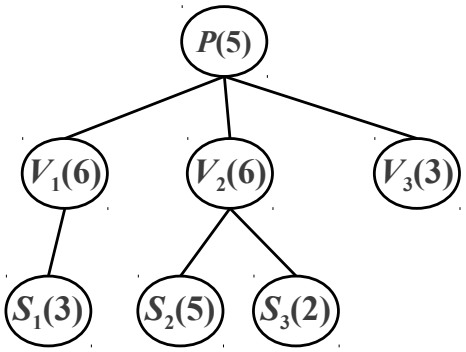
Self-study 1, Exercise 2

Consider a large corporation that is planning a team-building event. 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 *friendliness rating*, which is a real number. In order to facilitate meeting people that one does not interact with on a daily basis, the planners of the event *do not want both an employee and his or her immediate supervisor to attend*. Respecting this constraint, the goal is to produce a participant list that *maximizes the sum of the friendliness ratings* of the participants (which we will call the *friendliness rating* of the participant list).

You are given the tree that describes the structure of the corporation, for example, using the left-child, right-sibling representation as described in Section 10.4 of CLRS3, see Figure 10.10 (Section 10.3 of CLRS4, see Figure 10.7). Each node of the tree holds the name of an employee and that employee's friendliness rating.



1. *(10 points)* Consider the example company structure in the figure. First, consider inviting the president P. Then, which highest level employees can we consider to invite or not to invite and what is the highest friendliness rating of participant list that we can achieve? Now, consider not inviting P. Again, which highest level employees can we consider to invite or not to invite and what is the highest friendliness rating of the participant list that we can achieve?

2. *(20 points)* Write a pseudocode of a dynamic programming algorithm to solve this problem. The algorithm should return the maximum friendliness rating of a participant list. Another algorithm should then be provided to print the names of the invited participants (using the recorded information in the company tree—you should extend the nodes of the tree with necessary fields). It is probably most natural to write a memoized algorithm for this problem.

3. *(10 points)* How many times each node of the tree is visited by your algorithm (except for the root and its children)? What is the running time of your algorithm?

4. *(20 points)* Consider a company with extremely friendly bosses. More specifically, each employee is friendlier than the sum of the friendliness ratings of *all* of his subordinates (immediate and not immediate, i.e., the sum of the ratings in the whole subtree rooted at that employee). Prove that you can now use a greedy algorithm to solve the problem. Describe in a few words how the gready algorithm would work.

5. *(10 points)* Analyze the running time of your greedy algorithm. Is it faster than the dynamic algorithm? What if we consider exact (non-asymptotic) running time?