COMP2054
Spring Semester
Algorithms Data structures & Efficiency (**ADE**)

Introduction

Andrew Parkes (Room C78)

andrew.parkes@nottingham.ac.uk

http://www.cs.nott.ac.uk/~pszajp/

"ADE": Algorithms Data structures & Efficiency

- Covers
  - **A**lgorithms,
  - **D**ata Structures, and
  - **E**fficiency
- **Algorithm:** step-by-step procedure for solving a problem in a **finite** amount of time.
  - No "MAGIC steps" allowed! (No "Zeno machines")
- **Data structures** : lists, binary trees, heaps, Hashmaps, graphs, etc.
- **Efficiency**: We will start with methods to analyse, classify and describe the efficiency of algorithms
  - Needed in order to be able to select "best algorithms"

Rough Comments

- On average, the ADE module is less formal than the COMP2065-IFR. However:
  - Some portions (definitions and proofs) are formal – and should be treated as such
    - (Optional: think how to convert them to Lean.)
  - Some portions – interpretations and "every day usage" are less formal
    - E.g. targeted to what would a software development team want to know

ADE Assessments

- 25% "standard coursework"
  - THREE in-class **CLOSED BOOK IN-PERSON** exam-conditions tests of 30 minutes during the Monday lab (either written or on Moodle).
  - Final mark based on "best 2 of 3" !
  - Dates: To Be Confirmed (TBC). Provisionally:

    C/W 1: Mon 26 Feb
    C/W 2: Mon 18 March or Mon 25 March
    C/W 3: Mon 13 May

    Put these dates in your diary! There are no "late submissions" – but you ask for an EC for missed tests, and can miss one anyway as it is best 2 or 3
- 75 % ExamSys/written exam
  - Summer exam session (May/June)
  - **CLOSED BOOK, IN-PERSON** !!
  - Two hours ExamSys; possibly with "on-paper" addition

Tutorials

- ONE 1-hour tutorial per person
  - ADE sessions will just be some weeks, not all – I will notify you on Moodle and email
  - These are essential for understanding the material, and will help with C/Ws, etc.
  - Please use the hour that is assigned to you on your personal timetable

Labs

- ONE 1-hour tutorial per person
- Labs will be sessions for doing some formative exercises and getting help as needed in preparation for the C/Ws.
  - ADE labs will just be some weeks - not all – I will notify you on Moodle and email
- Please use the hour that is assigned to you on your personal timetable

Email Etiquette

- **Only send email from your University account**! (Except in exceptional circumstances).
  - Only this allows staff to know who is really sending. We cannot discuss anything with random external email addresses as they may not be you.
  - Such emails may well just be treated as spam and deleted
- Do not expect an immediate reply – 3 working days is the norm. After that, send a gentle reminder.
- Include the module code in the subject line
  - This is a good practice for all modules. Staff teach multiple modules; hence writing "I need help with your module" will just cause delays.
  - To check which emails need to be answered, I may well search by the module code.
- Include all relevant details. E.g. lecture and slide number, tutorial question, etc.
  - If it is about personal process, or scores, etc, then include your student ID number.
  - Please do not just write "Help, I'm stuck". It does not help us to help you !
  - Often the process of fixing the details leading to answering the question

Study Skills Suggestion

Personal View:

- Often problems arise from going too fast
- Likely to often need to backtrack or start again
  - Backtracking is normal; do not see it as a failure
- Learn to seek chances to identify when need to "fix a bug in your thinking"
  - Allows to make better overall progress
  - Rebuilding 'internal models' is positive!
    - (The main goal of "real learning"?)

"Exercises" & Hints

- Will often put "**Exercise**" in slides
  - Please do them! At least think about them
  - They are not assessed, but are vital
  - They might well be a vital part of an exam question
- Hints:
  - If stuck because of 'algebra' then "generate your own examples" simply insert small numbers. E.g. put a=2, etc.
  - **Strongly encourage to start with the very simplest examples and work 'upwards'**
    - Many problems arise from reluctance to focus on the basics first

What are Algorithms?

- Algorithm: a well-defined step-by-step procedure for solving a problem in a finite amount of time.

For example:

- A recipe for cooking pizza
- A set of instructions for assembling a piece of furniture
- A procedure for finding web pages containing keywords
- A procedure for computing n! given n
- Sorting and searching algorithms
  - E.g. how to sort a pack of cards.
- A deep learning algorithm for deciding whether to grant someone a mortgage

What are Data Structures?

