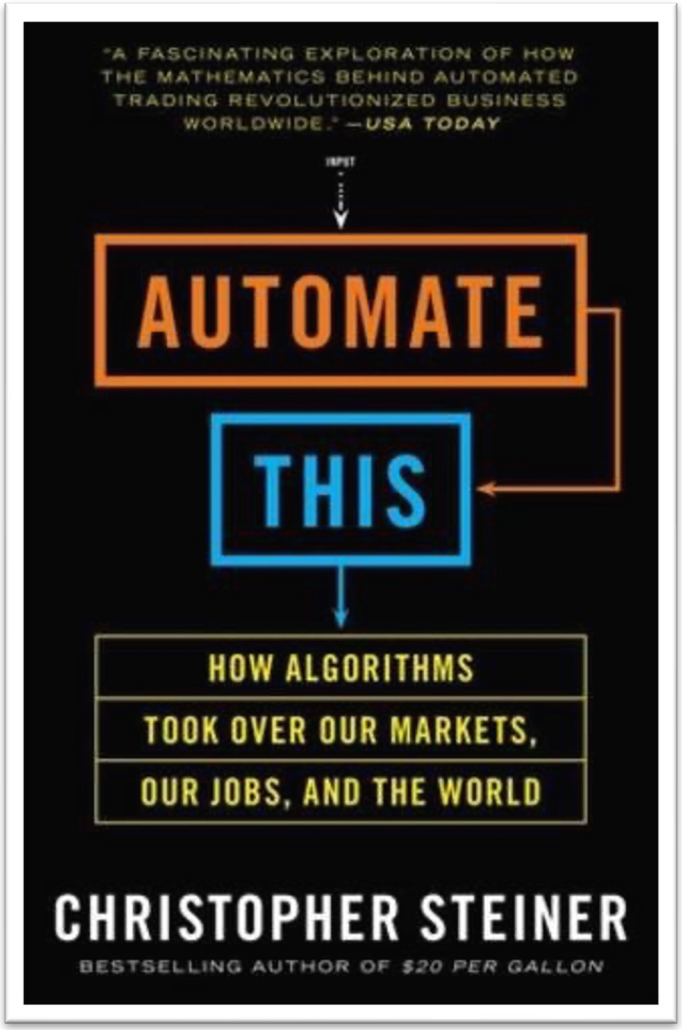
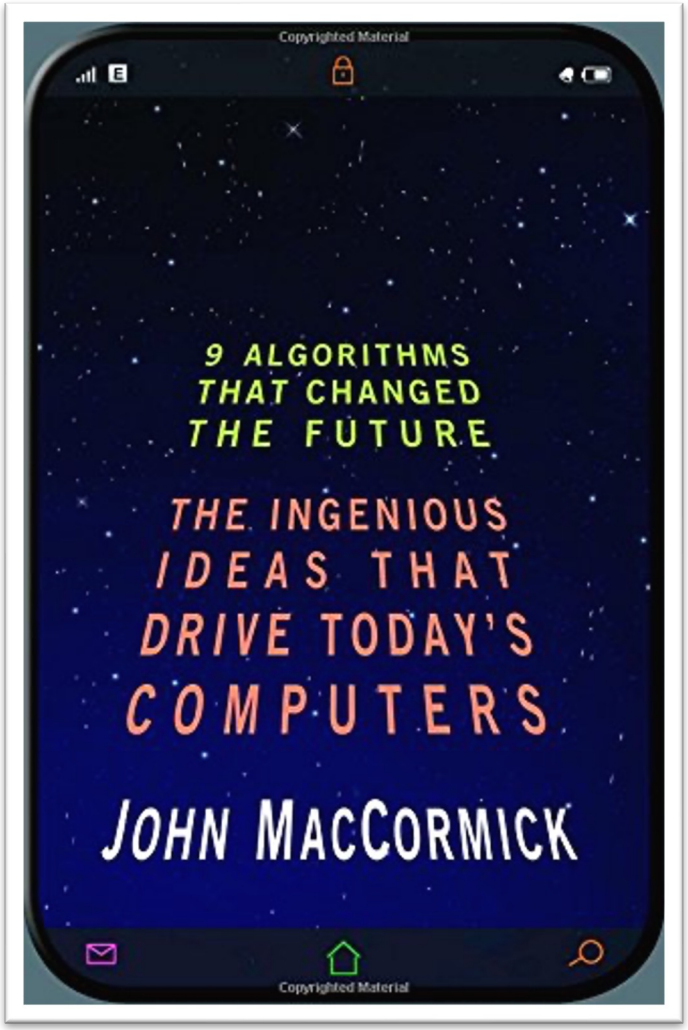
