

CS 535 Homework 9

Due: April 29 (Th), in class.

NOTE: Throughout this HW, you're asked to design a graph algorithm that takes *linear time*. What this means is if the input is a graph G with n vertices and m edges, then the running time of the algorithm should be $O(n + m)$ – that is, the running time is linear in the parameters n and m .

1. Consider the graph on page 289 of your book. Assume that this graph is represented using adjacency lists. Furthermore, for each node v , its adjacency list contains the nodes adjacent to v arranged in alphabetical order. For example, for the node labeled Garg, its list consists of Goodrich and then Tamassia.
 - a. Do a depth-first search on the graph starting at Garg. Make sure that you label each edge as a discovery edge or a back edge. Also show the resulting DFS tree.
 - b. Similarly, do a breadth-first search on the graph starting at Garg, labeling each edge as a discovery or cross edge. Output the BFS tree as well.
2. Suppose a CS curriculum consists of n courses, all of them required. As you know by experience, you typically can't just take any course you want because some courses have prerequisites. To better understand the sequencing of the courses, you decided to create a *directed* graph G which has a vertex for each course, and an edge from course v to course w if and only if v is a prerequisite for w .
 - a. Try this out! Using the Computer Science curriculum sheet (you can find one from the CS website), create the prerequisite graph for the *required* CS courses (it's the first group of courses). Ignore prerequisites like "junior status" or Math 211; that is, only consider the prerequisites that are CS courses themselves.
 - b. Given a general prerequisite graph G (not just the one you constructed from part a), design an algorithm that computes the minimum number of semesters necessary to complete the curriculum (assuming a student can take any number of courses in one semester). The running time of your algorithm should be linear in the size of G .
3. Given an undirected graph $G = (V, E)$ and a particular edge e of the graph, design a linear time algorithm that determines whether or not G has a cycle containing edge e .
4. Given a directed acyclic graph $G = (V, E)$ and two vertices s and t . Design an algorithm that returns the number of paths from s to t in G in linear time.