COMP108
Data Structures and Algorithms

Module Introduction

Professor Prudence Wong

pwong@liverpool.ac.uk

You can call me Prudence or formally Professor Wong

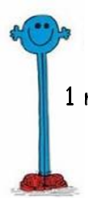
2022-23

Outline

- Puzzle
- Basic information
- Why COMP108?
- Module Aims
- Learning & Teaching Activities
- Assessments
- How to get help?
- More motivation to study Algorithms

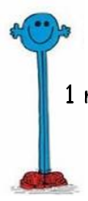
Crossing Bridge @ Night

- Each time, 2 persons share a torch
- They walk @ speed of slower person



Crossing Bridge @ Night

- Each time, 2 persons share a torch
- They walk @ speed of slower person



Can we do it in 17 mins?

Basic Module information

Professor Prudence Wong

- Contact: pwong@liverpool.ac.uk
- Office hours: Rm 3.18 Ashton Building, Thursdays 13:00-14:00

Demonstrators

- Mr Alexander Bird, Mr Richard Hogg, Mr Saad Qayyum, Mr Tuvey Oscar, Mr Khilan Santoki, Mr Benjamin Smith, Miss Yanhua Xu

References

T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. *Introduction to Algorithms.* The MIT Press.

M. T. Goodrich, R. Tamassia, M. H. Goldwasser. *Data Structures and Algorithms in Java.* Wiley.

Why COMP108?

Algorithm design and appropriate use of data structures

- foundation for efficient and effective programs
- want to process data quickly?
- want to process massive amount of data?

Pre-requisite for: COMP202, COMP218

- everybody takes COMP202

``Year 1 modules do not count towards honour classification . . .''

- Career Services: Employers DO consider year 1 module results
- The only results you can show if you want to apply for summer internships, year in industry placements, study abroad, etc.

Module Aims

To introduce the notation, terminology, and techniques underpinning the study of algorithms

To introduce basic data structures and associated algorithms

To introduce standard algorithmic design paradigms and efficient use of data structures employed in the development of efficient algorithmic solutions

Module Aims

To introduce the notation, terminology, and techniques underpinning the study of algorithms

What do we mean by ``efficient'' algorithms & data structures?

To introduce basic data structures and associated algorithms

To introduce standard algorithmic design paradigms and efficient use of data structures employed in the development of efficient algorithmic solutions

Module Aims

To introduce the notation, terminology, and techniques underpinning the study of algorithms

What do we mean by ``efficient'' algorithms & data structures?

To introduce basic data structures and associated algorithms

What are data structures, how to manipulate data efficiently?

To introduce standard algorithmic design paradigms and efficient use of data structures employed in the development of efficient algorithmic solutions

Module Aims

To introduce the notation, terminology, and techniques underpinning the study of algorithms

What do we mean by ``efficient'' algorithms & data structures?

To introduce basic data structures and associated algorithms

What are data structures, how to manipulate data efficiently?

To introduce standard algorithmic design paradigms and efficient use of data structures employed in the development of efficient algorithmic solutions

How to solve problems efficiently with algorithms & data structures?

Learning & Teaching Activities

Lectures (some live lectures may be replaced by pre-recorded videos)

- Mondays: 4:00pm-5:00pm
- Tuesdays: 9:00am-~~10:00pm~~ 10:00am
- Thursdays: 12:00pm-1:00pm

Learning & Teaching Activities

Lectures (some live lectures may be replaced by pre-recorded videos)

- Mondays: 4:00pm-5:00pm
- Tuesdays: 9:00am-10:00~~pm~~
- Thursdays: 12:00pm-1:00pm

Teaching materials (slides & lecture capture) on Canvas

- *When are they available?* Weekly, available on Mondays at 9:00am

Learning & Teaching Activities

Lectures (some live lectures may be replaced by pre-recorded videos)

- Mondays: 4:00pm-5:00pm
- Tuesdays: 9:00am-10:00pm
- Thursdays: 12:00pm-1:00pm

Teaching materials (slides & lecture capture) on Canvas

- *When are they available?* Weekly, available on Mondays at 9:00am

Lab & Tutorials (Weekly Submission)

- Week 1: no lab/tutorial --- familiarize with programming environment this week
- Weeks 2-3: 1-hour **tutorial**
- Weeks 4-9: 1-hour **lab session**
- Weeks 10-12: 1-hour **tutorial**

Learning & Teaching Activities

Lectures (some live lectures may be replaced by pre-recorded videos)

