Postgraduate New Student Welcome Afternoon

Eric Martin

School of Computer Science and Engineering
University of New South Wales
Sydney, Australia

23.02.2017
Welcome to CSE...

Congratulations for your admission to one of the CSE postgraduate coursework programs.

What we will do tonight:

- get to know each other and enjoy some drinks and snacks
- go over some important information
- examine a case study, *decision trees*, to stress out that IT develops technology from pure, elegant scientific ideas which are then harnessed by powerful engineering techniques, and encourage you to study with a mindset based on that observation
- examine how the courses fit together, by reflecting on the field and seeing the big picture
The exemption exams

- **They should be taken at the beginning of your studies**, not when you start your second semester.

- They apply to the following courses:
  - COMP9020  Foundations of Computer Science
  - COMP9021  Principles of Programming
  - COMP9024  Data Structures and Algorithms
  - COMP9032  Microprocessors and Interfacing
  - COMP9311  Database Systems
  - COMP9414  Artificial Intelligence
  - COMP9331  Computer Networks and Applications

- Make sure you are aware of the rules that govern **advanced standing and exemption** by the end of the week.

- To get advanced standing or exemption for COMP9024 or COMP9331, you first have to get advanced standing or exemption for COMP9021.
Important deadlines

- You have till the end of week 1 to enrol into a program.
- You have till the end of week 1 to make any change to your course selection.
- You have to pay your fees by the end of week 1; otherwise you will be disenroled.
- If you have any financial problem and cannot pay your fees then contact Commonwealth Support and Fees as soon as possible, by email (csandfees@unsw.edu.au) or phone (9385 3119).
- You have till the 31st of March to drop a course without financial penalty.

Attend the first lecture of as many courses as possible you are interested in and can enrol in, read the course outlines, so as to maximise the chances of making the right course selection.
How to select major and courses

Know yourself!

- Know what your interests are.
- Know your strengths and weaknesses.
- If you are passionate with your studies, and select major and courses based on your interests and your strengths, you will enjoy your studies and then get very good marks and then impress potential employers and then get a rewarding, well-paid job.
Constraints

- Be aware of program requirements:
  - You must take GSOE9820 Engineering Project Management, unless you have completed one management course as part of your undergrad studies, or unless you have extensive work experience.
  - You must take COMP9900 Information Technology Project.
  - You have to select at least 6 courses from the list *Advanced Disciplinary Knowledge Courses*, which expose you in one way or another to research.

- Know which courses are core in the majors of interest
- Be aware of prerequisites and corequisites
- Be aware that some courses are offered in either session 1 or session 2, or neither...
- Be aware of possible time clashes.

**Plan ahead, while still remaining flexible**
Options

- You can get advanced standing for completed courses that have not been credited towards a degree.
- You can get exemption for courses that are core in your program, but that you have done already.
- Be aware of the option of taking a UNSW free elective.
- Be aware of the option of substituting some CSE courses by some approved nonCSE courses, one (free or approved) nonCSE course for every 4 CSE courses.
- There is an option to take research projects in the last semester of study; this option is subject to a number of extra conditions.
Working on a course

- Every course has lectures. Some courses have labs.
- Every course requires at least 12 hours of study per week on average, with many courses requiring substantially more time.
- Being a full-time student is equivalent to having a demanding full time job where you often have to work lots of extra hours.
- You will mainly learn by tackling (often challenging) assignments, many of which require tens of hours of work.
- The understanding you will get from attending lectures will be shallow. Practical exercises and assignments will help you master the material, gain a deeper understanding.
- Assignments are essential to acquire technical and problem-solving skills.
- Work steadily, from the very first week. Start working on the assignments as soon as they are released.
- It is convenient to own a PC or a laptop and work at home, but you can book working stations in one of our labs.
Passing a course

- Every course has its own magic formula.
  - Some use arithmetic means, others harmonic means;
  - some require you to pass one or more assessment components; getting a total mark of 50 might then not be enough to pass.
- Read in detail the course outlines. They tell you everything on how the courses will be assessed.
- There is sometimes a mid-term exam.
- There is almost always a final exam, which can be written, or practical, or both. It is usually 2-3 hour long and might include multiple-choice questions, some discussion, some programming. . .
- If you have been working steadily during the whole session, exams should be pieces of cake.
- You should not aim at just “getting a pass.” Employers know what a grade of 50 or hardly more means. . .
The problem: get away with the pain of marking

Maybe that the grade of a programming assignment can be determined from its syntactic properties (after all, it seems that in humanities, an essay could be automatically marked using pattern matching techniques, looking for particular keywords...), such as:

- number of functions
- minimum number of lines in functions
- maximum number of lines in functions
- average number of lines in functions
- maximum indentation level
- average indentation level
- ...
Producing and classifying the data

With a perl script and a bash script

produce_data

we produce data

data.arff

that we classify with tools from the Weka system which implements many machine learning techniques.
Desirable properties of entropy

- No mess when everything is white or everything is black
- Maximum mess when 50% is white and 50% is black

