CPSC 335 (Wortman) - Spring 2018 - Midterm Study Guide

The exam will take place Wednesday June 13, in class. It will cover chapters 1-5 of the ADITA textbook, which corresponds to the material for weeks 1-2 in the syllabus.

Exam questions will be similar in nature to in-class exercises and quizzes. You may be asked to define terminology; write problem statements; prove efficiency classes; analyze pseudocode; execute the algorithms we have studied by hand, showing work; and design algorithms using the greedy pattern.

You may bring a memory aid ("crib sheet") to the exam. Your sheet must be a single piece of paper, letter/A4 sized, optionally double-sided, with your name and CWID on it, and you must turn it in with your exam packet.

The following material is fair game:

- Chapter 1 Introduction
    - The only testable material in this chapter is the concept of algorithm patterns; and the three-act structure of the book that implies that algorithms have limitations that can sometimes be overcome.
- Chapter 2 Fundamentals
    - 2.2 Terminology: data, problem, instance, solution, algorithm, pseudocode, implementation, the pseudocode clarity litmus test
    - 2.3 Pseudocode Checklist: be prepared to write pseudocode that is clear, correct, and terminates
    - 2.4 Algorithm Patterns: be prepared to design algorithms using the greedy pattern
- Chapter 3 Efficiency Analysis
    - 3.2 Functions Measuring Resources: resources, worst-case analysis, complexity functions
    - 3.3 Asymptotic Notation: definition of O, rationale for the definition
    - 3.4 The Big Eight Efficiency Classes: be able to recognize them and rank them by order of growth
    - 3.5 Experimental Analysis: know the terminology and issues; of course you cannot perform a coding experiment during the exam, that is the purpose of the projects
    - 3.6 Mathematical Analysis: computational models, the standard model, definition of "step," chronological step counting, proving efficiency classes by induction, proving efficiency classes with limits, proving efficiency classes with properties of O. Keep in mind that we have proved classes by properties of O far more frequently than using induction or limits.
    - 3.7 Amortized Analysis: at this point you should be familiar with the concept, understand the difference between a worst-case time bound and amortized time bound, and know which data structure operations carry amortized time bounds (essentially just adding to the back of a vector, and other operations built on top of adding to the back of a vector)
- Chapter 4 Essential Data Structures

    - Data structures are a prerequisite to this course, so this chapter should be almost entirely review. You will not be tested on data structures specifically, but need to be comfortable enough with them to design efficient algorithms with clear pseudocode.
    - You should know when and how to use vectors, linked lists, binary search trees, and graphs; and to know which of their essential operations are $O(1)$, $O(\log n)$, and $O(n)$.
    - You do **not** need to memorize the specific API and function names in chapter 4; you only need to be able to communicate data structure operations that meet the clearness litmus test. For example "add x to the back of linked list L" is perfectly clear, you do not need to memorize the syntax "L.add_back(x)".
- Chapter 5 The Greedy Pattern
    - 5.2 American Change Making: know the problem and why this is a greedy algorithm
    - 5.3 Analyzing Greedy Algorithms: be prepared to analyze algorithms that make use of summation expressions, rectangular sums, and triangular sums
    - 5.4 Sequential Search: know the problem, algorithm, and "running best" pattern
    - 5.5 Greedy Sorting: know the sorting problem, definition of comparable and non-decreasing order, concept of an invariant, idea of sorted/unsorted zones, basic selection sort, in-place selection sort, time complexity of both algorithms
    - 5.6 Minimum Spanning Trees: graph terminology, MST problem, number of edges in a spanning tree, bridge theorem, greedy MST pattern, Prim-Jarnik algorithm
    - 5.7 Nonnegative Single-Source Shortest Paths and Dijkstra's Algorithm: graph terminology, distance and penultimate data structures, correctness of Dijkstra's algorithm, edge relaxation.