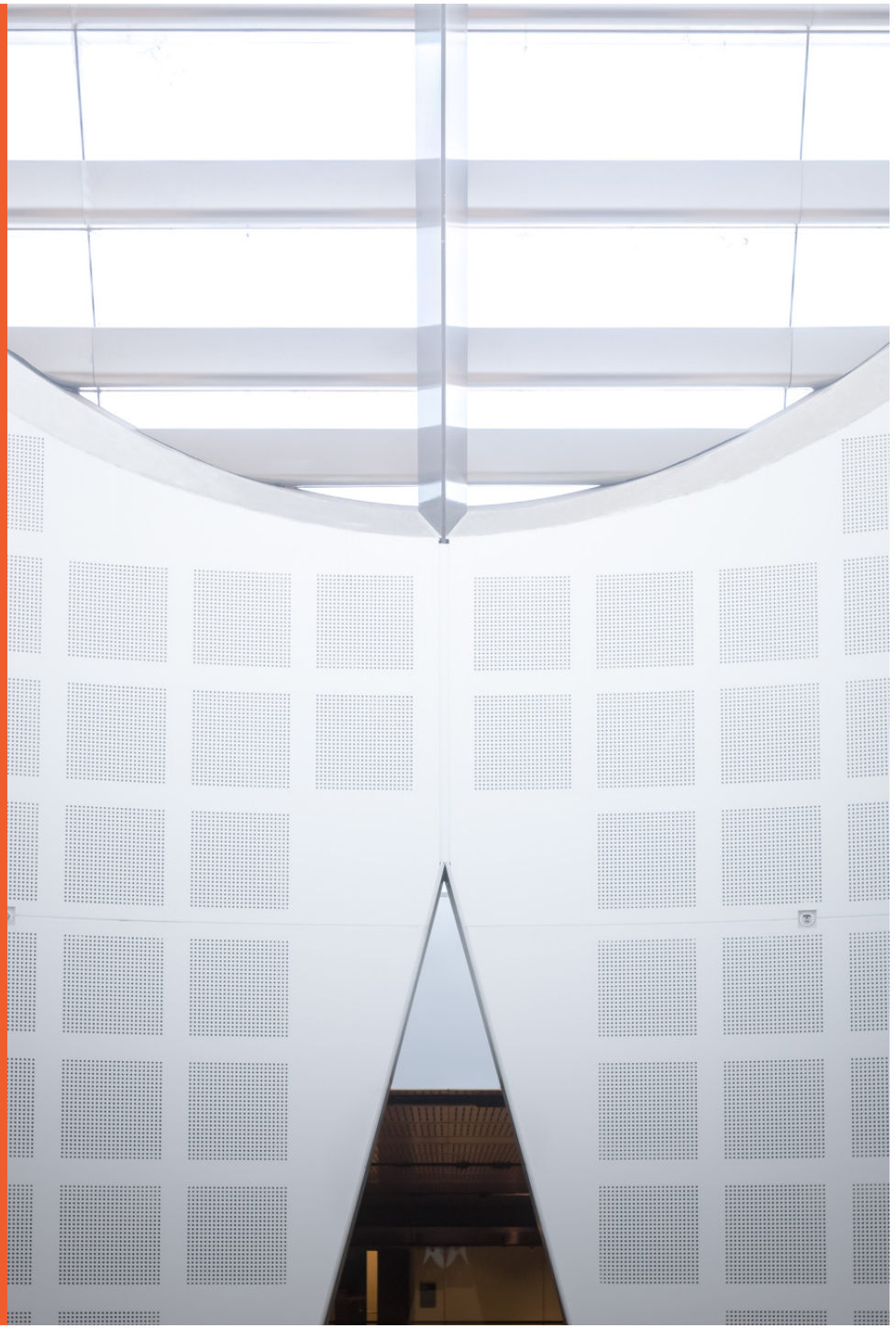
INFO1105/1905/9105
Data Structures
Week 13: Exam Review

Alan Fekete and Seokhee Hong
School of Information Technologies

THE UNIVERSITY OF SYDNEY



Reminder

- Fill out online Unit of Study Survey
  - Answer a few questions online at

    https://student-surveys.sydney.edu.au/students/
  - Use the free text to help us make the replacement units (COMP2123 and INFO1113) better for next years students
  - "Pay it forward"
- Check your asst1, quiz and lab marks

Week 13 lab

- Please prepare by coming with your solution for “exam revision” exercises 9 and 10
  - look under “revision material” link on eLearning

Where next (enrolment for 2018)

- Massive curriculum changes across the university
- For School of IT, guidance and plans checked by undergrad advisor
  - see reception (level 2 of SIT building) to make appointment
- Units of interest include
  - COMP3027/3927 Algorithm Design (replaces former comp2007; more clever algorithms, and ways to invent them yourself))
  - SOFT2201 Software Construction and Design (new unit, follows from the OO and Java aspects of info1105)
  - COMP2022/2922 Prog Languages, Logics and Models (lots of changes in content from 2017; including coverage of functional programming)
  - SOFT2412 Agile Software Development Practices (new unit, focus on group processes in Java settings)
  - DATA2001 Data Science: Big Data and Data Diversity (new unit, part of new major offered in new degrees)

Interested to learn some Python?

