Course Outline

Contents

- Course Details
- Course Summary
- Assumed Knowledge
- Course Aims
- Student Learning Outcomes
- Teaching Strategies
- Teaching Rationale
- Assessment (https://webcms3.cse.unsw.edu.au/COMP2521/18x1/resources/13789/edit#assessment)
- Course Schedule (https://webcms3.cse.unsw.edu.au/COMP2521/18x1/resources/13789/edit#schedule)
- Resources for Students (https://webcms3.cse.unsw.edu.au/COMP2521/18x1/resources/13789/edit#resources)
- Student Conduct
- Course Evaluation and Development

Course Details

Course Code	COMP2521
Course Title	Data Structures and Algorithms
Convenor	Angela Finlayson (/users/z9801583)
Admin	Angela Finlayson (/users/z9801583)
Classes	Lectures: Tue 14:00 - 17:00 Central Lecture Block 8 (K-E19-105) Thu 10:00 - 13:00 Ainsworth G03 (K-J17-G03) Timetable for all classes (/COMP2521/18x1/timetable)
Units of Credit	6
Course Website	http://cse.unsw.edu.au/~cs2521/18x1/ (http://cse.unsw.edu.au/~cs2521/18x1/)
Handbook Entry	http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP2521.html (http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP2521.html)

Course Summary

The goal of this course is to deepen your understanding of data structures and algorithms and how these can be employed effectively in the design of software systems. It is an important course in covering a range of core data structures and algorithms that will be used in context in later courses. You explore these ideas in lectures, tutorials, lab classes, and assignments. Assessment involves labs, practical exams, a practical final exam, and a theory exam. At the end of the course, we want you to be a solid programmer, with knowledge of a range of useful data structures and programming techniques, capable of building significant software systems in a team environment, and ready to continue with further specialised studies in computing.

Topics: An introduction the structure, analysis and usage of a range of fundamental data types and the core algorithms that operate on them, including: algorithm analysis, sorting, searching, trees, graphs, files, algorithmic strategies, analysis and measurement of programs. Labs and programming assignments in C, using a range of Unix tools.

Assumed Knowledge

The official pre-requisite for this course is either COMP1511 or COMP1917 or COMP1921.

We assume that:

- you can program in the C programming language, and are familiar with arrays, strings, pointers, dynamic memory allocation, recursion
- you are able to design, implement, debug, test and document small C programs (up to several hundred lines of code)
- you are familiar with the Linux environment on CSE computers

Course Aims

The aim of this course is to get you to **think like a computer scientist** . Certainly sounds like a noble goal, but what does it really mean. How does a *scientist* , let alone a computer scientist, actually think?

What many types of scientists try to do is to understand natural systems and processes. For example, a geologist tries to understand the structure of the earth, a biologist tries to understand living organisms, a chemist tries to understand materials and reactions, and so on. Computer scientists don't, as the name might suggest, simply try to understand the structure and behaviour of computers, but are more concerned with understanding software systems (and the interaction between the software and the hardware on which it runs). Also, unlike other scientists, computer scientists frequently build the objects that they study.

During this course, we'll be looking at ways of creating, analysing and understanding software. Ultimately, you should be able to answer the question "Is this piece of software any good?" and be able to provide sound reasons to justify your answer.

This course follows on from the introductory C programming courses COMP1511 or COMP1917 or COMP1921. We cover additional aspects of the C programming language that were not covered in these course, and also look at some programming tools which were not covered (in detail) earlier. However, this course is not simply a second C programming course. The focus is on the ideas and abstractions behind the data structures and algorithms that are used.

COMP2521 is a critical course in the study of computing at UNSW, since it deals with many concepts that are central to your future studies in the area. Whether you are studying Computer Science, Software Engineering, Bioinformatics, Computer Engineering, or even a discipline outside the realm of computing, understanding a range of algorithms and data structures and how to use them will make you a much more effective computing problem solver in the future.

Student Learning Outcomes

After completing this course, students will:

1. be familiar with fundamental data structures and algorithms
2. be able to analyse the performance characteristics of algorithms
3. be able to measure the performance behaviour of programs
4. be able to choose/develop an appropriate data structure for a given problem
5. be able to choose/develop appropriate algorithms to manipulate this data structure
6. be able to reason about the effectiveness of data structures and algorithms for solving a given problem
7. be able to package a set of data structures and algorithms as an abstract data type
8. be able to develop and maintain software systems in C that contain thousands of lines of code

This course contributes to the development of the following graduate capabilities:

