CSC 225 - Summer 2019

Course Introduction

Bill Bird

Department of Computer Science
University of Victoria

May 8, 2019

University
of Victoria

Flood Fill (1)

University
of Victoria

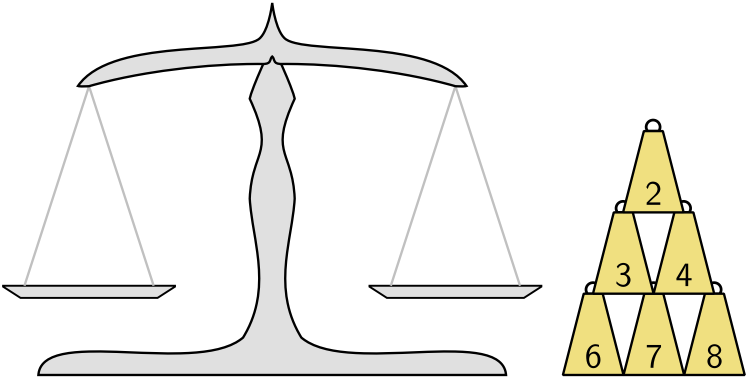
Problem: Given an image and a point somewhere in the image...

Flood Fill (2)

University
of Victoria

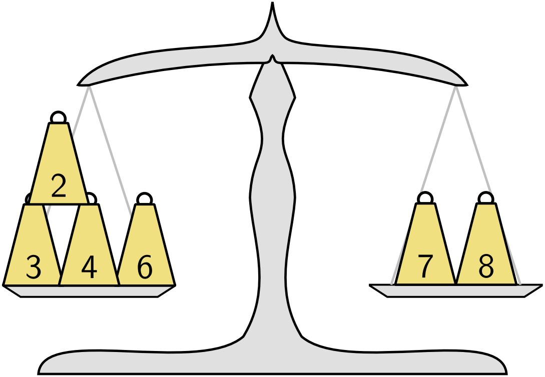
...fill the entire region containing the point with a solid colour.

Balancing a Scale (1)



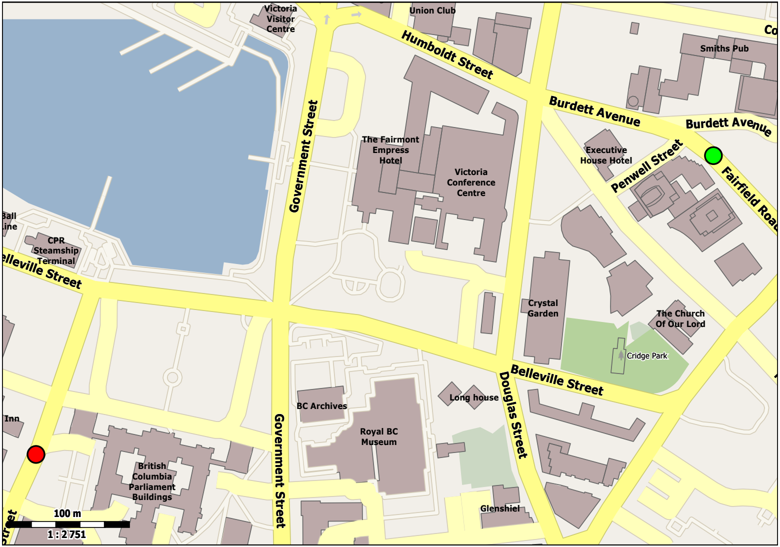
Problem: Given a scale and a collection of weights...

Balancing a Scale (2)



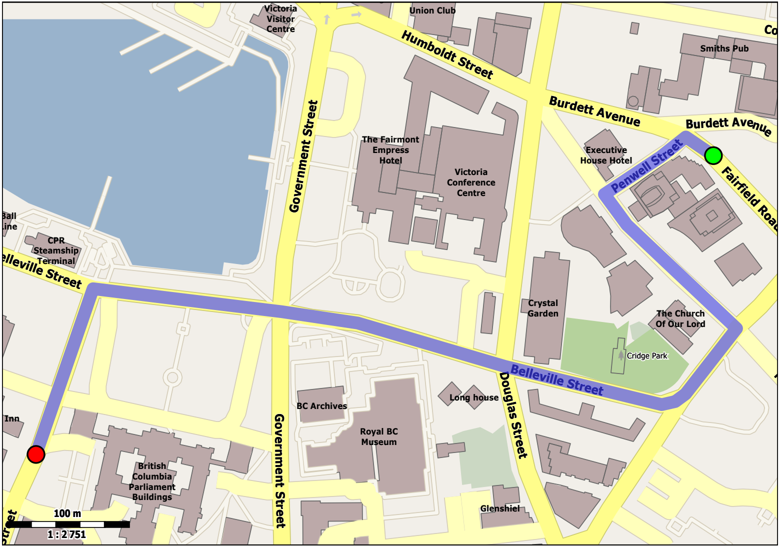
...place all of the weights on the scale such that the scale remains balanced.

