Chapter 10: Elementary Data Structures

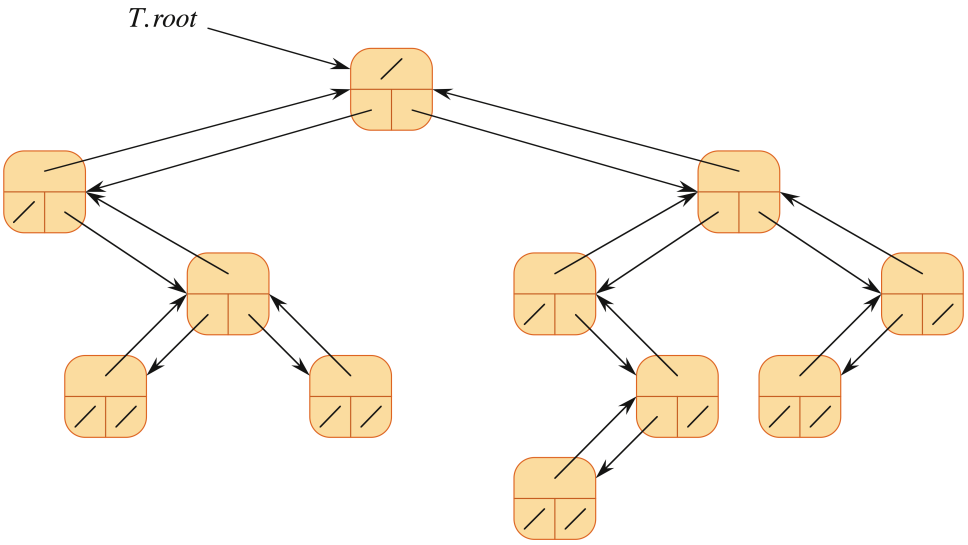
Section 10.3: Representing Rooted Trees

- Trees are composed by tree nodes.
- Each tree node has a key field and some other pointer fields pointing to other nodes. Number of pointer fields in a tree node may be different for different types of trees.
- A tree T has an attribute $T.root$: a pointer to the root of the tree.

Binary Trees

For each node x, there are 3 pointer fields and one data field

- $x.p$ is a pointer to x's parent.
- $x.left$ is a pointer to x's left child.
- $x.right$ is a pointer to x's right child.
- $x.data$ is a pointer to x's satellite data



```
public class BinaryTree<T> {
    private Node<T> root;

    public class Node<S> {
        private Node<S> parent;
        private Node<S> left;
        private Node<S> right;
        private S data; //arbitrary data

        //...
    }
    //...
}
```

Rooted Tree with Bounded Branches

We can represent a general tree that has a bounded number of branches by using an array of pointers. Let the bound be r (also know as the degree). A node in such a tree will have the following fields:

- $x.p$ is a pointer to x's parent.
- $x.child[1:r]$ is a pointer to x's children, up to r of them.
- $x.data$ is a pointer to x's satellite data

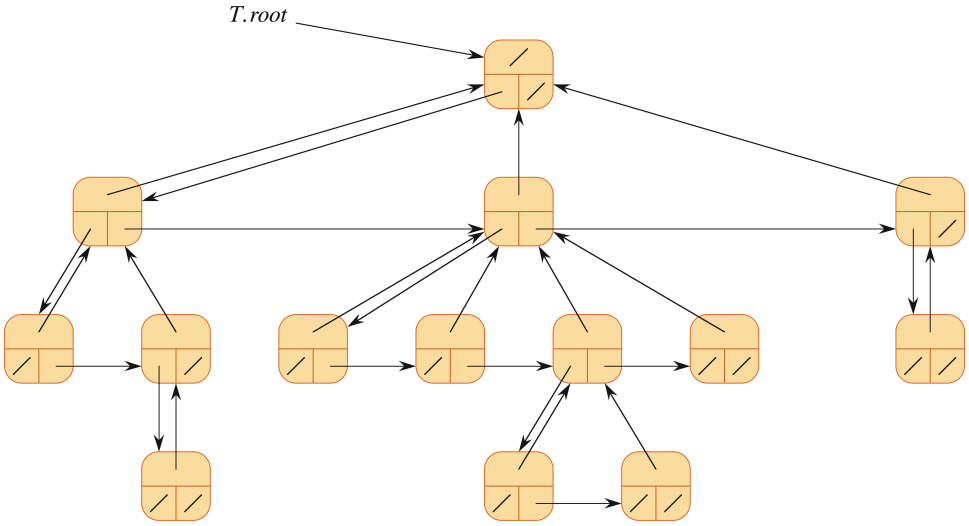
In general, this would be space inefficient as most of the child pointers will be NIL. But it provides an easy and fast way to access the i child of any node.

Assume that the array of child pointers starts at 0. We will denote r using the variable `degree` in the code below.

```
public class ArbitraryTree<T> {
    private Node<T> root;
    private int degree;

    public Tree(int degree) {
        this.degree = degree;
    }

    public class Node<S> {
        private Node<S> parent;
        private Node<S> child[];
        private S data; //arbitrary data

        //...
    }
    //...
}
```

Rooted Trees with Unbounded Branches

- Each node can have any number of children.
- Using left-child, right-sibling representation allows us to represent an arbitrary tree using only three pointers per node.
- Three pointer fields for each node x.
  - $x.p$ is a pointer to x's parent.
  - $x.left$ is a pointer to x's left-most child.
  - $x.right$ is a pointer to the sibling of x immediately to the right.



- Write the Java class declaration for the left-child right-sibling representation of an arbitrary tree.
- **Recommended Exercises**: 10.3-1, 10.3-2, 10.3-4.
- **Solution for Exercise 10.3-2**. We will develop a simple recursive solution for printing key (or data) of each node in a binary tree.

```
PrintAll (root)
// Print the data of each node in the subtree at root
1. if x == NIL then return
2. PrintAll(x.left)
3. print x.data
4. PrintAll(x.right)
```