CS2516
Algorithms and Data Structures II

CS2515 studied Algorithms and Data Structures:

- array-based and linked structure implementations
- stacks, queues, lists, sets, priority queues, binary trees, binary search trees, balanced trees, binary heaps, maps, hash tables
- algorithms for searching and updating BSTs, AVL Trees, Binary Heaps and Hash Tables
- worst case complexity for space and runtime
- implementation in Python

So what is CS2516?

CS2516

Course info

Lecturer: Ken Brown

Lectures: Tuesday, 12pm – 1pm, WGB G08
Thursday, 11am – 12pm, WGB G02

Lab class: Tuesday, 9:00 – 11:00, WGB G24

Assessment: 80% by final written exam (90 minutes)
20% by continuous assessment and in-class tests

Module Objective

"Students should gain expertise in the use and implementation of fundamental data structures and algorithms, and their application in the creation of efficient software."

Outline Syllabus

Revision of CS2515
Complexity and Analysis
Sorting and Selecting
Graphs
Sample algorithms in text processing, matching, and memory management

All programming examples and assignments will be using Python 3

Learning Outcomes

"On successful completion of this module, students should be able to:

- Apply data structures and algorithms appropriately in formulating solutions of meaningful computational problems;
- Implement data structures and algorithms in a modern programming language;
- Analyze simple algorithms;
- Evaluate algorithms on the basis of performance."

Assessment

90 minute written exam, worth 80%
Continuous assessment, worth 20%

You must pass the combined exam and CA

The continuous assessment will consist of submitted software solutions to exercises, at times to be decided.

There will be a repeat exam in August.

Self-directed Learning

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

There are 4 hours timetabled for CS2516.

That means you will need to spend at least 4 hours per week working on the material outside of scheduled 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, or you will not survive.

Textbook

There is no required textbook.

Recommendations – the same as for CS2515:

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

Other Textbooks

The following three textbooks are also highly recommended for more advanced algorithms material:

1. *Introduction to Algorithms (3e)*,
   Cormen, Leiserson, Rivest and Stein
   MIT Press

2. *Algorithm Design*,
   Kleinberg and Tardos
   Pearson Education

3. *The Algorithm Design Manual*,
   Skiena
   Springer

Revision of CS2515

CS2515

almost everything in Python is an object, stored in some location in memory

2 ways of implementing data structures:

array-based

linked structures

array-based implementations

an array is a sequence of items all of the same type, laid out one after another in a fixed-size chunk of memory (i.e. a sequence of bits)

because we know the size of each item, we can compute the location of any item in the array, and jump straight to it – constant time

if we want to add items beyond the fixed size, we have to create a new array of the right size and copy everything across

linked structure implementations

a node is an object containing references to an item and to other nodes

a collection of nodes referring to each other creates a data structure, but where each node can be placed anywhere in memory

to access a particular item, we need to iterate through a chain of references

there is no limit on the size (apart from the total memory in the system)

main linked structures in CS2515

linked list – a linear sequence

if we are given a location, easy to update

... but slow to search

binary search tree

reasonably fast to update and fast to search

... assuming the tree is balanced

AVL tree - a balanced binary tree

reasonably fast to update and fast to search

because it is always balanced

the two main types of tree in CS2515

binary search tree – use it for *search*

for each node, all left descendants are less, and all right descendants are greater

binary heap – use it for *priority queues*

for each node, all descendants have lower priority (and tree is full, except for maybe lowest level, which is full from left up to a point and then empty)

- **the most efficient implementation uses an array-based list, not a linked tree**

maps and hashing in CS2515

CS2515 used an array-based list

- **may have other lists in each cell**

use maps for fast lookup and storage

- **the key for a value helps tell you where it will be stored**

- **almost always fast lookups by controlling the size of the underlying array, but occasionally can be slow (long chains, large buckets, etc)**

The CS2515 exam

most of the CS2515 exam was bookwork

three questions:

Q1 was on linked lists and stacks

Q2 was binary heaps

Q3 was on maps and hashing

Q1 was done best, Q2&3 not so well

Main issues in the exam

Q1 final part:

A *MinStack* is an ADT which extends the Stack ADT with one additional method, *min()*, which simply reports the minimum valued element in the stack. Give a design for an efficient implementation of MinStack – ideally, this implementation should allow constant time operation (i.e. O(1)) for min(), and should not change the complexity of any of the other methods compared to original Stack.

Be careful in your answer – make sure that your solution *works*.

Main issues in the exam

Q2 was about Binary Heaps, not Binary Search Trees

1. **Read the question carefully**
2. **Revise the whole module – do *not* try to guess which topics will come up and ignore other topics**

Main issues in the exam

Q3 was about hashing and maps using array-based lists underneath – it was all bookwork, but from the last couple of lectures

1. **Revise the whole module – do *not* try to guess which topics will come up and ignore other topics**

the ideas are too complex to cram into your memory in a couple of weeks

you need to do the work during the term

if you try the exercises and implement the programs as we step through the module,

- demonstrators can explain things to you
- you will understand more deeply
- it will be easier to understand later lectures
- it will be much easier to memorise things

Next Lecture

Recursion