P3

Prompt:

Given a tree with non-negative weighted nodes, design a polynomial time algorithm to find the minimum cost vertex cover of the tree.

Algorithm:

Let Y be a vertex v indexed array that holds the cost of the minimum cost vertex cover where v is included.

Let N be a vertex v indexed array that holds the cost of the minimum cost vertex cover where v is not included.

```
Input: The root node of the tree and the MAYBE flag
Output: The minimum cost vertex cover of the tree
MinCost(v, f):
    Let Y be an empty array
    Let N be an empty array
    If f is YES then
        If Y[v] is not assigned then
            Set Y[v] = cost(v) + sum{MinCost(c, MAYBE) for c in v.children}
        EndIf
        return Y[v]
    Else if f is NO then
        If N[v] is not assigned then
            N[v] = sum{MinCost(c, YES) for c in v.children}
        EndIf
        return N[v]
    Else if f is MAYBE then
        If Y[v] is not assigned then
            Set Y[v] = cost(v) + sum{MinCost(c, MAYBE) for c in v.children}
        EndIf
        If N[v] is not assigned then
            N[v] = sum{MinCost(c, YES) for c in v.children}
        EndIf
        return min{Y[v], N[v]}
    EndIf
```

Claim:

The algorithm terminates in polynomial time.

Proof:

That the algorithm terminates is obvious given the tree is finite. The cost of a vertex's inclusion or exclusion from the minimum cost solution is cached it's found so future requests for that value occur in constant time. Therefore, recursion is the dominant runtime factor. Thus, the algorithm runs in polynomial time. □

Claim:

The algorithm outputs the minimum cost vertex cover of the tree.

Proof:

Let OPT(n, f) denote the minimum cost vertex cover for a tree where n is the root node of the tree and f is a flag indicating that n is YES included in the minimum cost vertex cover, n is not NO in the minimum cost vertex cover, or that n might be MAYBE in the minimum cost vertex cover.

Case 1: If f=YES, then n is included in the minimum cost vertex cover. So, we include the cost of n in OPT(n, YES). To that we add the minimum vertex cover cost of every child c of n which may or may not be included in the minimum vertex cover. Thus:

$$\text{OPT(n, YES)} = \text{cost(n)} + \text{sum}\{\text{OPT(c, MAYBE) for c in n.children}\}$$

Case 2: If f=NO, then n is not included in the minimum cost vertex cover. So we do not include the cost of n in OPT(n, NO). Since n is not in the minimum cost vertex cover set, every child c of n must be so that every edge is incident to at least one vertex that is. Therefore:

$$\text{OPT(n, NO)} = \text{sum}\{\text{OPT(c, YES) for c in n.children}\}$$

Case 3: If f=MAYBE, then n might be included in the minimum cost vertex cover. Since we don't know, we find the minimum cost vertex cover where n is included OPT(n, YES) and the minimum cost vertex cover where n is not included OPT(n, NO) and choose whichever is smaller. Thus:

$$\text{OPT(n, MAYBE)} = \min\{\text{OPT(n, YES)}, \text{OPT(n, NO)}\}$$

Thus, the recurrence relation:

$$\text{OPT(n, f)} = \begin{cases} \text{cost(n)} + \text{sum}\{\text{OPT(c, MAYBE) for c in n.children}\} & \text{if f=YES} \\ \text{sum}\{\text{OPT(c, YES) for c in n.children}\} & \text{if f=NO} \\ \min\{\text{OPT(n, YES)}, \text{OPT(n, NO)}\} & \text{if f=MAYBE} \end{cases}$$

□