Given 3 classes $C_1$, $C_2$ and $C_3$ in proportions $p_1$, $p_2$, $p_3$, separate $C_1$ from $C_2 \cup C_3$, and then separate the proportion $(p_2 + p_3)$ of guys in $C_2 \cup C_3$ from each other:

$$E(C_1, C_2, C_3) = E(C_1, C_2 \cup C_3) + (p_2 + p_3)E(C_2, C_3)$$
**Definition of entropy**

Only one function satisfies the previous constraints: given \( n \) classes \( C_1, C_2, \ldots, C_n \), in proportions \( p_1, p_2, \ldots, p_n \),

\[
E(C_1, C_2, \ldots, C_n) = \sum_{i=1}^{n} -p_i \log_2(p_i)
\]

\[
- p \log_2(p) - (1 - p) \log_2(1 - p)
\]
A particular case

When

\[ p_1 = p_2 = \cdots = p_n = \frac{1}{n} \]

\[ E(C_1, C_2, \ldots, C_n) \] is equal to

\[ \log_2(n) \]

that is, the number of bits needed to encode in binary any of the \( n \) numbers in \( \{0, 1, \ldots, n - 1\} \).

Hence in this particular case, \( E(C_1, C_2, \ldots, C_n) \) is equal to the minimal number of bits needed to encode which of the \( n \) classes \( C_1, C_2, \ldots, C_n \) a particular datum belongs to.
Letting a condition split the data

With a test such as “max_indentation_level > 4”, a proportion $p$ of data go left (those that satisfy this condition), and a proportion $(1 - p)$ of data go right (those that do not satisfy this condition).

\[ C_1, C_2, \ldots, C_n \]
\[ \begin{array}{c}
   p \\
   1 - p
\end{array} \]
\[ C'_1, C'_2, \ldots, C'_n \]
\[ C''_1, C''_2, \ldots, C''_n \]

The mess goes from

\[ E(C_1, C_2, \ldots, C_n) \]

to

\[ p \times E(C'_1, C'_2, \ldots, C'_n) + (1 - p)E(C''_1, C''_2, \ldots, C''_n) \]
How to split the data

Choose the condition which reduces the mess as much as possible (maximise the information gain)!

The mess has been completely reduced when the data belong to only one class, so this is the best binary split:

\[
C_1, C_2
\]

\[
p \quad 1 - p
\]

\[
C_1 \quad C_2
\]
Decision trees

Problems to address:

- Missing data, noisy data, huge amounts of data...
- Attributes of various nature, symbolic, numeric...
- Not overfitting the data, pruning the tree...

A simple, most elegant scientific idea, and more than two decades of engineered improvements to turn theory into an evolving technology

ID3    C4.5    C5.0
What is computer science and engineering?

It is useful to think of a field as having an **object of study** and possibly developing **applications** based on the **knowledge** acquired by studying this object.

The objects of study of *cosmology*, *biology*, *optics*, *chemistry*, and their applications, if any, are pretty clear.

What is the object of study of computer science and engineering?
When the field began

The study of *computation*, that is, essentially, the study of the so called *computable functions*.

Which of these functions are computable?

1. The function that given two integers as input, returns their sum.
2. The function that returns 1 if there will still be human beings on Earth in year 3,000, and returns 0 otherwise.
3. The function that given a C program that compiles and that receives no input, returns 1 if the program crashes, waits for input or outputs something after some time, and 0 if it loops.
4. The function that given an integer $n$ as input, returns the $n$th digit after the decimal point in the decimal expansion of $\pi$.
5. The function that given two real numbers as input, returns 1 if these numbers are equal and 0 otherwise.
Mathematical tools

General (graphs theory, probability theory, logic, complexity classes, linear algebra):

COMP9020 Foundations of Computer Science

Of specific interest:

GSOE9210 Engineering Decision Structures
A key result is the existence of a *universal* computable function, which allowed one to envision programmable devices to solve arbitrary (computable) problems, rather than devices dedicated to solving a unique (computable) problem.

**COMP4141 Theory of computation**

Some problems have a computable solution *in theory*, but not in any practical term. Complexity theory addresses these issues. Algorithmic theory proposes practical computable solutions to many problems in a rather systematic way, considering families of (generic) problems and families of (generic) solutions, possibly only approximated.

**COMP9101 Design and analysis of algorithms**
**COMP4121 Advanced and Parallel Algorithms**
**COMP6741 Parameterised and Exact Computation**
Application: the computer

The computer is one of our main tools, similarly to the fact that a telescope is one of the main tools of an astronomer. But whereas the latter is an application from optics, the former is an application of our very field (and others, such as electronics).

A computer has to be designed and built.

COMP9032 Microprocessors and interfacing
COMP9222 Digital circuits and systems
COMP9211 Computer architecture

It has to implement a universal computable function.

COMP9102 Programming languages and compilers
Making it easier to speak to a computer

It needs high level programming languages to implement the solutions to specific problems. These languages can be classified into families and studied at an abstract level.

**COMP9161 Concepts of Programming Languages**

In practice, we will use particular instances of the former and program in some specific language.