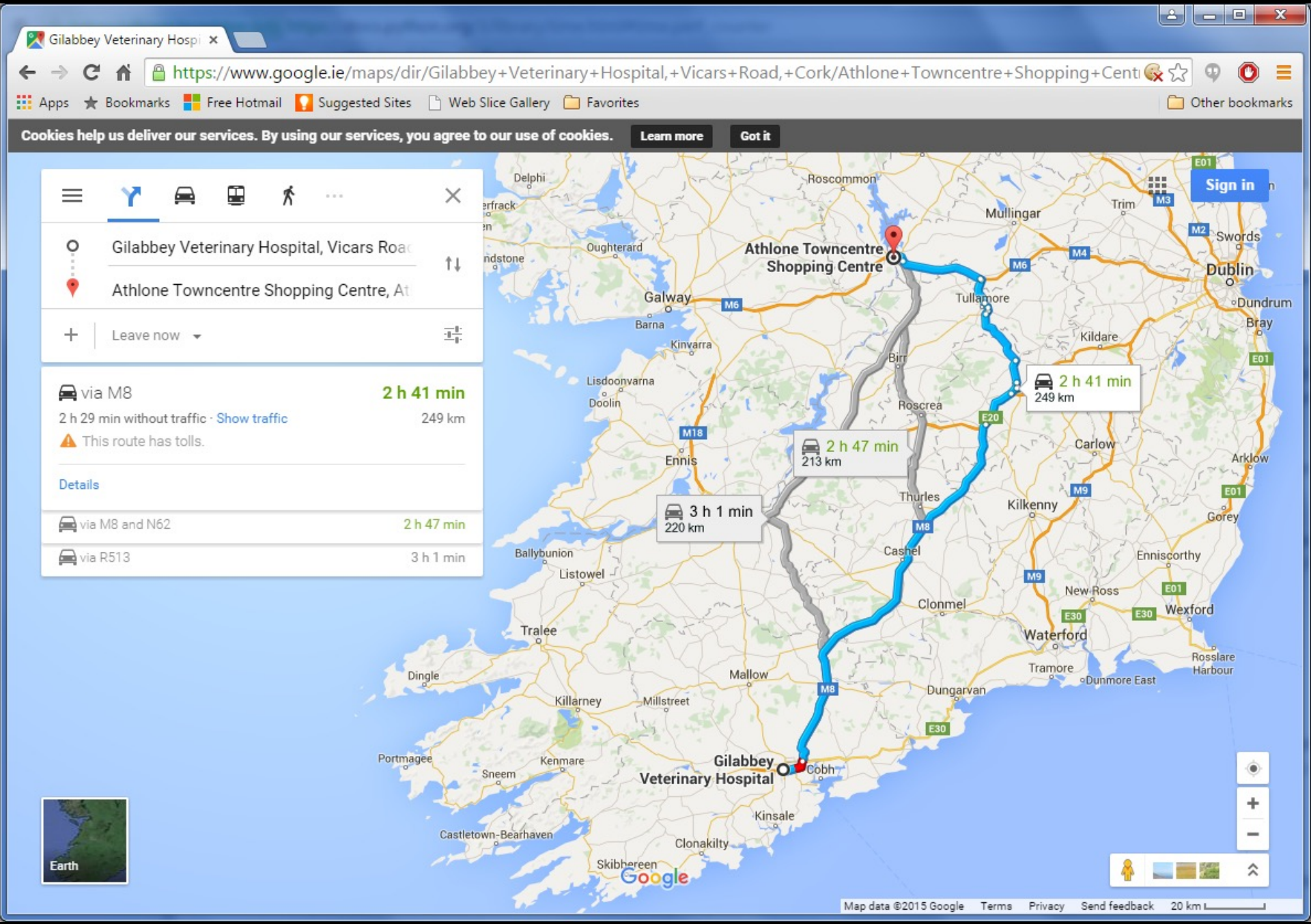
CS2515
Algorithms and Data Structures I


