CS140: Algorithms

Spring 2001

Assignment 6

This assignment is due by the start of class on Tuesday, Feb. 27. 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 (65 points if handwritten).

1. A universal sink in a directed graph is a node that has an in-edge from every other node in the graph and no out-edges.
   (a) Prove that a directed graph has at most one universal sink.
   (b) Suppose you are only allowed to access the input graph by queries of the from "Is there an edge from i to j?" Design an algorithm to determine if the graph has a universal sink. You may assume the graph has $n = 2^m$ nodes. You should only need $3n - m - 3$ queries in worst case.
   (c) Prove that $3n - m - 3$ queries are necessary by giving an adversary argument.
2. An *independent set* of a graph G is a subset W of the vertices of G such that no two vertices in W are connected by an edge in G. Finding a largest independent set of an arbitrary graph is NP-hard. The problem is solvable in polynomial time, however, if the input is restricted to be a forest.
   (a) Let F be a forest and let v be a leaf in F. Prove that v is in some largest set of F.
   (b) Suppose I give you a black box that will find a largest independent set of a forest with $n - 1$ or fewer nodes. Explain how to use this box to build an algorithm for finding independent sets in forests with n or fewer nodes. Be sure to describe any base cases. (How do you handle isolated nodes?)
   (c) How would you implement this algorithm? Carefully describe the data structures you'll use. You may specify the data structure used to represent the input. Analyze the running time of the algorithm implemented with your data structures.
3. In the *weighted independent set* problem, the input is a graph G in which each vertex is assigned a positive weight. The *weight* of an independent set W is the sum of the weights of the vertices in W. The objective is to find an independent set with largest weight. This problem is NP-hard but can be solved in polynomial time if we restrict the input to be a weighted tree.

(a) Give an example of a weighted tree T for which your algorithm from problem 1 fails to find a maximum weight independent set.

(b) Give an example of a tree T rooted at some node r such that the union of the maximum weight independent sets of the children of r is a maximum weight independent set T.

(c) Give an example of a tree T rooted at some node r such that the union of the maximum weight independent sets of the children of r is *not* a maximum weight independent set T. Give a maximum weight independent set of this tree.

(d) What information would you like about the subtrees of a rooted tree in order to determine the weight of a maximum weight independent set of the tree? (You should be able to do this with two functions on the nodes of the tree.)

(e) Design an algorithm that computes the functions you defined in the previous part.

(f) Show how to use this algorithm to determine the weight of a maximum weight independent set of a tree.

(g) Show how to modify your algorithms to compute a maximum weight independent set of an input tree.

(h) Briefly describe implementation details. Note that an $\Theta(n)$ algorithm is possible.