Informatics 2:
Introduction to Algorithms and Data Structures

Lecture 1: Overview of Course Content

John Longley

School of Informatics
University of Edinburgh

September 2021

Video lectures

Admin lecture: Practical arrangements. What will happen when?

This lecture: Overview of content. What is the course about?

General advice for viewing this and later lectures . . .

- Each lecture: several videos organized into a playlist.
  (Average 50-60 minutes.)
- Mix of videos from this year and last year.
- Captions: may sometimes correct small errors or omissions.
- Quiz questions within lectures: for your private use only.
- Do post questions or comments on the lectures to Piazza, mentioning lecture and slide number.
  Use fridayqs to submit questions for live Q+A sessions.

What are algorithms and data structures?

Informatics 2 – Introduction to Algorithms and Data Structures

Algorithms are basically *methods* or *recipes* for solving various problems. To write a *program* to solve some problem, we first need to know a suitable *algorithm*.

Data structures are ways of storing or representing data that make it easy to manipulate. Again, to write a program that works with certain data, we first need to decide how this data should be stored and structured.

Tasks calling for algorithms

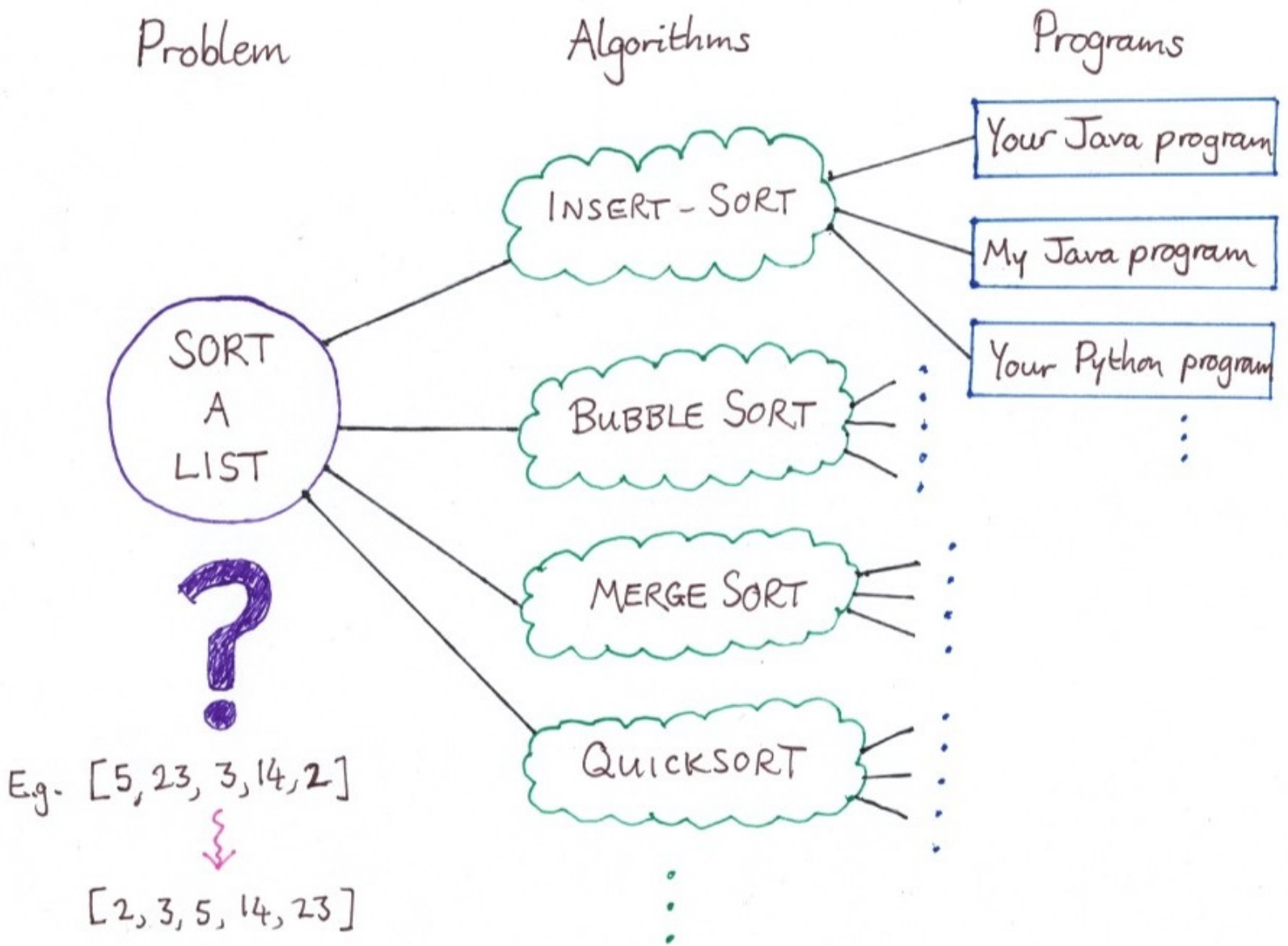
How would you efficiently ...

