COMP20003 Algorithms and Data Structures Summary

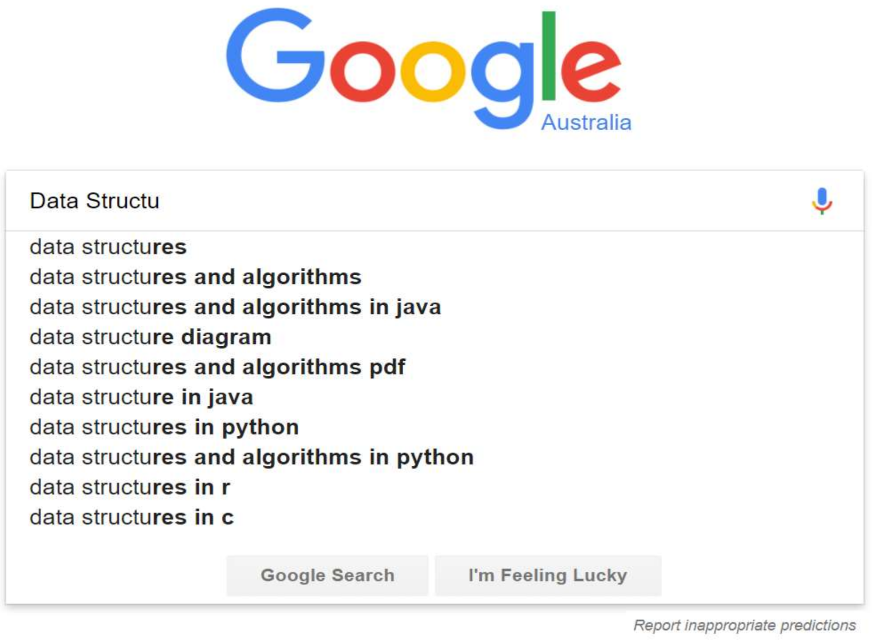
Nir Lipovetzky
Department of Computing and Information Systems
University of Melbourne

Semester 2

Different Majors in a CS subject:

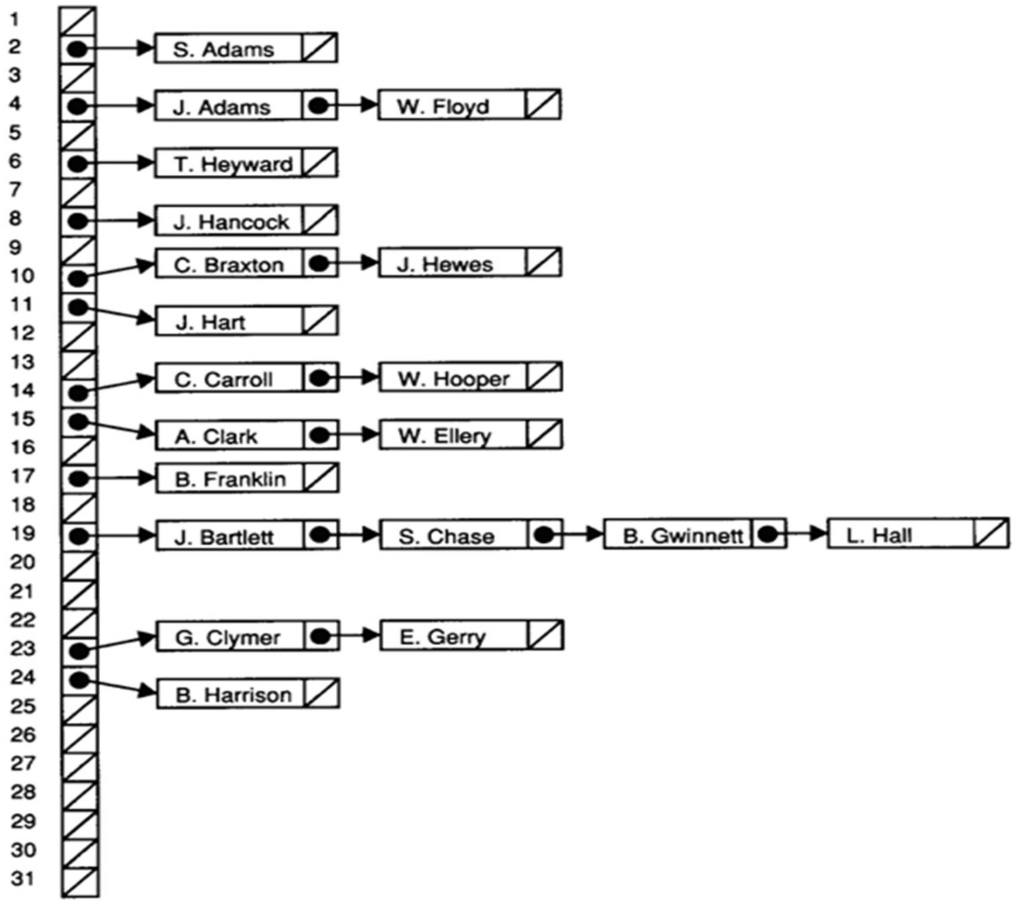
- What ultimately matters in this course is not so much where you end up relative to your classmates but where you, in Week 12, end up relative to yourself in Week 0

