Algorithms Lab 9

csci 2200, Laura Toma. Bowdoin College

In lab

1. Finish "Applications of BFS and DFS" handout.

2. Suppose you have an undirected graph given as an adjacency list, and you want to figure out if they form a *tree* (a tree is an undirected graph that's connected and has no cycles). How would you do it, and how fast?

3. The two-colorability problem from handout: Is it possible that the vertices of a given graph be assigned one of two colors, such that no edge connects vertices of the same color? (Note: this is equivalent to the question: is G bipartite?))

4. Suppose the degree requirements for a computer science major are organized as a DAG (directed acyclic graph), where vertices are required courses and an edge (x, y) means course x must be completed prior to beginning course y. Make the following assumptions:

   - All prerequisites must be obeyed.
   - There is a course, CPS1, that must be taken before any other course.
   - Every course is offered every semester (unlike our department).
   - There is no limit to the number of courses you can take in one semester (again, unlike our department).

   Describe an efficient algorithm to compute the minimum number of semesters required to complete the degree and analyze its running time.

Homework

1. (4.2.31 Sedgewick Wayne) In a directed graph, two vertices u and v are said to be in the same *strongly connected component (SCC)* if u can reach v and v can reach u.

   (a) Describe a linear time algorithm for computing the *strong component* containing a given vertex v.

   (b) On the basis of that algorithm, describe a simple quadratic time algorithm for computing the strong components of a directed graph G.

2. (CLRS 22.4-2) Give a linear-time algorithm that takes as input a directed acyclic graph $G = (V, E)$ and two vertices s and t, and returns the number of simple paths from s to t in G. For example, the DAG in Fig. 22.8 CLRS contains exactly four simple paths from p to v: pov, $poryv$, $posryv$ abd $psryv$. Your algorithms needs to only count the simple paths, not list then.

   Hint: dynamic programming on DAGs.

3. Assume you are given a DAG, and you want to compute "longest" paths; the edges do not have weights, and we use the sttandard convention that the length of a path is the number of edges on the path.

   (a) Describe how to compute the longest path in a DAG starting from a specified vertex.

   (b) Describe how to compute the longest path in a DAG.

   Hint: dynamic programming on DAGs.

4. (4.2.32 Sedgewick Wayne) (Hamiltonian paths in DAGs) Given a DAG, design a linear time algorithm to determine whether there is a directed path that visits each vertex exactly one.