- Sort 1000000000 names into alphabetical order? (Databases)
- Visit all web pages reachable from a given starting page? (Search engines)
- Find the shortest/fastest/cheapest route from A to B? (SatNav)
- Find the longest substring shared by two (long) strings? (Genetics)
- Tell whether a 100-digit number is prime or not? (Cryptography)

In some cases there's an 'obvious' method.

Often there's a non-obvious method that's much more efficient.

Problems, algorithms, programs



There were algorithms before there were computers

'Algorithms' are so named after Muhammad al-Khwārizmī, a 9th century Persian mathematician who wrote an important book on methods for arithmetic in decimal notation.
(E.g. $+$, $-$, long $\times$, long $/$, $\sqrt{\ }$.)

Even earlier, there was Euclid's *greatest common divisor* algorithm:

$$\begin{aligned}\text{GCD}\ (4851, 840) &= \text{GCD}\ (840, 651) = \text{GCD}\ (651, 189)\\ &= \text{GCD}\ (189, 84) = \text{GCD}\ (84, 21) = 21.\end{aligned}$$

But now that we have computers, algorithms are everywhere!

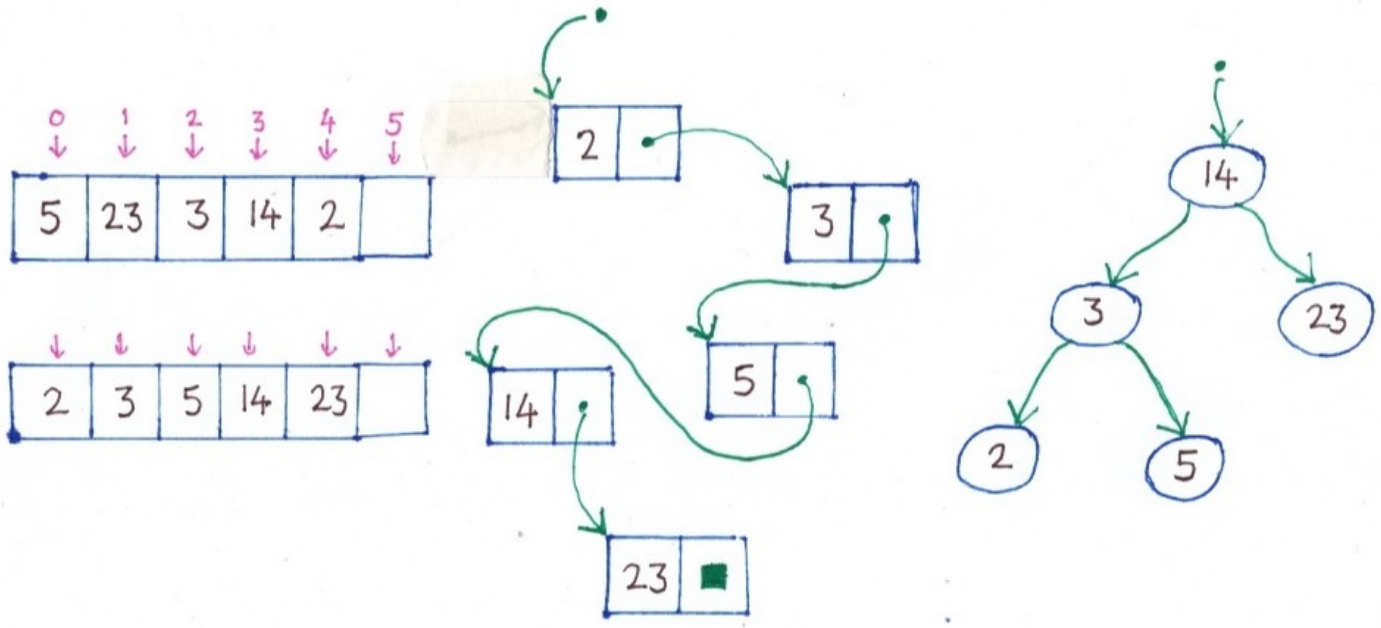
Quiz question

Which of the following is not a true statement about algorithms?