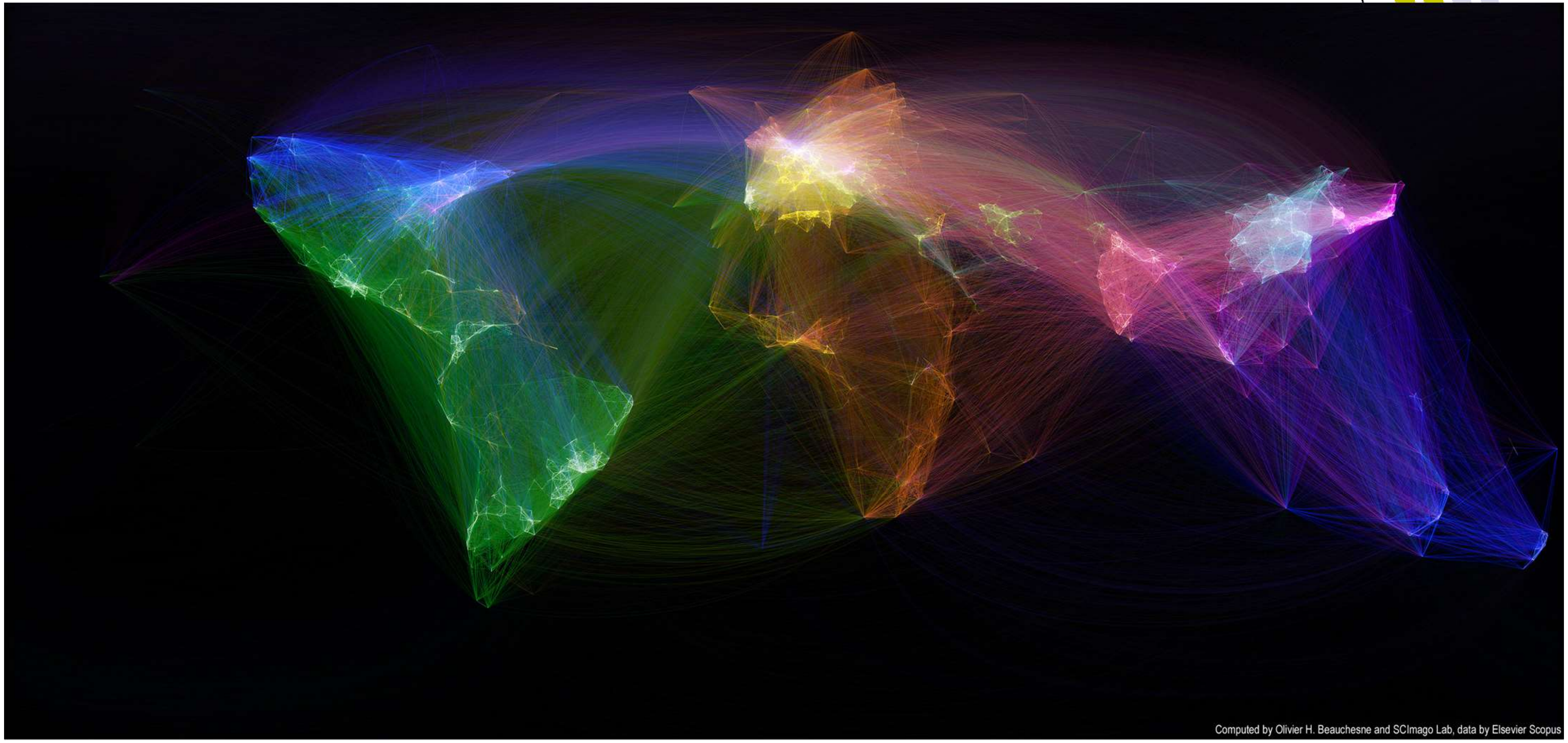
```
#
##
###
#####
######
#######
########
```



