In-Class Problems Week 5, Fri.

Problem 1.

**Definition.** The recursive data type binary-2PG of *binary trees* with leaf labels $L$ is defined recursively as follows:

- **Base case:** $(\text{leaf,} l) \in \text{binary-2PG}$, for all labels $l \in L$.
- **Constructor case:** If $G_1, G_2 \in \text{binary-2PG}$, then
  $$(\text{bintree,} G_1, G_2) \in \text{binary-2PG}.$$ 

The *size* $|G|$ of $G \in \text{binary-2PG}$ is defined recursively on this definition by:

- **Base case:**
  $$|\{\text{leaf,} l\}| ::= 1, \quad \text{for all } l \in L.$$ 
- **Constructor case:**
  $$|\{\text{bintree,} G_1, G_2\}| ::= |G_1| + |G_2| + 1.$$ 

For example, the size of the binary-2PG $G$ pictured in Figure 1, is 7.

![Figure 1](image_url) 

**Figure 1** A picture of a binary tree $G.$
(a) Write out (using angle brackets and labels \texttt{bintree}, \texttt{leaf}, etc.) the binary-2PG $G$ pictured in Figure 1.

The value of $\text{flatten}(G)$ for $G \in \text{binary-2PG}$ is the sequence of labels in $L$ of the leaves of $G$. For example, for the binary-2PG $G$ pictured in Figure 1,

$$\text{flatten}(G) = \langle \text{win, lose, win, win} \rangle.$$

(b) Give a recursive definition of $\text{flatten}$. (You may use the operation of \textit{concatenation} (append) of two sequences.)

(c) Prove by structural induction on the definitions of $\text{flatten}$ and size that

$$2 \cdot \text{length}(\text{flatten}(G)) = |G| + 1. \quad (1)$$

\textbf{Problem 2.}

For $T \in \text{BBTr}$, define

$$\text{leaves}(T) ::= \{S \in \text{Subtrs}(T) \mid S \in \text{Leaves}\}$$

$$\text{internal}(T) ::= \{S \in \text{Subtrs}(T) \mid S \in \text{Branching}\}.$$

(a) Explain why it follows immediately from the definitions that if $T \in \text{Branching},$

$$\text{internal}(T) = \{T\} \cup \text{internal(left}(T)) \cup \text{internal(right}(T)),$$

$$\text{leaves}(T) = \text{leaves(left}(T)) \cup \text{leaves(right}(T)). \quad \text{(trnlT)} \quad \text{(lVT)}$$

(b) Prove by structural induction on the definition of $\text{RecTr}$ that in a recursive tree, there is always one more leaf than there are internal subtrees:

\textbf{Lemma.} If $T \in \text{RecTr}$, then

$$|\text{leaves}(T)| = 1 + |\text{internal}(T)|. \quad \text{(If-vs-in)}$$

\textbf{Problem 3.}

\textbf{Definition.} Define the \textit{sharing binary trees} $\text{SharTr}$ recursively:

\textbf{Base case:} ($T \in \text{Leaves}$). $T \in \text{SharTr}$.

\textbf{Constructor case:} ($T \in \text{Branching}$). If $\text{left}(T), \text{right}(T) \in \text{SharTr}$, then $T$ is in $\text{SharTr}$.

(a) Prove size $(T)$ is finite for every $T \in \text{SharTr}$.

(b) Give an example of a finite $T \in \text{BBTr}$ that has an infinite path.

(c) Prove that for all $T \in \text{BBTr}$

$$T \in \text{SharTr} \iff T \text{ has no infinite path.}$$

(d) Give an example of a tree $T_3 \in \text{BBTr}$ with three branching subtrees and one leaf.

(e) Prove that

\textbf{Lemma.} If $T \in \text{SharTr}$, then

$$|\text{leaves}(T)| \leq 1 + |\text{internal}(T)|.$$
Hint: Show that for every $T \in \text{SharTr}$, there is a recursive tree $R \in \text{RecTr}$ with the same number of internal subtrees and at least as many leaves.

**Problem 4.** (a) Edit the labels in this size 15 tree $T$ so it becomes a search tree for the set of labels $[1..15]$.

![Tree Diagram]

(b) For any recursive tree and set of labels, there is only one way to assign labels to make the tree a search tree. More precisely, let $\text{num} : \text{RecTr} \rightarrow \mathbb{R}$ be a labelling function on the recursive binary trees, and suppose $T$ is a search tree under this labelling. Suppose that $\text{num}_{\text{alt}}$ is another labelling and that $T$ is also a search tree under $\text{num}_{\text{alt}}$ for the same set of labels. Prove by structural induction on the definition of search tree that

$$\text{num}(S) = \text{num}_{\text{alt}}(S)$$

for all subtrees $S \in \text{Subtrs}(T)$.

Reminder:

**Definition.** The Search trees $T \in \text{BBTr}$ are defined recursively as follows:

**Base case:** ($T \in \text{Leaves}$). $T$ is a Search tree.

**Constructor case:** ($T \in \text{Branching}$). If $\text{left}(T)$ and $\text{right}(T)$ are both Search trees, and

$$\max(\text{left}(T)) < \text{num}(T) < \min(\text{right}(T)),$$

then $T$ is a Search tree.