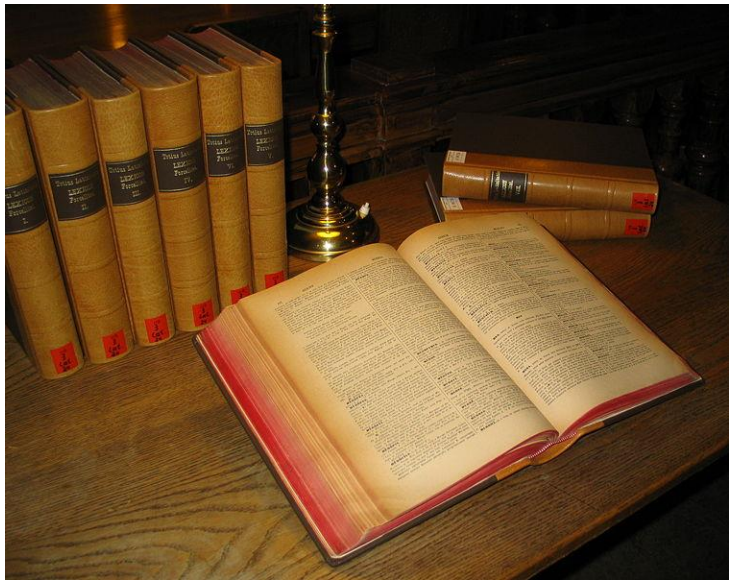
- Ways in which to store data in a well-defined and structured fashion to allow algorithms to:
  - make better use of it
  - maintain it more easily as the data changes

We will need to also define “better” and “more easily”

Data Structures in Everyday Life? …

Data Structure Example: Dictionary

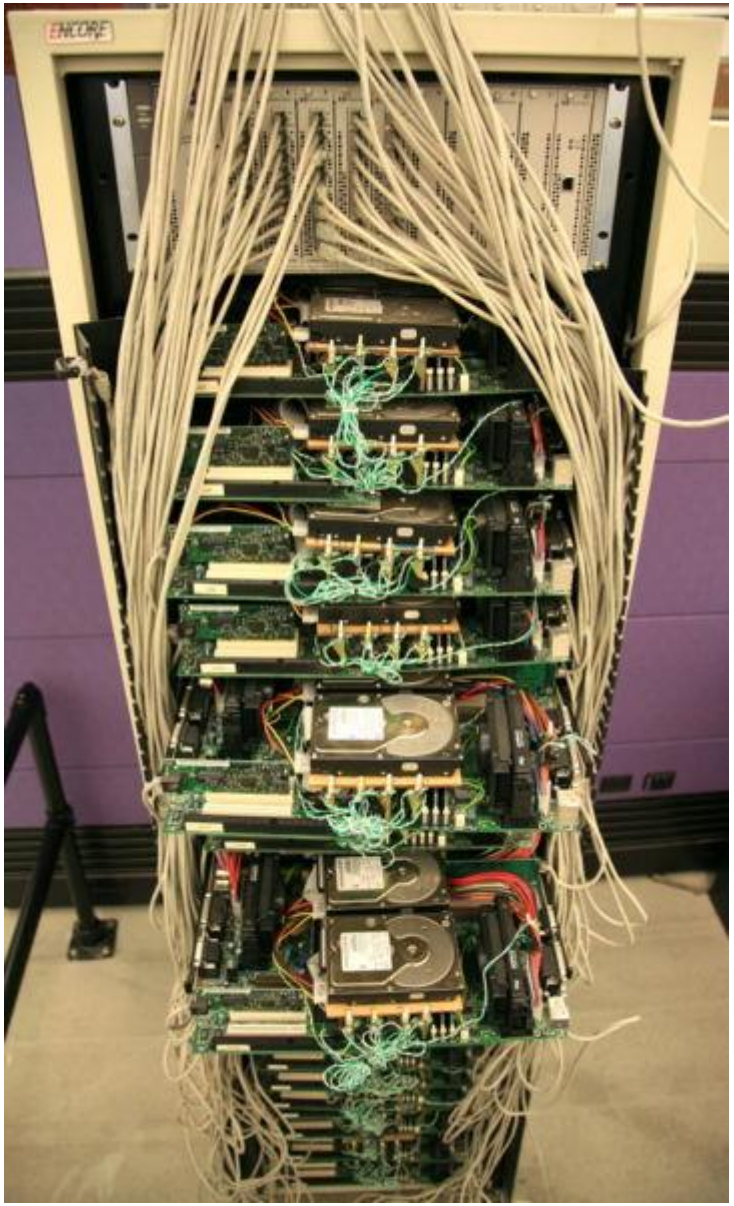
- Organising the data correctly helps to find it
- Predates computers by thousands of years
- Good for lookups, but bad for maintenance
  - Deletion leaves holes
  - Insertion is physical cut-and-paste
    - E.g. why a thesis might be double-spaced



