Pseudo code

To present an algorithm, you can either first describe it in prose and then refer to a summary in pseudo code, or you first refer to the pseudo code, followed by an explanation of the comprised steps. Algorithm 1 on page 1 gives an example of pseudo code.

Note that all terms, data types and meanings of variables need to be clear – either from the main text or specified in the pseudo code. For instance, write "rooted tree T" instead of just "T". Properly specify the input and output, even if you think it is clear from the main text, because often it is not, and even if so, pseudo code (like tables and figures) should be self-contained.

You can point to individual lines or blocks composed of several lines. This is also useful when you discuss the run time complexity of an algorithm, e.g., the for-loop in lines 12–13.

Level of detail. Be only as detailed and technical as necessary to not obfuscate the procedure but still clarify all relevant aspects. Whether a step needs to be broken into sub-steps often depends on the context. If its implementation is not obvious or a naive implementation would ruin the overall run time, break it down. In particular, breaking down trivial things like the selection of a minimum from a list or summing up its values, would only hinder the reader to understand the algorithm. In such cases, use standard mathematical formalism, such as $\min\{\cdots\}$ or $\Sigma\cdots$.

Notation. In general, when presenting algorithms, mathematical notation is preferable to programming notation, because it is standardized and commonly known by computer scientists. Feel free to combine it with prose, like in line 12 of Algorithm 1. In particular, there is no need to use ascii-art such as "`x_i`" rather than "x_i", or "`x*y`" rather than "$x \cdot y$". Use "$\longleftarrow$" for assignments to distinguish it from equality ("=") or mathematical definitions (":="). The latter might only be needed to *specify* a value but not to *assign* it to any variable, see, e.g. line 13.

Algorithm 1: Fitch-Hartigan for Zero-One Instances

Input: A rooted tree $T = (V, E)$ with each leaf $l \in V$ labeled with $L(l) \subseteq \mathcal{C}$.
Output: A labeling L with minimal Wagner parsimony weight $W(T, L)$.

1 **foreach** *character* $c \in \mathcal{C}$ **do**
  `/* Bottom-up phase */`
2 **foreach** *leaf* l **do**
3 $VU(l) \longleftarrow \{L_c(l)\}$
4 $VL(l) \longleftarrow \emptyset$
5 **foreach** *unlabeled node* u *whose children* $v_1, \ldots, v_{l(u)}$ *are labeled* **do**
6 **for** $b \in \{0, 1\}$ **do** $k(b) \longleftarrow |\{v_i \mid b \in VU(v_i)\}|$
7
8 $K \longleftarrow \max\{k(0), k(1)\}$
9 $VU(u) \longleftarrow \{b \mid k(b) = K\}$
10 $VL(u) \longleftarrow \{b \mid k(b) = K - 1\}$
  `/* Top-down refinement */`
11 assign any $b \in VU(r)$ to the root node r: $L_c(r) := b$
12 **foreach** *unrefined node* v *whose parent node* u *is already refined to* $L_c(u) = a$ **do**
13 refine the labeling of v to $L_c(v) \longleftarrow b$, with any $b \in B(v, a)$:

$$B(v,a) := \begin{cases} \{a\} & \text{if } a \in VU(v), \\ \{a\} \cup VU(v) & \text{if } a \in VL(v), \\ VU(v) & \text{otherwise.} \end{cases}$$

14 **return** L