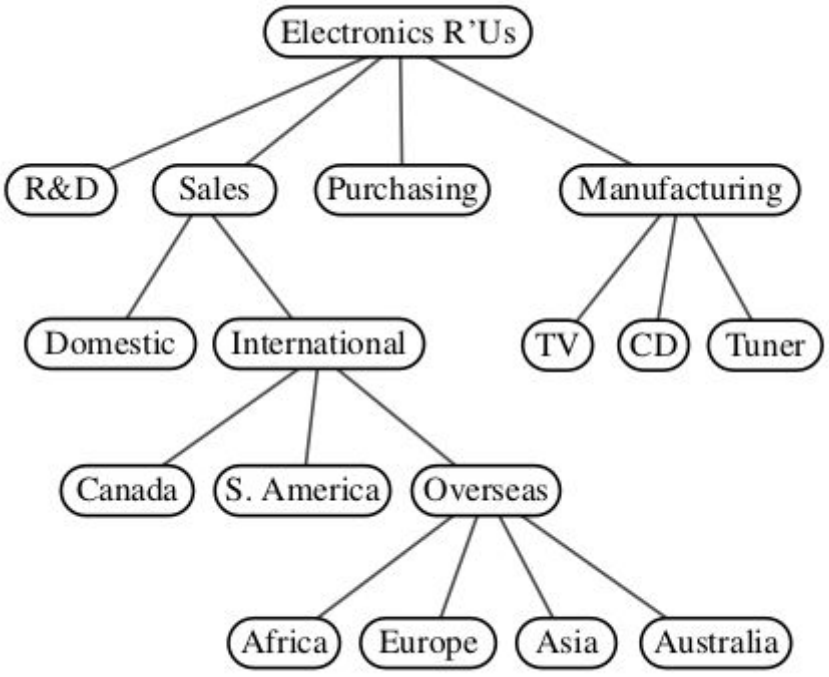
Tarea 15 Arboles/trees

Renteria Magaña Rayni Damian 17212178

A **tree** is an abstract data type that stores elements hierarchically. With the exception of the top element, each element in a tree has a parent element and zero or more children elements. A tree is usually visualized by placing elements inside ovals or rectangles, and by drawing the connections between parents and children with straight lines. We typically call the top element the root
of the tree, but it is drawn as the highest element, with the other elements being connected below (just the opposite of a botanical tree).



Normally, we define a **tree** T as a set of nodes storing elements such that the nodes have a parent-child relationship that satisfies the following properties:

• If T is nonempty, it has a special node, called the root of T , that has no parent.

• Each node v of T different from the root has a unique parent node w; every node with parent w is a child of w.

Note that according to our definition, a tree can be empty, meaning that it does not have any nodes. This convention also allows us to define a tree recursively such that a tree T is either empty or consists of a node r, called the root of T , and a (possibly empty) set of subtrees whose roots are the children of r.

Other Node Relationships

Two nodes that are children of the same parent are siblings. A node v is external if v has no children. A node v is internal if it has one or more children. External nodes are also known as leaves.

Ordered Trees

A tree is ordered if there is a meaningful linear order among the children of each node; that is, we purposefully identify the children of a node as being the first, second, third, and so on. Such an order is usually visualized by arranging siblings left to right, according to their order.

The Tree Abstract Data Type

As we did with positional lists , we define a tree ADT using the concept of a position as an abstraction for a node of a tree. An element is stored at each position, and positions satisfy parent-child relationships that define the tree structure. A position object for a tree supports the method:

p.element( ): Return the element stored at position p.

The tree ADT then supports the following accessor methods, allowing a user to navigate the various positions of a tree:

T.root( ): Return the position of the root of tree T, or None if T is empty.
T.is root(p): Return True if position p is the root of Tree T.
T.parent(p): Return the position of the parent of position p, or None if p is the root of T.
T.num children(p): Return the number of children of position p.
T.children(p): Generate an iteration of the children of position p.
T.is leaf(p): Return True if position p does not have any children.
len(T): Return the number of positions (and hence elements) that are contained in tree T.
T.is empty( ): Return True if tree T does not contain any positions.
T.positions( ): Generate an iteration of all positions of tree T.
iter(T): Generate an iteration of all elements stored within tree T.

Binary Trees

A binary tree is an ordered tree with the following properties:
1. Every node has at most two children.
2. Each child node is labeled as being either a left child or a right child.
3. A left child precedes a right child in the order of children of a node.

The subtree rooted at a left or right child of an internal node v is called a left subtree or right subtree, respectively, of v. A binary tree is proper if each node has either zero or two children. Some people also refer to such trees as being full binary trees. Thus, in a proper binary tree, every internal node has exactly two children. A binary tree that is not proper is improper.

A Recursive Binary Tree Definition

Incidentally, we can also define a binary tree in a recursive way such that a binary tree is either empty or consists of:

• A node r, called the root of T , that stores an element
• A binary tree (possibly empty), called the left subtree of T
• A binary tree (possibly empty), called the right subtree of T

The Binary Tree Abstract Data Type

As an abstract data type, a binary tree is a specialization of a tree that supports three additional accessor methods:

T.left(p): Return the position that represents the left child of p,
or None if p has no left child.

T.right(p): Return the position that represents the right child of p,
or None if p has no right child.

T.sibling(p): Return the position that represents the sibling of p,
or None if p has no sibling.

References

Goodrich, M. T., Goldwasser, M. H., & Tamassia, R. (2013). *Data structures and algorithms in Python*. Wiley.