Picture from https://en.wikipedia.org/wiki/Dictionary

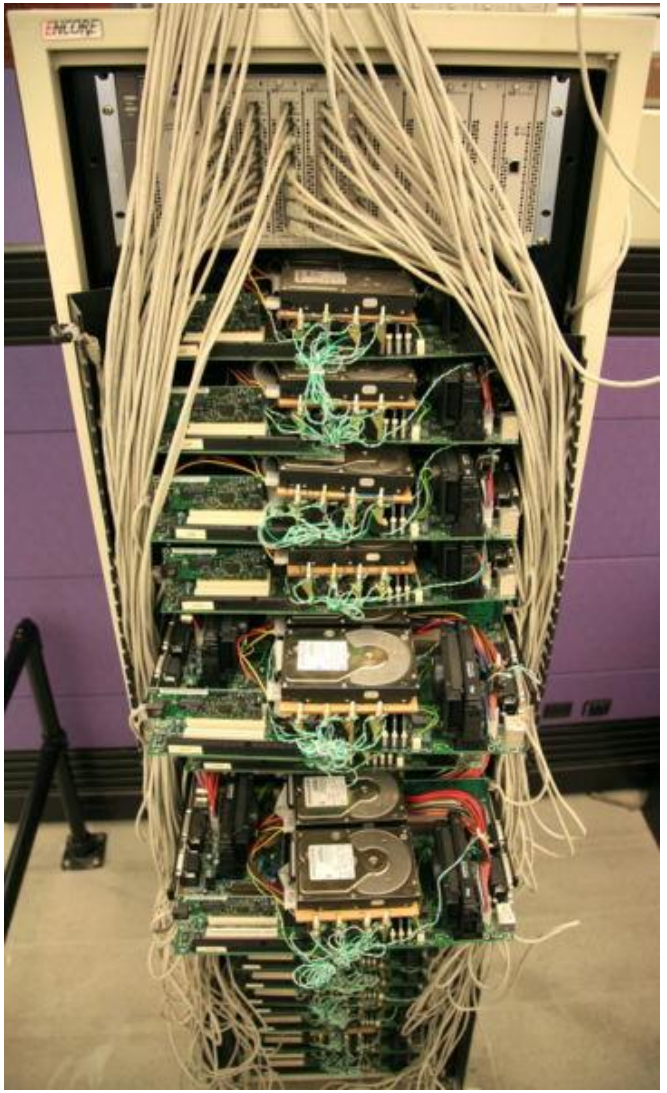
Data Structure Examples:

- Any guesses as to what this is?



Data Structure Examples: (ans)

- Any guesses as to what this is?
- Google's first (production) server



From https://en.wikipedia.org/wiki/History_of_Google

Aims of the ADE Module

Aim: understanding of issues involved in program design; good working knowledge of some common algorithms and data structures

Objectives:

- design data structures and algorithms which express this functionality in way that makes **efficient** use of
  - time
  - memory
  - **(and so indirectly: grid electricity usage, and battery usage)**
- be able to evaluate a given implementation in terms of its **efficiency**

Why care about efficiency?

Beware:

- Recent history is that hardware improved rapidly
  - This compensated for many poor programs
- 3GHz desktop with 2GB RAM on a small "first-year-style" program
  - almost any implementation will work 'instantly'

These might give a false sense that efficiency does not matter.

Why care about efficiency? (batteries!)

But:

- Clock speeds are 'stuck'
  - Not all programs can exploit multi-core, GPU, clouds, but must run on the single core
- Mobile devices are much less powerful
- Also 'inefficient' consumes more energy
  - **poorly written apps will drain the mobile device battery**
  - **2% (?) of world electricity goes in server farms**

Problem Oriented Approach

- The module has ‘maths’ and ‘theory’
- But these are designed to help with solving real problems/tasks
- What kinds of tasks? …

Example task 1

- "What method would you use to look up a word in a dictionary? (Correctly and efficiently)"

Example task 2
(optional offline exercise)

- "You are given a list of numbers. When you reach the end of the list you will come back to the beginning of the list (a circular list).
- Write the most efficient algorithm to find the minimum number in this list.
- Find any given number in the list.
- The numbers in the list are always increasing but you don't know where the circular list begins, e.g.: 38, 40, 55, 89, 6, 13, 20, 23, 36."

Example task 3
(optional offline exercise)

- "Write a function f(a, b) which takes two character string arguments and returns a string containing only the characters found in both strings in the order of a. Write a version which is order N-squared and one which is order N."

Example task 4

- "Describe the algorithm for a depth-first graph traversal."

Example task 5

- "Tree search algorithms. Write BFS and DFS code, explain run time and space requirements. Modify the code to handle trees with weighted edges and loops with BFS and DFS, make the code print out path to goal state."

