Course Outline

Contents

- Course Details
- Course Summary
- Assumed Knowledge
- Student Learning Outcomes
- Teaching Strategies
- Teaching Rationale
- Student Conduct
- Assessment
- Course Schedule
- Resources for Students
- Course Evaluation and Development

Course Details

Course Code	COMP2521
Course Title	Data Structures and Algorithms
Convenor	John Shepherd (/users/z9300035)
Admin	Kevin Luxa (/users/z5074984)
Classes	Timetable for all classes (/COMP2521/20T2/timetable)
Consultations	Consultation are online in Bb Collaborate
Units of Credit	6
Course Website	http://cse.unsw.edu.au/~cs2521/20T2/ (http://cse.unsw.edu.au/~cs2521/20T2/)
Handbook Entry	http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP2521.html (http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP2521.html)

Note

This term, all classes will be conducted online. Lectures will be replaced by topic-based videos, which you can watch in your own time; each video will cover a single topic and could be anywhere from 30 mins to 60 mins in length. The lecture time slots will be used for problem-solving sessions. Tutes/Labs will be held in the timetabled time slots, but will be done via Blackboard Collaborate (unless we find a better tool). Assignments and quizzes will be done online as usual. The Final Exam will most likely be held online.

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 lectres, tutorials, lab classes, and assignments. Assessment involves labs, tutes, quizzes, a final exam involving both practice and theory. 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.

Executive Summary

A summary of the critical things to know about COMP2521:

- attempt all of the quizzes, labs, tutorials, and assignments yourself
- always try to produce a better program than last time
- while viewing the lecture content, think critically about what's being said/shown
- the textbook is a useful reference source beyond this course
- assessment: labs: 18%, quizzes: 12%, assignments: 30%, final exam: 40%
- *enjoy the course!*

Now, please read the rest of this document.

Course Aims

The aim of this course is to get you to **think like a computer scientist** . This 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 understand natural systems and processes: a geologist, for example, 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 good?** and be able to provide sound reasons to justify your answer.

This course follows on from an introductory C programming course: COMP1511 (or COMP1911 plus a bridging course). We cover additional aspects of the C programming language that were not covered in those courses, 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 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.

Assumed Knowledge

Before commencing this course, students should be:

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

Installing Linux, possibly as a virtual machine, on your own computer would be a major bonus.

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

This course is taught the way it is because you learn computer science best by working through exercises ... lots of exercises. The tutes give small, theoretical exercises. The lab contain small practical exercises. The assignments are large practical exercises. If you complete all of these exercises yourself, you will have learned what we want, and will have no trouble passing the exam, and will be able to use all of your skills in your future programming endeavours.

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, we 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 three hours of 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.Note that we may *not* post the lecture slides before the lecture, to encourage you to concentrate on the fresh material! However, the lecture topics are available in the course outline.

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 we use in lectures, available online after 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.

Lab Classes

Lab classes aim to give you practice in problem-solving and program development. Each week, there will be one or two small exercises to work on. These exercises will be released in the week preceding the lab class. Labs will be done in pairs, and you and you partner should discuss the exercises *before* going to the online 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.Pairs will also be asked to do code reviews in the tutorials, to explain how they tackled a particular problem and describe interesting features of their 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.

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 a 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, at CSE you are responsible for ensuring that your assignment files are not accessible by anyone but you by setting correct permissions in your CSE home directory and for any code repositories you may use.

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.

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)

Special Consideration

If you are impacted from completing an assessment (assignment or exam), you should apply for Special Consideration through ..

- UNSW Special Consideration Portal (https://iaro.online.unsw.edu.au/special_consideration/home.login)

For an assignment, an extension may be granted, commensurate with the amount of time you have been affected. For an exam, you may be granted a Supplementary Exam. Whether consideration is granted depends on the severity of the problem and how well it is documented.

Students should familiarise themselves with the UNSW Special Consideration processes and policies

- Special Consideration (https://student.unsw.edu.au/special-consideration)

Assessment

AssessmentYour final mark in this course will be based on components from the assignment work, quizzes, labs, and the final exam.

