CS140: Algorithms

Fall 2000

Assignment 12

This assignment is due by 5 PM on Friday, Nov. 10. You may collaborate with your classmates to complete this assignment but you must reference your collaborators. In addition, you must do your own write-up. This assignment is worth 100 points (75 points if handwritten).

Your answers will be judged for correctness, clarity, and conciseness. When you give an algorithm or proof, you should begin with a brief (a few sentences) overview that identifies the key ideas behind your approach. Then fill in the (important) details.

1. Let $G = (V, E)$ be a directed acyclic graph with n vertices and m edges. Design an algorithm that lists the vertices of G in an order such that if $< u, v >$ is an edge of G then u precedes v in the list. To receive full credit your algorithm should run in time $O(n + m)$.

2. Let $G = (V, E)$ be a directed graph with n vertices and m edges. The transitive closure of G, denoted G^T, is the graph with vertex set V and edge set

$$E^T = \{<u, w> \mid \text{there is a path from } u \text{ to } w \text{ in } G\}.$$

Let $d(u, w)$ denote the length of the shortest path from vertex u to vertex w in G. The diameter of G is defined to be

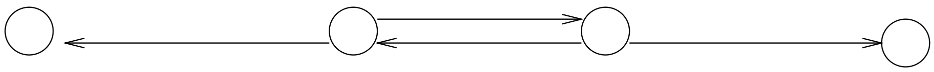
$$max\{d(u, w) | \text{ there is a path from u to v in } G\}.$$

An example of a graph and its transitive closure is shown in the figure on the next page. The diameter of the graph G in the figure is 2. What is the diameter of G^T?

   (a) Give an algorithm that takes as input the adjacency matrix M and diameter d of a directed graph G and outputs the adjacency matrix of G^T. To receive full credit your algorithm should run in time $O(d \cdot n^{\lg 7})$. You may assume that n is a power of 2. (Hint: Reduce the problem to a problem we've studied that has an algorithm with running time $O(n^{\lg 7})$.

   (b) Now suppose that G is acyclic. In this case we can assume, without loss of generality, that the adjacency matrix of G is upper triangular. Give an algorithm for finding the transitive closure of G in time $O(n^{\lg 7})$). (Hint: Divide and conquer)

G:



G^T: