Algorithms and Complexity: Homework #3

Due on February 15, 2019 at 3:10pm

Professor Bradford

Patrik Sokolowski

Problem 1

Chapter 3, Exercise 1.

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

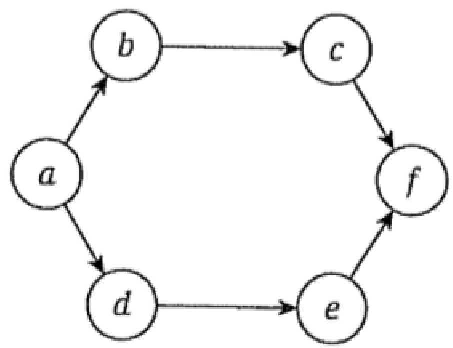

Figure 3.10 How many topological orderings does this graph have?

Picture Provided by Textbook (Algorithm Design by John Kleinberg)

6 Topological orderings:

a,b,c,d,e,f
a,b,d,c,e,f
a,b,d,e,c,f
a,d,e,b,c,f
a,d,b,c,e,f
a,d,b,e,c,f
(Topological Sort)

Problem 2

Chapter 3, Exercise 2.

Give an algorithm to detect whether a given undirected graph contains a cycle. If has a cycle it should output one. Running time shoud be O(m+n), n nodes and m edges.

Run BFS algorithm on an arbitrary node s, resulting in a list/tree T.
. If every edge of L (our list/tree) appears in the BFS tree, then L = T.
. so L is a list/tree which has no cycles.
. Otherwise there is an edge e=(v,w) that belongs to L but not T.
. Considering the least common ancestor u of v and w in T.
. We get the cycle from the edge e, together with u-v and v-w paths in T.

We can start with either BFS or DFS which would change our algorithm slightly depending on which is chosen.

Problem 3

Chapter 3, Exercise 3.
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 the G is a DAG, or (b) a cycle in G, thus esablishing that G is not a DAG. Running time should equal O(m+n) with n nodes and m edges.

The inductive proof contains the following algorithm to compute a topological ordering of G.

```
To compute a topological ordering of G:
Find a node v with no incoming edges and order it first
Delete v from G
Recursively compute a topological ordering of G−{v}
  and append this order after v
```

Picture Provided by Textbook (Algorithm Design by John Kleinberg)

Run BFS algorithm on an arbitrary node s, resulting in a list/tree T.
. If every edge of G (our list/tree) appears in the BFS tree, then G = T.
. so G is a list/tree which has no cycles.
. Return topological ordering of G (running topological algorithm above) establishing G is not a DAG.
. Otherwise there is an edge e=(v,w) that belongs to G but not T.
. Considering the least common ancestor u of v and w in T.
. We get the cycle from the edge e, together with u-v and v-w paths in T.
. Return that cycle, establishing G is not a DAG.

Problem 4

Chapter 3, Exercise 5.
SHow by the induction that in any binary tree the number of nodes with two children is exactly less than the number of leaves.

Proof by Induction:

. **Base Case:** root node with 0, 1, or 2 leaves
. Consider the root has no children, then it is a leaf and the tree has 1 leaf with no node with 2 children.
. Consider the root has a single leaf, then again the tree has 1 leaf anf no nodes with 2 children.
. Consider the root has 2 children that are leafs, then the tree has 1 node with 2 children and 2 leafs.

. **Inductive hypothesis:** T is a binary tree with 'n' nodes.
. The number of nodes with 2 children is exactly 1 less than the number of leaves.

. **Inductive step:** consider both cases:
. A parent with only 1 child
. A parent that is a leaf
. The number of leaves and nodes with 2 children both increment by 1.
. Proving I.H. to be true.

Problem 5

Chapter 3, Exercise 8.

claim: There exists a positive natural number c so that for all connected graphs G, it is the case that: $\frac{diam(G)}{apd(G)} \leq c.$

false!

Consider constant k,
Take a path to k-1 nodes, $\mathrm{u}_1, u_2, ..., u_{n-1}$
Attach additional n-k+1 nodes (attaching an edge to u)
. $\mathrm{v}_1, v_2, ..., v_{n-k+1}$
diameter of G now equals dist($\mathrm{v}_1, v_{k-1}) = k$
pairing both lists makes their distance $\leq$ k.
If we choose n-1 $> 2\mathrm{k}^2$ *we get apd(G) lowerthan* 3.
Given k $>$ 3c :
$\frac{diam(G)}{apd(G)} > \frac{3c}{3} =$ c
Which disproves the claim.