- Mondays: 4:00pm-5:00pm
- Tuesdays: 9:00am-10:00pm
- Thursdays: 12:00pm-1:00pm

Teaching materials (slides & lecture capture) on Canvas

- *When are they available?* Weekly, available on Mondays at 9:00am

Lab & Tutorials (Weekly Submission)

- Week 1: no lab/tutorial --- familiarize with programming environment this week
- Weeks 2-3: 1-hour **tutorial**
- Weeks 4-9: 1-hour **lab session**
- Weeks 10-12: 1-hour **tutorial**

Weekly quiz on Canvas (for revision only)

Learning & Teaching Activities

Lectures (some live lectures may be replaced by pre-recorded videos)

- Mondays: 4:00pm-5:00pm
- Tuesdays: 9:00am-10:00pm
- Thursdays: 12:00pm-1:00pm

Teaching materials (slides & lecture capture) on Canvas

- *When are they available?* Weekly, available on Mondays at 9:00am

Lab & Tutorials (Weekly Submission)

- Week 1: no lab/tutorial --- familiarize with programming environment this week
- Weeks 2-3: 1-hour **tutorial**
- Weeks 4-9: 1-hour **lab session**
- Weeks 10-12: 1-hour **tutorial**

Weekly quiz on Canvas (for revision only)

Assessments

- Examination 60%, Class Test 15%, Programming assignment 15%, Weekly submission 10%

Suggested Structure for your week

Day	Activities
Monday	Attend Lecture #1 at 4pm Study associated materials
Tuesday	Attend Lecture #2 at 9am Study associated materials
Wednesday/Thursday/Friday	Attend lab/tutorial working on the exercises
Thursday	Attend Lecture #3 at 12pm Study associated materials Take the (non-assessed) weekly revision quiz
Friday	Submit weekly lab/tutorial exercises

Continuous Assessments

If you miss all continuous assessments, you will need at least 67% in the exam to pass the module!

Continuous Assessments

If you miss all continuous assessments, you will need at least 67% in the exam to pass the module!

Continuous Assessments

- Weekly submission --- lab/tutorial exercises (10%) --- due Fridays 5:00pm
- Class Test (15%) --- Week 5: **Thursday 2nd March 2023, 12-1pm**
- Assignment (15%) --- due Week 9: **Friday 21st April 2023, 5pm**
  - Standard late penalty policy: 5 marks deducted every (calendar) day, maximum of 5 days late

Continuous Assessments

If you miss all continuous assessments, you will need at least 67% in the exam to pass the module!

Continuous Assessments

- Weekly submission --- lab/tutorial exercises (10%) --- due Fridays 5:00pm
- Class Test (15%) --- Week 5: **Thursday 2nd March 2023, 12-1pm**
- Assignment (15%) --- due Week 9: **Friday 21st April 2023, 5pm**
  - Standard late penalty policy: 5 marks deducted every (calendar) day, maximum of 5 days late

Programming

- We will implement (some of) the algorithms we learnt in this module
- Java will be used (see COMP122 for preparation)
- Key point is about how the algorithms work, only allowed to use basic data structures, most built-in methods not allowed.

How to get help?

If you get stuck, have a question, or want to learn more

- Ask questions during lectures
- Talk to me/demonstrators during labs/tutorials
- Contact me during office hours
- Email me your questions (please include ``COMP108'' in the email subject)
- Check solutions / examples on Canvas
- Read references

Discussion board on Canvas

Exemption from Late Penalty (ELP)

- Coursework affected: contact me and submit ELP, link on CS-UG-PGT-202223
  Canvas course: `https://liverpool.instructure.com/courses/62776/pages/assessments-information`

Extenuating circumstances (EC)

- Other EC: contact me and submit EC claim on
  `https://exc.liverpool.ac.uk/`

Plagiarism/Collusion

What is it?

- University Code of Practice on Assessment Appendix L Academic Integrity Policy
- *Plagiarism: when a student misrepresents, as his/her own work, work in the public domain, written or otherwise, of any other person (including another student) or of any institution.*
- *Collusion: active cooperation of two or more students to produce work, including knowingly allow academic work to be presented by another student as if it's his/hers, and providing work to another student.*

Plagiarism/Collusion

What is it?

