Problem 3. Constructing trees by using a fixed library of tree patterns

A *tree template* is a triple (T, a, c), where T represents a small tree and a and c are non-negative real numbers that describe the "bonding parameters" of T. Figure 3 gives an example of a set of six templates and an overlay of an example tree.

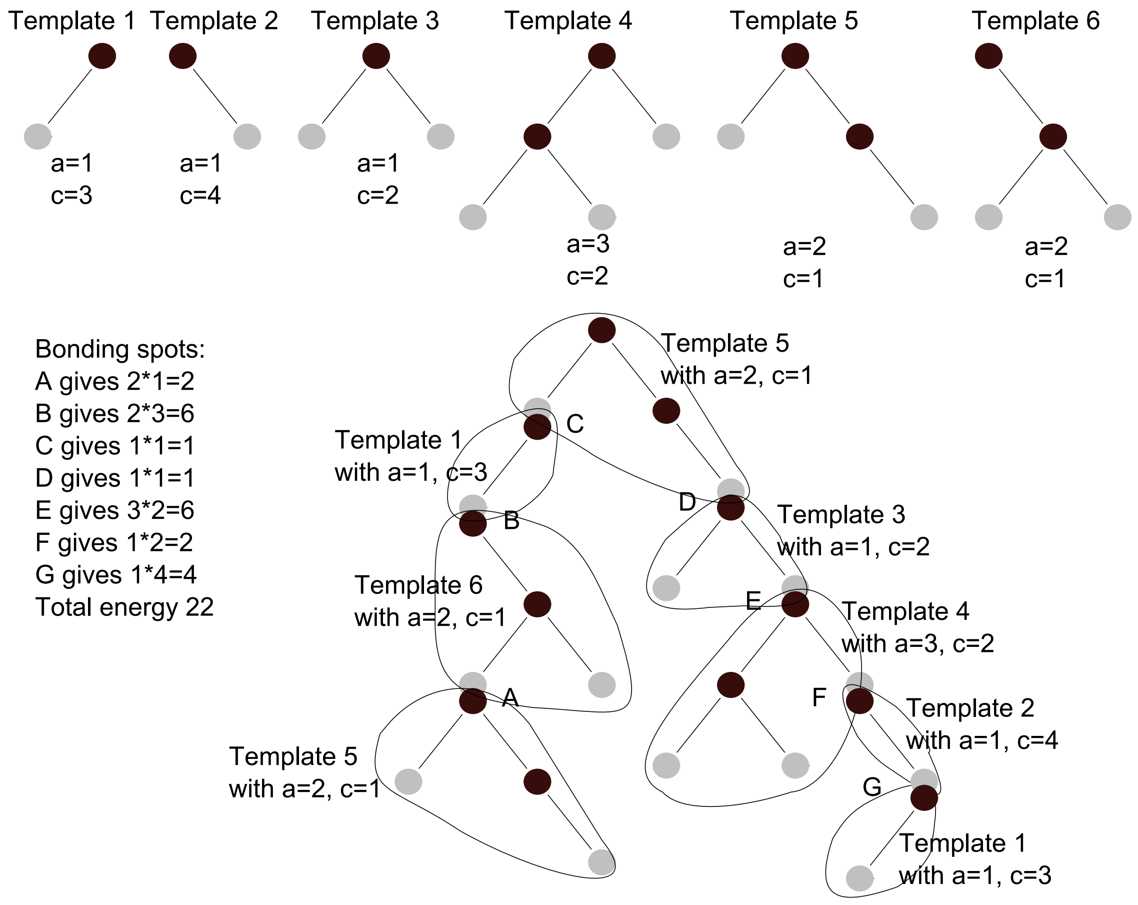

Figure 3: A set with 6 templates together with an overlay of an example tree. Notice that even though templates 1 and 2 are one another's mirror images, they have different parameters.

Given a set of templates and an arbitrary input tree, we want to compute an "optimal overlay" of the input tree by using the set of templates. An *overlay* of a tree Z by using a set S of templates is a collection of templates $(T_1, a_1, c_1) \in S$, $(T_2, a_2, c_2) \in S$, $(T_3, a_3, c_3) \in S$, …, $(T_k, a_k, c_k) \in S$ together with a mapping from the vertices of $T_1, T_2, T_3, \ldots, T_k$ to the vertices of Z such that:

1. If (u, v) is an edge in one of the templates T_i, then u and v map to an edge in Z. In other words, around the images of u and v, Z looks like T_i; that is, the mapping preserves the structure of each of the templates T_i. Figure 3 depicts an example of an overlay.

2. For each template T_i, if the root of T_i is mapped to a vertex x in Z, then *either* x is the root of Z *or* x is the image of exactly one leaf of exactly one of the other templates T_j. In the second

case, x is called a *bonding spot* and the *bonding energy* at x between T_i and T_j is equal to a_i times c_j. For example in Figure 3, at bonding spot B the bonding energy between template 6 with $a = 2$ and template 1 with $c = 3$ is equal to $2 \cdot 3 = 6$.

The *bonding energy* of an overlay is the sum of the bonding energy of all of its bonding spots.

You need to design an efficient algorithm that takes a set of m templates and a tree with n nodes as input and computes whether an overlay of the tree exists. If an overlay exists, then the algorithm should return an overlay of the tree for which the sum of all bonding energies is minimized. Assume that, for each template, the number of leaves is at most the total number (m) of templates. Analyze the running time of your algorithm in n and m. Partial credit will be given for algorithms that assume that $c = 1$ for each template.

Solution: Our solution uses dynamic programming. Let $\{(T_i, a_i, c_i) : 1 \leq i \leq m\}$ be the set of templates. For each node x in Z, let $cost_i(x)$ minimize the bonding energy of the layouts of the subtree of Z rooted in x that contain template T_i and map its root to x. If no such layout exists then $cost_i(x) = \infty$. Let $leaves_i(x)$ be the set of vertices in Z that are the images of the leaves of T_i. Then

$$cost_i(x) = \sum_{l \in leaves_i(x)} \min_{1 \leq j \leq m} (cost_j(l) + a_j c_i).$$

This recurrence relationship leads to a straightforward dynamic programming solution. For each of the n nodes in the tree we need to evaluate the recurrence which costs $O(m^2)$ (by our assumption $leaves_i(x)$ has size at most m). This gives a running time of $O(nm^2)$.

By defining

$$cost'_i(x) = \min_{1 \leq j \leq m} (a_j c_i + cost_j(x))$$

we obtain the recurrence

$$cost'_i(x) = \min_{1 \leq j \leq m} (a_j c_i + \sum_{l \in leaves_j(x)} cost'_j(l)).$$

This recurrence leads to an $O(nm^2)$ dynamic programming solution as well.