COMP26120 examination revision guidance

Here is guidance on what parts of the course textbook (Algorithm Design, Goodrich and Tamassia, Wiley) are relevant to your exam revision for the summer examination. The material of both semesters is examinable, as is the laboratory, tutorial and lectured material as well as topics covered in more detail in the course textbook.

A full list of revision topics is below:

For the January exam you need cover only the topics from the first semester. These are marked red

* You will need to be able to compute and compare complexity measures for algorithms, using the ‘Big O’ notation, as described in Chapter 1.
* You are expected to understand amortization and calculate amortized performance as described in Section 1.5
* Chapter 2 covers the basic material on stacks, queues, trees and hashing. This is examinable.
* Chapter 3: Sections 3.1, 3.2 only.
* Sorting: You will need to know about a range of sorting algorithms, their performance and applicability. These are mainly in Chapter 4 of the book, but some are elsewhere.
* Chapter 5 is a survey of some fundamental algorithmic techniques. All three techniques covered here are relevant. For the January exam only ‘divide-and-conquer’ techniques will be covered.
* Graphs and graph algorithms: All of Chapter 6 is relevant, particularly adjacency matrix and adjacency list representations, and BFS and DFS traversal. In Chapter 7, we cover Dijkstra’s algorithm in detail (7.1.1) and Floyd’s algorithm (7.2.1).
* You will also need to know the material on the laboratory exercises in Semester 2, in particular that on Graph navigation and on Knapsack problems.

The relevant parts of the book for Knapsack problems are Sections 5.1.1 (fractional case), 5.3.3 (dynamic programming) and 13.5.2 (branch and bound). You should make sure you understand both the Branch-and-bound and the Dynamic programming solutions, as in the laboratory exercise.

Revision for Knapsack

Revising for Knapsack

* Be able to describe the 0/1 Knapsack problem in detail
* Understand how enumeration works, and limitations of enumeration (complexity)
* Understand how greedy works and limitations (not exact for 0/1 problems) and its complexity
* Understand how branch-and-bound works in detail for the 0/1 problem
  + Understand how bounds are calculated, and how they are used to make the search efficient
  + Know the worst-case complexity
  + Understand about what instances would be easy for branch-and-bound and why
* Understand how and why dynamic programming works in detail
  + Complexity of the dynamic programming approach (how this relates to table size)
  + Which instances are easy for DP and which are hard
  + How the actual list of items packed are stored and reported by DP.
* Applications of knapsack problems
* The fractional knapsack problem — solvable (exactly) by greedy. Why?

Student Questions

***If you were a student, how would you prepare for the exam?***

Follow the advice given. Go through the lecture slides. Use the course textbook. Look back over relevant labs. Do the past examination papers. Ask by email if there is anything I still don’t understand after all that.

***What are you expecting of the students to learn from this unit?***

This is detailed on the webpage for the course. We are interested that students understand some core algorithms, are able to design their own algorithms, and able to reason about the complexity and correctness of algorithms.

***How are is this reflected in the exam? What are you looking for in the exam?***

Questions are asked about the algorithms we have studied, and about correctness and complexity. We probe understanding by asking to explain particular cases and to explain how fixes or variations of algorithms compare in terms of complexity.

***What would be an ideal exam?***

I think the current exam is ideal. We try to make the questions clear, fair and able to distinguish understanding. Our marking scheme is detailed. The only way to do it better would be to interview every student individually (as is done in Italy) ;-)