Algorithms
Computer Science 140 & Mathematics 168
Spring 2012
Homework 4a
Due Thursday, February 9

1. **[25 Points] The Millisoft Party Problem!** You've decided to accept a job as senior algorithm designer at Millisoft. One afternoon, Gill Bates comes to you with the following problem. "I'm throwing a company party," Gill says excitedly, "And I need your help! As you know, Millisoft has a hierarchical structure. You can think of it as a tree. The president, that's me, is at the root of the tree. Oh boy, I love being at the root!" You take a sip of your luke-warm diet coke (which Millisoft provides for free - what a perk!) and listen patiently as Gill continues. "Below the root are supervisors, below them are managers, below them are team leaders, etc., etc., until you get down to the leaves - the summer interns. The tree is not necessarily binary; some non-leaf nodes may have one "child", others two, and others even more. Anyhow, to make the party fun, I thought it best that we don't invite an employee along with their immediate boss (their parent in the tree). Also, I've personally assigned every employee a positive real number (actually it's an arbitrary precision floating point number, but nevermind that!) called their *coefficient of fun*. My objective is to invite employees so as to maximize the total sum of the coefficients of fun of all invited guests, while not inviting an employee with his or her immediate boss."

   (a) Describe a recursive algorithm for this problem in clear English or clear pseudo-code. Assume that the tree is represented as a collection of nodes with links from parents to children and also from children to parents. The tree is passed to you by giving you a pointer to the root.

   (b) Describe a DP algorithm for this problem. You may assume that each of the n nodes in the tree has a unique number between 1 and n associated with it. You may also assume that you have a function that will give you a linked list (or an array - whichever you wish) of all of the leaves in the tree. (This function simply uses depth-first search from CS 60 to locate all of the leaves. In case you've forgotten, the running time of depth-first search on a tree is $O(n)$, so this function is quick!) Finally, you may be asking yourself "what is the shape of this DP table?". It might be useful to make the tree itself the DP table! That is, you can just write values at the nodes of the tree rather than in an array.

(c) What is the asymptotic running time of your DP algorithm? Explain.

(d) How can you modify your algorithm to find the optimal solution in which Gill gets invited to his own party?

(e) After using your DP for several years, Gill Bates comes to your office one day and tells you: "This year, I just want to find the largest number of people that I can invite such that we never invite an employee and their immediate boss. This is analogous to every employee having 1 as their coefficient of fun." Of course, you could solve this problem with your DP, but Gill suspects that there is a greedy algorithm that would solve this problem optimally. In other words, your task is to take as input a tree representing the company hierarchy and compute the largest number of employees (nodes) that can be selected such that no two adjacent nodes (i.e. a node and its child) are chosen. Describe a *greedy algorithm* for this problem (in either pseudo-code or clear English), prove its correctness using strong mathematical induction, and give its worst-case running time. A greedy algorithm, in this case, is one that visits vertices one-by-one (in some order of your choosing) and makes binding decisions on whether or not to invite that vertex before moving on to the next vertex. (You might be tempted to have your greedy algorithm take a bunch of vertices at the same time. Doing so will make your proof of correctness harder. We strongly urge you to define your algorithm so that it chooses one vertex at a time.)