MA/CSSE 473 – Design and Analysis of Algorithms

Homework 7 78 points total

Submit to HW 07 drop box

When a problem is given by number, it is from the 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.

4.2.1 (quicksort example)
4.2.4 (quicksort sentinel)
4.2.6 (increasing arrays in quicksort)
4.3.5 (binary search linked list?)
4.4.2 (LeafCounter algorithm)
4.6.1 (one-dimensional closest-pair divide-and-conquer)
4.6.3 (implement closest-pair divide-and-conquer)
4.6.8 (case that leads to $O(n^2)$ behavior for quickhull)
4.6.10 (reasonably efficient shortest-path)

Problems to write up and turn in:

1. (6) 4.2.2 (Quicksort partition scan properties)
2. (10) 4.2.7 (average case for quicksort) Feel free to look up a solution, understand it, and write it in your own words (and symbols). The Weiss Data Structures book (Section 8.6.2) is one possible source. You should write a reasonable amount of detail.
3. (6) 4.2.8 (Negatives before positives)
4. (8) 4.2.9 (Dutch National Flag) [do it with a one-pass algorithm if you can]
5. (8) 4.2.11 (nuts and bolts). In addition to writing the algorithm, write and try to solve a recurrence for average-case efficiency.
6. (15) 4.4.7 (Construct binary tree from inorder and postorder traversals) **See details below**.
7. (5) 4.5.9 (Analyze Pan's matrix multiplication algorithm)
8. (10) 4.6.2 (Recurrence/analysis for simpler divide-and-conquer closest points algorithm)
   To solve the recurrence, try backwards substitution (a review of this technique is on pages 475-476)
9. (5) 4.6.6 (Find p_{max} analytically)
10. (5) 4.4.10 (Chocolate Bar)

Details for 4.4.7

(a) 3 points. Instead of drawing the tree, 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. Your job is to 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
```