"A FASCINATING EXPLORATION OF HOW THE MATHEMATICS BEHIND AUTOMATED TRADING REVOLUTIONIZED BUSINESS WORLDWIDE." —USA TODAY
INPUT
AUTOMATE
THIS
HOW ALGORITHMS
TOOK OVER OUR MARKETS,
OUR JOBS, AND THE WORLD
CHRISTOPHER STEINER
BESTSELLING AUTHOR OF $20 PER GALLON


Copyrighted Material
9 ALGORITHMS
THAT CHANGED
THE FUTURE
THE INGENIOUS
IDEAS THAT
DRIVE TODAY'S
COMPUTERS
JOHN MACCORMICK
Copyrighted Material

Photograph:
Matthew Chattle/Rex/Shutterstock

Photograph: Henry Nicholls/Reuters


Gilabbey Veterinary Hospi
https://www.google.ie/maps/dir/Gilabbey+Veterinary+Hospital,+Vicars+Road,+Cork/Athlone+Towncentre+Shopping+Centi
Apps
Bookmarks
Free Hotmail
Suggested Sites
Web Slice Gallery
Favorites
Other bookmarks
Cookies help us deliver our services. By using our services, you agree to our use of cookies.
Learn more
Got it
Sign in
Gilabbey Veterinary Hospital, Vicars Roa
Athlone Towncentre Shopping Centre, At
Leave now
via M8
2 h 41 min
2 h 29 min without traffic · Show traffic
249 km
This route has tolls.
Details
via M8 and N62
2 h 47 min
via R513
3 h 1 min
2 h 41 min
249 km
2 h 47 min
213 km
3 h 1 min
220 km
Athlone Towncentre Shopping Centre
Gilabbey Veterinary Hospital
Roscommon
Mullingar
Trim
Swords
Dublin
Dundrum
Bray
Galway
Barna
Kinvarra
Oughterard
Tullamore
Kildare
Birr
Lisdoonvarna
Doolin
Roscrea
Carlow
Arklow
Ennis
Thurles
Kilkenny
Gorey
Ballybunion
Listowel
Cashel
Enniscorthy
Clonmel
New Ross
Wexford
Tralee
Waterford
Rosslare Harbour
Dingle
Mallow
Tramore
Dunmore East
Killarney
Millstreet
Dungarvan
Portmagee
Kenmare
Sneem
Cobh
Kinsale
Castletown-Bearhaven
Clonakilty
Skibbereen
Google
Earth
Map data ©2015 Google
Terms
Privacy
Send feedback
20 km

Algorithms

- An algorithm is a set of instructions which state how a task is to be performed
  - A knitting pattern
  - Instructions for setting up a new Xbox console
  - A recipe for cooking spaghetti bolognaise
  - Directions for walking from Boole Library to 1st year Computer Science lab in Western Gateway Building
  - The operations for turning an MP3 file into a sequence of sounds produced by an MP3 player

