Review for Exam 2

The exam will be closed notes, closed book, and no calculators.

The 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.

1. 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 a balanced binary tree (e.g. Red-Black tree)? Why use one?
2. Heapsort and Priority Queues
   - What are the typical operations of a priority queue?
   - What are the time complexity 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 *siftdown, heapify*, and *heapsort* do? What are their complexity?
3. 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, open addressing, random hashing)
4. Sorting
   - What are the different sorting algorithms: bubble, insertion, selection, mergesort, stoogesort, shellsort, quicksort, heapsort, bucket sort, radix sort.
   - What is the complexity of each (if known).
   - Which ones: are comparison sorts, are divide and conquer, sort in-place, are stable, exchange adjacent items, are linear
   - understand why algorithms that just exchange adjacent items are $\Omega(n^2)$ for the worse case?

- Understand the proof showing that all comparison sorts are $\Omega(n \lg n)$ for the worse case?
- Under what conditions is it possible for so-called linear sorts to be worse than comparison sorts.

5. Dynamic Programming

- What is DP and when is it used?
- memoization
- Applications: Matrix Chain, LCS, Cheapest path, 0-1 Knapsack, Pretty Printing
- What does it mean for a problem to have *optimal substructure*?