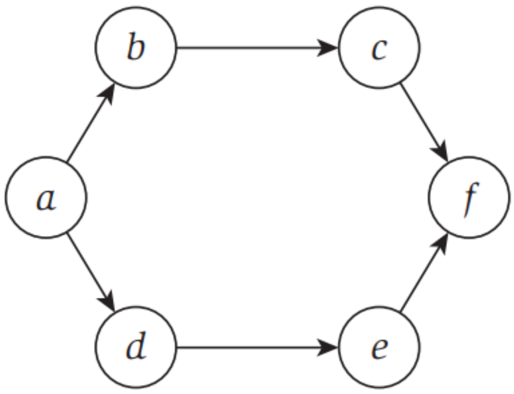

Figure 3.10 How many topological orderings does this graph have?

There 6 topological orderings this graph have

1. a b d c e f
2. a b d e c f
3. a b c d e f
4. a d b c e f
5. a d b e c f
6. a d e b c f

Question3

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.

Here is the pseudocode of the judgement:

```
// G: A directed graph
DAG_Judge(G){
    Create queue Q;// for topogical order
    Create ArrayList A;// record the topogical answer
    // O(n)
    for (all Nodes in G){
        if (there is not nodes which in-degree is 0){
            return("Not a DAG");
        }else{
            if(node in-degree == 0){
                Q.add(node);
            }
        }
    }
    // O(n + m)
    while(Q is not null){
        node = Q.pop();// will probably pop n nodes
        A.add(node);
        count += 1;
        // will probably pass m edges
        for(outnode in all the outnodes of node){
            outnode in-degree -= 1;
            if(outnode in-degree == 0){
                Q.add(outnodes);
            }
        }
    }
    // Judgement
    if(A.size() == G node number){
        return(A);
    }else{
        return("Not a DAG");
    }
}
```