**COMP9021 Principles of programming**
**COMP9024 Data structures and algorithms**
**COMP6771 Advanced C++ Programming**
Making a computer do what you want

To organise all data and programs than can be run on a computer and execute programs efficiently, a special program, the **operating system**, needs to be designed and implemented.

**COMP9201 Operating systems**
**COMP9242 Advanced Operating Systems** *

Operating systems must also be able to let many machines share the work

**COMP9243 Distributed Systems** *
Managing the complexity of very large programs

Algorithmic theory is about coming up with very smart solutions to small, well defined problems. But many problems do not require so much clever solutions, but mainly a huge number of lines of code, with the assurance that the result meets the specifications, is robust, error-free, secure, etc. This leads to the field of **Software engineering**.

Dealing with writing good software that is easy to read and maintain, and using techniques that make it secure and reliable:

- COMP9181 Language-based software safety *
- COMP9153 Algorithmic Verification *

Dealing with writing large programs:

- COMP9041 Software construction: techniques and tools

More generally, managing projects and teams (in general):

- GSOE9820 Engineering Project Management
Writing special kinds of programs

Programs that draw geometric or animated figures.

- COMP9415 Computer graphics
- COMP9018 Advanced graphics

Programs that play games

- COMP4431 Game Design Workshop
- COMP4432 Game Design Studio

Programs that fulfil the needs of businesses (which requires of course understanding these needs, hence programming is only one aspect of the following courses; in particular, designing appropriate systems is crucial).

- COMP9321 Web applications engineering
- COMP9322 Service-oriented architectures
- COMP9323 e-Enterprise project
The computer and the brain

Many believe that the brain is a computer; others believe that the brain can be simulated by a computer (anything the former can do, the latter can do as well). So let us try and let computers do what we do. This leads to the field of Artificial intelligence.

We can try and come up with a (simplified) model of the brain and write programs that simulate its activity.

COMP9444 Neural networks *

One can also completely ignore the workings of the brain and just focus on what we want the computer to achieve. Human beings are able to complete such a broad range of tasks.

COMP9414 Artificial intelligence
Focusing on particular tasks...

How can computers gain knowledge, how can we make them reason?

COMP4418 Knowledge Representation and Reasoning *

How can computers prove theorems?

COMP4161 Advanced topics in software verification *

How can computers build representations and analyse what they “see”?

COMP9517 Computer vision *

How can they make a robot move and execute the right set of actions?

COMP4411 Experimental robotics *

How can they make a robot plan its actions?

COMP9431 Robotic software architecture *
...and on more particular tasks

How can they build a model of their environment, that evolves as the latter changes?

COMP9417 Machine Learning and Data Mining *
COMP9418 Advanced Topics in Statistical Machine Learning *
Are we still in the realm of *computable functions* and their applications?

More likely, our object of study and the applications we develop on the basis of the knowledge we gain on that object has something to do with information.

*From wikipedia: Information is any kind of event that affects the state of a dynamical system. In its most restricted technical sense, it is an ordered sequence of symbols. As a concept, however, information has many meanings. Moreover, the concept of information is closely related to notions of constraint, communication, control, data, form, instruction, knowledge, meaning, mental stimulus, pattern, perception, and representation.*

Information is an object of study in physics and communication theory, with *entropy* as a key notion. Some aspects of it are our object of study.
How do computers communicate information to each other or to or from other devices, and in particular, using the Internet and using wireless technology?

COMP9331 Computer networks and applications
COMP9332 Network routing and switching
COMP9333 Advanced computer networks
COMP9334 System capacity planning *
COMP9335 Wireless mesh and sensor networks *
COMP9336 Mobile data networking *
COMP9337 Securing wireless networks *
COMP6733 Internet of things experimental design studio *
Communicating information (2)

How do we schedule the exchange of information?

COMP9151 Foundations of concurrency
COMP9152 Comparative concurrency semantics
COMP6752 Modelling Concurrent Systems *

How do we make sure that the exchange of information is secure?

COMP6441 Security engineering and cyber security
COMP9447 Security engineering workshop
Storing and extracting information

How do we efficiently store, search for, retrieve, and perform fundamental operations on pieces of information, and in particular, pieces of information stored on the web?

COMP9311 Database systems
COMP9315 Database systems implementation *

How can we compress the representation of a piece of information as much as possible and search from the compressed form?

COMP9319 Web data compression and search *

How do we deal with and exploit massive amounts of data?

COMP9313 Big Data Management
Discovering information

How do we discover information from data?

COMP9318 Data warehousing and data mining *

How do we make sure that the information we extract is relevant?

COMP6714 Information Retrieval and Web Search *

Discovering information from biological data:

BINF9010 Bioinformatics Methods and Applications
BINF9020 Computational Bioinformatics
And (possibly) turning some of that...

...into a successful business:

GSOE9220 Launching a start-Up
Conclusion

- CSE courses are demanding, but ultimately rewarding
- Work very hard, starting Monday!
- Devote some time to every course at least every second day, and preferably every day
- Never think that you will catch up later, seek help regularly whenever you need

Have a great time as a CSE student!