DP and Graph Algorithms
Problem Set 2 – CS 6515/4540 (Fall 2025)

Answers to problem set 2, question 5.

5 DP: Minimum Lights

The solution for this problem is to identify the smallest possible vertex cover of the given tree. Another way to look at it is if we have identified the maximum independent set (MIS) of this tree then taking the complement of the vertices in the MIS, i.e. $(V \backslash MIS)$, will give us the smallest vertex cover. We can construct a MIS in $O(|V|)$, therefore we can produce a complement set, i.e. the vertex cover in $O(|V|)$ as well.

Below, I will outline a dynamic programming based solution to directly compute the min vertex cover.

I assume that I have access to the children and grandchildren of any given node of the tree, this is feasible in $O(|V|)$ and can be done as a pre-computation step.

1. Define,
$$T[i] = \text{size of the min vertex cover of the tree rooted at node i}$$
Given this our final answer is $T[root]$.

2. When computing $T[i]$ either node i will be a part of the min vertex cover or it will not. Based on this we define 2 scenarios:
   - If i is in the cover then we have covered all edges to the children of i and we only need to look at the $\sum T[c]$ for all children of i. The vertex cover in this case would be $[(\cup_{c \in children[i]} VC[c]) \cup i]$. $VC[c]$ gives the vertices included in the vertex cover of a subtree rooted at c.
   - If i is not in the cover, then we need to cover all edges from i to its children. We look at $\sum T[gc]$ for all grandchildren of i and then add all |children[i]| to this result. The vertex cover in this case would be $[(\cup_{gc \in grandchildren[i]} VC[gc]) \cup (\cup_{c \in children[i]} c)]$

   Therefore, the recurrence becomes:
$$T(i) = \min(1 + \sum_{c \in children[i]} T[c], \sum_{gc \in grandchildren[i]} T[gc] + |\text{children}[i]|)$$
   Base case, for a tree rooted at a leaf node, there are no edges so cost will be 0.

3. The running time of this algorithm is $O(|V|)$ as each node will be visited only 3x in a single run of the algorithm. 1x for filling in the value and 2x when recursing for its parent and grandparent.