- New mostly-online 2cp unit OLET1307 will start in 2018
  - simpler aspects of Python programming
  - target for data science activities
- We want volunteers to try out some material over summer vacation
- No money, no credit, lots of patience needed
- If interested, please email alan.fekete@sydney.edu.au

Agenda

- *Focus of the Unit*
- Exam Coverage
- Exam Structure
- Exam Advice

From CUSP Unit Handout

"The unit will teach some powerful ideas that are central to quality software: data abstraction and recursion. It will also show how one can analyse the scalability of algorithms using mathematical tools of asymptotic notation. Contents include: both external "interface" view, and internal "implementation" details, for commonly used data structures, including lists, stacks, queues, priority queues, search trees, hash tables, and graphs; asymptotic analysis of algorithm scalability, including use of recurrence relations to analyse recursive code. This unit covers the way information is represented in each structure, algorithms for manipulating the structure, and analysis of asymptotic complexity of the operations. Outcomes include: ability to write code that recursively performs an operation on a data structure; experience designing an algorithmic solution to a problem using appropriate data structures, coding the solution, and analysing its complexity."

INFO1105 Learning Outcomes

- 1. Ability to analyse scalability of algorithms using mathematical tools of asymptotic notation. They will be able to analyse recursive structures by setting up and solving asymptotic recurrence relations.
- 2. 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.
- 3. Ability to write code that recursively performs an operation on a data structure.
- 4. Understanding of basic algorithms related to data structures, such as algorithms for sorting, tree traversals, and graph traversals.
- 5. Experience designing an algorithmic solution to a problem, coding it, and analysing its complexity.

Agenda

- Focus of the Unit
- *Exam Coverage*
- Exam Structure
- Exam Advice

What is examinable?

- Everything from the lectures, the referenced sections of the textbook, the labs, the weekly challenge tasks, the quizzes, the assignments
  - except when explicitly labeled as non-examinable
- If it happened during this unit, you are expected to know about it!

Data Structure Content (begin)

- List (index, positional, iterator)
  - implemented by array, linked list variants
- Stack, Queue, Deque
  - implemented by array, linked list variants
- Trees (including Binary Trees)
  - properties and definitions
  - traversals
  - implemented by linked nodes
- Priority Queue
  - implemented by heap-structured binary tree, in array
  - use in sorting

Learning Outcomes 2 and 4

Data Structure Content (continued)

- Map
  - implemented as hashtable variants (separate chaining, open addressing with different probing rules)
  - Sorted Map
    - implemented as binary search tree
- Graph
  - traversal algorithms
  - implemented as edge list, adjacency list, adjacency map, adjacency matrix
  - variants (directed, weighted) and their algorithms strong connectivity, (transitive closure, topological sorting, shortest paths)
- Sorting algorithms
  - those based on priority-queue
  - others (bubblesort, mergesort, quicksort)

Performance reasoning content

- Concept of worst-case runtime as function of input size
- Concept of worst-case space cost as function of input size
- Big-Oh notation for asymptotic growth of a function
- Know big-Oh costs for runtime and space of each of the algorithms that we covered on particular implementations
- Know how to determine big-Oh worst-case runtime, based on code structure
  - deal with loops
  - deal with recursion by finding the recurrence equation
  - know solutions of the recurrence equations that we covered

Learning Outcome 1

Coding content

- Use interface, extends and implements, to code ADT and implementation
- Work with linked structures
  - code to change inter-object references
  - understand NullPointerException and how to avoid it
  - draw diagrams of the structure and how it changes
- Clever patterns for generality eg use of generic types, Comparable, Comparator
- How to code recursion
  - especially on tree structures
  - including code that modifies the tree

Learning Outcome 3

Design content

- Given a task, choose data structures and use them appropriately to solve the task
  - being correct is the most important requirement
  - being efficient is also desirable
  - eg Asst1, revision task in week 12 lecture,

