11.3 Tree Traversal

Universal Address Systems

A way to totally order the vertices of and ordered rooted tree. We do this recursively:

- Label the root with the integer 0. Then label its \( k \) children (at level 1) from left to right with 1, 2, 3, \ldots, \( k \).

- For each vertex \( v \) at level \( n \) with label \( A \), label its \( k_v \) children, as they are drawn from left to right, with \( A.1, A.2, \ldots, A.k_v \).

![Diagram of a tree with labels]

The lexicographic ordering for this tree is 0 < 1 < 1.1 < 1.1.1 < 1.1.2 < 1.2 < 1.3 < 2 < 2.1 < 2.2 < 2.3 < 3 < 3.1 < 3.1.1 < 3.1.2

Traversal Algorithms

Preorder Traversal

Let \( T \) be an ordered rooted tree with root \( r \). If \( T \) consists only of \( r \), then \( r \) is the preorder traversal of \( T \). Otherwise, suppose that \( T_1, T_2, \ldots, T_n \) are the subtrees at \( r \) from left to right in \( T \). The preorder traversal begins by visiting \( r \). It continues by traversing \( T_1 \) in preorder, then \( T_2 \) in preorder, and so on, until \( T_n \) is traversed in preorder.

**Algorithm** preorder(\( T \) : ordered rooted tree)

1: \( r = \) root of \( T \)
2: list \( r \)
3: for each child \( c \) of \( r \) from left to right do
4: \( T(c) = \) subtree with \( c \) as its root
5: preorder(\( T(c) \))
6: end for

Inorder Traversal

Let \( T \) be an ordered rooted tree with root \( r \). If \( T \) consists only of \( r \), then \( r \) is the inorder traversal of \( T \). Otherwise, suppose that \( T_1, T_2, \ldots, T_n \) are the subtrees at \( r \) from left to right. The inorder traversal begins by traversing \( T_1 \) in inorder, then visiting \( r \). It continues by traversing \( T_2 \) in inorder, then \( T_3 \) in inorder, \ldots, and finally \( T_n \) in inorder.
Algorithm *inorder*(*T*: ordered rooted tree)

1: \( r = \text{root of } T \)
2: if \( r \) is a leaf then
3: \( \text{list } r \)
4: else
5: \( l = \text{first child of } r \) from left to right
6: \( T(l) = \text{subtree with } l \) as its root
7: \( \text{inorder}(T(l)) \)
8: \( \text{list } r \)
9: for each child \( c \) of \( r \) except for \( l \) from left to right do
10: \( T(c) = \text{subtree with } c \) as its root
11: \( \text{inorder}(T(c)) \)
12: end for
13: end if

Postorder Traversal

Let \( T \) be an ordered rooted tree with root \( r \). If \( T \) consists only of \( r \), then \( r \) is the *postorder traversal* of \( T \). Otherwise, suppose that \( T_1, T_2, \ldots, T_n \) are the subtrees at \( r \) from left to right. The *postorder traversal* begins by traversing \( T_1 \) in postorder, then \( T_2 \) in postorder, \ldots, then \( T_n \) in postorder, and ends by visiting \( r \).

Algorithm *postorder*(*T*: ordered rooted tree)

1: \( r = \text{root of } T \)
2: for each child \( c \) of \( r \) from left to right do
3: \( T(c) = \text{subtree with } c \) as its root
4: \( \text{postorder}(T(c)) \)
5: end for
6: \( \text{list } r \)

Infix, Prefix, and Postfix Notation

Complicated expressions, such as compound propositions, combinations of sets, and arithmetic expressions can be represented by ordered root trees. For arithmetic expressions:

- Internal vertices: operations
- Leaves: variables or numbers
Infix Notation

Inorder traversal of the binary tree representing an expression produces the original expression with the elements and operations in the same order as they originally occurred, except unary operations. The fully parenthesized expression is the *infix form*.