Graduate Capability	**Acquired in**
scholarship: understanding of their discipline in its interdisciplinary context	lectures, assignments
scholarship: capable of independent and collaborative enquiry	lab work, assignments
scholarship: rigorous in their analysis, critique, and reflection	tutorials
scholarship: able to apply their knowledge and skills to solving problems	tutorials, lab work, assignments
scholarship: ethical practitioners	all course-work, by doing it yourself
scholarship: capable of effective communication	blog, tutorials
scholarship: digitally literate	everywhere in CSE
leadership: enterprising, innovative and creative	assignments
leadership: collaborative team workers	lab work, assignments
professionalism: capable of operating within an agreed Code of Practice	all prac work

Teaching Rationale

Computer Science is, to a large extent, a practical discipline, and so COMP2521 has an emphasis on practice. Lectures will include exercises where we examine the practice of developing and analysing programs. Tutorials aim to develop analysis and understanding via practical case studies. Lab Classes also provide practice in program development and analysis. Assignments provide large case studies of software development.

Teaching Strategies

COMP2521 involves lectures, tutorials, labs, assignments and a text book.

Lectures aim to convey basic information about the course content and to model the practices and techniques involved in software development (i.e. we do demos). The most important components of the course, however, are the tutorials, labs and assignments. Tutorials aim to clarify and refine the knowledge that you got from lectures and from reading the text book and notes. Labs and assignments are where you get to put together and practise all of the ideas from the lectures, tutes and text. The only way to develop the skills to do effective software development is by practising them. If you slack off on the assignments and lab exercises (or, worse, rely on someone else to do them for you), you're wasting the course's most valuable learning opportunities.

The University requires us to assess how well you have learned the course content, and the primary approach to achieving this is via a final exam. An exam is the ultimate summative assessment tool; it gives you a chance, at the end of the course, to demonstrate everything that you've learned. Labs and assignments are a *learning* tool, not an assessment tool, so, in an ideal world, I would have them as pure learning exercises and award no marks for them. However, to give a more concrete incentive to do them (in a timely fashion), there are marks tied to them.

Lectures

Each week there will be 2 three hour lectures during which theory, practical demonstrations and case-studies will be presented. Lectures convey a small amount of information about the course content, but their main aim is to try to stimulate you to think about concepts and techniques. Feel free to ask questions at any stage, but otherwise please respect the right of other people around you who are trying to listen and (shhhhhh) keep quiet.

Text vs Slides

- **textbook** : contains all material for the course (and more), available in UNSW Bookstore at start of semester, describes material in lots of detail and is very well-written
- **slides** : the material I use in lectures, available online after (sometimes before) the lecture.

Tutorials

Tutorials aim to clarify ideas from lectures and to get you to think about design/analysis issues. There will be a number of exercises set for each tutorial class. The aim of the class is *not* to simply get the tutor to give you the answers; the aim is to focus on just one or two of the exercises and work through them in detail, discussing as many aspects, alternative approaches, fine details, etc. as possible. You must be active and ask questions in tutorials. Ideally, students should run the entire tute themselves, with the tutor being a moderator and occasionally providing additional explanations or clarifications. You may not get through all the exercises each week. That is ok. They can be used as additional private study material.

Lab Classes

Lab classes aim to give you practice in problem-solving and program development. Each week, there will be a few small exercises to work on. These exercises will be released in the week preceding the lab class. Except for in the first week, labs will be done in pairs, and you and you partner should discuss the exercises *before* going to the lab, to maximise the usefulness of the class. The exercises will need to be submitted (for our records) and will be assessed by your tutor. During the lab, your tutor will provide feedback on your approach to the problem and on the style of your solution.

Assignments

In the assignments, you will work on more substantial (hundreds of lines of code) programming exercises. The first assignment is an individual assignment; the second will be completed in groups. We expect all members of a pair or group to contribute to the assignments; part of your assignment mark will be tied to this. As noted above, assignments are the primary vehicle for learning the material in this course. If you don't do them, or simply copy and submit someone else's work, you have wasted a valuable learning opportunity.

Assessment

Item	Topics	Due	Marks
Prac Exams	Various Topics	Week 5 and 8	10% (5% each)
Assignment1	ADTs	Week 3	10%
Assignment2	Graphs	Week 8	15%
Labs	All topics	All Weeks	10%
Final Exam	All topics	Exam period	55%

Course Schedule

Week	Lectures	Tutes + Labs	Textbook Chapter	Assignments	Prac Exams
1	Introduction; Revision of data structures + ADTs	Linked List Revision		Ass1 out	-
2	Algorithmic Complexity, Recursion, Trees	ADTs		-	-
3	Sorting	Trees		Ass1 due	-
4	Graphs Representation and Traversal	algorithmic complexity and sorting		Ass 2 out	-