Learning Outcome 5

Agenda

- Focus of the Unit
- Exam Coverage
- *Exam Structure*
- Exam Advice

Exam Structure

- Two hours writing plus 10 minutes reading at the start
- Write answers on the question booklet in spaces
- Bring in one double-sided A4 sheet
  - can be handwritten or typed
  - leave it with your exam
- no calculators
- Do 5 questions worth in total 100 points
  - some common questions, but one different question for info1105 vs INFO1905 vs INFO9105
    - info1105 do qs 1-5
    - info1905 do qs 1-4 and 6
    - info9105 do qs 1-4 and 7
    - so 7 questions on the paper, but you do not choose which to answer!
- Worth 50% of overall INFO1105/1905/9105 grade

Question 1

- 20 points
- Know the data structures and operations
  - Based on the whole semester lecture content
- 5 medium-length answer parts (show the steps, draw a diagram, etc), each worth 4 points
- Eg: “Consider [data structure in some state]. Draw the state after [doing something]” or “Write the sequence of steps/labels when you perform [operation on data structure]”

- Very similar in style to the quiz review questions

Question 2

- 20 points
- performance analysis
- Q2(a) 10 points
  - fill in table by giving big-Oh cost of particular operations on particular implementation data structures
- Q2(b) 5 points
  - give big-Oh cost of a code fragment involving loops etc
- Q2(c) 5 points
  - give big-Oh for solution of a recurrence equation

Question 3

- 20 points
- 3 code-related questions, worth 5, 5, and 10 points
- Based on the lecture content, practical lab work, and assignments
- Eg “Consider [data structure]. Write some Java code to go in this class, to perform […]”, “Explain [some aspect of Java coding that arose in practical work or was discussed in lectures]”
- Note: most of the marks are for the approach and key steps, not for details of Java syntax or semantics

- See revision questions for some examples
  - you *will* be asked to code a method on a tree; this is best done using recursion, so practice this

Question 4

- 20 points
- a design question
- Eg “Consider [situation]. Show how to design software that does [task]. Your answer should describe the data structures that you would use, and indicate by pseudocode the calculations that you would perform. Also give (and justify) the big-Oh worst-case runtime for your method”
  - You can use Java syntax in pseudocode, but you do not have to
  - A clearly described correct but inefficient solution will gain at least half the marks

- See lecture of week 12 (but exam will involve dealing with only one method), and also revision question 10, for examples

Question 5 (Only for INFO1105!)

- 20 points
- 4 questions, worth 5 points each
- More questions (like Q1 but on more complex structures or operations) about the data structures we have covered

- Very similar in style to the 5 sets of quiz review questions, and the paper-based lab questions

Question 6 (Only for INFO1905!)

- 20 points
- 4 questions, worth 5 points each
- Questions on the data structures or analysis techniques covered in advanced hour of labs
  - amortized analysis
  - average-case analysis
  - binomial queue
  - rotations eg move-to-front tree
  - AVL tree, red-black tree
  - treap
  - splay tree
  - skiplist
  - sorting lower bound, quickselect
- Focus on knowing the content, not on solving fresh situations

- Similar in style to asst X, asst Y.

Question 7 (Only for INFO9105!)

- 20 points
- Eg “Consider [a situation]. Write a brief report, evaluating [proposal]” or “Consider [a situation]. Write a brief report proposing approach for [issue]”
- Similar in style to info9105-only aspects in Asst1 and Asst2

Agenda

- Focus of the Unit
- Exam Coverage
- Exam Structure
- *Exam Advice*

Reminder

- In order to pass INFO1105/1905/9105, you must
  - score at least 40 out of 100 points on the exam,
  - and also get at least 50 marks overall in the unit

Exam technique

- Plan how you will allocate time (wisely)
  - Use “reading time” to check your understanding (ask questions for clarification)
  - Also to plan time allocation to questions (approx 1 min per point!)
- Answer everything (get the “easy marks”)
  - Even if you don’t know the answer, show that you have some relevant knowledge
- For design question, better to be simple and clear and correct
  - only try fancy, efficient approach if you are really confident you can succeed
- Write clearly and efficiently
  - Start with outline/bullet points, then expand if you have time
  - No need for fancy style
  - Handwriting needs to be easy to read!
- If you need more space, use blank pages but leave a forwarding pointer in the provided space (where the marker will be looking)

Illness

- If you are unwell, and it seems that you won't be able to demonstrate your knowledge/skill properly, then you can **request special consideration**
- Follow the same procedure as during semester (get medical person to fill out special USyd form, scan and attach when you fill in the online form, within 3 days)
- Usual outcome: an alternate test, a few weeks later
- If you become sick during the exam itself, raise hand and speak to the invigilator
- The University goal is to get a fair assessment of what you have achieved

Pragmatic Advice

- Find the room location before the exam day itself!
- Come in plenty of time (don't rely on public transport running smoothly on the day of the exam)
- Bring spare pens
- Bring water
- Have clothing in layers
- **Have your student id** and put it on the desk
- Switch off mobile phone and put it under the desk
- Breathe
- Relax

Good Luck