MA/CSSE 473 – Design and Analysis of Algorithms

Homework 1 (9 problems to turn in, 62 points total) Updated for Summer, 2017

These are to be turned in to a drop box on Moodle. You should either do the assignment on your computer, or write it and scan it (Phone pictures are okay provided that they are very easy to read). Include your name and the number of the assignment in the filename, and also at the top of the first page. The problems should be in assignment order. 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. If your submission contains multiple pictures, please insert all of them into a single document (for example, a Word document)

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 several of the assignments, a couple of the required problems (for example, problems 7 and 8 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 about how to do them.

A look ahead to HW 2: Partly because it is due very 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 do them so soon.

Assess your background preparation for this course

The http://www.rose-hulman.edu/class/csse/csse473/201740/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. In 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/csse473/201740/Homework/230-problems.pdf

General notes about assigned problems

All textbook problems in all assignments are from Levitin 3rd Edition unless otherwise indicated. When a problem is given by number, it is from the textbook. 1.1.2 means "Exercise 2 from section 1.1".
Problem umbers in brackets are the corresponding problems from the 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.

[Numbers in brackets refer to the 2nd edition of Levitin].

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 ($1 \le m, n \le 10$)
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)

Be sure to do the reading that is in the Preparation column of the Schedule page for days 1 and 2.

Problems to write up and turn in (x.y.z numbered problems are from Levitin):

This is a somewhat eclectic collection of problems that serves many purposes. It should "get you back into" thinking about algorithms and writing about them. It should remind you of some things you should review from previous courses (e.g., recursive algorithms and mathematical induction). The "puzzler" problems (1, 2, 7) should stretch your brain a bit.

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

   Questions and answers from previous terms' Piazza:

   Q: I'm not entirely sure what it means to toggle every i-th locker. Does it mean:
   1st round: toggle 1st, 2nd, 3rd ... nth
   2nd round: toggle 1st, 3rd, 5th ... (2n+1)th
   3rd round: toggle 1st, 4th, 5th ... (3n+1)th
   Or:
   1st round: toggle 1st, 2nd, 3rd ... nth
   2nd round: toggle 2nd, 4th, 6th ... 2nth
   3rd round: toggle 3rd, 6th, 9th ... 3nth
   A: The latter. The second round lockers numbered $2k$ for $k = 1,2,3,...$ are toggled, the third round lockers numbered $3k$, etc. Note that n was the total number of lockers, so we never go above n.

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

   Previous questions and answers from Piazza:

   Can two people cross the bridge in opposite directions so long as one of them has the flashlight? **TA answer:** No. The darkness constraint means that anyone crossing the bridge should be accompanied by the flashlight for their entire crossing (either in their hand or their partner's).

3. 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 (6) (graph properties, based on adjacency matrix, adjacency list) See pages 28-30. Be sure to do part (b).

5. 1.4.5 (5) (Free tree → rooted tree) The algorithm is given a reference to a node of the free tree. 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 the nodes that are adjacent to it. The algorithm should return a reference to the root node of the rooted tree that it constructs. **There is an example at the end of this document.**

Previous questions and answers from Piazza:

Q: I am reading the description of free trees and rooted trees and the only difference, at least in the book, seems to be that in rooted trees, a node has been selected as the root. This would make this problem very easy, but it is also why I think there is something I am missing.
A: One major difference: A free tree is an undirected graph; a rooted tree is a directed graph. Ina rooted tree there is an edge from a parent vertex to each of its children, but there is no edge from a child to its parent. Unfortunately, the textbook does not say this explicitly. It does imply it as it has a discussion of moving up and down in the tree. If the graph is undirected, there wouls be no notion of "up" or "down"

6. 1.4.10 (6) (anagram checker) Note that it says "anagram", not "palindrome".

Previous questions and answers from Piazza:

Q: For problem 6, the anagram checker, do we know how expensive removing a character from a string is? I have an idea to get faster than O(n^2), but I want to know if the cost of the removals will cancel out the gain in speed.
A: If strings are represented in "normal" ways, the worst case for removing a character from a string of length n is $\Theta(n)$. Best case $\Theta(1)$, average case $\Theta(n)$. If we represent strings in a more complex way (such as the AVL tree with rank, as in the CSSE 230 EditorTrees project). The worst and average can be $\Theta(\log n)$. But the time overhead of rebalancing is so great that you would only get faster performance for really long strings.