Data Structu

data structu**res**
data structu**res and algorithms**
data structu**res and algorithms in java**
data structu**re diagram**
data structu**res and algorithms pdf**
data structu**re in java**
data structu**res in python**
data structu**res and algorithms in python**
data structu**res in r**
data structu**res in c**


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
S. Adams
J. Adams
W. Floyd
T. Heyward
J. Hancock
C. Braxton
J. Hewes
J. Hart
C. Carroll
W. Hooper
A. Clark
W. Ellery
B. Franklin
J. Bartlett
S. Chase
B. Gwinnett
L. Hall
G. Clymer
E. Gerry
B. Harrison


Computed by Olivier H. Beauchesne and SCImago Lab, data by Elsevier Scopus

Let's review our CS backpacking experience...

Complexity

- Ideas behind complexity:
  - $n \geq n_0$
  - Estimate
  - Count the most expensive operation
  - Big-O notation
  - Big-Θ, Big-Ω
- Fibonacci example
- Expressing complexity as recurrences.

C programming: Dynamic memory allocation

- Use of `malloc()` and friends.
- Arrays and pointers, addresses in memory (pointer arithmetic).
- String processing.

Data structures

- Linear data structures:
  - Arrays, linked lists
  - Queues, Stacks (abstract data structures)
- Branching data structures:
  - Trees, balanced trees.

- Complexity of operations for higher-order data types depends on implementation.

Searching

- Worst case *vs.* average case
- Balanced trees:
  - AVL-tree
  - 2,3,4-tree (etc)
- Hashing:
  - Probabilistic behavior
  - Collision resolution

Sorting

- Distribution counting:
  - not comparison-based
- Divide and conquer sorting algorithms:
  - Quicksort: hard split, easy join
  - Mergesort: easy split, hard join
- Master theorem for divide-and-conquer algorithms.

A couple of single-purpose data structures and algorithms

- Priority Queues
  - Heap
- Union-find
  - Different representations
  - Simple array representation
- Topological sort:
  - Source removal algorithm

Graphs

- Representations
- Traversals
- Path problems:
  - Path finding
  - Single source shortest paths
  - All pairs single source shortest paths
  - Minimum spanning trees

When faced with an algorithmic problem...

- Use big-O to choose
- Sort something
  - Is there an O(n^2) that can be O($nlogn$) or O(n+K)
- Divide and conquer
  - Can the problem be broken down in to instances of the same problem?
  - Partitioning, Merging
  - Master Theorem
- Greedy
  - Does it produce best?
  - Do we have the data structures to make it fast?
- Dynamic Programming
  - Fibonacci Memoization (Intro Class)
- Graphs
  - Can we model the problem as a graph?
  - Run out of Memory or Time?
  - Directed, undirected
  - Weighted, not weighted
  - DFS – CC, SCC, Traversal, etc.
  - DFS,BFS,ID,A* – paths
  - Shortest paths: Dijkstra
  - Min. Spanning tree: Prim
  - Sort, Sources, Sinks

Changing Data structures might help...

- Array
  - Sorted, Unsorted
- Linked List
  - Sorted, Unsorted
- Binary Search Tree
  - Plain, Balanced: AVL Tree
- Hash Table
  - Chaining, Linear probing,
  Double hashing
- Disjoint Sets
  - As Array
  - As trees with path compression
- Priority Queue
  - Heap
  - Sorted Array
  - Unsorted Array
- Graphs
  - Adjacency Matrix
  - Adjacency List

How is the exam structured?