5	Graphs Algorithms	more sorting	-	Prac Exam 1
6	Symbol Tables, Binary Search Trees	Graphs	-	-
7	Balanced Binary Search Trees, Hash Tables	Graphs algorithms	-	-
8	Revision	Balanced Trees and HashTables	Ass2 due	Prac Exam 2

Resources for Students

COMP2521 follows the contents of the pair of books:

- *Algorithms in C, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching* (3rd Edition)
  by Robert Sedgewick, Addison-Wesley
- *Algorithms in C, Part 5: Graph Algorithms* (3rd Edition)
  by Robert Sedgewick, Addison Wesley

These two books are available as a bundle from the UNSW bookshop. They are expensive, but are useful well beyond this course, and will serve as a useful reference on the bookshelf of any serious programmer.

You may also be able to find on-line resources related to the text books. Robert Sedgewick has a series of videos on the topics in this course, but unfortunately they all seem to be in Java (which he has used for the new edition of his book). If you find any useful on-line resources, please let me know and I will add them to the Resources section of the course web site (with credit to the finder).

This website also has links to the auxiliary material/documentation that you'll need for the course. Solutions for all tutorial questions and lab exercises will also be made available. I'll review quiz and assignment solutions in the lectures.

Student Conduct

The **Student Code of Conduct** (Information (https://student.unsw.edu.au/conduct) , Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)) sets out what the University expects from students as members of the UNSW community. As well as the learning, teaching and research environment, the University aims to provide an environment that enables students to achieve their full potential and to provide an experience consistent with the University's values and guiding principles. A condition of enrolment is that students *inform themselves* of the University's rules and policies affecting them, and conduct themselves accordingly.

In particular, students have the responsibility to observe standards of equity and respect in dealing with every member of the University community. This applies to all activities on UNSW premises and all external activities related to study and research. This includes behaviour in person as well as behaviour on social media, for example Facebook groups set up for the purpose of discussing UNSW courses or course work. Behaviour that is considered in breach of the Student Code Policy as discriminatory, sexually inappropriate, bullying, harassing, invading another's privacy or causing any person to fear for their personal safety is serious misconduct and can lead to severe penalties, including suspension or exclusion from UNSW.

If you have any concerns, you may raise them with your lecturer, or approach the School Ethics Officer (mailto:ethics-officer@cse.unsw.edu.au) , Grievance Officer (mailto:grievance-officer@cse.unsw.edu.au) , or one of the student representatives.

Plagiarism is defined as (https://student.unsw.edu.au/plagiarism) using the words or ideas of others and presenting them as your own. UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Plagiarism and Academic Integrity (https://student.unsw.edu.au/plagiarism)
- UNSW Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism. In particular, you are also responsible that your assignment files are not accessible by anyone but you by setting the correct permissions in your CSE directory and code repository, if using. Note also that plagiarism includes paying or asking another person to do a piece of work for you and then submitting it as your own work.

UNSW has an ongoing commitment to fostering a culture of learning informed by academic integrity. All UNSW staff and students have a responsibility to adhere to this principle of academic integrity. Plagiarism undermines academic integrity and is not tolerated at UNSW. Plagiarism at UNSW is defined as using the words or ideas of others and passing them off as your own.

If you haven't done so yet, please take the time to read the full text of

- UNSW's policy regarding academic honesty and plagiarism (https://student.unsw.edu.au/plagiarism)

The pages below describe the policies and procedures in more detail:

- Student Code Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)

- Student Misconduct Procedure (https://www.gs.unsw.edu.au/policy/documents/studentmisconductprocedures.pdf)
- Plagiarism Policy Statement (https://www.gs.unsw.edu.au/policy/documents/plagiarismpolicy.pdf)
- Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf)

You should also read the following page which describes your rights and responsibilities in the CSE context:

- Essential Advice for CSE Students (https://www.engineering.unsw.edu.au/computer-science-engineering/about-us/organisational-structure/student-services/policies/essential-advice-for-cse-students)

Course Evaluation and Development

Student feedback on this course, and on the effectiveness of lectures in this course, is obtained via electronic survey (MyExperience) at the end of each semester. Student feedback is taken seriously, and continual improvements are made to the course based in part on this feedback. Students are strongly encouraged to let the lecturer in charge know of any problems as soon as they arise. Suggestions and criticisms will be listened to openly, and every action will be taken to correct any issue or improve the students' learning experience.

Resource created 4 years ago (Thursday 02 November 2017, 09:59:16 AM), last modified 4 years ago (Saturday 11 November 2017, 02:26:57 PM).

Comments (/COMP2521/18x1/forums/search?forum_choice=resource/13789) (/COMP2521/18x1/forums/resource/13789)

There are no comments yet.