- University Code of Practice on Assessment Appendix L Academic Integrity Policy
- *Plagiarism: when a student misrepresents, as his/her own work, work in the public domain, written or otherwise, of any other person (including another student) or of any institution.*
- *Collusion: active cooperation of two or more students to produce work, including knowingly allow academic work to be presented by another student as if it's his/hers, and providing work to another student.*

Related procedures

- Declaration is required when students submitting work
- Submission will be checked automatically
- Cases of suspected academic misconduct will be reported

Plagiarism/Collusion

What is it?

- University Code of Practice on Assessment Appendix L Academic Integrity Policy
- *Plagiarism: when a student misrepresents, as his/her own work, work in the public domain, written or otherwise, of any other person (including another student) or of any institution.*
- *Collusion: active cooperation of two or more students to produce work, including knowingly allow academic work to be presented by another student as if it's his/hers, and providing work to another student.*

Related procedures

- Declaration is required when students submitting work
- Submission will be checked automatically
- Cases of suspected academic misconduct will be reported

Penalties

- Range from mark deduction to suspension/termination of studies
- Last year, 11 COMP108 students had assignments awarded a mark of 0
- Don't take chances!

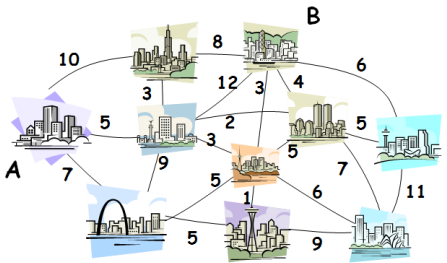
More motivation to study algorithms

Why do we study algorithms?

A valid solution may not be an optimal solution

Given a map of n cities & traveling cost between them.

What is the cheapest way to go from city A to city B?



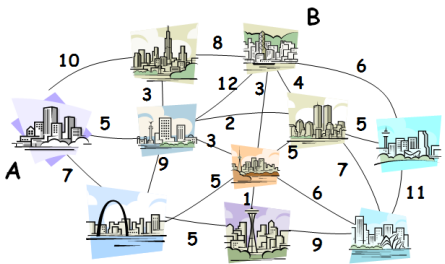
Find any path from A to B

- Not necessarily the cheapest
- How to find the cheapest?

Why do we study algorithms?

The obvious solution to a problem may not be efficient

Given a map of n cities & traveling cost between them.
What is the cheapest way to go from city A to city B?



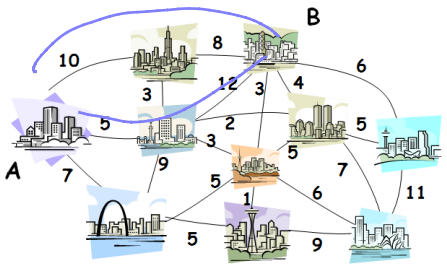
Simple brute-force solution

- Compute the cost of each path from A to B
- Choose the cheapest one

Why do we study algorithms?

The obvious solution to a problem may not be efficient

Given a map of n cities & traveling cost between them.
What is the cheapest way to go from city A to city B?



Simple brute-force solution

- Compute the cost of each path from A to B
- Choose the cheapest one

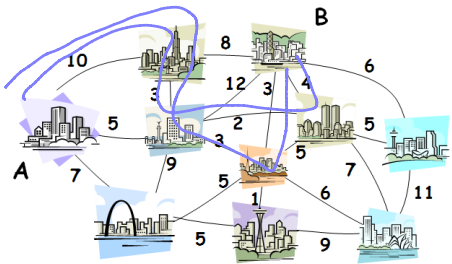
How many paths involving 1 intermediate city?

- 2

Why do we study algorithms?

The obvious solution to a problem may not be efficient

Given a map of n cities & traveling cost between them.
What is the cheapest way to go from city A to city B?



Simple brute-force solution

- Compute the cost of each path from A to B
- Choose the cheapest one

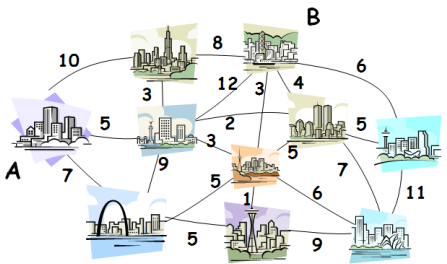
How many paths involving 3 intermediate city?

- 2 + 3 + 6 = 11

Why do we study algorithms?

The obvious solution to a problem may not be efficient

Given a map of n cities & traveling cost between them.
What is the cheapest way to go from city A to city B?