- Approximately 3 minutes per mark.
- Budget your time!

Questions

- Programming
  - (1 question, 18 marks)
  - Q1: Write 2 parts / functions.
- Algorithms and Data Structures
  - (4 questions, selected from topics above)

- *Note: There is some overlap between topics!*

What kinds of questions will be asked?

- The flavor is similar to:
  - Tutorial questions
  - Midterm test
  - Quizzes

- There is a range of difficulty.

What kinds of questions will be asked?

- Kinds of questions:
  - Working through problems, *e.g.* “Run algorithm *X* over the following data.”
  - What if… *e.g.* Modify the classic data structure, so that…
  - How would *Y* work for a dense graph? For a sparse graph?
  - What is the complexity of…?

Study tip

- Ask yourself questions.
- Give yourself problems to work through.
- Engage!

- Often you are asked to explain *briefly*.
- But not always.

- Show your workings for possible partial credit.
- Show comments for partial credit on the programming questions.

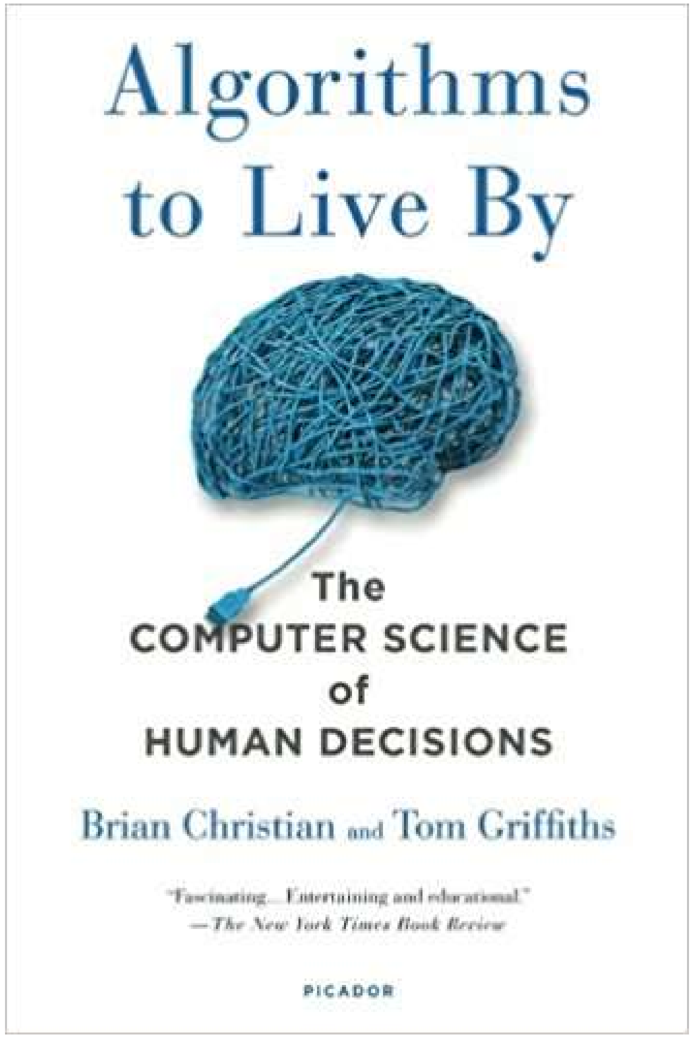
- I don’t look at anything on the left hand page unless you call my attention to it.

Reminders

- Question & Answer session:
  - Thursday 24 October, 4 P.M.
  - Office 6.19, level 6 Doug McDonell
  - Bring your questions, in priority order.

- Please remember to fill in your SES survey.

A Take away recommendation:


Algorithms to Live By
The COMPUTER SCIENCE of HUMAN DECISIONS
Brian Christian and Tom Griffiths
"Fascinating... Entertaining and educational."
—The New York Times Book Review
PICADOR

"In this remarkably lucid, fascinating, and compulsively readable book, Christian and Griffiths show how much we can learn from computers. We've all heard about the power of algorithms—but *Algorithms to Live By*actually explains, brilliantly, how they work, and how we can take advantage of them to make better decisions in our own lives."
—Alison Gopnik, coauthor of *The Scientist in the Crib*