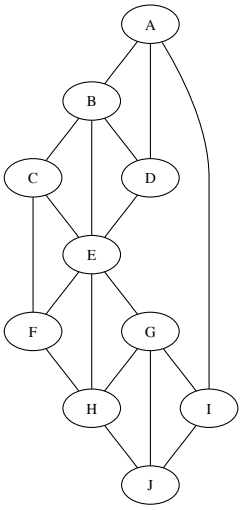
CS255, Spring 2014, SJSU
Homework 7

Fernando Lobo

Due: May 7, 2014

Problem 1



Suppose we run DFS on the undirected graph above, starting on vertex A. Also, assume that when the algorithm explores the neighbors of a given vertex, it does so in alphabetic order. Trace the execution of DFS by writing the discovery and finishing time for every single node in the graph. (**Note: there won't be partial credit in this question. You'll either get full score or no score, so check your solution very carefully.**)

Problem 2

What's the running time of DFS on a graph $G = (V, E)$ represented by an adjacency matrix? Justify your answer.

Problem 3

Prove that every Directed Acyclic Graph (DAG) has at least one vertex with no incoming edges. (Assuming of course that the DAG is non-empty, i.e., it has a least one vertex.)

Problem 4

The statement in Problem 3 suggests that we can find a topological sort of a DAG by finding a vertex with no incoming edges, and deleting it from the graph. Then repeat this procedure until the graph is empty. The order in which the vertices are deleted constitute a topological sort of the nodes. Show how you can implement this algorithm in linear time? What happens to your algorithm if the input graph has a cycle?

Problem 5

Suppose you want to find a maximum spanning tree of a weighted connected undirected graph $G = (V, E)$. In other words, you want a to find a spanning tree whose weight is as large as possible. Specify an algorithm to solve this problem and give its running time. Justify your answer. (You can give your answer in plain english, without any pseudocode.) [Hint: think about transforming the input graph into some other graph, and then apply an algorithm that you already know.]

Problem 6

You are given a weighted connected undirected graph $G = (V, E)$ with $|E| = |V|$. Suppose that among all the edges of the graph, there is an edge (u, v) whose weight is strictly larger than any other edge weight of the graph. Can (u, v) belong to some minimum spanning tree of the graph G? Justify your answer.

Problem 7

The Set Cover Problem is defined as follows. Given a set U of n elements, a collection $S_1, \ldots, S_m$ of subsets of U, and an integer k, does there exist a collection of at most k of these sets whose union is equal to U?

For example, suppose $U = \{1, 2, 3, 4, 5\}$, the collection of subsets of U is $\{1, 2, 3\}$, $\{2, 4\}$, $\{3, 4\}$, $\{4, 5\}$, and $k = 2$. This particular instance of the Set Cover Problem has a YES answer because the union of $\{1, 2, 3\}$ with $\{4, 5\}$ gives the set U.

(a) The above description is formulated as a *decision problem*, i.e., as a problem returning YES or NO. Suppose you had access to an algorithm that solves the Set Cover Problem as formulated above. How could you solve the corresponding optimization problem? In other words, the problem of finding the fewest number of subsets whose union is U?

(b) Prove that the Set Cover Problem is NP-Complete.
(**Hint: reduce Vertex Cover to Set Cover.** Given an instance of Vertex Cover specified by a graph $G = (V, E)$ and an integer k, transform it into an instance of Set Cover by making the set of edges E correspond to the set U, and for each vertex $i \in V$ add a set S_i to the collection of subsets of U where each S_i consists of all the edges in G that are incident on vertex i.)