Greedy Algorithm Example

In an attempt to get people in the diverse parts of her corporation to know each other, Gillian Bates, the president of Macrohard Corporation, has decided to hold some company parties. The company is hierarchical, and thus each of Macrohard's n employees except Gillian has exactly one immediate supervisor. The main requirement Gillian has imposed for each party is that among the guests, there cannot be two employees such that one is the subordinate of the other, either directly or indirectly. The personnel director has provided you with a list of the k employees whose schedules indicate that they are free for this Wednesday's party. Give an efficient algorithm to select the largest possible subset of these employees as guests subject to Gillian's requirements.

It is easy to see a greedy algorithm that works. For each k, work up the tree and mark all nodes. If you get to a node that is already marked, stop. If you get to Gillian, stop. When you are through, invite all potential guests that are not marked.

The running time is $O(n)$, because we visit each node in the tree at most once.

We will prove correctness using these facts. Let x and y be any two potential invitees, such that x is a subordinate of y.

1. Any subordinate of x is also a subordinate of y.
2. Any superior of x is either a subordinate of y or a superior of y.
3. By 1 and 2, given any legal guest list that contains y, we could invite x instead of y and maintain legality.

So, let's prove that the greedy algorithm works. We'll assume that we have our greedy solution, $\textsc{Greedy} = \{e_i, \ldots, e_x\}$, and an alternative legal solution, $\textsc{Opt} = \{e_j, \ldots, e_y\}$. These are both subsets of the set of k potential guests, $P = \{e1, e_2, \ldots, e_k\}$. We will try and modify $\textsc{Opt}$ until it looks like $\textsc{Greedy}$.

Suppose $\textsc{Opt}$ contains employee x and $\textsc{Greedy}$ contains employee y, where x is y's superior. Clearly, we could uninvite x and invite y instead, by 3 above, and maintain legality.

Suppose $\textsc{Opt}$ contains employee x and $\textsc{Greedy}$ contains employee y, where y is x's superior. This could not happen, because our algorithm always took the lower person.

So, for each employee x in $\textsc{Opt}$ that is not in $\textsc{Greedy}$, we replace him/her with a subordinate y that we included in $\textsc{Greedy}$. There is always such a person, because we only omitted potential invitees that had underlings who were potential invitees. We could not replace two x, x' in $\textsc{Opt}$ with one person y in $\textsc{Greedy}$ because of the tree structure: all x, x' have mutually exclusive sets of underlings.

We do this until $\textsc{Opt} \subseteq \textsc{Greedy}$, and we are done.