

Mid-semester test notes.

What was being tested?

- Writing simple recursive programs for tree data-structures;
- An awareness of complexity issues (eg what is the worst-case complexity, and how is it affected by the shape of the tree);
- An understanding of the Dynamic Programming Technique and how to apply it.

The height of a tree:

$$\text{height}(t) = 1 + \text{maximum} \{ \text{height}(t \rightarrow \text{LChildPtr}), \text{height}(t \rightarrow \text{RChildPtr}) \}$$
$$\text{height}(\text{NULL}) = 0$$

maximum value in a binary tree:

$$\text{maxVal}(t) = \text{maximum} \{ t \rightarrow \text{item}, \text{maxVal}(t \rightarrow \text{LChildPtr}), \text{maxVal}(t \rightarrow \text{RChildPtr}) \}$$
$$\text{maxVal}(\text{NULL}) = -1$$

These are easy to turn into recursive functions.

Note that in both cases the complexity is $O(n)$, and that the complexity does not depend on the shape of the tree, since (for height) all subtrees must be examined, and (for maxVal) all items must be examined to see if they are the maximum.

Common mistakes:

- The results of recursive calls were not used;
- The wrong type was returned, or no value returned at all;
- The 1+ was missed out (in height);
- The current item was not included in the maximum (in maxVal);
- Some people assumed a binary search tree.

The sum of items in a binary tree satisfying a condition:

$$\text{sumBetween}(t, a, b) = t \rightarrow \text{item} +$$
$$\text{sumBetween}(t \rightarrow \text{LChildPtr}, a, b) + \text{sumBetween}(t \rightarrow \text{RChildPtr}, a, b),$$

if $t \rightarrow \text{item}$ satisfies the condition,

$$\text{sumBetween}(t, a, b) =$$
$$\text{sumBetween}(t \rightarrow \text{LChildPtr}, a, b) + \text{sumBetween}(t \rightarrow \text{RChildPtr}, a, b),$$

if $t \rightarrow \text{item}$ does not satisfy the condition

$$\text{sumBetween}(\text{NULL}, a, b) = 0$$

This is now easy to turn into a recursive function.

Note that the complexity is $O(n)$, and that the complexity does not depend on the shape of the tree, since each and every item must be examined to see whether it satisfies the condition.

Common mistakes:

- The results of recursive calls were not used;
- The wrong type was returned, or no value returned at all;
- The case analysis was not done on $t \rightarrow \text{item}$.

The product of items in a binary tree satisfying a condition:

$$\text{productBetween}(t, a, b) = t \rightarrow \text{item} + \\ \text{productBetween}(t \rightarrow \text{LChildPtr}, a, b) + \\ \text{productBetween}(t \rightarrow \text{RChildPtr}, a, b), \\ \text{if } t \rightarrow \text{item} \text{ satisfies the condition,}$$
$$\text{productBetween}(t, a, b) = \\ \text{productBetween}(t \rightarrow \text{LChildPtr}, a, b) + \\ \text{productBetween}(t \rightarrow \text{RChildPtr}, a, b), \\ \text{if } t \rightarrow \text{item} \text{ does not satisfy the condition}$$
$$\text{productBetween}(\text{NULL}, a, b) = 1$$

This is now easy to turn into a recursive function.

Note that the complexity is $O(n)$, and that the complexity does not depend on the shape of the tree, since each and every item must be examined to see whether it satisfies the condition.

Common mistakes:

- The results of recursive calls were not used;
- The wrong type was returned, or no value returned at all;
- The case analysis was not done on $t \rightarrow \text{item}$.