MA/CSSE 473 Fall 2014 Exam 2 specification (Last updated October 25, 2014)

Resources allowed:

1. Calculator (not a smart phone, iPod, etc. that happens to have a calculator.
    a. The only necessary operations are +, *, -, /, exponentiation, and you probably won't need those.
2. One 8.5" x 11" page of notes (handwritten on one side).
3. No other electronic devices, including phones, MP3 players, anything with headphones or earbuds.

Material Covered

- **Background material from 230:** Algorithms and analysis for implementing stacks, queues, linked lists, sequential lists, binary trees, binary search trees, binary heaps, dictionaries. Recursion and mathematical induction. Writing and solving simple recurrence relations, analysis of nested loops as in Weiss chapter 5 and its exercises. Fibonacci numbers. Sequential and binary search. Well-known sorting methods: Insertion, selection, merge, quick, heap. Binary tree traversals: preorder, inorder, postorder, level order. Formal definitions of O(N), Θ(N), etc.
- **HW 1 - HW 12** (including the "not to turn in" problems)
- **Assigned readings** from Chapters 1-11 from the textbook plus Sections 12.1, 12.2, as well as documents posted on Moodle that are listed on the Schedule page, and material from PowerPoint slides for all sessions. (Chapters 1-5 from the textbook as well as documents posted on Moodle that are listed on the Schedule page for Sessions 1-16), and material from PowerPoint slides for sessions 1-18.
  More details below on specific algorithms you should know.

Question types: In order to minimize the amount of typing you must do, the exam will have several multiple-choice and T-F-IDK questions. For questions where you have to write something, You do not have to have perfectly formatted mathematics. English and "mathematical pseudocode" is fine.
For example, Theta(3N^2 + (N^2 + N)/sqrt(N-4) or (1/N)*Sum(k-1, k=2 .. N).

T-F-IDK. Below you will find several statements. A statement is true (T) if it is always true. It is false (F) if there is at least one counterexample (sometimes false). You may also choose IDK to indicate that you do not know the answer. *There is some value (to me and you) in knowing what you don't know.*

Point values: Correct answer: 4, incorrect answer: -1, IDK: 1, blank: 0. So if you get seven questions correct, choose IDK for three, get two wrong, and leave one blank, your score is 7*4 +3*2 – 2*1 + 1*0 = 32 out of 52. What if you are not sure of an answer? If you are totally guessing an answer, you are probably better off pointwise if you choose IDK. If you have a feeling that one "real" answer is correct, you are probably better off guessing that answer.

Can you come up with a reasonably efficient algorithm to … ? Many of the homework problems (especially the puzzle problems) have been this type of problem.

Does this proposed algorithm for problem _____ work?

What is the worst-case running time for the following algorithm?

Which of these algorithms is an example of <design technique>? (design_technique may be something like.Greedy, Divide-and-conquer, Dynamic Programming, Decrease by a constant factor. The problem will ask you to choose among a list of several algorithms. The problem could also go the other way; giving you the name of an algorithm that we studied and asking you to choose among a list of algorithm design techniques.

Do you know the details of this algorithm? Some questions will be checking to see whether you know and understand the details of specific algorithms. Here are examples of algorithms that you should be able to explain and discuss the results of what happens when they are implemented with specific input data.

Examples (there are more examples in the background material section above)**: (the first 6 bullets are form Dasgupta)**

- The algorithms from the list, in the similar document for Exam 1. While I will not ask detailed questions specifically about these algorithms unless they are also listed below, I may ask you to compare or contrast some feature of one of them with one of the more recent algorithms, or there could be a "Name an algorithm from this course that …" question.
- Understand (not memorize) the proof of the RSA algorithm.
- Binary tree search and insertion (L 4.5)
- Nim (binary digital sum)
- Merge sort (L 5.1)
- Quick sort (L 5.2) Know the basic approach to solving average case recurrence)
- Big integer multiplication (L 5.4)
- Fast Matrix Multiplication (L 5.4)
- Recursive closest pair (L 5.4)
- Recursive convex hull (L 5.5)
- Gaussian Elimination (L 6.2)
- LU Decomposition (L 6.2)
- Matrix Inverse and Determinant computation (L 6.2)
- AVL trees (big-picture ideas, not details of specific rotations)
- Insertion and balancing: 2-3 trees (L 6.3)
- Heap insertion and maximum element deletion (L 6.4)
- Heapsort (L 6.4)
- Horner's Rule for left-to-right polynomial evaluation (L 6.5)
- Left-to-right binary exponentiation (L 6.5)
- Sorting by Counting (L 7.1)
- Horspool and Boyer-More algorithms (L 7.2)
- General ideas of KMP string search, but not the details of shift table calculation and use.
- B-tree insertion and size calculation (L 7.4)
- Dynamic programming for calculation of Fibonacci numbers and binomial coefficients.
- Optimal static Binary search tree, including "gaps" frequency info, (as in the HW problem, the slides, and the Reingold/Hanson excerpt that is posted on Moodle) (a limited version is in section L 8.3)
- Warshall's Algorithm for finding the transitive closure of a graph (L 8.4)
- Floyd's Algorithm for All-pairs shortest Path problem (L 8.4)

Concepts, definitions, etc.

- Concepts from the exam 1 list may be included, but not as the main emphasis of a problem, only for comparison or "name an algorithm that …"
- Master Theorem for divide-and-conquer algorithms. No need to memorize; I will give you the formulas (L 5.1)
- Binary tree traversals and properties (L 5.3)

- Transform and conquer – Levitin's classification of the three types of transformations.
- Presorting (L 6.1)
- Reducing the solution of a problem in one domain to solving a problem in (possibly) another domain (L 6.6)
- Hashing: the basic ideas, separate chaining vs Open Addressing, Approaches to collision resolution (this is mostly review from CSSE 230) (L 7.3 and there is additional information in the PowerPoint slides)
- Basics of the Dynamic Programming approach (L 8.1)