Basic Algorithms

Tree Algorithms, Additional Problems.

For all the problems, before writing pseudo-code, **determine what is the formula you need to compute at each node**; be precise. Also, once you have written the recursive procedure, remember that you also need an initial call to the recursive procedure.

1. Consider a tree in which each node \( v \) has a field \( v.clr \) which has the value \( R \) (red) or \( G \) (green). If \( v.clr = G \) we will say that \( v \) is green, and if \( v.clr = R \) we say that \( v \) is red. By completing the following procedure, compute, for each node \( v \) in a tree \( T \), in the field \( v.clr._cnt \), the number of green nodes in \( v \)'s subtree.

**Recursive formulation:**

\[
\text{ClrCnt}(v);
\]

\[
\text{for each child } w \text{ of } v \text{ do}
\]

\[
\text{ClrCnt}(w);
\]

\[
\text{end (* for *)}
\]

end

Initial call/Driver:
2. Use the same setting as in problem 1, but now count the number of green leaves in the subtree rooted at node \( v \), storing the result in the field \( v.grn_lvs \).

**Recursive formulation:**

\[
\text{GrnLfCnt}(v); \\
\text{for each child } w \text{ of } v \text{ do} \\
\quad \text{GrnLfCnt}(w); \\
\text{end (* for *)} \\
\text{end}
\]

**Initial call/Driver:**
3. Define the Greenness of a node $v$, $\text{grnss}(v)$ to be the number of green nodes in the subtree rooted at $v$ divided by the number of nodes in the subtree rooted at $v$.

a. Compute the greenness of each node $v$, storing the result in the field $v.\text{grnss}$, and using whatever additional fields may be useful. Remember that the values in any additional field must be computed by your procedure.

**Recursive formulation:**

```plaintext
GrnNss(v);

for each child $w$ of $v$ do

    GrnNss(w);

end (* for *)
```

end

**Initial call/Driver:**
b. For non-leaf nodes $v$, in the field $v.grnnst\_chld$ compute a pointer to the child $w$ with the maximum $w.grnnss$ value among all of $v$’s children. Try to do this with a 1-pass algorithm. The solution will require adding some additional lines of code to the solution for part a.

**Recursive formulation:**

```plaintext
GrnNstChld(v);

for each child $w$ of $v$ do

    GrnNstChld(w);

end (* for *)

end

Initial call/Driver:
4.a. A green path is defined to be a path in which all the nodes are green. For each node \( v \), in the field \( v.lngst_grn_pth \), compute the length in nodes of the longest green path descending from \( v \) (so if \( v \) is red, the length is zero).

**Recursive formulation:**

\[
\text{GrnPthLngth}(v); \\
\text{for each child } w \text{ of } v \text{ do} \\
\quad \text{GrnPthLngth}(w); \\
\text{end} (* \text{ for } *) \\
\text{end}
\]

**Initial call/Driver:**
b. For each node \( v \), in the file \( v.lngst\_alt\_pth \), compute the length in nodes of the longest path descending from \( v \) such that the colors of the nodes along the path alternate.

**Recursive formulation:**

\[
\text{LngstAltPath}(v); \\
\text{for each child } w \text{ of } v \text{ do } \\
\quad \text{LngstAltPath}(w); \\
\text{end (* for *) } \\
\text{end}
\]

**Initial call/Driver:**
5.a. For each node $v$ compute the length in nodes of the longest and second longest alternating paths descending from $v$, putting the values in the fields $v.\text{lngst}$ and $v.\text{2lngst}$ respectively. Note that for a leaf node $x$, $x.\text{2lngst} = 0$, as there is only one path descending from $x$, namely the “path” consisting of the single node $x$. To simplify your code you may assume you have a function $\text{2ndLargest}\{\cdots\}$ which takes any number of arguments and returns the second largest value; e.g. $\text{2ndLargest}\{3, 5, 7, 2\} = 3$.

**Recursive formulation:**

\begin{verbatim}
2LgstAltPth(v);

for each child $w$ of $v$ do

2LgstAltPth(w);

end (* for *)

end

Initial call/Driver:
\end{verbatim}
b. For each node \( v \), in the field \( v.lngst_grn_pth2lf \), compute the length in nodes of the longest green path descending from \( v \) to a leaf; if there is no such path, define its length to be zero. Be careful in figuring out the correct test here. Try different cases: what needs to happen when \( v \) is a leaf, and if \( v \) is not a leaf, what needs to happen if \( v \) is red, and what needs to happen if there is no suitable path in the subtree rooted at \( v \)'s child \( w \).

**Recursive formulation:**

\[
\text{LngstGrnPth2Lf}(v); \\
\text{for each child } w \text{ of } v \text{ do} \\
\quad \text{LngstGrnPth2Lf}(w); \\
\text{end (* for *)} \\
\text{end} \\
\text{Initial call/Driver:}
\]