CS 312 Study Guide for Midterm and Final

The tests are closed book, but you may bring one single sided page of notes with a font no smaller than Times 12 to the test. The spirit of this is a note page to put on equations or other items which are harder to memorize. It is not meant for trying to cram all the slides, book chapters, or knowledge from the course on a single sheet. You should know most of that without needing a sheet. Your note page will be handed in with the test. *For all algorithms mentioned, you should be able to figure out time and space complexity.* You should be prepared to answer questions from the following topic lists:

Midterm

Analyzing algorithms: correctness, complexity, can we do better
Asymptotic Analysis: Big O, Θ, and Ω.
Basic arithmetic algorithms and complexity: Addition, multiplication, modular arithmetic, modular exponentiation, modular inverse/division, Euclid's GCD
Primality testing - Fermat's little theorem, probabilistic nature
Divide and Conquer – Paradigm/How is speed up attained, mergesort, quicksort, multiply, matrix multiply, convex hull, selection (median)
Master Theorem – How to use it, what does it tell us, geometric series intuition
Recurrence relations – What do solutions tell us, be able to solve homogenous and non-homogenous LTI recurrence relations, use change of variables for divide and conquer recurrence relations
Graph connectedness – Graph representation, DFS, search tree and pre/post order values, cycle-detection, DAGs, linearization, finding strongly connected components
Graph paths – BFS, Dijkstra's algorithm, priority queue implementations/complexity, negative edges, Bellman-Ford, shortest DAG paths
Greedy algorithms – Paradigm/philosophy, MST with Kruskal's/Prim's, Huffman

Final

The final is comprehensive with heavy emphasis on topics covered since the midterm.
Topics from the 2nd half of the semester include:
Dynamic Programming – Philosophy and how to use it
Longest increasing subsequence, binomial coefficient, edit distance, knapsack with and w/o repetition, chain matrix multiply, Floyds, Recursion and Memoization
Linear Programming – Setting up an LP from an objective and constraints, putting an LP in standard form, Simplex algorithm (you don't need to know code, but you should be able to walk through a detailed simplex example, given an LP)
Intelligent Search – Backtracking, Branch and Bound, A*, beam search, bounding functions TSP with B&B, reduced cost matrix, State space search: partial path and include-exclude
Complexity classes – P vs NP, NP-complete
Bounded approximation algorithms, approximation factor, k-cluster
Local search – Basic algorithm and properties, local minima, perceptron learning algorithm, genetic algorithms
Analysis – Empirical analysis, Average case analysis (high level)
Stochastic Algorithms – Monte Carlo vs Las Vegas algorithms, amplification of stochastic advantage
Overall summary – When to use which the basic 312 algorithms/Paradigms