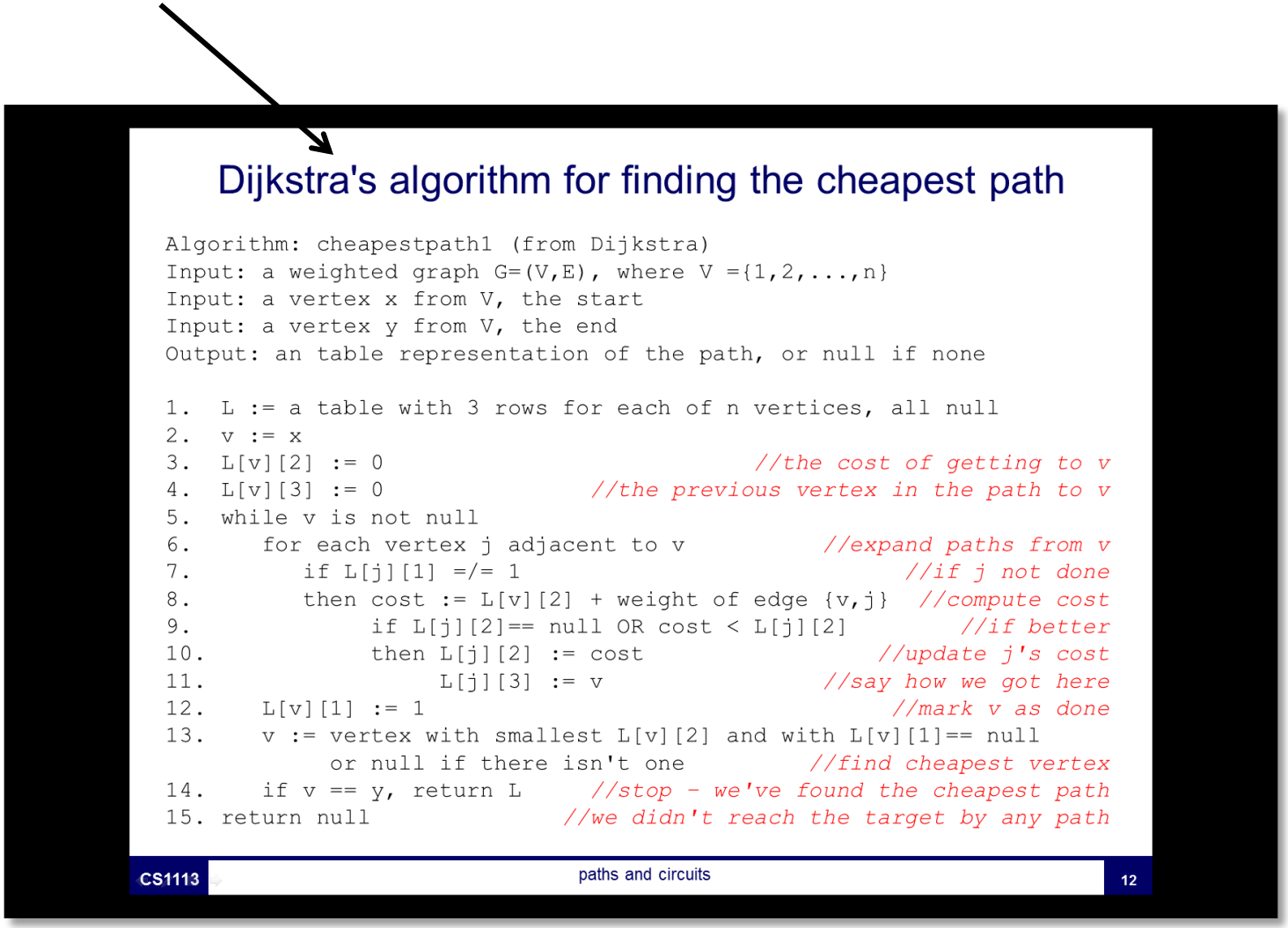
An algorithm is an ordered, deterministic, executable, terminating set of instructions

Algorithms

Algorithms: the abstract specification of the processes that are running on all of our computers, phones, games consoles, databases, autopilots, banking systems, autonomous vehicles, networks, ...

Algorithms process information maintained in data structures, and produce actions and results.

algorithm



Dijkstra's algorithm for finding the cheapest path

```
Algorithm: cheapestpath1 (from Dijkstra)
Input: a weighted graph G=(V,E), where V ={1,2,...,n}
Input: a vertex x from V, the start
Input: a vertex y from V, the end
Output: an table representation of the path, or null if none

1.  L := a table with 3 rows for each of n vertices, all null
2.  v := x
3.  L[v][2] := 0                              //the cost of getting to v
4.  L[v][3] := 0                   //the previous vertex in the path to v
5.  while v is not null
6.     for each vertex j adjacent to v          //expand paths from v
7.        if L[j][1] =/= 1                          //if j not done
8.        then cost := L[v][2] + weight of edge {v,j}  //compute cost
9.             if L[j][2]== null OR cost < L[j][2]        //if better
10.            then L[j][2] := cost                  //update j's cost
11.                 L[j][3] := v                  //say how we got here
12.    L[v][1] := 1                                  //mark v as done
13.    v := vertex with smallest L[v][2] and with L[v][1]== null
            or null if there isn't one          //find cheapest vertex
14.    if v == y, return L      //stop - we've found the cheapest path
15. return null                //we didn't reach the target by any path
```


We'll see a more efficient version of this algorithm later this year which relies on using an efficient *data structure* underneath

Data Structures

Data structures: the frameworks we use to maintain the data that are processed by the algorithms.

Inside data structures, we use algorithms to manipulate the data efficiently.

Abstract Data Types

Abstract Data Types: higher level patterns for interacting with data structures – patterns for reading data from the structure, for removing data, and for adding new data.

