Review for Final Exam

The exam will be closed notes, closed book, and no calculators. Exam may include true/false, multiple choice, short answer, and short proofs. When doing proofs, you must explain all of your steps.

Suggestion: carefully review all lab problems and class notes. Reread relevant sections in text.

Topics before Midterm

1. Proof by induction - review homework problems.
2. Asymptotic Notation
   - Experimental calculation of complexity. How do you measure the complexity?
   - Know the definitions of $\Omega, \Theta, \omega, O$ and o.
   - Know how to use the definitions in a proof.
   - Know how to use limits to determine complexity of a function.
   - Know how basic functions such as $f(n) = n, n^k, e^n, \lg n, n!$, etc compare. Be able to use L'Hopital's Rule.
   - Know how to do basic manipulation of exponentials and logs.
   - Know how to sum arithmetic series and geometric series.
3. Recurrences
   - Substitution method (guess and check with induction)
   - Iteration method and telescoping.
4. Binary Trees
   - What is a binary tree. How is it constructed. How do you implement the basic operations using recursion? (getHeight, printSorted, insert, remove, etc).
   - What is an AVL tree? What are it's properties?
   - Why are AVL trees used?
5. Heapsort and Priority Queues
   - What are the trade-offs of the various ways of implementing a priority queue?
   - What is a heap, how is it stored, what is its height?

- What is the heap property?
- What do the methods *heapify, build-heap*, and *heapsort* do? What are their complexity?

6. Hashing
   - What is hashing?
   - What are examples of hash functions?
   - What is a collision detection strategy? What are some examples? (e.g. chaining, linear probing, rehashing, open addressing, random hashing)

Topics after Midterm

1. Master Equations and Recurrences
   - Know how to use the Master Equation to prove bounds on recurrences.
   - Know when the Master Equation will not work.
2. Sorting in general
   - Know the different sorting algorithms: mergesort, insertion sort, heapsort, quicksort
   - Know the different approaches such as divide and conquer, comparison sorts, bucket sorts
   - How do sorts behave on already sorted lists, reverse ordered lists, etc.
   - What is the big-Oh bound for the different sorts.
3. Quicksort
   - What is the algorithm. How does the *partition* method work.
   - What is the worst case complexity? Average case?
   - How can quicksort be improved, e.g. median of 3?
4. Comparison Sorts
   - Understand the proof showing that all comparison sorts are at best $O(n \lg n)$
5. Radix and Bucket Sort
   - How does radix sort work?
   - What is its complexity?
6. Dynamic Programming
   - When is DP effective?

- Defining the subproblem
- Determining the recursion
- memoization
- Applications: Matrix Chain, LCS, Cheapest path, 0-1 Knapsack, Pretty Printing

7. Greedy Algorithms

   - What is a greedy algorithm?
   - Why use non-optimal greedy algorithms?
   - What is the greedy choice property and how do you prove that a problem satisfies it?
   - What is the optimal substructure and how do you prove that a problem satisfies it?
   - Applications: cheapest path, activity selection, Huffman codes, fractional knapsack problem

8. Graphs

   - Definitions
   - Breadth First Search Trees
   - Depth First Search Trees
   - Topological Sorting
   - Articulation Points and Bi-connected Graphs
   - Minimum Spanning Trees: Prim's Algorithm, Kruskal's Algorithm
   - Single Source Shortest Path - Dijkstra's Algorithm
   - All Pairs Shortest Path - Floyd-Warshall Algorithm