Tutorial 4: Dynamic Programming on Trees

Problem 1. *You work for a commuter rail agency in City R. The agency wants to open coffee shops at railroad stations. Your task is to design a program that finds the most economical way to do that.*

The rail network is represented by a tree $T = (V, E)$ rooted at vertex R. The agency wants to open coffee shops at some stations so that for every station u there is a coffee shop at u or one of the stations adjacent to u. In other words, the requirement on the set of stations/vertices C that host a coffee shop is as follows:

> *For every $u \in V$, we have (i) $u \in C$, or (ii) the parent p_u of u is in C (unless $u = R$, then this option is not available since u has no parent), or (iii) at least one of the children of u is in C.*

The cost of opening a coffee shop at station u is $c_u > 0$.

Design a DP-algorithm that finds the cost of the cheapest solution.

1. *Define subproblems.*
2. *Define a dynamic-programming table and explain the meaning of its entries.*
3. *Write the initialization step of your algorithm.*
4. *Write the recurrence formula for computing entries of the table.*
5. *Explain the formula.*
6. *Find the running time of your algorithm.*

Hint: *Consider a feasible solution C. Let u be some vertex and T_u be the subtree rooted at u. Is it necessarily true that the restriction of C to T_u is a feasible solution for subtree T_u?*

For a subset $S \subseteq V$ and a vertex $u \in V$, let us say that u is *covered* by S if either $u \in S$, or $v \in S$ for some neighbor of u. A subset $S \subseteq V$ such that for every $u \in V$, u is covered by S, is called a *dominating set* of T. The problem asks us to find a dominating set of T of minimum cost.

We define $A[u]$ to be the minimum cost of a dominating set of T_u, $B[u]$ to be the minimum cost of a dominating set S of T_u such that $u \in S$, and $C[u]$ to be the minimum cost of a dominating set of T_u with no requirement for covering u. More precisely, $C[u]$ is the minimum cost of a subset $S \subseteq V(T_u)$ such that for every $v \in V(T_u) - \{u\}$, v is covered by S.

If u is a leaf, we have $A[u] = c_u$, $B[u] = c_u$ and $C[u] = 0$. If u is not a leaf, we have

$$A[u] = \min \begin{cases} c_u + \sum_{v \text{ child of u}} C[v] \\ \min_{v \text{ child of u}} \left(B[v] + \sum_{w \neq v \text{ child of u}} A[w] \right). \end{cases}$$

The first case is for when u belongs to the minimum cost dominating set. In this case, we add u's cost and recurse on $C[v]$ for its children since they have been covered by having selected u, thus there is no requirement for covering them. The second case is for when u does not belong to the minimum cost dominating set. In that case one of u's children must be selected to cover u, therefore we recurse on $B[v]$ for that child and on $A[w]$ for the rest. Similarly, we find

$$B[u] = c_u + \sum_{v \text{ child of u}} C[v]$$

and

$$C[u] = \min \begin{cases} c_u + \sum_{v \text{ child of u}} C[v] \\ \sum_{v \text{ child of u}} A[v]. \end{cases}$$

We compute $A[R]$ to solve our problem. There are $3n$ subproblems, three for every vertex of T. In solving a subproblem associated with u, we visit u's children at most three times to solve their associated subproblems. We thus visit every edge at most 9 times, getting a running time linear in the size of T.