10.4 Representing rooted trees

The methods for representing lists given in the previous section extend to any homogeneous data structure. In this section, we look specifically at the problem of representing rooted trees by linked data structures. We first look at binary trees, and then we present a method for rooted trees in which nodes can have an arbitrary number of children.

We represent each node of a tree by an object. As with linked lists, we assume that each node contains a *key* attribute. The remaining attributes of interest are pointers to other nodes, and they vary according to the type of tree.

Binary trees

Figure 10.9 shows how we use the attributes p, *left*, and *right* to store pointers to the parent, left child, and right child of each node in a binary tree T. If $x.p = \text{NIL}$, then x is the root. If node x has no left child, then $x.left = \text{NIL}$, and similarly for the right child. The root of the entire tree T is pointed to by the attribute $T.root$. If $T.root = \text{NIL}$, then the tree is empty.

Rooted trees with unbounded branching

We can extend the scheme for representing a binary tree to any class of trees in which the number of children of each node is at most some constant k: we replace the *left* and *right* attributes by $child_1, child_2, \ldots, child_k$. This scheme no longer works when the number of children of a node is unbounded, since we do not know how many attributes (arrays in the multiple-array representation) to allocate in advance. Moreover, even if the number of children k is bounded by a large constant but most nodes have a small number of children, we may waste a lot of memory.

Fortunately, there is a clever scheme to represent trees with arbitrary numbers of children. It has the advantage of using only $O(n)$ space for any n-node rooted tree. The ***left-child, right-sibling representation*** appears in Figure 10.10. As before, each node contains a parent pointer p, and $T.root$ points to the root of tree T. Instead of having a pointer to each of its children, however, each node x has only two pointers:

1. $x.left\text{-}child$ points to the leftmost child of node x, and
2. $x.right\text{-}sibling$ points to the sibling of x immediately to its right.

If node x has no children, then $x.left\text{-}child = \text{NIL}$, and if node x is the rightmost child of its parent, then $x.right\text{-}sibling = \text{NIL}$.

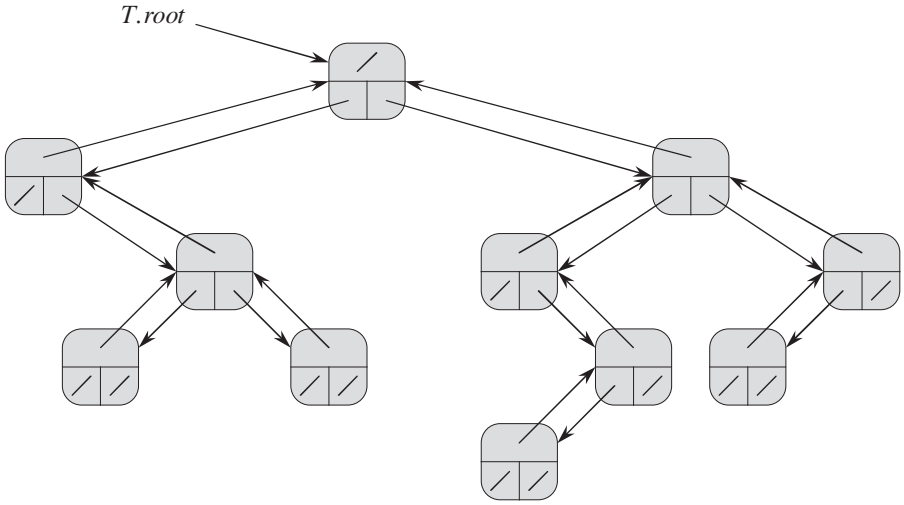

Figure 10.9 The representation of a binary tree T. Each node x has the attributes $x.p$ (top), $x.left$ (lower left), and $x.right$ (lower right). The *key* attributes are not shown.

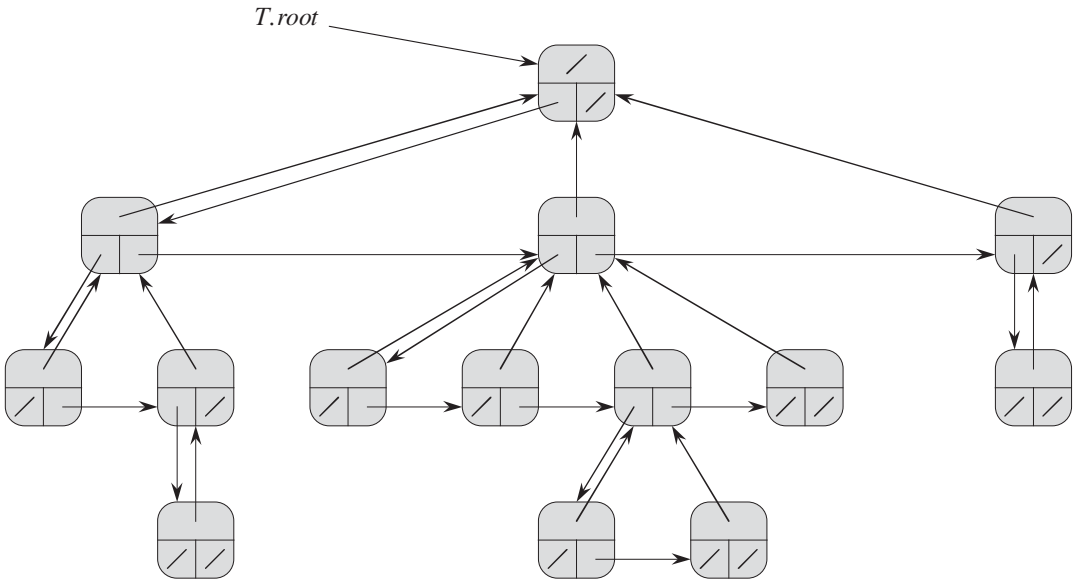

Figure 10.10 The left-child, right-sibling representation of a tree T. Each node x has attributes $x.p$ (top), $x.left\text{-}child$ (lower left), and $x.right\text{-}sibling$ (lower right). The *key* attributes are not shown.