Structural and tree Induction

Margaret M. Fleck

3 April 2009

Here are some written-up examples of structural and tree induction, to go with lecture 27.

1 Trees

A *full binary tree* is a tree in which each node has either two children or no children. Full binary trees (and their roots) can be defined recursively as follows:

- A single node is a full binary tree (its root).
- Suppose X and Y are full binary trees. Define a new tree $X \cdot Y$ which contains all of X and Y, plus a new root node x. x is connected by edges to two children: the root notes of X and Y. Then $X \cdot Y$ is also a full binary tree.

The top node in the tree is known as the *root*. Nodes with no children are known as *leaves*. The height of a tree is the number of edges in the longest path from one of the leaf nodes to the root.

Suppose we store numbers in the nodes of a full binary tree. The numbers obey the *heap property* if, for every node X in the tree, the value in X is at least as big as the value in each of X's children. I claim that:

Claim 1 *If a tree has the heap property, then the value in the root of the tree is at least as large as the value in any other node of the tree.*

Let's let $v(X)$ be the value at node X and let's use the recursive structure of trees to do our proof.

Proof by structural induction.

Base: If a tree contains only one node, obviously the largest value in the tree lives in the root!

Induction: Suppose that the claim is true for trees X and Y. We need to show that the claim is also true for $T = X \cdot Y$.

Let p, q, and r be the root nodes of X, Y, and T. p and q are the children of r. Since T has the heap property, $v(r) \geq v(p)$ and $v(r) \geq v(q)$. By the inductive hypothesis, $v(p)$ is $\geq$ the value in each node of X and $v(q)$ is $\geq$ the value in each node of Y. So $v(r) \geq$ the value in each node of X and each node of Y. The nodes of T are the nodes of X, plus the nodes of Y, plus r. So $v(r)$, the value at the root node of T, is at least as big as the value at any node of T.

In the inductive step, notice that we split up the big tree (T) at its root, producing two smaller subtrees (X) and (Y). Some students try to do induction on trees by grafting stuff onto the bottom of the tree. This frequently does not work, especially as you get to examples in more advanced courses. Therefore, we will take off points if you do it on homework or tests.

2 An example with 2D points

Consider the following recursive definition of a set S of 2D points:

1. $(3, 5) \in S$
2. If $(x, y) \in S$, then $(x + 2, y) \in S$
3. If $(x, y) \in S$, then $(-x, y) \in S$
4. If $(x, y) \in S$, then $(y, x) \in S$

What's in S? Starting with the pair specified in the base case $(3, 5)$, we use rule 3 to add $(-3, 5)$. Rule 2 then allows us to add $(-1, 5)$ and then $(1, 5)$. If we apply rule 2 repeatedly, we see that S contains all pairs of the form $(2n + 1, 5)$ where x is a natural number. By rule 4, $(5, 2n + 1)$ must also be in S for every natural number n.

We then apply rules 2 and 3 in the same way, to show that $(2m+1, 2n+1)$ is in S for every natural numbers m and n. We could make this more formal using a proof by induction, but let's not bother.

The big missing piece of this analysis is showing that S doesn't contain any other points other than those with odd coordinates. Think of this as proving a set equality. Let T be the set of 2D points with odd coordinates. We've shown that $T \subseteq S$ and we still need to show that $S \subseteq T$. We can do this by structural induction.

> Proof by structural induction that all elements of S have both coordinates odd.
>
> Base: Both coordinates of $(3, 5)$ are odd.
>
> Induction: Suppose that both coordinates of (x, y) are odd. We need to show that both coordinates of $(x + 2, y)$, $(-x, y)$, and (y, x) are odd. But this is (even in the context of this course), obvious.

In this case, the second half of the proof was very easy because we used structural induction. Two things were important. First, it was critical that we did do this second half, otherwise we couldn't claim to have completely nailed down what's in S. Second, we needed generate the outline of structural induction and verify the base case and inductive step.

3 Another tree example

Here's another example of structural induction on trees.

Let T be a full binary tree. Suppose that $n(T)$ is the number of nodes in T and $h(T)$ is the height of T. Let's show that $n(T) \leq 2^{h(T)+1} - 1$.

> Proof by structural induction.
>
> Base: Trees containing a single node have no edges and therefore height 0. Notice that $n(T) = 1$ and $2^{h(T)+1} - 1 = 2^1 - 1 = 1$.
>
> Induction: Let X and Y be full binary trees. Suppose that the claim is true for X and Y. We need to show that the claim is also true for $T = X \cdot Y$.

The number of nodes in T is $n(T) = 1+n(X)+n(Y)$. Because the claim holds for X and Y, we know that this is $\leq 1 + (2^{h(X)+1} - 1) + 2^{h(Y)+1} - 1 = (2^{h(X)+1} + 2^{h(Y)+1}) - 1$. This, in turn, is $\leq 2 \cdot \max(2^{h(X)+1}, 2^{h(Y)+1}) - 1 = 2 \cdot 2^{\max(h(X),h(Y))+1} - 1$ So $n(T) \leq 2 \cdot 2^{\max(h(X),h(Y))+1} - 1$

Now, the longest path down from the root of T is the longest path down from the roots of X and Y, plus one more edge. That is, $h(T) = 1 + \max(h(X), h(Y))$.

So $n(T) \leq 2 \cdot 2^{\max(h(X),h(Y))+1} - 1 = 2 \cdot 2^{h(T)} - 1 = 2^{h(T)+1} - 1$