CS 374 **Spring 2015**

Homework 7

Nikhil Unni (nunni2)

1.

$$\text{mon}(r, \text{Yes}) = \begin{cases} w(r) & \text{if } \|\text{children}(r)\| = 0 \\ w(r) + \sum_{\text{child} \in \text{children}(r)} \min(\text{mon(child, No)}, \text{mon(child, Yes)}) & \text{else} \end{cases}$$

$$\text{mon}(r, \text{No}) = \begin{cases} 0 & \text{if } \|\text{children}(r)\| = 0 \\ \sum_{\text{child} \in \text{children}(r)} \text{mon(child, Yes)}) & \text{else} \end{cases}$$

The recurrence is based on the fact that, for any given root node, if it is not included in the minimum vertex cover, then all of its children **must** be, because otherwise the edges inbetween them would go uncovered. Conversely, if the root is included, then we must then just find the minimum cover for our children, which can be either included or excluded, whichever has a smaller final value.

The function can be called by min(mon(r, Yes), mon(r, No)) for any arbitrary root node of the tree.

We would memoize on both the vertex and whether or not its included. If the vertices were indexed, then we could have a 2D array of size $n \times 2$, where n is the number of vertices. The algorithm would build bottom up from the leaf nodes towards the root. Overall, the algorithm would have a time complexity of $O(\|V\| + \|E\|)$, as it visits each node a maximum of 2 times, and at each vertex it does work of the size of its children. Since the structure is a tree, all of this "children work" amortizes to $O(\|V\|)$ throughout the execution of the algorithm.