Algorithm 1 Enumerate all trees for given taxon set

1: **function** RESOLVE($root$)
Require: phylogenetic node, $root$, the root of an unresolved phylogenetic tree
Ensure: phylogenetic forest: all possible resolutions of tree rooted at $root$
2: $forest \leftarrow \emptyset$
3: **if** $root$ is leaf **then**
4: $forest \leftarrow list(root)$
// $root$ is already the root of all possible subtrees
5: **else**
6: $subsetPairs \leftarrow$ GENERATE_SUBSETS($root_{descendents}$)
7: **for all** $(l, r) \in subsetPairs$ **do**
8: $subtree_l \leftarrow$ RESOLVE(l)
9: $subtree_r \leftarrow$ RESOLVE(r)
10: $forest \leftarrow$ CREATE_NODE($subtree_l$, $subtree_r$)
11: **end for**
12: **end if**
13: **return** $forest$
14: **end function**

Algorithm 2 Enumerate subset bifurcations of ordered input set

1: **function** GENERATE_SUBSETS($forest, length$)

Require: Phylogenetic forest (essentially an *ordered* set of nodes) of size > 2, as well as size of said forest

Ensure: List of tuple pairs of all bifurcations of forest set

2: **if** $length < 3$ **then** ▷ This is degenerate case. Returning this just to match types.

3: $subsetPairs \leftarrow (\emptyset, n)$

4: **else if** $length == 3$ **then** // This is the base case.

5: $subsetPairs \leftarrow$
[([forest[0]], forest[1:]),
([forest[1]], [forest[0], forest[2]]),
([forest[2]], forest[:2])] // Split notation above follows Python.

6: **else** // Call recursively from here.

7: $subsetPairs \leftarrow \emptyset$

8: **for** $i \leftarrow 0, n$ **do** // For each single member, m, create tuple $(m, forest - m)$.

9: $subsetPairs \leftarrow subsetPairs$ **cons**

10: **if** $length - i - 1 > 2$ and $i < length/2$ **then**

11: **for** $(lhs, rhs) \in$ GENERATE_SUBSETS($forest_d escendents$[]last $length - i$ descendents of forest, length - i - 1) **do**

12: **end for**

13: **end if**

14: **end for**

15: **end if**
return $subsetPairs$

16: **end function**