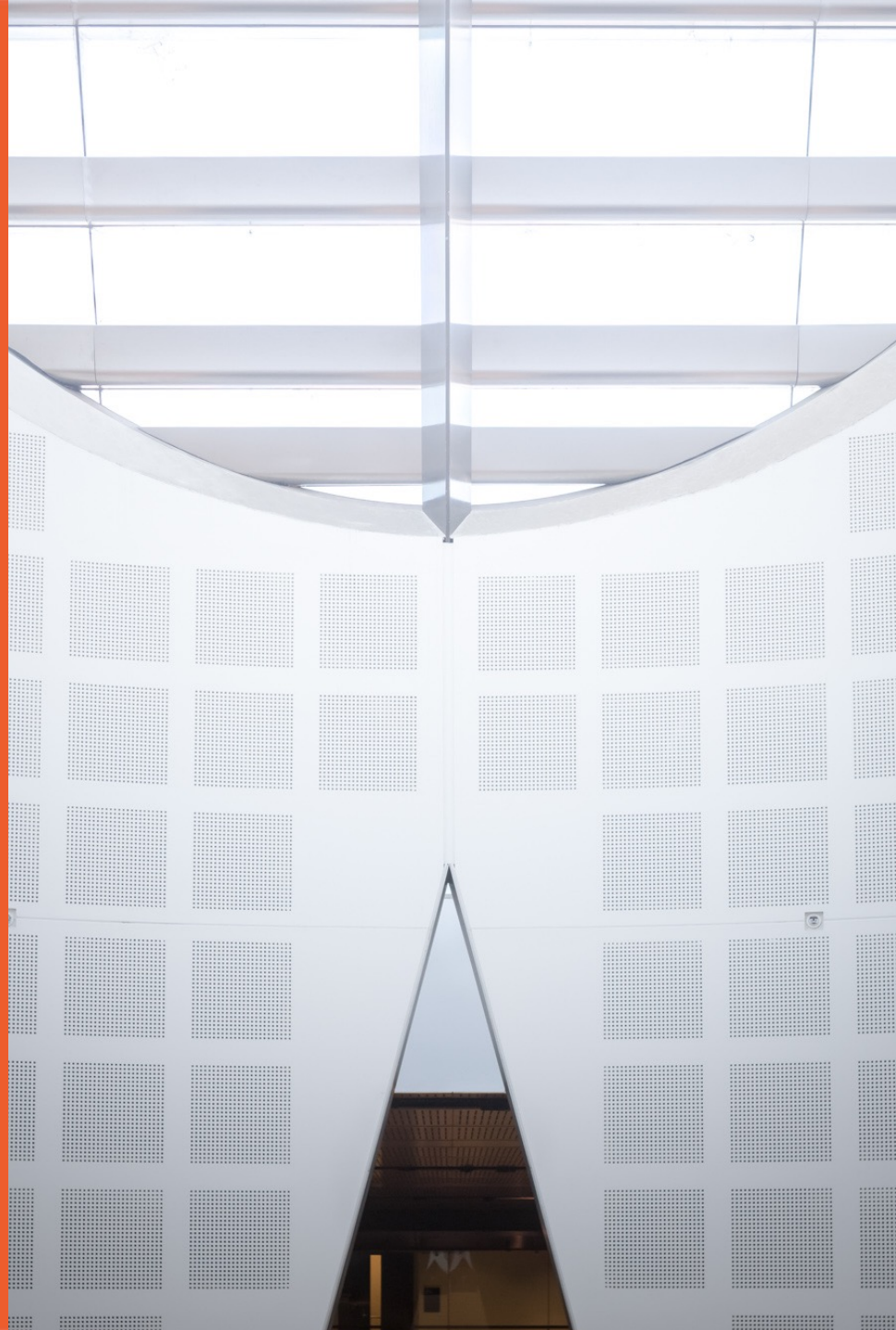
COMP2123

Week 13: Recap and Exam Review

Dr. André van Renssen

School of Computer Science



THE UNIVERSITY OF SYDNEY

Quick announcements

Fill out online Unit of Study Survey

- https://student-surveys.sydney.edu.au/students/
- Use the free text to help us make this better for next years students. “Pay it forward”

Examples of changes based on last year’s feedback:

- wrote the guide on how to approach algorithmic problems
- designed programming exercises

Week 13 Quiz

Quiz 10 will be about the final. Unlike previous quizzes you will be able to attempt it multiple times.

It's available until the start of the exam.