Example task 6 (optional offline exercise)

- "There is an array A[N] of N numbers. You have to compose an array Output[N] such that Output[i] will be equal to multiplication of all the elements of A[N] except A[i]. For example Output[0] will be multiplication of A[1] to A[N-1] and
Output[1] will be multiplication of A[0] and from A[2] to A[N-1].
Solve it without division operator and in O(n)."

Example task 7

- How long it would take to sort 1 billion numbers?
- Come up with a rough estimate.
- "Billion" is 10^9 (US and modern UK usage)
  In old UK usage it was 10^{12}

- **<u>EXERCISE: Offline – try to estimate it using "back of an envelope" style</u>**

Previous “Example tasks” are irrelevant?

- **If you think the previous were irrelevant then be aware:**
- **They were taken (mostly) from “Google Interview Questions: Software Engineer”**
  - (http://blog.seattleinterviewcoach.com/2009/02/140-google-interview-questions.html#software_engineer but page is no longer active)

Interview Question

- Question: “There is an array A[N] of N numbers. … Solve it … in O(n).”

- What happens if your answer is
  - “what does ‘O(n)’ mean?”
  - ☹

Rough Context within "Programming"

- "Low-level" Programming (Modules in C, Assembly)
  - for, while loops etc
- **ADE: Algorithm Efficiency**
  - **trees, search algorithms, etc**
  - **code: "small but difficult" (each line might be 'hard')**
- Object-oriented design (Java / C++ modules)
  - software reuse, inheritance, etc
- Software systems architecture design (DMS?)
  - designing software components and interactions, etc
  - code: "easy but large" (design is hard, but each line tends to be 'easy')
- Algorithm Correctness – e.g. COMP2065-IFR
  - Proving correctness and/or termination
  - Requires "formal reasoning about programs" – VERY challenging for real-world programs

Rough Contents of the ADE portion

Basics/Theory:

- **Algorithm analysis, "big-Oh" and its family**
- (Revision) Abstract vs. Concrete Data Types
- (Revision?) Stacks, queues, lists, trees.
- Recurrence relations. "Master Theorem".

Sorting Algorithms:

- Bubble-sort, …, mergesort, quicksort, "stability", efficiency

Standard data structures

- Priority queues, Heaps, Hash tables

Search:

- Maps, hashmaps, binary search trees, balanced vs. unbalanced trees

Graphs & Algorithms:

- Minimum Spanning Trees, "dynamic programming", Shortest Paths (Dijkstra's algorithm, and Floyd-Warshall),

Maths needed includes:

- Properties of inequalities:
  - Reasoning about efficiency involves comparisons
- Properties of logarithms and exponents
  - Used to describe efficiency
    - (Generally don't need sin/cos etc.).
- Arithmetic and Geometric Series
- (Function fitting
  - "Regression" a.k.a. "trend lines in Excel")
- Proof by induction
- (Simple) Propositional and predicate logic

(A lot of this should be "revision".)
Tutorials and other material will support these skills.

Maths needed includes: 'comparisons"

- **Properties of inequalities**:

$a < b$ implies $2a < 2b$

$a \leq b$ implies $2a \leq 2b$

$a \leq b$ implies $a + c \leq b + c$

$a < b$ implies $-a > -b$

$a \leq b$ implies $-a \geq -b$

Etc.

Maths needed includes: (logs!)

- **Properties of exponentials**: (examples)

  $a^{(b+c)} = a^b\, a^c$ $\qquad$ $a^0 = 1$

  $a^{bc} = (a^b)^c$ $\qquad$ $a^b / a^c = a^{(b-c)}$

- **Properties of logarithms:**

  $b = a^{\log_a b}$ ("definition of 'log base a' ")

  Offline future exercises:
  derive each of the following

  $b^c = a^{c*\log_a b}$

  $\log_b(xy) = \log_b x + \log_b y$

  $\log_b(x/y) = \log_b x - \log_b y$

  $\log_b x^a = a \log_b x$

  $\log_x a = \log_x b \log_b a$

  Quick self-test: What is $\log_8(2)$?

  (Yes, we do need this!)

Maths: "Geometric and Arithmetic Series"

- **(Offline) Without using a calculator What are**:

"Geometric"

1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 + 256 + 512

"Arithmetic"

1 + 3 + 5 + 7 + 9 + 11 + 13 + 15 + 17 + 19

Summary

- Main points of ADE part:
  - Develop habit of thinking about efficiency
    - E.g. Develop skills (mostly 'big-Oh' and family) to describe efficiency, and reason about efficiency
- But note:
  - This does not mean you always have to write the most efficient program
  - The skills (big-Oh family) will not be a panacea; they do not answer all questions about efficiency (but this does not mean they are irrelevant)

Notes

- Last year there was a module with code COMP2054
  - it was the Autumn portion of COMP2009-ACE – not the same as this module !