Algorithms
Computer Science 140 & Mathematics 168
Instructor: B. Thom
Spring 2004
Homework 7b
Due on 3/9/04 (beginning of class)

1. [**30 Points**] **Optimizing Party Fun!**

   After your success at Snapple, you've decided to accept a job as senior algorithm designer at the well-known My-I'm-Soft Corporation. One day, the President of My-I'm-Soft, 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, My-I'm-Soft 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 My-I'm-Soft 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 never-mind 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 soft. My-I'm-Soft 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 an efficient dynamic programming algorithm for this problem. Make sure to describe it in enough detail that it is clear how to fill in the values (no strategy, data structure, or approach is off limits). What is the asymptotic running time of your algorithm?

   (c) How can the program be changed to make sure Gill is invited to his own party?

   (d) Develop a greedy algorithm for the case when all coefficients of fun are equal to one and argue its correctness. (P.S. Don't worry if your greedy and DP algorithms have the same asymptotic run-time, although it is worth convincing yourself that you understand why, in this problem, this is the case!).

   (e) Provide a counter-example where the greedy choice would fail when all coefficients were not necessarily the same value.

2. [**25 Points**] **Detecting Cycles!**

(a) Consider an *undirected graph* with V vertices and E edges. The graph may be disconnected. That is, the graph may consist of a number of separate disconnected components. Describe how depth-first search can be used to determine whether or not a graph has a cycle in linear—i.e. $O(V + E)$—time. You should assume that the graph is represented by an adjacency list. Prove your algorithmic modification is correct—why it will detect some cycle in the graph (if and only if it exists). Note: if a graph is disconnected, the algorithm should return true when at least one of its components has a cycle. Finally, analyze its runtime, demonstrating that it has the desired asymptotic behavior. (This problem might require a bit of cleverness in the algorithm's data structures in order to get the $O(V + E)$ time!)

(b) Does your your algorithm necessarily work on *directed* graphs?

(c) Does your above algorithm work if you use breadth-first search instead of depth-first search? Explain briefly.

(d) Show how the topological sorting algorithm can be used to determine whether a *directed* graph contains a cycle (the graph may be disconnected). The algorithm should work in $O(V + E)$ time assuming the graph is represented as an adjacency list. Again, explain your algorithmic modification, justify its correctness, and analyze its runtime.

3. **[25 Points] Diameters.**

In this problem we will consider connected undirected graphs with no cycles. The graphs are simply represented with adjacency lists. The distance between two vertices in such a graph is just the *minimum* number of edges on a path between the vertices. The *diameter* of the graph is the *maximum* distance between all pairs of vertices. *Make sure that you understand this definition well before proceeding!*

Describe an algorithm that utilizes one of the reachability algorithms discussed in class to compute the diameter of a connected undirected acyclic graph in $O(V)$ time. You should justify why your algorithm is correct and carefully analyze and explain its running time.