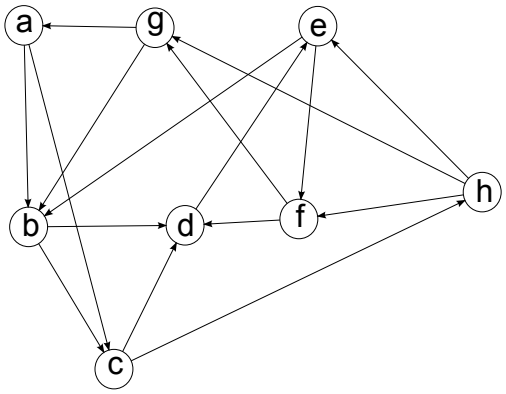
CS 473: Fundamental Algorithms, Spring 2013

Discussion 2

January 24, 2013

2.1. DFS example.

Consider the following graph.



Draw the **DFS** tree rooted at d for the above graph. Use alphabetic ordering to break ties. Label the vertices of the tree with their $\text{pre}(v) : \text{post}(v)$ time. Add in the remaining edges of the graph and label them as forward (F), backward (B), and cross (C) edges. Sort the vertices by their post visit order.

2.2. Post numbering of SCCs has to be done carefully.

Let G be a directed graph and consider a specific depth first search of G. Let S and S' be strong connected components in G. Suppose (S, S') is an edge in G^{SCC} we saw in lecture that $\text{post}(S) > \text{post}(S')$ (recall that $\text{post}(S)$ was defined as $\max_{u \in S} \text{post}(u)$). However it is not true that for every $u \in S$ and $u' \in S'$, $\text{post}(u) > \text{post}(u')$. Describe an example to show that this claim is not true.

2.3. Prove/disprove: Smallest post number implies a vertex is in the sink.

Let G be a directed graph and G^{SCC} its strong connected component meta-graph (which is a DAG). Prove or disprove the following. For any **DFS** of G the vertex with smallest post-visit number is in a sink component of G^{SCC}.

2.4. No no no! I want two cycles, not one.

Let $G = (V, E)$ be an undirected graph with n vertices ($|V| = n$) and m edges ($|E| = m$). Give an $O(n)$ time algorithm to check if G has at least *two* distinct cycles and output them if it does. Assume that the graph is represented using adjacency lists. Note that m can be much larger than n so the algorithm should not check all edges. *Hint:* What is the structure of a minimal connected graph that has two cycles? Use **DFS**.

2.5. Lights on!

There are n light bulbs in a garden. These bulbs can be turned on manually by flipping on the switches at the light posts. Also, each light post can broadcast turn-on signals to some other pre-defined light posts in the garden, turning them on. When a light post is turned on, it will automatically broadcast a turn-on signal to its pre-defined light posts.
This signal broadcasting is directional. If a broadcasts to b, it is not necessarily true that b also broadcasts to a.
So one can manually flip on some of the switches to the light posts, and those light posts will broadcast a turn-on signal to other light posts. These will in turn be switched on and broadcast signals to their own pre-defined set of light posts, and so on.
Given each light post in the garden and the respective light posts to which they broadcast, derive a linear time algorithm for finding the minimum number of switches needed to be flipped to light up the whole garden. (Linear time means $O(n + m)$ where n is the number of light posts and m is the number of broadcast associations between them).

Source: *ACM ICPC 2010 World Finals Warmup 2*

Example Case: Number of lights : 5 , Number of broadcast associations: 4
Associations : 1=>2, 1=>3 3=>4, 5=>3
Answer: Minimum number of flips required : 2 ,
Turning on switches 1 and 5 should light up the whole garden
Hints:

(I) Model the problem using directed graphs.
(II) What is the solution if the graph in question is strongly connected?
(III) What is the solution in general?

2.6. Topological Ordering.

(A) Given a DAG, how can one decide if it has only a single topological ordering that is consistent with it?
(B) Given a DAG you are given an access to an oracle that can tell you for any two vertices of the graph, in which order they appear in the "true" topological ordering of the vertices of the graph. Give an algorithm that using this ***comparison*** oracle, add edges to the graph such that the resulting DAG has a unique topological ordering of its vertices.
(C) What is the minimum number of edges one has to add to a DAG to make it have only one unique topological ordering? Show how to compute these edges using only $O(n \log n)$ calls to the comparison oracle.
(D) (Harder.) Show an algorithm that calls the oracle $O(k^2)$ times if one can add just k edges and make the DAG have a single unique topological ordering.