Weighted Rooted Small Parsimony Problem (WRSPP)

Input: Rooted binary tree T with m leaves labeled by elements n-character string.
$k \times k$ scoring matrix δ (indicating cost of transforming between states)

Output: Labeling of non-leaf vertices with n-character string minimizing the parsimony score.

Sankoff's Algorithm (1975)

Solves the Weighted Rooted Small Parsimony Problem (WRSPP) using dynamic programming.

For each character, you must complete the following dynamic programming process:

Define $s_t(v)$ to be minimum parsimony score of subtree rooted at v if vertex v has state t for the character in question.

Initialization: If v is a leaf, then $s_v(k) = \begin{cases} 0 & \text{if } v \text{ has state } k \\ \infty & \text{otherwise} \end{cases}$

Recurrence: Suppose v has children u and w. Then,

$$s_t(v) = \min_i \{s_i(u) + \delta(i,t)\} + \min_j \{s_j(w) + \delta(j,t)\}$$

Fill out from leaves to root of the tree.

Return: The optimal score (for the character) will be the value $s^* = \min_i s_{root}(i)$.

To get the labelings, you must traceback through the tree as follows:

- Root state is $c^* = argmin_i s_{root}(i)$.
- Follow pointers down the tree to pick the states at each vertex.

To get the total optimal score across all characters, you must add together the optimal scores for each character.

To get the complete labeling of internal nodes, just concatenate together the labelings for each character.

Example using Sankoff's Algorithm

	A	T	G	C
A	0	3	4	9
T	3	0	2	4
G	4	2	0	4
C	9	4	4	0

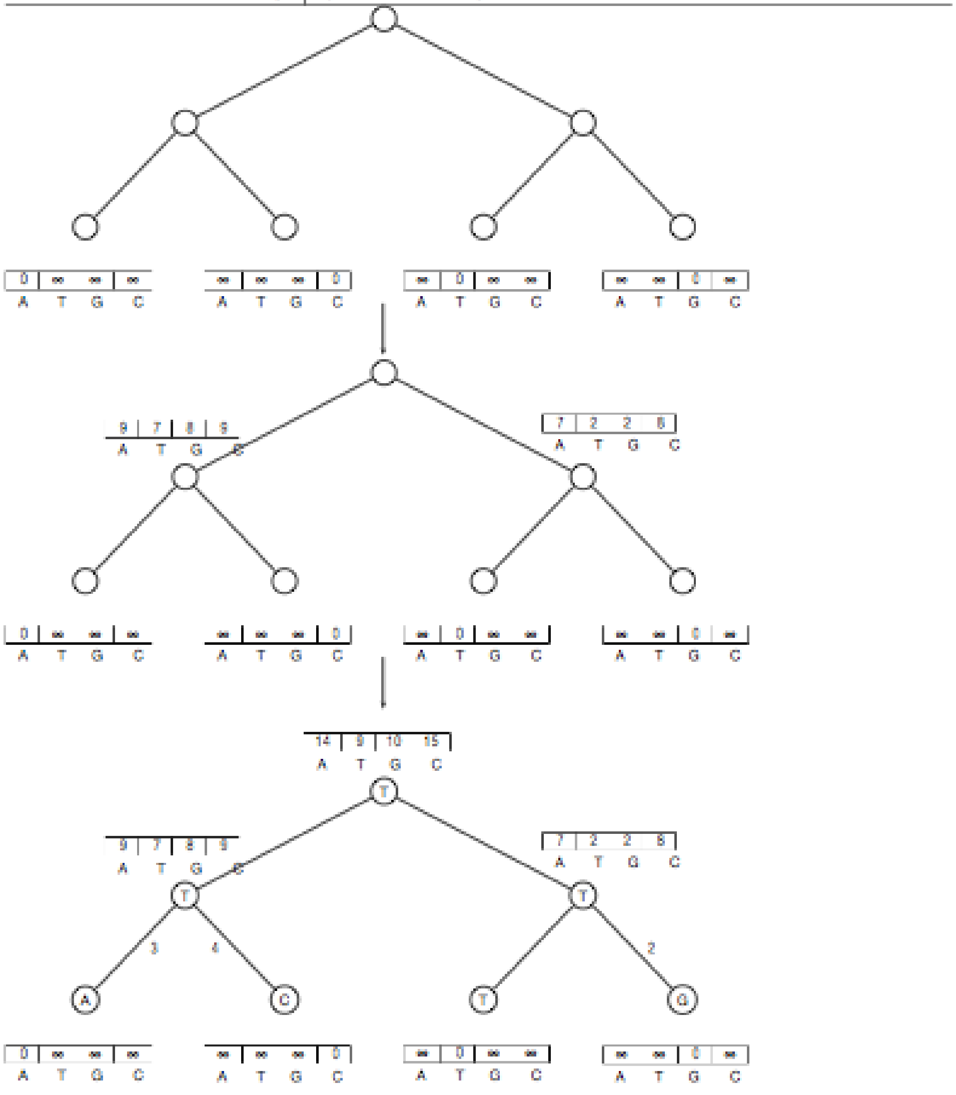

Figure 10.18 An illustration of Sankoff's algorithm. The leaves of the tree are labeled by A, C, T, G in order. The minimum weighted parsimony score is given by $s_T(\text{root}) = 0 + 0 + 3 + 4 + 0 + 2 = 9$.

Unweighted Rooted Small Parsimony Problem (URSPP)

Input: Rooted binary tree T with m leaves labeled by elements n-character string.

Output: Labeling of non-leaf vertices with n-character string minimizing the parsimony score (total number of changes in the tree).

Fitch's Algorithm (1971)

Solves Unweighted Rooted Small Parsimony Problem.

For each character, you must complete the following process:

Step 1: Assign a set $S(v)$ of letters to every vertex v, traversing from leaves to root.

- For each leaf ℓ, $S(\ell)$ = observed character.
- For vertex v with children u and w: $S(v) = \begin{cases} S(u) \cap S(w) & \text{if non empty intersection} \\ S(u) \cup (Sw) & \text{otherwise} \end{cases}$

Step 2: Assign labels to each vertex, traversing the tree from root to leaves.

- Assign root r arbitrarily from $S(r)$
- For all other v:
  - If its parent's label is in its set $S(v)$, give it that label
  - Else, choose an arbitrary letter from the set $S(v)$

To get the complete labeling of internal nodes, just concatenate together the labelings for each character.

Example using Fitch's Algorithm

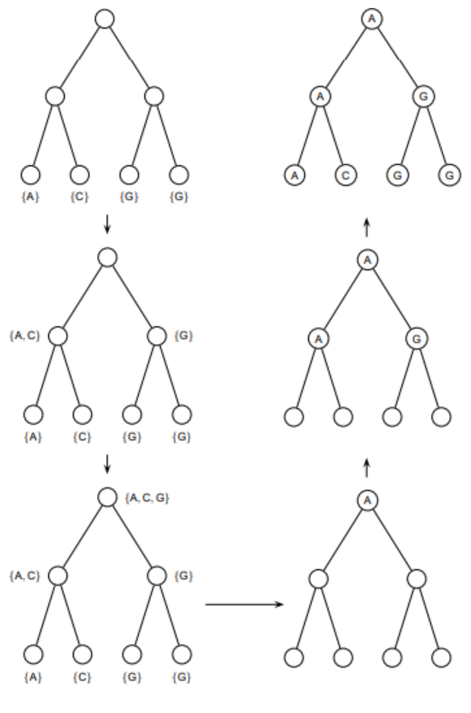

Figure 10.20 An illustration of Fitch's algorithm.