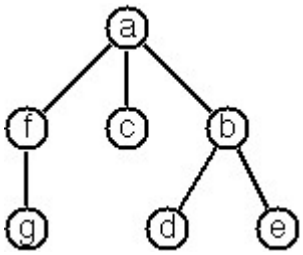
Prolog Site

Search this site

Prolog Problems >

5. Multiway Trees

Solutions can be found here.



A multiway tree is composed of a root element and a (possibly empty) set of successors which are multiway trees themselves. A multiway tree is never empty. The set of successor trees is sometimes called a forest.

In Prolog we represent a multiway tree by a term t(X,F), where X denotes the root node and F denotes the forest of successor trees (a Prolog list). The example tree depicted opposite is therefore represented by the following Prolog term:

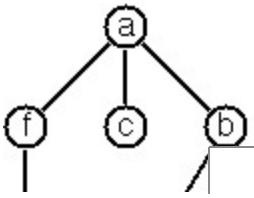
```
T = t(a,[t(f,[t(g,[])]),t(c,[]),t(b,[t(d,[]),t(e,[])])])
```

5.01 (*) Check whether a given term represents a multiway tree
Write a predicate istree/1 which succeeds if and only if its argument is a Prolog term representing a multiway tree.
Example:
?- istree(t(a,[t(f,[t(g,[])]),t(c,[]),t(b,[t(d,[]),t(e,[])])])).
Yes

5.02 (*) Count the nodes of a multiway tree
Write a predicate nnodes/1 which counts the nodes of a given multiway tree.
Example:
?- nnodes(t(a,[t(f,[])]),N).
N = 2

Write another version of the predicate that allows for a flow pattern (o,i).

5.03 (**) Tree construction from a node string
We suppose that the nodes of a multiway tree contain single characters. In the depth-first order sequence of its nodes, a special



character ^ has been inserted whenever, during the tree traversal, the move is a backtrack to the previous level.

By this rule, the tree in the figure opposite is represented as: `afg^^c^bd^e^^^`

Define the syntax of the string and write a predicate tree(String,Tree) to construct the Tree when the String is given. Work with atoms (instead of strings). Make your predicate work in both directions.

5.04 (*) Determine the internal path length of a tree

We define the internal path length of a multiway tree as the total sum of the path lengths from the root to all nodes of the tree. By this definition, the tree in the figure of problem 5.03 has an internal path length of 9.

Write a predicate ipl(Tree,IPL) for the flow pattern (+,-).

5.05 (*) Construct the bottom-up order sequence of the tree nodes

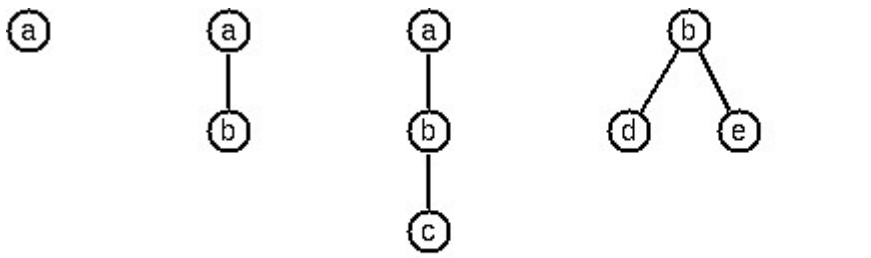
Write a predicate bottom_up(Tree,Seq) which constructs the bottom-up sequence of the nodes of the multiway tree Tree. Seq should be a Prolog list.

What happens if you run your predicate backwords?

5.06 (**) Lisp-like tree representation

There is a particular notation for multiway trees in Lisp. Lisp is a prominent functional programming language, which is used primarily for artificial intelligence problems. As such it is one of the main competitors of Prolog. In Lisp almost everything is a list, just as in Prolog everything is a term.

The following pictures show how multiway tree structures are represented in Lisp.



Note that in the "lispy" notation a node with successors (children) in the tree is always the first element in a list, followed by its children. The "lispy" representation of a multiway tree is a sequence of atoms and parentheses '(' and ')', which we shall collectively call "tokens". We can represent this sequence of tokens as a Prolog list; e.g. the lispy expression (a (b c)) could be represented as the Prolog list ['(', a, '(', b, c, ')', ')']. Write a predicate tree_ltl(T,LTL) which constructs the "lispy token list" LTL if the tree is given as term T in the usual Prolog notation.

Example:
?- tree_ltl(t(a,[t(b,[]),t(c,[])]),LTL).
LTL = ['(', a, '(', b, c, ')', ')']

As a second, even more interesting exercise try to rewrite tree_ltl/2 in a way that the inverse conversion is also possible: Given the list LTL, construct the Prolog tree T. Use difference lists.

Subpages (1): Solutions-5