CS381 **Homework 4 Sketch of Solution** **Fall 2012**

Question 1. The crucial observation is this: For any leaf v, a solution that does not include v can be replaced by another (equally good) solution in which v has been included and its parent kicked out. This suggests a solution whose high-level description is as follows:

1. Initialize S to be empty.
2. Repeat the following until the graph becomes empty:
   (a) Add to S every vertex v that is a leaf.
   (b) Delete from T the leaves and the parents of leaves. (Note that T may become disconnected as a result of these deletions, i.e., it can become a forest of trees rather than a single tree.)
3. Output S.

The following is an algorithm that implements the above idea in $O(n)$ time by traversing the tree without modifying it, merely marking the nodes that belong to S during that traversal.

1. Initially none of the vertices is marked as being in S.
2. A postorder traversal of the tree is done, and at the moment of assigning a postorder number to a node v a decision is also made on whether to include it in S or not, according to the following criterion: Unless at least one child of v has been marked as being in S, v is marked as being in S. Hence a leaf is in S, as is a node none of whose children have been marked as being in S by the postorder traversal.

Question 2. For every edge (u, v), either both u and v are ignored (if one of them is in S) or both are included in S. At least of of them must be in $\hat{V}$, because the definition of $\hat{V}$ requires it. Therefore in the worst case 2 vertices are being added to S when only one of them is in $\hat{V}$. Therefore $|S| \leq 2|\hat{V}|$.

Question 3. Initially all jobs are marked as *needy*. As the algorithm proceeds, a job that gets a machine assigned to it becomes marked as *not needy*. A machine i is said to be *compatible* with a job J_k if $l_k \leq i \leq r_k$. The algorithm is then as follows.

1. Go through the machines in the order $1, 2, \ldots, m$ and, for each such i do the following:
   (a) Compute the set of jobs (call it S_i) that are still needy, and are compatible with machine i.
   (b) From the set S_i, pick the job J_k that has the smallest r_k, assign machine i to J_k, and mark J_k as being not needy. (Of course if S_i is empty then there is no such J_k and machine i remains unused.)

The intuitive rationale for the above greedy choice of J_k is that, among the jobs in S_i, J_k is the job that is most at risk of remaining permanently needy (once the machine number exceeds its r_k).

Question 4. For every leaf v, $\mathit{With}[v] = w[v]$ and $\mathit{Without}[v] = 0$. For every non-leaf v, the following holds:

$$\mathit{With}[v] = w[v] + \sum_{x \in L[v]} \mathit{Without}[x]$$

$$\mathit{Without}[v] = \sum_{x \in L[v]} \max\{\mathit{With}[x], \mathit{Without}[x]\}$$

The above suggests computing the *With* and *Without* vectors in a postorder traversal of the tree, because once we have the *With* and *Without* values for the children of a node v, we can compute them for the node v in $O(|L[v]|)$ time, for a total time of $\sum_v |L[v]| = O(n)$.