Looking back

- Lecture 1a - Introduction
- Lecture 1b - Analysis
- Lecture 2 - Lists
- Lecture 3 - Trees
- Lecture 4 - Binary Search Trees
- Lecture 5 - Priority Queues
- Lecture 6 - Hashing
- Lecture 7 - Graphs
- Lecture 8 - Graph Algorithms
- Lecture 9 - Greedy Algorithms
- Lecture 10 - Divide and Conquer I
- Lecture 11 - Divide and Conquer II

We covered a lot of ground!

Core concept 1: Abstraction layers

Abstract Data Type

Data Structure

Computer code

Problem definition

Algorithm

Computer code

Core concept 2: Algorithm analysis

A principled framework for evaluating algorithms:

- measuring performance of resource use
- proving correctness

These should inform your design and implementation choices

Learning outcomes

1. Proficiency in organising, presenting and discussing professional ideas [...]
2. Using mathematical methods to evaluate the performance of an algorithm.
3. Using notation of big-O to represent asymptotic growth of cost functions.
4. Understanding of commonly used data structures, including lists, stacks, queues, priority queues, search trees, hash tables, and graphs. This covers the way information is represented in each structure, algorithms for manipulating the structure, and analysis of asymptotic complexity of the operations.
5. Understanding of basic algorithms related to data structures, such as algorithms for sorting, tree traversals, and graph traversals.
6. Ability to write code that recursively performs an operation on a data structure.
7. Experience designing an algorithmic solution to a problem, coding it, and analysing its complexity.
8. Ability to apply basic algorithmic techniques (e.g. divide-and-conquer, greedy) to given design tasks.

Beyond this unit of study

SCS offers many algorithmic units:

- COMP2022 Models of Computation (S2)
- COMP3027 Algorithm Design (S1)
- COMP3530 Discrete Optimization (S2)
- COMP5045 Computational Geometry (S1)

Sydney Algorithms and Computation Theory group:

- weekly seminar on Algorithms research
- do a research project with us
- we are always looking for motivated honours students

What is examinable?

Everything from the lectures, the referenced sections of the textbooks, the tutorials, the quizzes, the assignments. Exceptions to this rule:

- when explicitly labeled as non-examinable.
- probabilistic analysis of randomized algorithms

In general though, if it happened during this unit, you are expected to know about it!

Focus on the things we put most emphasis on, as seen in tutorials and assignments

Final Exam Structure

2 hours writing plus 10 minutes reading

4 questions worth in total 60 points

Worth 60% of overall COMP2123 grade

Final exam has a 40% barrier

Do's and Don'ts

Open book exam:

- Can refer to slides, tutorial solutions, assignment solutions, books used in the unit
  - Making a 2-page summary is highly recommended
- Can't use the internet to look things up
- **Never** copy text verbatim from anywhere, including the slides (few points and potential academic dishonesty)
  - If you refer to anything from the permitted material, write in your own words

Type your answers and submit it as an assignment in **Canvas**

Handwritten/scanned answers will **not** be accepted.

Start your submission with your student ID

- Don't include your name

Problem 1

10 points

Analysis of given algorithms

Easy problem. Make sure you nail it!

Problem 2

10 points

Tracing algorithms/data structures on a given example

Easy problem. Make sure you nail it!

Problem 3

20 points

Design or modify an ADT

Medium/Hard problem.

Remember to:

- Describe your approach
- Prove correctness
- Analyze complexity (if there's a space requirement, don't forget to analyze this as well)

Problem 4

20 points

Design an algorithm that solves a problem

Medium/Hard problem.

Remember to:

- Describe your algorithm
- Prove correctness
- Analyze complexity

Problem 3 & 4

Check if you're supposed to use a specific technique:

- "design a greedy algorithm"
- "design a divide and conquer algorithm"

(Using a different technique will cost you a significant number of marks, but may still be better than a poorly explained incorrect attempt)

Let the running time requirement guide you:

- If we ask O(1) time, this limits your options considerably
- If we ask O(n) time, you can't sort the input

Exam technique

Read all questions to see which ones you can answer quickly

Plan how you will allocate time (wisely)

Start with easy problems and move to harder ones

Write clearly and efficiently

- Start with outline/bullet points, then expand if you have time
- No need for fancy style or overly formal

Figures take a lot of time to draw

- Describing in text is faster (a graph is a bunch of vertices and edges)
- **No scanned handwriting or paragraphs of text in figures allowed**

Pragmatic Advice

- Practice submitting a file in Canvas!
- Be alone in your room to avoid distractions
- Let housemates know when your exam is to avoid distractions
- Bring water
- Have clothing in layers
- Start your submission with your student ID (can prepare file in advance)
- Do not write your name on the exam (marking is anonymous)
- Breathe
- Relax
- Do not contact teaching staff during the exam

Good Luck!!!