Introduction

Computational Thinking with Algorithms

What is an algorithm?

- A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer (Oxford English Dictionary)
- Algorithms can be thought of like a “recipe” or set of instructions to be followed to achieve the desired outcome

Algorithms & programming

A typical programming task can be divided into two phases:

- Problem solving phase
  - produce an ordered sequence of steps to solve the problem
  - this sequence of steps is called an algorithm
- Implementation phase
  - implement the program in some programming language

- This module addresses the problem solving aspect

The word “Algorithm”

- Al-Khwarizmi (c. 780 to 850) was a Persian mathematician
- His name was translated into Latin as “Algoritmi”
- English word “algorithm” first appeared in the 17th century

Simple example

- Algorithm to make a cup of tea?

Tea making algorithm

1. Fill kettle
2. Boil kettle
3. Put tea bag in cup
4. Pour water
5. Add milk

Why are algorithms important?

- Just as algorithms that you follow affect your daily life, so do algorithms that run on computational devices
- E.g.
  - GPS uses a “shortest-path” algorithm
  - Secure online shopping and banking requires encryption algorithms
  - Delivery services use proprietary algorithms to assign parcels to trucks, and to determine the order in which drivers should deliver parcels
  - Email services use classification algorithms to label suspect messages as spam
  - Can you think of other examples?

Computer algorithms

- While humans can use imprecise algorithms to accomplish tasks, the same cannot be said for computers
- E.g. when driving to your workplace, your algorithm may contain a rule like: “If traffic is heavy, take an alternate route”
- We know intuitively what is meant by “heavy traffic”, but a computer must be explicitly programmed to deal with this scenario
- Therefore, algorithms for computational devices must be described precisely enough that a computer can run them successfully, achieving the desired outcome

A brief history of algorithms

- c. 300 BC - Euclid’s algorithm
- c. 820 - Al-Khawarizmi writes about algorithms for solving linear equations and quadratic equations
- 1945 – Merge sort developed by John von Neumann
- 1959 – Dijkstra’s algorithm
- 1973 – RSA encryption algorithm
- 1998 – PageRank algorithm published by Larry Page (Google)
- 2017 – Deepmind’s AlphaGo beat Ke Jie, the world No.1 ranked Go player

Will discuss some of these algorithms in more detail later

Machine Learning

- “Machine Learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed” (Arthur Samuel, 1959)
- Machine Learning is a process whereby a computer program learns from experience to improve its performance at a specified task (Mitchell, 1997).

What sort of tasks?

- Targeted advertising
- Movie recommendations
- Product recommendations
- Identifying suspicious credit card transactions
- Detecting cancer on scans
- Email spam filters
- High-frequency trading
- Other examples???

Design criteria for algorithms

- A well-designed algorithm should:
  - Produce a “correct” solution for a given input
  - Use computational resources efficiently

Correctness

- What criteria should a correct solution satisfy?
- Easy to define correctness in many (simple) real world problems, e.g.
  - Is a book by author X present on this bookshelf (boolean true/false answer)
  - What is the shortest route between two points (one unique solution)
  - Sort pupils in a class by increasing height (one correct answer)
- More difficult to define correctness in other real world problems, e.g.
  - Which route will take the least time to traverse (answer depends on factors other than geometry, i.e. traffic conditions)
  - Handwriting recognition (is that character a number 5 or a letter S)?

Correctness

- In this module we will focus mainly on problems which have knowable solutions
- Incorrect solutions are acceptable occasionally, so long as the expected probability of returning an incorrect solution can be quantified/controlled
- For many practical applications, algorithms have been developed which have a guaranteed margin of error, rather than a guarantee of complete correctness
- E.g. testing if a number is prime (the RSA cryptosystem relies on algorithms which determine whether a number is prime)

Efficiency

- If a routing algorithm could produce a route that avoided all congestion, but took an hour to do so, would it be useful?
- An algorithm that produces a correct solution but takes a long time to run may be useless in practice. This is the reason that more efficient algorithms that compute approximate solutions with guaranteed margins of error are often adopted in practice

Efficiency

- Factors which may be taken into consideration when evaluating the efficiency of an algorithm:
    - Time (real world/ CPU time)
    - Memory (RAM) usage
    - Storage (Hard disk) usage
    - Number of read/write operations (especially on disk)
    - Network usage
    - Random bit requirements (some algorithms require a source of random numbers)
- We will focus mostly on time efficiency in this module
- E.g. “how does the time taken for an algorithm to execute vary with the size of the input?”