Lists, trees, and recursive algorithms for them

Definition. $L$ is a list if either

- $L$ is \texttt{NULL}
- or $L$ is a pointer to a data structure with fields data and next, where next is a list.

Definition. $T$ is a tree if either

- $T$ is \texttt{NULL}
- or $T$ is a pointer to a data structure with fields data and children, where children is an array (alternatively, a list) of trees.

Lists

\textbf{In} List $L$

\textbf{Out} The sum of all data-values in $L$

\begin{verbatim}
function SumIterative(L)
    result = 0
    while $L \neq \texttt{NULL}$ do
        result = result + $L$.data
        $L = L$.next
    end while
    return result
end function
\end{verbatim}

\begin{verbatim}
function SumRecursive(L)
    if $L == \texttt{NULL}$ then
        return 0
    end if
    return $L$.data + SumRecursive($L$.next)
end function
\end{verbatim}
Trees

**In** Tree $T$

**Out** The sum of all data-values in $T$

```
function Sum(T)
    if T == NULL then
        return 0
    end if
    result = T.data
    for C in T.children do
        result = result + Sum(C)
    end for
    return result
end function
```

Note that this scheme can be easily adapted to compute min / max, product, and other similar functions. For example, see the following code for min.

**In** Tree $T$

**Out** The minimum of all data-values in $T$

```
function Min(T)
    if T == NULL then
        return $\infty$
    end if
    result = T.data
    for C in T.children do
        result = min(result, Min(C))
    end for
    return result
end function
```

Sometimes, one has to compute more than needed during recursion in order to solve the problem. Consider the following problem

**In** Tree $T$

**Out** The second smallest data-value in $T$ (or $\infty$ if there are less than two vertices in $T$)

It can be solved using the following recursive algorithm
In Tree $T$

Out The pair $(a, b)$, where $a$ and $b$ are the smallest and the second smallest data-values in $T$, respectively

function TwoMin(T)
    if $T == NULL$ then
        return $(\infty, \infty)$
    end if
    min = $T$.data
    second_min = $T$.data
    for $C$ in $T$.children do
        $(Cmin, Csmin) = \text{TwoMin}(C)$
        second_min = min(max(min, Cmin), second_min, Csmin)
        min = min(min, Cmin)
    end for
    return (min, second_min)
end function