Algorithms Section 2
Spring 2003
Homework 5a
Due Thursday, February 20

1. **[20 Points] The Millisoft Party Problem!** After your success at Snapple, you've decided to accept a job as senior algorithm designer at the well-known Millisoft Corporation. One day, the President of Millisoft, 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. 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 real number (actually it's a double precision floating point, 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) The first algorithm that Gill thought up was to simply enumerate all possible subsets of his employees, throw out those subsets that include an employee and his or her boss, find the score for each remaining subset, and finally choose the best one. There are 1000 employees at Millisoft. Millisoft has also just purchased a Crayfish YMP that can process one trillion (10^{12}) subsets per second. How long will it take to find the optimal solution using this brute force approach on the Crayfish?

   (b) Describe in detail an efficient dynamic programming algorithm for this problem. What is the asymptotic running time of your algorithm?

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

2. **[10 Points] Greed Must be used with Extreme Caution!** In class we talked about the problem of finding the maximum number of non-conflicting courses from a given set of courses. We argued that if the courses are sorted in order of completion time, then the following greedy algorithm is guaranteed to find the maximum number of non-conflicting courses: Choose the course with earliest completion time. Cancel out all courses that conflict with that course. Now repeat the process for the remaining courses.

   Professor I. Lai of the Pasadena Institute of Technology has proposed the following four alternative greedy algorithms for this problem.

   Algorithm 1: Sort the courses by ending time as before. Now, run our original greedy algorithm in the opposite direction. That is, choose the course that ends **latest**.

Then cancel out all courses that conflict with that course. Now repeat the process for the remaining courses.

Algorithm 2: Sort the courses by increasing starting time (rather than ending time). Now, choose the course that starts first. Then cancel out all courses that conflict with that course. Now repeat the process for the remaining courses.

Algorithm 3: Forget about sorting the courses. Choose a course of shortest duration (that is the course that has the least length). Then cancel out all courses that conflict with that course. Now repeat the process for the remaining courses.

Algorithm 4: Don't sort the courses. Choose a course that conflicts with the fewest other courses, breaking ties arbitrarily. Then cancel out the courses that conflict with that course. Now repeat the process for the remaining courses.

Show that each of these greedy algorithms does not always find an optimal solution. (Showing this for Algorithm 4 is optional and is worth 10 Bonus Points.) To show that an algorithm does not always find an optimal solution, you need only demonstrate a specific example and show that the given algorithm could select a particular solution which is not the best one.