

QUIZ 2

Introduction to Computer Science (COMP 250)

Fri. March 18, 2011

Professor Michael Langer

STUDENT NAME: _____ ID: _____

1. (4 points)

Solve the recurrence:

$$t(n) = t\left(\frac{n}{2}\right) + n + 2.$$

You may assume n is a power of 2, and $t(1) = 0$.

SOLUTION:

$$t(n) = t\left(\frac{n}{2}\right) + n + 2 \quad (1)$$

$$= t\left(\frac{n}{4}\right) + \frac{n}{2} + 2 + n + 2 \quad (2)$$

$$= t\left(\frac{n}{8}\right) + \frac{n}{4} + 2 + \frac{n}{2} + 2 + n + 2 \quad (3)$$

$$= t\left(\frac{n}{2^k}\right)n + \frac{n}{2} + \dots + \frac{n}{2^{k-1}} + 2k \quad (4)$$

When $k = \log(n)$, we get:

$$= t(1) + n + \frac{n}{2} + \dots + \frac{n}{2^{k-1}} + 2\log(n) \quad (5)$$

$$= n + \frac{n}{2} + \dots + 4 + 2 + 2\log(n) \quad (6)$$

$$= \sum_{i=1}^{\log n} 2^i + 2\log(n) \quad (7)$$

$$= (2^{\log n+1} - 1)/(2 - 1) - 1 + 2\log(n) \quad (8)$$

$$= 2n - 2 + 2\log(n) \quad (9)$$

Marking scheme:

- 1 point if you are correct up to (2).
- 2 points if you are correct up to (5).
- 3 points if you are correct up to (7).
- 4 points if you are correct up to (8). We did not take off marks if you stopped there.

If you didn't make it to (2), but you did get the term $2 + 2 + 2 + \dots + 2 = 2\log n$, then we gave 1 point.

2. (5 points)

- (a) State the formal definition of “ $t(n)$ is $\Omega(g(n))$ ”.
- (b) Prove that $t(n)$ is $\Omega(n^2)$, where

$$t(n) = \frac{n^2}{2} + 3 \log n - 40.$$

SOLUTION:

- (a) From the notes: the sequence $t(n)$ is $\Omega(g(n))$ if there exists two positive numbers n_0 and c such that, for all $n > n_0$, $t(n) \geq c g(n)$.
- (b) There were two ways to do it. The first way:

$$t(n) = \frac{n^2}{2} + 3 \log n - 40 \tag{10}$$

$$\geq \frac{n^2}{2} - 40 \text{ for } n \geq 1 \tag{11}$$

Since we are looking for a lower bound, let's try a constant $c < \frac{1}{2}$, specifically take $c = \frac{1}{4}$. We want to find an n_0 such that, for all $n \geq n_0$,

$$\frac{n^2}{2} - 40 \geq \frac{n}{4} \tag{12}$$

or equivalently

$$\frac{n^2}{4} \geq 40 \tag{13}$$

We see $n_0 = 13$ does the job, since $13^2 = 169 > 160 = 4 * 40$.

Marking Scheme:

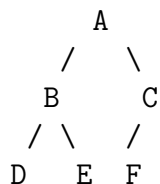
- 2 points for (a):
 - 1 point taken off if missing the “there exists” quantifier on c or n_0 , but it was otherwise correct
 - 1 point taken off if missing the “for all $n \geq n_0$ ” quantifier, but otherwise correct.
- 3 points for (b):
 - 1 point for (10), i.e. dropping the $3 \log n$ term
 - 1 point for choosing a $c < \frac{1}{2}$
 - 1 point for solving for n_0 .

There was a second way to do it, which was to guess $c = \frac{1}{2}$ and then find an n_0 such that $3 \log n - 40 > 0$ for all $n > n_0$. Choosing $n_0 = 2^{\frac{40}{3}}$ does the job.

3. (5 points)

(a) Draw the binary tree whose in-order traversal is DBE AFC and whose pre-order traversal is ABDECF.

SOLUTION:

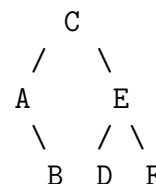
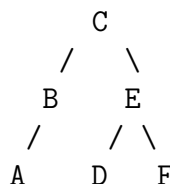
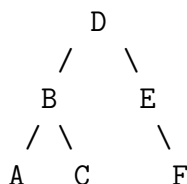
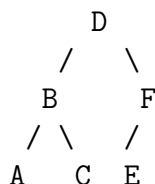


(b) What is the *post-order* traversal of this tree?

SOLUTION: DEBFCA

(c) Draw all *binary search trees* of height 2 that can be made from *all* the letters ABCDEF, assuming the natural ordering.

SOLUTION:



Marking Scheme:

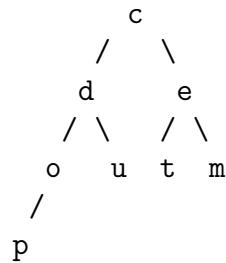
- 2 points for (a).
- 1 point for (b). If you got (a) wrong but you gave a tree with all letters, then we gave you the point here if you gave the correct postorder traversal.
- 2 points for (c).

4. (6 points)

Recall that a *heap* is a *complete binary tree* such that each node is less than its children.

- (a) Use the `upHeap()` method to build a heap out of the characters of the word: `computed`. (If you don't know that method, then use your own for part marks only.)

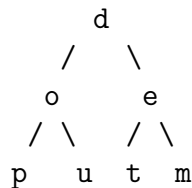
SOLUTION: (2 point)



There are other possible solutions. (This is the solution if you use the `upHeap` method.)

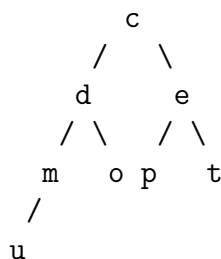
- (b) What is the result of applying the `removeMin()` method to the heap in (a) ?

SOLUTION: (1 point)



- (c) Give an example of a heap that uses the characters of the word `computed`, and that has the following additional property: for any node with two children, the left child is less than the right child.

SOLUTION: (2 points)



The easiest way to solve this one is to sort the elements from smallest to largest. The resulting sequence, if written as a array, defines a heap.

- (d) Give an example of a complete binary tree that contains the characters of the word `computed` and is also a *binary search tree*. (Such a tree is not a heap.)

SOLUTION: (1 point)