An ADT does not specify how an underlying data structure should be implemented.

Algorithms: three questions

Does it do what it is supposed to do?

How long will it take?

How much space (memory) does it need to run?

What is the point?

1. Your programs must be *correct*
   - not just bug-free code, but implementing an algorithm that does exactly what it is supposed to do.

2. Your programs should be *efficient*
   - programs that take too long to run are useless
   - understand the implication of choosing different library functions
   - understand the implication of choosing different design patterns
     - what may be easier for you to code may be much worse to run ...

Yawn – machines are getting faster every year – who cares about efficiency?

1. faster machines means users expect to solve bigger problems
2. some problems have known limits on their efficiency

What is the point (part 2)?

3. You must be able to advertise your code to other programmers *in terms they will understand*
   - state what abstract data types you are using
     - Programmers then know how to interact with them
   - state what data structures and algorithms you have used to implement them
     - Programmers then know how efficient different interactions would be

4. You should only use external code as specified
   - understand the specified ways to interact with the data
   - don't interact with it in other ways
     - abandons correctness and efficiency
   - be disciplined in your coding

... and a fourth question

Does it do what it is supposed to do?

How long will it take?

How much space does it need to run?

How do we implement it ?

We are going to use Python 3.

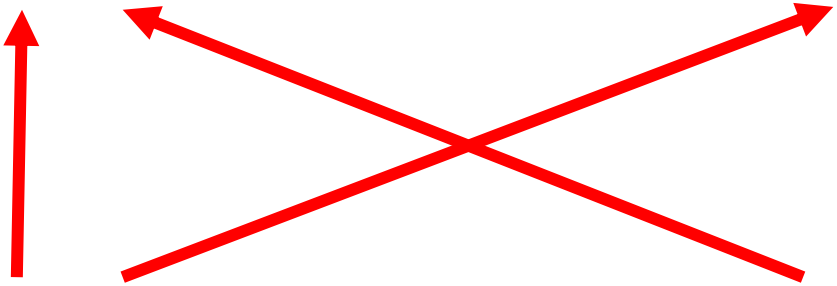
- understand the library functions
- learn how to implement structures efficiently
- the module (and Python 3) is based around the use of Classes and Objects

The module is a mix of theory and practice.

Four things you need to do in CS2515

Understand the concepts of data structures, and the processes for algorithms that manipulate them.

Understand the impact that different choices will have on the time software takes to run



Write and test programs to implement data structures and algorithms, and software that uses these programs.

Understand O(.) notation and be able to apply it carefully and correctly to algorithms and designs.

You must be able to program.
You must learn how to do O(.) analysis

How will you be assessed?

In CS2515, assessment will be by written exam, and by class tests.

The written exam will check that you understand the concepts, that you can apply known algorithms, that you can write new efficient algorithms using data structures to solve problems, and that you can do O(.) analysis on them.

If you don’t learn how to program these techniques, you will struggle badly in the exam. You will also struggle badly on the next module, CS2516, in Semester 2.

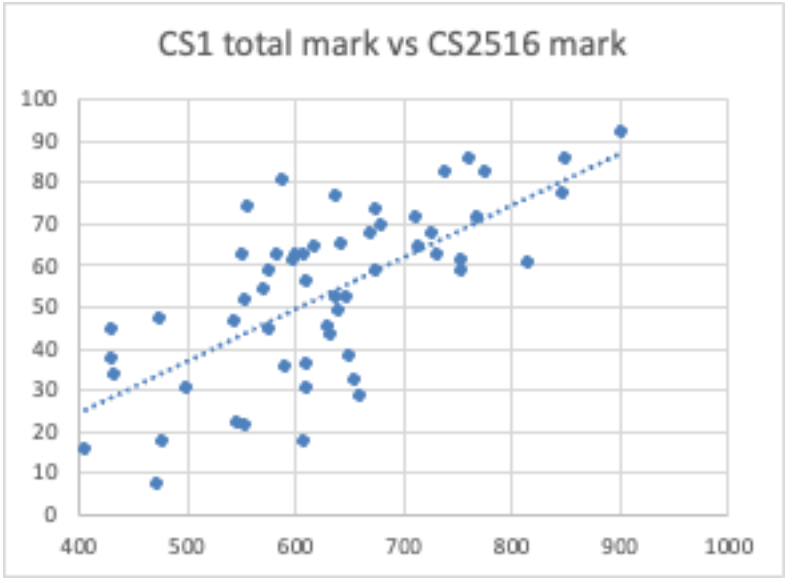
Programming Labs

There are weekly programming labs in CS2515. You will not be asked to submit code solutions for grading.

