MA/CSSE 473 – Design and Analysis of Algorithms

Homework 6 (includes an implementation problem) 36 + 40 points total

These are to be turned in as hard copy. You can write solutions out by hand, or write them on your computer and print them. If there are multiple pages, please staple them together.

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.

4.1.1 (divide-and-conquer array max for unsorted array)
4.1.2 (divide-and-conquer array max/min for unsorted array)
4.1.4 (logarithm base in the Master Theorem)
4.1.5 (Simple application of the Master Theorem)
4.1.7 (Mergesort stability)
4.1.9 (O(n log n) algorithm to count inversions in an array)

Problems to write up and turn in:

1. (6) (RSA attacks) Read about various ways of attacking the RSA cryptosystem.
   Write about two attacks that interest you. Explain how they work.
   One place you can look is http://en.wikipedia.org/wiki/Rsa ,
2. (15) (Miller-Rabin test) Let N = 561. For how many choices of **a** is $a^{560} \equiv 1 \pmod{561}$? How many of these liars are "outed" by the Miller-Rabin test.? [Hint: writing a few lines of code my help you here].
3. (15) (RSA decoding). If small primes are used, it is computationally easy to "crack" RSA codes.
   Suppose my public key is N=703, e= 41. You intercept an encrypted message intended for me, and the encrypted message is 361. What was the original message?

Implementation problem

(40) 3.4.9 (Magic squares)

You may work with another student on this one if you wish. As before, if you don't want to work alone but do not know who to work with, let me know before Monday's class and I'll try to help you find someone.

Your implementations should be in a language that I can easily compile and run. These languages definitely include Java, C, C++, Python, and Scheme. If you want to use another language, talk with me in advance about it.

For each n that has a magic square, you should find all of them and count them. But you should only print (or write to a file) the first solution that you find.

Good coding and documentation style should be used.

You should submit a ZIP file containing your code to the Magic Squares drop box on ANGEL. If you work with someone else, put both students' names in the title of your submission. Package your code in a way that makes it as easy as you can to actually run the code once it is unzipped. For example, if you use Eclipse, include the entire Eclipse project.

A paper document (or documents) that include(s):

- A high-level description of each algorithm
- Your code for both algorithms (8 point font is fine)
- Instructions for running the code (if it is not obvious form context)
- Information about the largest value of N for which each algorithm works in under a minute.