INFO1105: DATA STRUCTURES

Semester 2, 2016 | 6 Credit Points | Mode: Normal-Day
Coordinator(s): Alan Fekete

WARNING: This unit is an archived version! See Overview tab for delivered versions.

1. INTRODUCTION

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.

2. LEARNING OUTCOMES

Learning outcomes are the key abilities and knowledge that will be assessed in this unit. See assessment summary table below for details of which outcomes are assessed where. Outcomes are listed according to the course goals that they support.

Design (Level 2)

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.

Engineering/IT Specialisation (Level 2)

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.

For further details of course goals related to these learning outcomes, see online unit outline at http://cusp.eng.usyd.edu.au/students/view-unit-page/alpha/INFO1105 .

3. ASSESSMENT TASKS

ASSESSMENT SUMMARY

Assessment name	Team-based?	Weight	Due	Outcomes Assessed
Quiz	No	20%	Multiple Weeks	1, 2
Tasks	No	10%	Multiple Weeks	2, 3, 4, 5
Assignment 1	No	8%	Week 8 (Friday, 5 pm)	1, 3, 5
Assignment 2	No	12%	Week 12 (Friday, 5 pm)	1, 2, 3, 5
Final Exam	No	50%	Exam Period	1, 2, 3, 4, 5

ASSESSMENT DESCRIPTION

Quiz*: Multiple quizzes during tutorial times throughout the semester. Each quiz will be computer-based or paper-based. Late submission is not possible; in case of special consideration, extensions or alternative sittings will not be offered; instead, reweighting should be applied.

Tasks*: Multiple programming exercises throughout the semester, automatically graded. Late submission is not possible; in case of special consideration, extensions or alternative sittings will not be offered; instead, reweighting should be applied.

Assignments: Practical individual work that will include the use, design, implementation, and analysis of data structures. Except through special consideration, late work will be penalized by 20% of the available maximum mark, for each day or part day after the work is due.

Final Exam: Covering all aspects of the unit of study.

ASSESSMENT GRADING

Final grades in this unit are awarded at levels of HD for High Distinction, DI (previously D) for Distinction, CR for Credit, PS (previously P) for Pass and FA (previously F) for Fail as defined by University of Sydney Assessment Policy. Details of the Assessment Policy are available on the Policies website at http://sydney.edu.au/policies . Standards for grades in individual assessment tasks and the summative method for obtaining a final mark in the unit will be set out in a marking guide supplied by the unit coordinator.
It is a policy of the School of Information Technologies that in order to pass this unit, a student must achieve at least 40% in the written examination. For subjects without a final exam, the 40% minimum requirement applies to the corresponding major assessment component specified by the lecturer. A student must also achieve an overall final mark of 50 or more. Any student not meeting these requirements may be

given a maximum final mark of no more than 45 regardless of their average.

IMPORTANT: POLICY RELATING TO ACADEMIC DISHONESTY AND PLAGIARISM.

All students must submit a cover sheet for all assessment work that declares that the work is original and not plagiarised from the work of others.

In assessing a piece of submitted work, the School of IT may reproduce it entirely, may provide a copy to another member of faculty, and/or to an external plagiarism checking service or in-house computer program and may also maintain a copy of the assignment for future checking purposes and/or allow an external service to do so.

See Policies section below for other policies relating to assessment and progression.

4. ATTRIBUTES DEVELOPED

Attributes listed here represent the course goals designated for this unit. The list below describes how these attributes are developed through practice in the unit. See Learning Outcomes and Assessment sections above for details of how these attributes are assessed.

Attribute	Method
Design (Level 2)	Transfering problem statments into design, utilizing popular data structures and algorithms to solve problems.
Engineering/IT Specialisation (Level 2)	Fundamental generic IT/computer science skills: the ability to analyse and manipulate data structures and the ability to design algorithms for this purpose.

For further details of course goals and professional attribute standards, see the online version of this outline at http://cusp.eng.usyd.edu.au/students/view-unit-page/alpha/INFO1105 .

5. STUDY COMMITMENT

Activity	Hours per Week	Sessions per Week	Weeks per Semester
Lecture	2.00	1	12
Laboratory	2.00	1	13
Independent Study	8.00		13

Standard unit of study workload at this university should be from 1.5 to 2 hours per credit point which means 9-12 hours for a normal 6 credit point unit of study. For units that are based on research or practical experience, hours may vary. For lecture and tutorial timetable, see University timetable site at: web.timetable.usyd.edu.au/calendar.jsp

6. TEACHING STAFF AND CONTACT DETAILS

COORDINATOR(S)

Name	Room	Phone	Email	Contact note
Prof Fekete, Alan			alan.fekete@sydney.edu.au	

LECTURERS

Name	Room	Phone	Email	Contact note
Dr Stavrakakis, John			john.stavrakakis@sydney.edu.au	

7. RESOURCES

RECOMMENDED REFERENCES

Michael Goodrich, Roberto Tamassia, Michael Goldwasser, *Data Structures and Algorithms in Java (6th edition)*. Wiley, 2014.

COURSE WEBSITE(S)

Sydney elearning site: http://elearning.sydney.edu.au

8. ENROLMENT REQUIREMENTS

ASSUMED KNOWLEDGE

None.

PREREQUISITES

INFO1103 OR INFO1903.

PROHIBITIONS

INFO1905.

9. POLICIES

IMPORTANT: School policy relating to Academic Dishonesty and Plagiarism.

In assessing a piece of submitted work, the School of IT may reproduce it entirely, may provide a copy to another member of faculty, and/or to an external plagiarism checking service or in-house computer program and may also maintain a copy of the assignment for future checking purposes and/or allow an external service to do so.

Other policies

See the policies page of the faculty website at http://sydney.edu.au/engineering/student-policies/ for information regarding university policies and local provisions and procedures within the Faculty of Engineering and Information Technologies.

10. WEEKLY SCHEDULE

Note that the "Weeks" referred to in this Schedule are those of the official university semester calendar https://web.timetable.usyd.edu.au/calendar.jsp

Week	Topics/Activities
Week 1	Introduction; administrivia
	Data abstraction; Map and List
Week 2	Linked lists; Iterators
Week 3	Stacks and queues
	Recursive operations on data structures
Week 4	Trees
	Big-Oh notation
Week 5	Priority Queue; Heap
Week 6	Binary search tree
Week 7	Graph
Week 8	Graph
	Assessment Due: Assignment 1
Week 9	Hash table; Set
Week 10	Applications
Week 11	Sorting
Week 12	spare for catchup
	Assessment Due: Assignment 2
Week 13	Unit of Study Review
Exam Period	Assessment Due: Final Exam