

# Introduction to Algorithms

Semester 2, 2020

Kris Ehinger

# Why study algorithms?

- Know how to choose best algorithm for a problem
- Analyse efficiency
- Understand why it works -- and when it won't

# What is an algorithm?

- A set of steps to accomplish a task:
  - A series of steps to compute the product of two integers
  - A series of steps to prepare data for entry in a database
  - A recipe for cooking a meal
  - A procedure for doing laundry
  - A series of steps to drive a car to a destination
  - A procedure for applying to graduate school
  - Etc.

# What is a *computer* algorithm?

- An algorithm with the following properties:
  - Precisely defined steps/conditions
  - Defined input
  - Defined output
  - Correct
    - Exactly correct?
    - Correct to within some  $\epsilon$
  - Terminates within a reasonable period of time

# Algorithms

- Al-Khwarizmi (Baghdad, 9<sup>th</sup> century)
- “Founder of algebra”
- Textbook on arithmetic:
  - Arabic numerals
  - Decimal positional number system
  - Addition and multiplication
- Textbook on algebra:
  - Extract square roots
  - Solve linear and quadratic equations
- Contributions in astronomy, geography, trigonometry



# Example: Addition

- What algorithm do you use to add two numbers?

$$\begin{array}{r} 17 \\ + 25 \\ \hline 42 \end{array}$$

↑ ↑ ↑

# Example: Addition

- How would a computer add two numbers?
- Simple option: ripple carry adder
- Recall that computers represent numbers in binary:

$$\begin{array}{r} 3 \\ + 6 \\ \hline 9 \end{array} \quad \longrightarrow \quad \begin{array}{r} 0011 \\ + 0110 \\ \hline 1001 \end{array}$$

# Example: Addition

[https://www.youtube.com/watch?v=OpLU\\_bhu2w](https://www.youtube.com/watch?v=OpLU_bhu2w)



# Efficiency

- How long does it take to add two numbers?
- Depends on the speed of the “processor”:
  - Computer >> human >> dominos
- Depends on how large the numbers are
  - Number of columns (decimal places or bits)
  - Bigger numbers also require more space (for all three “processors”)
- **Efficiency:** how do the time/space requirements of the algorithm scale with the input?

# Correctness

- **Correct:** algorithm returns the desired output every time, for every possible (legal) input
- Proving that an algorithm is correct can be difficult
- Proving an algorithm incorrect is often easy – just come up with one case where it won't work!

# Classifying algorithms

- Algorithms can be classified by task:
  - Numeric
  - Sorting
  - Searching
  - Routing
  - Scheduling
  - Etc.

# Classifying algorithms

- Algorithms can be classified by approach:
  - Brute force
  - Divide and conquer
  - Decrease and conquer
  - Greedy
  - Etc.

# Classifying algorithms

- Algorithms can be classified by solution type:
  - Exact
  - Approximate
  - Heuristic

# Summary

- What is an algorithm?
- How can we classify algorithms?
- Correctness and efficiency