MA/CSSE 473 – Design and Analysis of Algorithms

Homework 1 (8 problems to turn in, 49 points total) Updated for Summer, 2014

These are to be turned in to a drop box on Moodle. Include your name and the number of the assignment in the filename and also at the top of the first page. If your submission contains multiple documents (for example a Word document and a Maple document), please submit a single ZIP or RAR document that contains all of them. You may write your solutions by hand and scan them if you wish. There is an easy-to-use network scanner in F-217. It will email the scan to you. If you are not near campus and do not have a scanner where you are, I recommend buying one. You can get a decent scanner for about $60.

Most of the problems to turn in for this assignment are relatively short and simple. But during the first week there is a lot of reading, and there are also many not-for-turnin problems related to the "review reading" that you should at least think about a little bit.

On many assignments, a couple of the problems to be turned in (for example, Problem 7 in this assignment) may require a day or more from the time you first encounter them until the time when the light comes on and you see how to do them. So be sure to read the problem statements right away and start thinking how to do them as soon as you can.

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

A look ahead to HW 2: Partly because it is due early in the term, I made HW 1 rather short. HW 2 is longer, and is due soon after HW 1. So ideally you should do a few problems from HW 2 before HW 1 is due. But because some students have so many adjustments at the beginning of the term, I am not requiring you to have them done so soon.

Assess your background preparation for this course

The http://www.rose-hulman.edu/class/csse/csse473/201440/Homework/BackgroundMaterial.html document in the homework folder lists some material from the prerequisite courses and the level of understanding that I hope you will have coming into this course. Different students have had different versions of CSSE230 and MA375, so you should be okay if you are at the suggested level for about 2/3 of the topics. If you are at the suggested level for fewer than half of the topics, you may have quite a bit of remedial work to do for 473. As you can see from the list, most of these topics are reviewed in the 473 textbook. I some cases you will also want to go back to your 230 or 375 textbook for more details. I have also provided some old 230 problems and links to some of my PowerPoint slides from 230 as an additional resource. The problems are at http://www.rose-hulman.edu/class/csse/csse473/201440/Homework/230-problems.pdf

General notes about assigned problems

All textbook problems in all assignments are from Levitin 3rd Edition unless otherwise indicated.

Problem numbers [in brackets] are the corresponding problems from Levitin 2nd Edition

Note that there are hints for most of the exercises at the end of the Levitin book (just before the index). I suggest that you try to do each one without the hint, then read the hint if you get stuck.

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

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

1.1.2 [1.1.2] (algorithms patentable?)
1.1.4 [1.1.4] (algorithm for sqrt(n)
1.1.6 [1.1.5] (practice Euclid, estimate speedup of Euclid vs. brute force algorithm)
1.1.7 [1.1.6] (prove that the main step of Euclid works)
1.1.8 [1.1.7] (Euclid with largest number second)
1.1.9 [1.1.8] (smallest, largest number of Euclid divisions ($1 \leq m, n \leq 10$)
1.1.10a [1.1.9a] (Extended Euclid algorithm)
1.2.1 [1.2.1] (cabbage, wolf, goat)
1.2.3 [1.2.3] (triangle area formulas: which ones are algorithms?)
1.2.5 [1.2.5] (binary representation of integers)
1.3.1 [1.3.1] (Comparison Counting Sort)
1.3.2 [1.3.2] (known search algorithms)
1.3.3 [1.3.3] (string-matching algorithm)
1.3.4 [1.3.4] (Königsberg bridges)
1.3.10 [1.3.10] (intersection of line segments)
1.4.1 [1.4.1] (efficient delete in an array – this book's "array" behaves more like a Java ArrayList)
1.4.3 [1.4.3] (push, pop, enqueue, dequeue)
1.4.6 [1.4.6] (height of a binary tree)
1.4.7 [1.4.7] (inefficient implementations of "priority queue")
1.4.9 [1.4.9] (choose best data structure)

Post to the *Introduce Yourself* discussion forum on Piazza.

You can find out many "secret facts" about me and other students. But you should also post something about yourself. Also, feel free to respond to what others have written.

Problems to write up and turn in (numbered problems are from Levitin):

Be sure to do the reading for days 1 and 2. [Numbers in brackets refer to the 2nd edition of Levitin].

1. 1.1.12 [1.1.11] (5) (locker doors)

2. 1.2.2 [1.2.2] (5) (four people and a flashlight)

3. 1.3.9 [1.3.9] (5) (are n given points on circumference of the same circle?). Input: a list of coordinates, output: boolean. You can be brief, but do not be so vague that I cannot tell whether you really know how to do this.

4. 1.4.4 [1.4.4] (6) (graph properties, based on adjacency matrix, adjacency list)

5. 1.4.5 [1.4.5] (5) (Free tree → rooted tree) The algorithm is given a reference to the free tree and a reference to the node that is to be the root. Assume that the free tree is represented As a graph with adjacency lists, and that each node of the rooted tree has a list of its children.

6. 1.4.10 [1.4.10] (3) (anagram checker) Note that it says “anagram”, not “palindrome”.

7. (5) (combinatorial practice, thanks to former RHIT faculty member Salman Azhar) Each element of nonempty set A is a set of four distinct random digits, each digit in range 1..9 (inclusive) 1 and 9. Example: A = {{2, 4, 7, 5}, {1, 4, 8, 3}, {1, 3, 5, 9}}. Construct set B as follows. For each element of A, we construct a single element of B as the sum of the 24 four-digit numbers that are all possible permutations of that element of A. What is the smallest possible value of the GCD (greatest common divisor) of all of the elements of B?

8. (10) Read through the review of Mathematical induction that is linked from the Resources column of Session 2 on the Schedule Page.

This is an enhanced composite of some materials that I used when we used to emphasize induction more in CSSE 230. Even if you already feel quite comfortable with both ordinary and strong induction, be sure to read the part on how not to do induction. Other sources on induction: Weiss 7.2, Grimaldi Chapter 4.

Let the Fibonacci numbers be as defined on page 80 [78] of Levitin. On pages 80-82 [79-80], the author derives an explicit formula (2.9)[(2.11)] for the nth Fibonacci number. Another approach (the one Weiss uses) is to produce the formula "by magic" and use mathematical induction to prove it. That "proof by induction" approach is what you are to do for this problem. Be sure that in your induction proof you make it explicit what the induction hypothesis is and how it is used.

Hint: you might first want to prove (induction is not needed for this part) the simple relationship among 1, φ, and φ^2, namely $1+ \varphi = \varphi^2$, which implies the same relationship among φ^k, φ^{k+1}, and φ^{k+2}. Note that φ and $\hat{\varphi}$ are defined on page 80 [80] of Levitin.

A few notes on writing up problems for this course:

- “Produce an algorithm” should always be interpreted as “produce an efficient algorithm” unless a problem states otherwise. Of the there is a trivial and horribly inefficient algorithm, which will probably not earn you full credit.
- A few of the questions in the book can (perhaps) be answered with a number, or with a word or two. In most cases that is not sufficient; I want to know how you arrived at your conclusion.
- You can usually present algorithms as high-level pseudocode (or as real code in a programming language if you wish, though that sometimes takes a lot more time). Go into enough detail to convince me that you are not glossing over any hard parts, but you do not have to give all of the details for the easy parts. Pseudocode, just like real code, often needs comments and explanations. The burden of convincing me that your solution works is on you. I will tell the graders (in summer versions of this course, I will be the grader for most assignments) that if they cannot quickly understand your solution, they should give you a small percentage of the points and move on.