1. Typically, the same algorithm can be implemented in many different programming languages.
2. Some old algorithms are still considered to be very efficient.
3. Some algorithms work OK on small examples, but don't scale up to larger ones.
4. In order to design an *algorithm* for tackling some problem, you first need to write a *program* that solves it.

Data structures

How might you store a set of numbers (e.g. $\{5, 23, 3, 14, 2\}$) in a computer? In an array (sorted or not)? As a linked list? As a tree?



Which of these make it easy . . .

- To test whether a given number (e.g. 11) is in the set?
- To add or remove a member of the set?

Our choice of data structure may depend on which of these operations are most important for us.

What makes computers go faster?

Combination of . . .

- Hardware technology (e.g. processor speeds)
- Parallel processing (doing several things at once)
- Compiler techniques (e.g. optimization of machine code)
- Advances in algorithms / data structures.

Even on 'old' algorithmic problems, new advances are still being made, sometimes leading to dramatic speed-ups on practical tasks.

- 2017: Major breakthrough in *minwise hashing* problem (from 1997) has led to order-of-magnitude improvement in near-duplicate detection, e.g. in search engines (Wang/Wang/Shrivastava/Ryu).
- 2018: Ideas of *adaptive sampling* give exponential speed-up on many optimization problems, e.g. taxi dispatch (Singer/Balkanski).

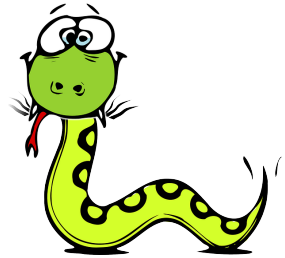
In this sense, algorithms / data structures are a technology.

People like Google$^{\mathrm{TM}}$ certainly care about them, and ask questions about them at internship interviews!

Rough outline of course (order slightly simplified)

- Simple examples of efficient/inefficient algorithms.
- How can we measure how 'good' an algorithm is? Approach via asymptotic analysis.
- Sorting algorithms: InsertSort, MergeSort, QuickSort, ...
- Basic data structures: Ways of implementing lists, stacks, queues, sets, dictionaries, ...
- Algorithms on graphs: depth-first and breadth-first search, topological sorting, shortest paths.
- Dynamic programming: A way to avoid repeating work. Applications, e.g. seam carving for images.
- Algorithms/data structures for language processing (e.g. of Java or Python source code). Grammars, syntax, parsing.
- What are the limits of algorithms and computation? Glance at complexity theory (intractable problems, P vs. NP) and computability theory (unsolvable problems, Turing machines, halting problem).

Programming thread: Python



The course will emphasize the relevance of algorithms/data structures to practical programming.

We'll be using Python as our programming language, and one of our goals is to teach you this. (Used in many later-year courses).

We'll issue Python lab sheets, for you to work through at the computer at your own speed. Customized to this course: tie in closely with lecture material. Contain exercises for your own use (not submitted or marked).

Online support available to help you with these sheets.

Two of the three courseworks will involve Python programming.

Other resources

Learn page: reached from `www.learn.ed.ac.uk`.
Everything will be made available here: lecture videos and slides, tutorial sheets, lab sheets, courseworks, link to discussion forum.

Textbooks:

- Cormen, Leiserson, Rivest and Stein, *Introduction to Algorithms* (3rd edition). The classic. Also useful for Year 3 ADS; wealth of more advanced material.
- Goodrich, Goldwasser and Tamassia, *Data Structures and Algorithms in Python*. Programmer-oriented: starts with intro to Python! Closely matches first 2/3 of course.
- Sedgewick and Wayne, *Algorithms* (2016 edition). Java-based.
- Anything else you find helpful? Let us know!

Reading for this lecture: CLRS Chapter 1.

ENJOY THE COURSE!