8.4. TREE TRANSVERSAL

8.4. Tree Transversal

8.4.1. Transversal Algorithms. In order to motivate this subject, we introduce the concept of Polish notation. Given a (not necessarily commutative) binary operation $\circ$, it is customary to represent the result of applying the operation to two elements $a, b$ by placing the operation symbol in the middle:

$$a \circ b.$$  

This is called infix notation. The Polish notation consists of placing the symbol to the left:

$$\circ a b.$$  

The reverse Polish notation consists of placing the symbol to the right:

$$a b \circ.$$  

The advantage of Polish notation is that it allows us to write expressions without need for parenthesis. For instance, the expression $a \ast (b + c)$ in Polish notation would be $\ast a + bc$, while $a b + c$ is $+ \ast a b c$. Also, Polish notation is easier to evaluate in a computer.

In order to evaluate an expression in Polish notation, we scan the expression from right to left, placing the elements in a stack. Each time we find an operator, we replace the two top symbols of the stack by the result of applying the operator to those elements. For instance, the expression $\ast + 2 3 4$ (which in infix notation is "$(2 + 3) \ast 4$") would be evaluated like this:

<table>
<thead>
<tr>
<th>expression</th>
<th>stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ast + 234$</td>
<td>4</td>
</tr>
<tr>
<td>$\ast + 23$</td>
<td>3 4</td>
</tr>
<tr>
<td>$\ast + 2$</td>
<td>5 4</td>
</tr>
<tr>
<td>$\ast + 2$</td>
<td>3 4</td>
</tr>
<tr>
<td>$\ast$</td>
<td>20</td>
</tr>
</tbody>
</table>

An algebraic expression can be represented by a binary rooted tree obtained recursively in the following way. The tree for a constant or variable $a$ has $a$ as its only vertex. If the algebraic expression $S$ is of

---

1A stack or last-in first-out (LIFO) system, is a linear list of elements in which insertions and deletions take place only at one end, called top of the list. A queue or first-in first-out (FIFO) system, is a linear list of elements in which deletions take place only at one end, called front of the list, and insertions take place only at the other end, called rear of the list.
the form $S_L \circ S_R$, where $S_L$ and $S_R$ are subexpressions with trees $T_L$ and $T_R$ respectively, and $\circ$ is an operator, then the tree $T$ for $S$ consists of $\circ$ as root, and the subtrees $T_L$ and $T_R$ (fig. 8.13).

![Figure 8.13. Tree of $S_1 \circ S_2$.]

For instance, consider the following algebraic expression:

$$a + b \times c + d \uparrow e \times (f + h),$$

where $+$ denotes addition, $\times$ denotes multiplication and $\uparrow$ denotes exponentiation. The binary tree for this expression is given in figure 8.14.

![Figure 8.14. Tree for $a + b \times c + d \uparrow e \times (f + h)$.]

Given the binary tree of an algebraic expression, its Polish, reverse Polish and infix representation are different ways of ordering the vertices of the tree, namely in preorder, postorder and inorder respectively.

The following are recursive definitions of several orderings of the vertices of a rooted tree $T = (V, E)$ with root $r$. If $T$ has only one vertex $r$, then $r$ by itself constitutes the preorder, postorder and inorder transversal of $T$. Otherwise, let $T_1, \ldots, T_k$ the subtrees of $T$ from left to right (fig. 8.15). Then:

1. **Preorder Transversal**: $\text{Pre}(T) = r, \text{Pre}(T_1), \ldots, \text{Pre}(T_k)$. 
2. *Postorder Transversal*: \( \text{Post}(T) = \text{Post}(T_1), \ldots, \text{Post}(T_k), r \).

3. *Inorder Transversal*. If \( T \) is a binary tree with root \( r \), left subtree \( T_L \) and right subtree \( T_R \), then: \( \text{In}(T) = \text{In}(T_L), r, \text{In}(T_R) \).