Tree Induction, Big-O, and Algos Questions — CS 173 Staff

Last Edited on 7/14/24 at 15:00

Problem 1

The Fibonacci trees T_n are a special sort of binary tree defined recursively as follows.

1. T_1 and T_2 are binary trees with only a single vertex.
2. For any $n \geq 3$, T_n consists of a root node with T_{n-1} as its left subtree and T_{n-2} as its right subtree.

Use induction to prove that the height of T_n is $n-2$, for any $n \geq 2$.

Here is some scaffolding to help you get started.

Solution: We prove by induction. Our induction variable is ______ and it represents the ____________ of/in the tree. *We are proving a property of certain types of trees, so use a property of trees as your induction variable.*

Base Case(s): *How many do you need? Remember, the number depends on your inductive step. So don't feel afraid to come back and revise this.*

Induction Hypothesis: *Be specific, don't just refer to "the claim." Similar to the base cases, part of this depends on your inductive step. So come back to this as needed and rewrite.*

Inductive Step: *Make sure it's clear where you use the inductive hypothesis!*

Problem 2

Here is a grammar G with start symbol S and terminal symbols a and b:

$$S \rightarrow \varepsilon \mid a\ S\ b\ S \mid b\ S\ a\ S$$

For a string s, let $A(s)$ denote the number of a's in s and similarly let $B(s)$ denote the number of b's in s. Use induction to prove that any string s with $A(s) = B(s)$, that is any string with an equal number of a's and b's, can be generated by G. You may use the following fact (and the equivalent fact swapping a's and b's) without proof:

Fact: Suppose the number of a's in a string s is one more than the number of b's in s. Then we can divide s into a string $s = xay$ where x and y are strings such that x, and therefore also y, both have an equal number of a's and b's.

The same scaffolding as above will also help you get started!

Problem 3

According to the $\ll$ ordering, order the following functions:

$$(2n)!, 2^n, 5n, 3^n, 12\log(n), n^3, n\log(n), n^2\log(n), n^2$$

Problem 4

Given the following recurrence, fill in the following key facts. Assume that n is a power of 4.

$$T(1) = 7$$
$$T(n) = 2T\left(\frac{n}{4}\right) + n$$

1. The height:
2. The number of leaves:
3. Total work (sum of the nodes) at level k:

Problem 5

Prof. Flitwick claims that for any functions f and g from the reals to the reals whose output values are always >1, if $f(x) \ll g(x)$ then $\log(f(x)) \ll \log(g(x))$. Is this true? Briefly justify your answer.

Problem 6

Find the big-Θ run time of the following recurrence:

$$T(1) = c$$
$$T(n) = 2T\left(\frac{n}{4}\right) + dn^2$$

Problem 7

Consider the following algorithm. Suppose that removing the k-th element of a list takes $O(k)$ time.

```
CHURN(a_1, ..., a_n: list of real numbers, n ≥ 2):
    if (n = 2):
        return |a_1 − a_2|
    else:
        best = −∞
        for k = 1 to n:
            val = CHURN(a_1, ..., a_{k−1}, a_{k+1}, ..., a_n)
            best = max(best, val)
        return best
```

(i) Give an English description of what CHURN computes. *Stuck? Try on a short input of a few integers.*

(ii) Let $T(n)$ denote the runtime of CHURN on a list of size n. Give a recursive definition of $T(n)$.

Stuck? First find the big-Θ run time of each line. Identify which portions are the recursive and non-recursive work and combine that into a recurrence.

(iii) Draw the first 2-3 levels of a recursion tree of $T(n)$.

(iv) How many leaves are there in the recursion tree for $T(n)$? What is the total work of the leaf nodes?