Item	Topics	Due	Marks	Contributes to
Quizzes	All topics	Weeks 2,3,4,5,7,8,9,10	12%	1,2,3,4,5,6,7
Assignment1	Trees, text	Week 5	15%	4,5,7,8
Assignment2	Graphs, strings	Week 9	15%	4,5,7,8
Labs	All topics	Weeks 1,2,3,4,5,7, ~~8~~,9, ~~10~~	18%	1,3,4,5
Final Exam	All topics	Exam period	40%	1,2,3,4,5,6,7,8

Each quiz contributes 2 marks, and we will use your best 6 quiz marks to determine the mark out of 12. Similarly each lab contributes 5 marks, and we will use your best 5 lab marks to determine the mark out of 18.

There is a hurdle on the Final Exam; very poor performance in the exam will result in a fail, even if all your other assessment marks have been satisfactory. The following formula describes precisely how the mark will be computed and how the hurdle will be enforced:

```
labs        = mark for lab exercises         (out of 18)
quizzes     = mark for quizzes               (out of 12)
ass1        = mark for assignment 1          (out of 15)
ass2        = mark for assignment 2          (out of 15)

finalExam   = finalExam                      (out of 40)
okExam      = finalExam >= 17/40

mark        = quizzes + labs + ass1 + ass2 + exam
grade       = HD|DN|CR|PS  if mark >= 50 && okExam
            = FL            if mark <  50
           = UF            if mark >= 50 && !okExam
```

In other words, if you don't learn how to program by doing the prac work yourself, you'll score poorly in the final exam, and this will result in you not passing the course, even if your overall mark is at least 50.

Course Schedule

This is a tentative schedule for topics in the course. It is subject to change as the term progresses.

Week	Lectures	Tutes	Labs	Assignments	Quizzes	Notes

1	Course intro, C Revision, Algorithm Analysis, ADTs	Welcome, C revision	Lists, Performance	-	-	-
2	Trees, Tree ADT, Binary Search Trees (BST)	Algorithm Analysis, ADTs	ADTs, Debugging, GDB	Ass 1 released	Quiz 1	-
3	Balanced Trees, Search Tree Algorithms	Tree Structures, Assignment 1	Tree Algorithms	-	Quiz 2	-
4	Graphs, Graph ADT, Graph Algorithms (i)	Balanced Trees	Tree Algorithms	-	Quiz 3	-
5	Graph Algorithms (ii)	Graphs (i)	Graph Algorithms	Ass 1 due	Quiz 4	-
6	Flexibility Week	-	-	Ass 2 released	-	-
7	Hashing, Heaps, Tries	Graphs (ii), Assignment 2	Graph	-	Quiz 5	-
8	Sorting	Hashing, Heaps, Tries	~~Hashing~~	-	Quiz 6	-
9	String Processing	Sorting	Sorting	-	Quiz 7	-
10	Course Review	String algorithms	~~String algorithms~~	Ass2 due	Quiz 8	-

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, published by Addison-Wesley
- *Algorithms in C, Part 5: Graph Algorithms* (3rd Edition)
  by Robert Sedgewick, published by 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 we 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 will need for the course. Solutions for all tutorial questions and lab exercises will also be made available. We will review quiz and assignment solutions in the lectures.

Course Evaluation and Development

This course is evaluated each session using the myExperience system.

In the previous offering of this courses, students noted that the move to completely online learning was challenging, especially getting in touch with tutors and not getting timely assistance in labs or help sessions.

Based on their comments, we will change the way tutes/labs run and will add live-streamed problem-solving sessions in the lecture time slots.

Resource created about a year ago (Sunday 26 April 2020, 10:38:17 AM), last modified 11 months ago (Tuesday 28 July 2020, 05:07:16 PM).

Comments

(/COMP2521/20T2/forums/search?forum_choice=resource/44742)

(/COMP2521/20T2/forums/resource/44742)

There are no comments yet.