MA/CSSE 473 – Design and Analysis of Algorithms

Homework 10 (~~84~~ points total)

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 (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.

6.1.2	(closest numbers in an array with pre-sorting)
6.1.3	(intersection with pre-sorting)
6.1.10	(open intervals common point)
6.1.11	(anagram detection)
6.2.8ab	(Gauss-Jordan elimination)
6.3.9	(Range of numbers in a 2-3 tree)
6.5.3	(efficiency of Horner's rule)
6.5.4	(example of Horner's rule and synthetic division)

Problems to write up and turn in:

1. (20) 5.6.10a (moldy chocolate) This problem may be harder than it looks at first.
   Perhaps it is no accident that the problem is in section 5.6, and also very close to Chapter 6.
   "Transform and conquer" is a good way to find a complete solution. However,
   If you can't solve the general case, solve some cases that you can solve, and talk about what you tried for other cases.
2. (10) 6.1.7 (to sort or not to sort)
3. (10) 6.2.8c (compare Gaussian elimination to Gauss-Jordan)
4. (6) 6.3.7 (2-3 tree construction and efficiency)
5. (25) (sum of heights of nodes in a full tree) In this problem, we consider completely full binary trees with N nodes and height H (so that $N = 2^{H+1} - 1$)

   (a) (5 points) Show that the sum of the heights of all of the nodes of such a tree can be expressed as
   $$\sum_{k=0}^{H} k\, 2^{H-k} \; .$$
   (b) (15 points) Prove by induction on H that the above sum of the heights of the nodes is N - H - 1. You may base your proof on the summation from part (a) (so you don't need to refer to trees at all), **or** you may do a "standard" binary tree induction based on the heights of the trees, using the definition that a non-empty binary tree has a root plus left and right subtrees. I find the tree approach more straightforward, but you may use the summation if you prefer.
   (c) (3 points) What is the sum of the *depths* of all of the nodes in such a tree?
   (d) (2 points) How does this apply to Heapsort analysis?
   Example of height and depth sums: Consider a full tree with height 2 (7 nodes).
   Heights: root:2, leaves: 0. Sum of all heights: 1*2 + 2*1 + 4*0 = 4.
   Depths: root: 0, leaves: 2. Sum of all depths: 1*0 + 2*1 + 4*2 = 10.
6. (10) 6.4.11 (spaghetti sort)
7. (3) 6.5.9 (Use Horner's rule for this particular case?)