EECS560 **Exam 1 Review Guide** **Fall, 2018**

Administrative

- **When:** 2:30pm-3:45pm, Thursday, September 27, 2018
- **Where:** 1136 Learned

Read carefully all of the following items....

- Unless otherwise stated, all material from the assigned readings, lectures and lecture notes, practice and lab assignments are fair game for exams.
- The exam will be closed book and closed notes.
- No calculators, cell phones, head phones, or electronic devices of any sort will be allowed. No such devices should be out in the open.
- Once you start the exam, you will not be excused from the room for any reason unless you turn in your exam. Once it is turned in, you cannot come back and continue working on it.
- You must write legibly and show all your work clearly for credit. Partial credit will only be given to meaningful answers.
- You will be graded according to your approach to the problems, mathematical rigor, and quality of your solutions. A correct but inefficient algorithm or data structure will receive very little credit in this course.

Exam Coverage

Basic Computation Model and Performance Analysis

- Understand the RAM computational model and be able to state its assumptions and use them to analyze the performance of algorithms.
- Understand the definitions and basic properties of different asymptotic relations including big-O, big-Ω, and big-Θ. For two given functions f(n) and g(n), be able to prove, or disprove, an asymptotic relation between these functions using the definitions and/or the properties of the corresponding asymptotic relation.
- For a given algorithm and/or program segment, be able to compute the best-case and worst-case complexity in closed-form and/or asymptotic forms using (1) detailed analysis, (2) basic operation(s), and (3) dominating steps as discussed in class.
- For two algorithms A_1 and A_2 with complexity $T_1(n)$ and $T_2(n)$, be able to compute the smallest input size n_0 such that $\forall\ n \geq n_0$, one algorithm will always outperform the other algorithm.
- Understand the importance of efficient algorithms and be able to compare the performance of different algorithms and estimate their resource consumption based on their complexity functions and input size.

时间的计算
输入size计算

ADT: Dictionary and Hash Tables

- Understand the basic structures and properties of a Dictionary.
- Understand the structures and implementations of open and closed hash tables.
- Understand the advantages and disadvantages in using open and closed hash tables.
- Understand the design and general properties of a “good” hash function.
- Understand and be able to compute the worst-case and average-case complexity of all basic dictionary operations of a hash table.

- Understand the design and general properties of different collision resolution schemes including separate chaining and open addressing. For a set of records, be able to construct and illustrate the hash table with a given collision resolution scheme.
- Understand the significance of load factor in the design and maintenance of hash tables.
- Understand the importance and application of prime numbers in closed hashing.
- Understand the importance and application of rehashing.

Trees and Their Implementations

- Understand the basic structures and properties of trees.
- Understand the basic structures and properties of different classes of special trees and their relations to performance analysis of tree-based ADTs.
- Given a set of n records to be organized using a tree-based data structure, be able to compute the maximum, and minimum, height, number of nodes, and number of leaves of this data structure.
- Understand and be able to perform basic tree traversals, including pre-, post-, in-, and level- order traversals. Given a tree-based data structure representing a set S, be able to use the tree traversal algorithms to explore the structure and retrieve information associated with the set/data structure.
- Given a set of records S being organized using a binary tree structure, understands and be able to store and reconstruct the corresponding binary tree structure using binary tree traversals.
- Given a binary or general tree, understand and be able to implement and illustrate the given tree structure using array, pointer, and hybrid data structures.