MA/CSSE 473 – Design and Analysis of Algorithms

Homework 5 (and a head-start on a HW6 implementation problem)

33 points total

Note that this assignment is smaller and simpler than some other assignments, with the intention that you will begin working on the implementation problem for Assignment 6.

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.

3.2.2 (best/worst case for sequential search)
3.2.9 (Substrings that begin/end with specific characters) Brute force is $\Theta(N^2)$
3.3.2 (closest straight-line post office)
3.3.3 (Manhattan distance)
3.4.2 (Hamiltonian Circuit)

Problems to write up and turn in:

1. (10) 3.2.3 (Gadget drop) Let N be the total number of floors, and F the number of the lowest floor on which a gadget fails when dropped from there.

   Assume that due to weight, volatility, or some other factor, there is a high cost (C) for each floor that a gadget must be carried up. Also a cost (T) for each test to determine whether the drop caused the gadget to fail after each drop.

   First, give big-Theta worst-case running times in terms of N, C, and T for the obvious algorithm that tries every floor in succession (that algorithm only needs one gadget). Then design and analyze a more efficient algorithm).

   Can you think of any flaws in this general approach to testing?
2. (2) 3.2.4 (String matching count)
3. (5) 3.2.6 (String matching worst case). Searching for a pattern of length n in a string of length m.
4. (3) 3.3.5 (brute-force k-dimensional closest-point algorithm)
5. (2) 3.3.8 (dealing with collinear points in brute-force complex hull algorithm)
6. (4) 3.4.1 (traveling salesman analysis)
7. (2) 3.4.5 (simple cost matrix for an assignment problem whose optimal solution does not include smallest element)
8. (5) 3.4.6 (partition problem)

(Continued on next page)

Implementation problem (part of HW6, due Day 14):

(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.