But it is really important that you work steadily through these exercises and learn how to code the solutions and observe the results, and keep up with the module as it goes along.

Do not be tempted to skips labs, or walk out early, or copy code obtained from online sources. If you don't go through the process of trying to code the solutions yourself, you will not learn how to do it, and your understanding of the concepts will be weak.

1st year relative to CS2516



The importance of being precise

For this module, and for working with data structures and algorithms in general, it is really important that you be precise and unambiguous in everything you write.

Small changes, or alternative interpretations, can make a massive difference to correctness and efficiency of code.

You should have learned how to be clear, precise and unambiguous in CS1112 and CS1113.

When you are answering exam questions, the detail of what you write down is important.

- vague pseudocode or hand-waving arguments about complexity will not get the marks

Course info

Lectures:

Wednesday, 2pm – 3pm, WGB G01

Friday, 10am – 11am, WGB 107

Lab classes: a 2 hour programming lab, interactive session

Thursday, 4pm to 6pm, WGB G24 (&G26)

Assessment: 80% final written exam (90 minutes)

20% continuous assessment and in-class tests

All course material will be posted on Canvas.

Outline Syllabus

Motivation and initial examples
Basics of algorithm analysis
Arrays and LinkedLists
Lists, Stacks, Queues, Priority Queues, Sets and Dictionaries
Binary trees and balanced trees
Search and traversal algorithms for trees
Binary heaps
Hashing

Applications (in the programming labs)

Plagiarism

1. Plagiarism is presenting someone else's work as your own. It is a violation of UCC Policy and there are strict and severe penalties.

2. You must read and comply with the UCC Policy on Plagiarism www.ucc.ie/en/exams/procedures-regulations/

3. The Policy applies to *all* work submitted, including software.

4. You can expect that your work will be checked for evidence of plagiarism or collusion.

Do not submit work:
- *copied from previous years' students*
- *scraped from a website*

5. In some circumstances it may be acceptable a reuse a small amount of work by others, but *only* if you provide explicit acknowledgement and justification.

6. If in doubt ask your module lecturer *prior* to submission. Better safe than sorry!

Self-directed Learning

The university assumes you will be spending at least 8 hours per week on a 5 credit course, not counting revision at the end of the year.

There are 4 hours timetabled for CS2515.

That means you will need to spend at least 4 hours per week working on the material outside timetabled classes. Most of this will be re-reading the lecture notes, and implementing solutions.

The module is a mix of theory and practice – *you will need to practice regularly*, or you will not survive.

The Lectures

The lectures will be delivered in person in the stated rooms. Occasionally some lectures will be delivered as video, with later in-person meetings to discuss the concepts.

Many of the lectures make extensive use of the whiteboard. You need to interact during the lecture – suggest solutions or strategies to solve problems, and help to complete exercises – you will learn more by participating.

Do not expect the in-person lectures to be recorded – apart from anything else, the system cannot capture the whiteboard

Slides will be made available after the lecture.

Textbook

There is no required textbook.

The following would be good books to buy or to consult:

1. Data Structures and Algorithms in Python,
   Goodrich, Tamassia & Goldwasser
   Wiley

2. Problem Solving with Algorithms and Data Structures
   Miller & Ranum

3. Data Structures and Algorithms with Python
   Lee & Hubbard
   Springer

Python

We will be using Python 3. If you are weak in python programming, you need to get practice now. Don't leave it until November.

Make sure you keep up-to-speed with the labs and exercises for CS2513

- CS2513 introduces object-oriented programming which we use in Algorithms and Data Structures
- the early weeks of that module are critical for being able to do the labs in CS2515, and for being able to understand the design of the data structures

Discrete Foundations

We will be assuming almost everything taught in CS1112 and CS1113: sets, functions, relations, logic, basic algorithms, graphs, trees, counting, complexity, Big-Oh notation, ...

Make sure you have that material to hand throughout the module. If you did not study those modules last year, please contact me today.

Remember the theme of those modules:

precise communication

Next lecture ...

THURSDAY 4pm

Introduction to Objects and Classes

Followed by 1-hour lab on programming with objects and classes