Pre-Order Traversal of a Tree

Idea: To visit all vertices in the subtree rooted at \( v \), \textit{pre-order} traversal uses the rule: "Visit \( v \) and then (recursively) visit the subtrees of \( v \)."

procedure \textsc{PRE}(v):
\{visit \( v \}\)
\textbf{for} each child \( w \) of \( v \) do \textsc{PRE}(w)
\textbf{end}

Example: Starting at the root, assuming children are visited in alphabetical order, pre-order visits: \( a, b, d, c, e, f, h, i, g \)
The Ordering Implied by a Traversal

Because the *for* loop does not specify the order children are visited, many traversals of the tree may satisfy the preorder and post-order definitions. In practice, a particular representation of the tree in memory will give rise to a natural "standard" order.

For example, if we are using the LMCHILD-RSIB representation, a unique traversal of the tree results from replacing the *for* loop by a *while* loop that follows the RSIB links from first to last; call this *first-to-lastPRE*:

```pascal
procedure first-to-lastPRE(v):
{visit v}
    w := LMCHILD(v)
    while (w is not nil) do begin
        first-to-lastPRE(w)
        w := RSIB(w)
    end
end
```
Post-Order Traversal

*Post-order* traversal is a recursive traversal like pre-order traversal, except a vertex is visited after visiting its children.

**Example:** Starting at the root, assuming children are visited in alphabetical order, post-order does: \(d, b, e, h, i, f, g, c, a\)
Example: Height of a Vertex

The basic recursive traversal of a tree in pre-order or post order can be adapted to perform many data gathering tasks on the tree, such as its height (number of edges on a longest path from \( v \) to a leaf).

function HEIGHT(\( v \)):

\[
    h := 0 \\
    \text{for each child } w \text{ of } v \text{ do } h := \text{MAXIMUM}(h, \text{HEIGHT}(w)+1) \\
    \text{return } h
\]

end
Level-Order Traversal

**Idea:** Visit the children of a vertex \( v \) before going deeper into the subtrees, so that vertices are visited from highest to lowest level.

```plaintext
procedure LEV(v)
    Initialize a queue to contain \( v \).
    while queue is not empty do begin
        \( v := \) DEQUEUE
        {visit \( v \})
        for each child \( w \) of \( v \) do ENQUEUE(\( w \))
    end
end
```
Example: Assuming that children are visited in alphabetical order, level order starting at the root is \(a, b, c, d, e, f, g, h, i\).
The Order Implied by A Level-Order Traversal

Like pre-order, the for loop does not specify the order in which children are visited. Again, if we are using the LMCHILD-RSIB representation, a natural unique traversal of the tree results from replacing the for loop by a while loop that follows RSIB links from left to right:

\[
\begin{align*}
  w & := \text{LMCHILD}(v) \\
  \text{while (} w \text{ is not } \text{nil} \text{) do begin} \\
  \quad \text{ENQUEUE}(w) \\
  \quad w := \text{RSIB}(w) \\
  \text{end}
\end{align*}
\]
Example: MIN-HEIGHT of a Vertex

The basic traversal of a tree in level order can be adapted to perform many data gathering tasks on the tree, such as its MIN-HEIGHT (number of edges on a shortest path from $v$ to a leaf).

function MIN-HEIGHT($v$)
  $h := 0$
  Initialize a queue to contain the pair $v,h$.
  while $v$ is not a leaf do begin
    $v,h :=$ DEQUEUE
    for each child $w$ of $v$ do ENQUEUE($w,h+1$)
  end
  return $h$
end