MA/CSSE 473 – Design and Analysis of Algorithms

Homework 8 50 points total Updated for Summer, 2015

Submit to HW 08 drop box

When a problem is given by number, it is from the Levitin textbook. 1.1.2 means "problem 2 from section 1.1"

Problems for enlightenment/practice/review/challenge (not to turn in, but you should think about them):

How many of them you need to do serious work on depends on you and your background. I do not want to make everyone do one of them for the sake of the (possibly) few who need it. You can hopefully figure out which ones you need to do.

4.4.5 [4.3.5] (binary search linked list?)
5.3.2 [4.4.2] (LeafCounter algorithm)
5.5.1 [4.6.1] (one-dimensional closest-pair divide-and-conquer)
5.5.4 [4.6.3] (implement closest-pair divide-and-conquer)
5.5.9 [4.6.8] (case that leads to O(n^2) behavior for quickhull)
5.5.12 [4.6.10] (reasonably efficient shortest-path)

Which permutation immediately follows 37246510 in lexicographic order? Show how you use the algorithm from class to get your answer.

Problems to write up and turn in:

Note that my python code for Euclid, Extended Euclid, and modexp are linked from the schedule page. Feel free to use them.

1. (15) 5.3.8 [4.4.7] (Construct binary tree from inorder and postorder traversals) **See details below**. Like most problems in this course, this can be a pencil-and-paper exercise, although you are welcome to write actual code if you wish.
2. (5) 5.3.11 [4.4.10] (Chocolate bar puzzle)
3. (5) 5.4.9 [4.5.9] (Analyze Pan's matrix multiplication algorithm)
4. (10) 5.5.3 [4.6.2] (Recurrence/analysis for simpler divide-and-conquer closest points algorithm)

   To solve recurrence, try backwards substitution
   (review of this technique: pages 481-482 [475-476]).
5. (5) 5.5.7 [4.6.6] (Find p_{max} analytically)

These problems (or ones like them) would be on the midterm exam if we had one:

6. (5) If the permutations of the numbers 0-7 are numbered from 0 to 8!-1, what is the (lexicographic ordering) sequence number of the permutation 37246510? **Show how you get it.**
   Example sequence numbers: 01234567 has sequence number 0, 01234576 has sequence number 1, 01234657 has sequence number 2, …, 76543210 has sequence number 8! - 1.

7. (5) Which permutation of 01234567 has sequence number 25000 (zero-based) in lexicographic order? Show how you get it.

Details for 5.3.8 [4.4.7]

(a) 3 points. Instead of drawing the tree, you can simply list the order of its preorder traversal.
(b) 2 points. Come up with as small an example as you can.
(c) 10 points. I am changing the input and output specifications from what is given in the problem. The elements in the tree will be characters, not numbers. The same character cannot appear in two different nodes of the tree. An input to the algorithm will be one (even-length) string. The first half of the string is the inorder traversal of a binary tree, and the second half of the string is the postorder traversal of the same tree. Output should be the preorder traversal of the tree.

For definiteness, I will show you my top-level Python code and its output. You can use my code, adapt it to another language, start from scratch in any language, or simply write very clear pseudocode. You must present your algorithm in a way that makes it easy for the grader to determine whether it is correct. Include in your submission at least your code or pseudocode for `buildTree`, and the preorder traversal of the tree built from the string `'itfwGLOAIRsHMTySehtfiGOLIRAwMHyheSTs'`.

Note: At first this may appear to be very complicated, but it does not have to be so. The body of my buildTree function is two lines of code; one of them is a call to the recursive function that actually builds the tree. My recursive procedure's body is 6 lines; preOrder's body is two lines.

Top-level code:

```
def processTraversalString(s):
    try:
        print (preOrder(buildTree(s)))
    except ValueError:
        print ('Inconsistent traversal strings')

processTraversalString('OCDCOD')
processTraversalString('PODYEPOYED')
processTraversalString('ASBLUFHSALUFHB')
processTraversalString('URPYMGUYPMRG')
processTraversalString('bctdcbda')
processTraversalString('BCADECEABD')
processTraversalString('XYTMPAUCLFJKYMAUPTXJFLKC')
```

Output:

```
DOC
DOPEY
BASHFUL
GRUMPY
Inconsistent traversal strings
Inconsistent traversal strings
CXTYPMUAKLFJ
```