

Computational Thinking and Algorithms 159.171 An Introduction

Amjed Tahir

a.tahir@massey.ac.nz

Previous contributors: Catherine McCartin & Giovanni Moretti

159.172: Computational Thinking and **Algorithms**

Focus: on Algorithms and their properties

Assessment

- 40% from 2 x Assignments
- 20% from Lab exercises
- 40% Final exam with minimum 40% to pass

Intensive: it's a short course – around 8 weeks

Programming competence will be assumed

understanding what your programs do is essential

+ Weekly Tutorial

Topics in the paper

- 1. Python Basics
- 2. Python Lists
- 3. Recursive functions
- 4. Abstract Data Types& Python Classes
- 5. Linked Lists
- 6. Searching and Sorting Algorithm Analysis

- 7. Python Dictionaries
- 8. Trees and Search Trees
- 9. Heaps
- 10. Hashing

What is Computer Science?

is it about the study of computers?

"Computer Science is no more about computers than astronomy is about telescopes"

is it learning to write programs?

Programming is an important tool...

"Science is not about tools, it is about how we use them, and what we find out when we do"

Muḥammad ibn Mūsā al-Khwārizmī

Muḥammad ibn Mūsā al-Khwārizmī

(Arabic: أمحمد بن موسى الخوارزمي; c. 780 - c. 850)

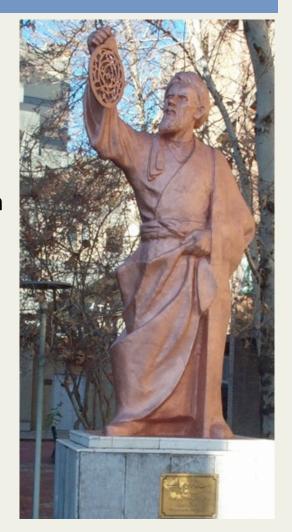
formerly Latinized as Algoritmi,

a Persian[1] mathematician, astronomer, and geographer during the Abbasid Caliphate, a scholar in the House of Wisdom in Baghdad (Iraq).

[1] Persia – now known as Iran

Photo: Statue of al-Khwarizmi at Amirkabir University of Technology in Tehran, Iran. Photo: M. Tomczak; Creative Commons license. http://www.es.flinders.edu.au/~mattom/science+society/lectures/ illustrations

from https://en.wikipedia.org/wiki/Muhammad_ibn_Musa_al-Khwarizmi



What is an algorithm?

"An algorithm is an ordered set of unambiguous, executable steps that defines a terminating process."

If we can specify an algorithm to solve a problem, we can automate its solution.

Can we always do this?

There are problems that are unsolvable,

i.e. problems for which no algorithmic solution can exist.

There are problems that are intractable,

i.e. problems for which any algorithmic solution will take too long to execute.

There are problems we don't yet know how to solve algorithmically.

An ordered set...

- the order in which steps are executed is clearly defined
- in the simplest case clear ordering to the operations, need to know which operation to do first, and at each step, which operation to perform next
- parallel algorithms? distributed algorithms?

An ordered set ...

- the order in which steps are executed is clearly defined
- simplest case clear ordering to the operations, need to know which operation to do first, at each step, which operation to perform next
- parallel algorithms? distributed algorithms?

... of unambiguous, executable steps ...

- can determine uniquely and completely what to do
- executable operation one that can be completed successfully using some computational process (effectively computable)

An ordered set ...

- the order in which steps are executed is clearly defined
- simplest case clear ordering to the operations, need to know which operation to do first, at each step, which operation to perform next
- parallel algorithms? distributed algorithms?

... of unambiguous, executable steps ...

- unambiguous operation can determine uniquely and completely what to do
- executable operation one that can be completed successfully using some computational process (effectively computable)

... terminating process

- observable result(s) must be produced after a finite number of steps
- non-terminating computational processes? maintaining, monitoring?

Algorithms: another expression

An algorithm is an effective method that can be expressed within a **finite amount of space and time** and in a **well-defined formal language** for calculating a function.

Starting from an **initial state** and **initial input (perhaps empty)**, the instructions describe a computation that, when executed, **proceeds through a finite number of well-defined successive states**, eventually **producing** "**output**" and terminating at a final ending state.

The transition from one state to the next is **not necessarily deterministic**; some algorithms ... incorporate random input.

from https://en.wikipedia.org/wiki/Algorithm

Solving problems algorithmically

Need to find good ways to model real-world problems (data models)

Use abstraction, concentrate on the important things, leave out unnecessary details.

Programming language constructs to represent data models (which data structures)

Techniques for manipulating the data models, or associated data structures, to obtain solutions to problems (programs)

Different programming languages

- have different primitive (built-in) data types
- have different built-in operations on those primitive data types

Studying the behaviour of algorithms

What makes a good algorithm? Correctness. Efficiency. Simplicity.

How do we decide if an algorithm is correct? How do we determine that an algorithm is efficient?

Implementing algorithms

Need different representations for different purposes. Levels of abstraction between human and hardware.

Human = high-level encoding, pseudo-code (structured English)Hardware = low-level encoding, machine language (binary)

Creation of Software

Algorithms are usually small parts of a complete program - programs are usually components of a larger system.

Development of a software system has several phases

- Problem definition and specification
- Design
- Implementation
- Integration and system testing
- Installation and field testing
- Maintenance

A simple example:

```
def tallest(classlist):
    (tallest, name) = classlist[0]
    for (height, person) in classlist:
        if height > tallest:
            tallest = height
            name = person
    return(tallest, name)
```

How much "work" does this algorithm do for a classlist of size n?