CS 624: Analysis of Algorithmns

Assignment 2

Due: Monday, February 8, 2021

1. Exercise 2.1 in the Lecture 3 handout (on page 5 of the handout).
2. Exercise 6.5-8 (page 166). Note that the problem asks you to give an algorithm that runs in $O(\log n)$ time. So you not only have to give the algorithm, you also have to show that it really does run in $O(\log n)$ time.
3. Exercise 6.5-9 (page 166).
4. Exercise 6.1 in the Lecture 3 handout (on page 13 of the handout).
5. Exercise 7.3-2 (page 180).
6. (This is only for extra credit.) Problem 7-4 (page 188). This requires careful explanation. It's the kind of reasoning that is very important in software engineering.
7. Prove that if a binary tree has depth n, then it has at most 2^n leaves.

   Please be careful about this. The result is not "obvious". And you certainly can't assume that the tree is "all filled out[1]", or that a tree that is "all filled out" has the maximum possible number of leaves. That's in fact what you are trying to prove, so you can't assume it is true.

   There are several ways to prove this. They all involve using mathematical induction. I think the simplest way is as follows, and this is how I want you to do it (and **please** don't try to do it any other way):

   (a) The inductive hypothesis should be a sequence of statements, one for each possible depth of the tree. For instance,

   - *If a binary tree has depth 5, then it has ...*

   Write down a few of these statements, each one completely (i.e., without "...").

   (b) Now finish writing the proof by induction, using these statements as the inductive hypothesis. The proof is not hard, but you really have to be careful.

   And let me just say it once again: you can **not** assume that any of the trees you are considering is "filled out completely". If you do, the proof is automatically wrong, and I won't even read it.

   (c) And one more thing to be careful about. I have seen students do something like this: if we assume that the inductive hypothesis is true for some number k, then we can take a binary tree of depth k and add one more node (carefully!) to get a binary tree of depth $k + 1$. And we can then continue the reasoning from there.

   This will not work. Why? Please write an explanation of why this reasoning would not constitute a proof.

[1] Our text calls a binary tree that is "all filled out" a *complete* binary tree. However, this term is not entirely standard.