Infix form: \((6 - 4) \ast (3 + (5 - 1))\)

Prefix Notation

The *prefix form* of an expression can be obtained by traversing its rooted tree in preorder. Binary operator precedes its two operands. Evaluate an expression in prefix form by working right to left.

Prefix form: \(* - 64 + 3 - 51\)

Postfix Notation

The *postfix form* of an expression can be obtained by traversing its rooted tree in postorder. Binary operator follows its two operands. Evaluate an expression in postfix form by working left to right.

Postfix form: \(64 - 351 - +\ast\)

11.3 pg. 783 # 1

Construct the universal address system for the given ordered rooted tree. Then use this to order its vertices using the lexicographic order of their labels.
The lexicographic ordering is $0 < 1 < 1.1 < 1.2 < 2 < 3$.

11.3 pg. 783 # 7
Determine the order in which a preorder traversal visits the vertices of the given ordered rooted tree.

The preorder traversal is: $a, b, d, e, f, g, c$.

11.3 pg. 783 # 11
Determine the order in which an inorder traversal visits the vertices of the given ordered rooted tree.
The inorder traversal is: \(d, b, i, e, m, j, n, o, a, f, c, g, k, h, p, l\).

1.3 pg. 783 # 13

Determine the order in which a postorder traversal visits the vertices of the given ordered rooted tree.

The postorder traversal is: \(d, f, g, e, b, c, a\).
11.3 pg. 784 # 17

a) Represent the expressions \((x + xy) + (x/y)\) and \(x + ((xy + x)/y)\) using binary trees.

\[\begin{align*}
(x + xy) + (x/y) & \\
+ & \\
+ & \\
x & \\
* & \\
+ & \\
x & \\
y & \\
\end{align*}\]

\[\begin{align*}
+ & \\
x & \\
/ & \\
* & \\
x & \\
y & \\
\end{align*}\]

b) Write these expressions in prefix notation.

\((x + xy) + (x/y)\) in prefix is: \(+ + x * x y/x y\)

\(x + ((xy + x)/y)\) in prefix is: \(+x/ + *x y x y\)

c) Write these expressions in postfix notation.

\((x + xy) + (x/y)\) in postfix is: \(x x y + * + x y/ +\)

\(x + ((xy + x)/y)\) in postfix is: \(x x y * x + y/ +\)

d) Write these expressions in infix notation.

\((x + xy) + (x/y)\) in infix is: \(((x + (x * y)) + (x/y))\)

\(x + ((xy + x)/y)\) in infix is: \((x + ((x * y) + x)/y))\)

11.3 pg. 784 # 23

What is the value of each of these prefix expressions?

a) \(- * 2/8 4 3\)

\(- * 2/8 4 3\)

\(- * 2 2 3\)

\(-4 3\)

\(1\)

b) \(↑ - * 3 3 * 4 2 5\)

\(↑ - * 3 3 * 4 2 5\)

\(↑ - * 3 3 8 5\)

\(↑ -9 8 5\)

\(↑ 1 5\)

\(1\)
c) \[ + - \uparrow 3 2 \uparrow 2 3/6 \ - \ 4 2 \]
\[ + - \uparrow 3 2 \uparrow 2 3/6 \ - \ 4 2 \]
\[ + - \uparrow 3 2 \uparrow 2 3/6 \ 2 \]
\[ + - \uparrow 3 2 \uparrow 2 3 3 \]
\[ + - \uparrow 3 2 8 3 \]
\[ + - 9 8 3 \]
\[ + 1 3 \]
\[ 4 \]

d) \[ * + 3 + 3 \uparrow 3 + 3 3 3 \]
\[ * + 3 + 3 \uparrow 3 + 3 3 3 \]
\[ * + 3 + 3 \uparrow 3 6 3 \]
\[ * + 3 + 3 729 3 \]
\[ * + 3 732 3 \]
\[ * 735 3 \]
\[ 2205 \]