**Study Guide for Final Exam in Algorithms**

**Comment:** This provides a list of some topics which I feel are important to study when preparing for the final exam. These topics were selected by making a “one pass” trip through the slides that I have prepared and listing many of the more important topics that I feel should be reviewed when studying for the final exam. The purpose of this study guide is to provide you with a shorter checklist to use to make sure you have studied most of the more important items in each chapter. A longer and more comprehensive study guide for this class consists of the slides prepared for this class and the problems that were assigned as homework.

**Chapter 1:**

* Advantages & disadvantages using experiments to evaluate the performance of an algorithm.
* Understand the RAM model use in the design and analysis of sequential algorithms, including what operations are assumed to require constant time.
* Understand how to compute the running time for algorithms and algorithm segments in terms of the input size of the data.
* Ability to evaluate the complexity of the growth rate of algorithms and algorithm segments in terms of Big-Oh, Big-Omega, and Big Theta notation ((e,g., O(n), Ω(n), and Θ(n) symbols).
* Knowledge of more standard math summation formulas, logarithm identities, exponential identities, etc. that are needed in the textbook, examples covered, and assigned problems.
* Understand and be able to use proof methods for theorems such as direct proof, contrapositive proof, proof by contradiction, and proof by counterexample, proof by induction, etc. A few questions may require you to prove some theorem we have covered in class or some fact.
* Methods for using experimentation to evaluate the performance of algorithms.

**Chapter 2:**

* An understanding of the more important types of ADT implementation used for a different data structures (e.g., stack, queue, tree, etc.) and why the performance and use of these ADTs are important. A detailed listing or discussion of the various methods implemented by each of these is not required.
* Possible questions may be asked about the different algorithms covered for the various data structures. More emphasis should be given to the algorithms that demonstrate important techniques or concepts.
* One example of “more important algorithms” that may appear on the final exam are the algorithms that involve the use of amortized analysis of running time.
* Topics and algorithms such as the various traversals, representing arithmetic expressions, total order relations, priority queue, selection sort, insertion sort,
* Algorithms and concepts involving heaps and heap sort.
* Binary tree algorithms and concepts.

**Chapter 3:**

* Binary search trees
* Basic aspects of (2,4) trees and their more basic algorithms.
* Basic aspects of Red-Black trees and their more basic algorithms

**Chapter 4:**

* Merge-Sort Algorithm and its complexity
* Quick-Sort Algorithm and its in-place implementation.
* Worst case and expected running time of quick-sort
  + The analysis of the expected running time would be a more challenging question for stronger students.
* Lower bound for comparison-based sorting. Basic understanding of what this says.
  + Proof of this result would be a more challenging question for better students.
* Understanding of how sets can be represented and understanding of basic algorithms needed
* Bucket –Sort and Radix-Sort – understand why not a comparison-based sort and basic algorithms.
* Stable sorts and lexicographic ordering concepts
* Quick-Select algorithm
  + Analysis of expected running time could be a more advanced question for top students

**Chapter 5: Fundamental Techniques:**

* Understand the three major algorithm design paradigms covered In this chapter – greedy method, Divide and Conquer, and Dynamic Programming
* Should be able to identify several problems that can be solved by each of these paradigms.
* Should be able to describe an algorithm that illustrates each of these paradigms.
* Understand algorithms covered in chapter using greedy method, including “Making Change”, “Fractional knapsack”, Task Scheduling,
* TO BE CONTINUED