Dynamic Programming Practice Problems

Problem 1: Planning a Company Party

You are consulting for the president of a corporation that is planning a company party. The company has a hierarchical structure; that is, the supervisor relation forms a tree rooted at the president. The personnel office has ranked each employee with a *conviviality* rating, which is a real number. In order to make the party the most fun for all of the attendees, the president does not want both an employee and his or her immediate supervisor to attend.

You are given the tree that describes the structure of the corporation, where each node represents an employee, and a node's children represent the employees under his or her direct supervision. Each node of the tree holds, in addition to the child pointers, the name of the employee and the employee's conviviality rating.

(a) Give a dynamic programming algorithm to make up a guest list that maximizes the sum of the conviviality ratings of the guests. Analyze the running time of your algorithm.

Hints:

- Construct your solution as a *binary choice*.
- Even the president of the company may end up not invited if that turns out to be optimal.

Problem 2: Dynamic Programming

It's the end of the semester, and you're taking n courses, each with a final project. Each project will be graded on a scale of 1 to $g > 1$, where a higher number is a better grade. Your goal is to maximize your average grade on the n projects. (Note: this is equivalent to maximizing the *total* of all grades, since the difference between the total and the average is just the factor n.)

You have a finite number of hours $H > n$ to complete all of your course projects; you need to decide how to divide your time. H is a positive integer, and you can spend an integer number of hours on each project. Assume your grades are deterministically based on time spent; you have a set of functions $\{f_i : 1, 2, \ldots, n\}$ for your n courses; if you spend $h \leq H$ hours on the project for course i, you will get a grade of $f_i(h)$. The functions f_i are nondecreasing; spending more time on a course project will not *lower* your grade in the course.

Given these functions, decide how many hours to spend on each project (again, in integer numbers of hours) to maximize your average grade. Your algorithm should be polynomial in n, g, and H.

To help get you started, think about the (i, h) subproblem. Let the (i, h) subproblem be the problem that maximizes your grade on the first i courses in at most h hours. Clearly, the complete solution is the solution to the (n, H) subproblem. Start by defining the values of the $(0, h)$ subproblems for all h and the $(i, 0)$ subproblems for all i (the latter corresponds to the grade you will get on a course project if you spend *no* time on it).

Argue that the problem has optimal substructure, define the recursive formula (i.e., the value of any (i, h) subproblem, where $0 \leq i \leq n$ and $0 \leq h \leq H$), describe an algorithm for iteratively computing the value of the optimal solution, and describe an algorithm for recreating the optimal solution (i.e., mapping your H hours to your n projects).