Q: Is the word case sensitive? **A:** The algorithm is simpler if the answer is "yes", so I will say "yes". You can consider "anagram" to simply mean "same characters, possibly different order".

7. (15) (combinatorial practice (thanks to former RHIT faculty member Salman Azhar) . This problem is partly an exercise in careful reading. Not an easy problem. Each element of nonempty set A is a set of four distinct random digits; each digit is in range 1..9 (inclusive). Example: A = {{2, 4, 7, 5}, {1, 4, 8, 3}, {1, 3, 5, 9}}. Construct set B from A as follows. For each element of A (that element is a set of four single-digit numbers), construct the corresponding element of B as the sum of the 24 four-digit integers that are all of the possible permutations of that element of A. So if A is a set of k sets, B is a set of k numbers. Find (and show how you get it) the smallest value of the GCD (greatest common divisor) of all of the elements of all possible instances of B. Note that the GCD of the elements in a particular set B depends on which sets are in the particular A from which B is derived. So we are looking for

min({gcd(B) : B is the set of the sums of all permutations derived from each of 4-element sets in A}),
where the minimum is taken over *all* A's which are nonempty sets of 4-element sets as described above.

Note: If the answer to this problem is a number S, there are two things to show:
1. S is a factor of every GCD that cones from every set A.
 A brief mathematical argument is needed for this part.
2. That there actually is a specific set A that leads to S as the GCD.
 Show me such an A (there are many possibilities).

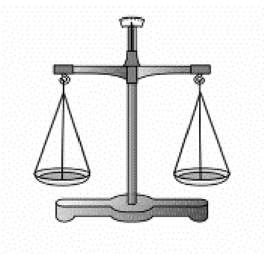
8. (10) Read through the review of Mathematical Induction that is linked from the Resources column of Session 2 on the Schedule Page. Summer: watch the induction videos (Day 4 in 2017)

This is an enhanced composite of some materials that I used in the days when we emphasized 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 n^{th} 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 explicitly say what the induction hypothesis is and how it is used in the proof of the induction step.

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.

9. (5) Fake 8. There are 8 identical-looking coins, but one of them is lighter than the others. Our only tool is a pan balance, which will tell us whether the coins in the two pans have equal weight, or whether the left or right side is heavier. What is the minimum number of weighings required to guarantee that we know which is the fake coin? Briefly explain your algorithm. Hint: A balance pan can hold more than one coin at a time.



Previous questions and answers from Piazza:

Q: I am assuming this problem means what is the minimum number of weighings needed to know which coin is lightest in a worst case scenario. (I.E. one is not counting the possibility of the weigher getting lucky). Am I correct to assume this? Also I am assuming that by one coin being lighter, it means that coin is the fake coin. **A:** Yes, and yes.

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. If there is an efficient algorithm, and you submit a trivial and much less efficient algorithm, it will probably not earn you full credit.

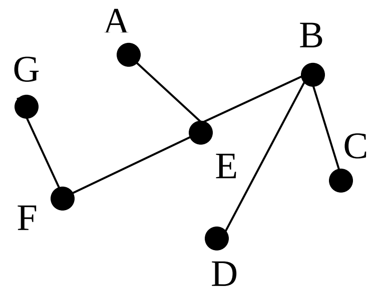
- A few of the questions in the book can (perhaps) be answered with a number, or with a word or two. In most cases that simple answer is not sufficient; I want to know how you arrived at your conclusion.

- You can usually present algorithms in English or 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 many assignments) that if they cannot quickly understand your solution, they should give you a small percentage of the points and move on.

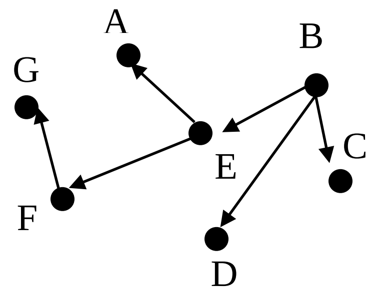
Reminder: Turn in this assignment to the HW 1 Drop box on Moodle by 11:55 on its due date.

For 5 points extra credit, you can also complete an optional survey concerning your experience with this assignment.

Example for Problem 5



Input: Free Tree



Output: Rooted tree with root at

The algorithm you must write takes as its arguments
 a free tree
 one of its nodes,
and returns a rooted tree with the given node as its root.
The edges of the free tree are undirected.