Simple brute-force solution

- Compute the cost of each path from A to B
- Choose the cheapest one

How many paths involving 5 intermediate city?

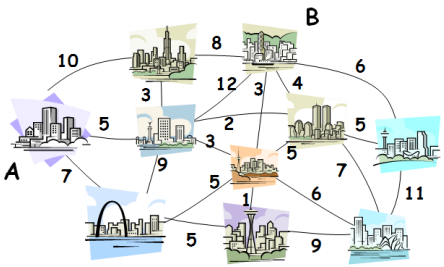
- TOO MANY!

Why do we study algorithms?

The obvious solution to a problem may not be efficient

Given a map of n cities & traveling cost between them.

What is the cheapest way to go from city A to city B?



Simple brute-force solution

- Compute the cost of each path from A to B
- Choose the cheapest one

When *n* is large

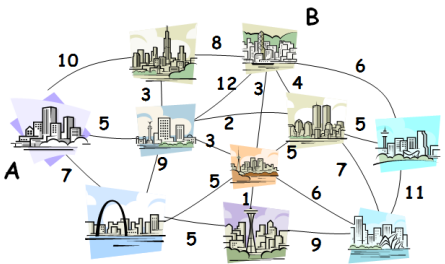
- too much time to check all paths
- We need more sophisticated solutions

Why do we study algorithms?

The obvious solution to a problem may not be efficient

Given a map of n cities & traveling cost between them.

What is the cheapest way to go from city A to city B?

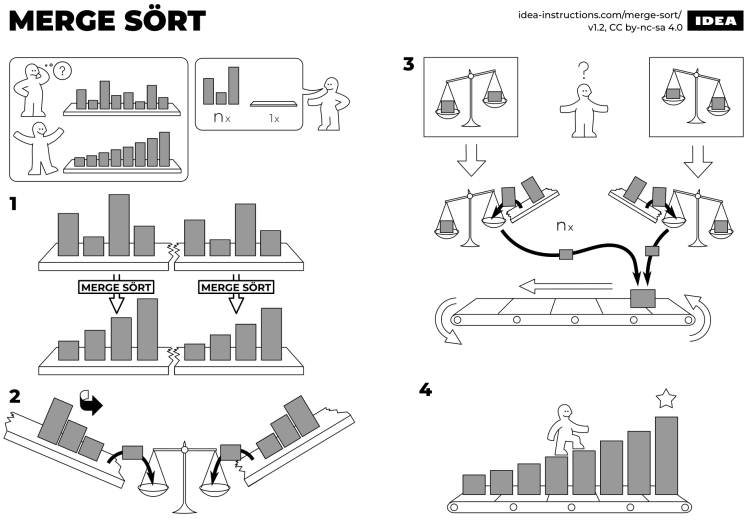


Simple brute-force solution

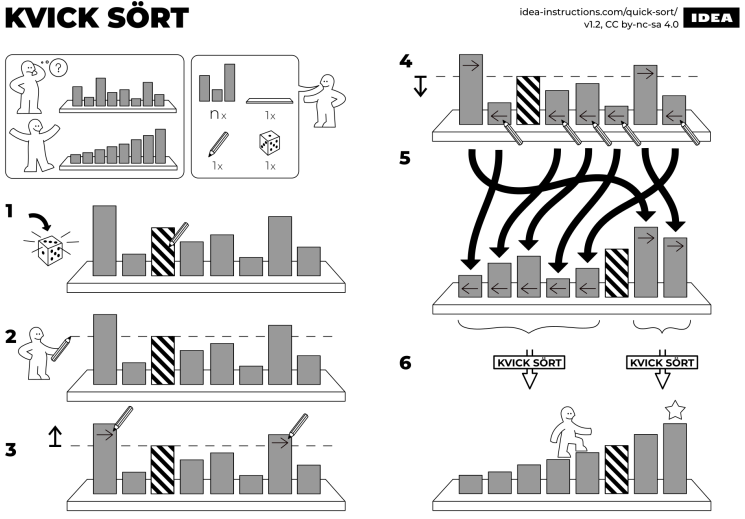
- Compute the cost of each path from A to B
- Choose the cheapest one
- Too much time for large *n*

There is an algorithm, called Dijkstra's algorithm, that can compute this shortest path efficiently.

Algorithm is fun



Algorithm is fun



Summary: Module information

This week: Familiarize yourself with programming environment
(in the lab or on your computer) .

We will start tutorials next week.

This week's topic: Understanding and writing pseudo code

For note taking