CS 417, Winter 08, Final Review

1. Big O, Ω, Θ notation, runtime analysis (Chap 2)
   (a) Basic ordering of common functions
   (b) Application of definition of O, Ω, Θ.
   (c) best-case, worst-case, average case
   (d) Given a simple algorithm in pseudo-code, analyze the runtime
2. Graphs (Chap 3)
   (a) Graph basics – different ways to store graphs, directed vs undirected, relationships between nodes & edges
   (b) Graph Algorithms. You do not need to memorize the basic algorithms (they will be provided), but you should be familiar with how to modify them for specific problems. Be familiar with the following algorithms & applications:
      i. BFS
      ii. DFS
      iii. Topological Sort
      iv. Shortest Path
      v. Connected Components (algorithm not provided)
3. Greedy Algorithms (Chap 4).
   (a) Be familiar with both the algorithms, and their proofs. You will *not* be asked to simply regurgitate one of these proofs – but you may be asked to sketch a similar proof.
      i. Interval Scheduling
      ii. Interval Partitioning (All Interval Scheduling)
      iii. Minimize Lateness
   (b) In particular, be familiar with the following proof techniques:
      i. Exchange Argument
      ii. Stay Ahead Argument
      iii. Structural Argument
      iv. Finding counter-examples to show how *bad* greedy strategies can fail.
   (c) Understand how to use the following algorithms. You do not need to know their proofs, but at least have an idea of why they are correct
      i. Coin Changing
      ii. Prim's Minimum Spanning Tree
      iii. Dijkstra's Shortest Path
   (d) Huffman codes

i. Basic idea of a prefix code, how to use it, how to calculate compressed file length
ii. Be able to apply the Huffman algorithm
iii. Understand the concept of *inversions* and how we applied them in analyzing prefix codes.
iv. You do *not* need to know the full proof of correctness for the Huffman algorithm.

4. Divide And Conquer (Chap 5).
   (a) Go from code to recurrence
   (b) Solve a recurrence (eg., with Master Theorem)
5. Dynamic Programming (Chap 6).
   (a) Show that greedy can fail
   (b) Using principle of optimality to make a recurrence
   (c) From recurrence to code – iterative version, memoize version
   (d) Know why "naive" recursion is bad
   (e) Understand some examples
      i. Minimum Stamp Problem
      ii. Weighted Interval Problem
      iii. RNA folding
      iv. String Similarity
      v. Knapsack
6. NP (Chap 8).
   (a) defn of decision problem
   (b) Why do we care about class P?
   (c) The class NP
      i. Understand defintion
      ii. How to solve a problem in NP
      iii. Be able to show a language is in NP
      iv. Know some examples
   (d) NP-complete problems
   (e) Relationship between P, NP, NP-complete, EXP