Path Finding (1)



Problem: Given a road map, a start point (green) and an end point (red)...

Path Finding (2)



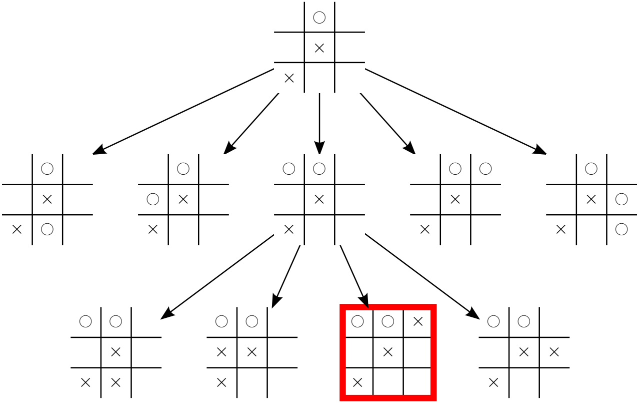
...find a path from the start point to the end point.

Tic-Tac-Toe (1)

	○	
	×	
×		

Problem: Given a partially-played tic-tac-toe board (with O to move)...

Tic-Tac-Toe (2)



…compute the best move for O to make.

The 16-Puzzle (1)

2	15	4	8
13	3	14	11
6		1	12
7	9	5	10

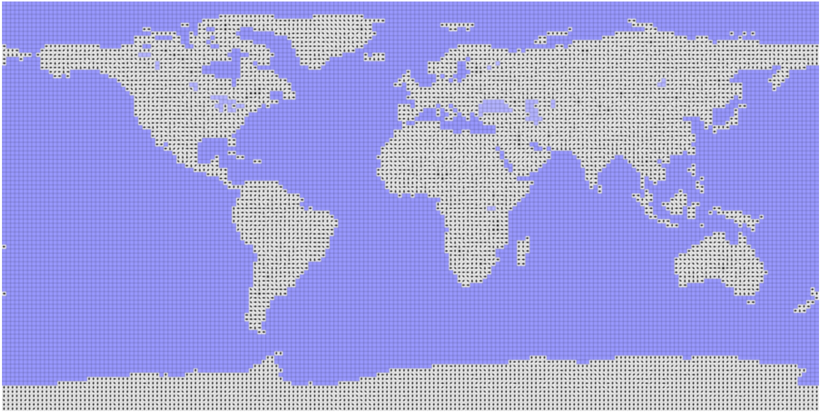
Problem: Given a scrambled 16-Puzzle, where tiles can be moved by sliding them horizontally or vertically into an empty square...

The 16-Puzzle (2)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

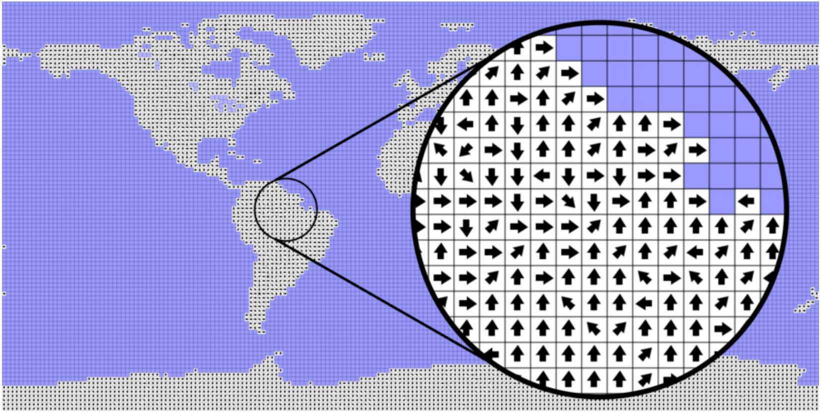
...solve the puzzle by finding a sequence of moves that result in the configuration above.

River Routing (1)



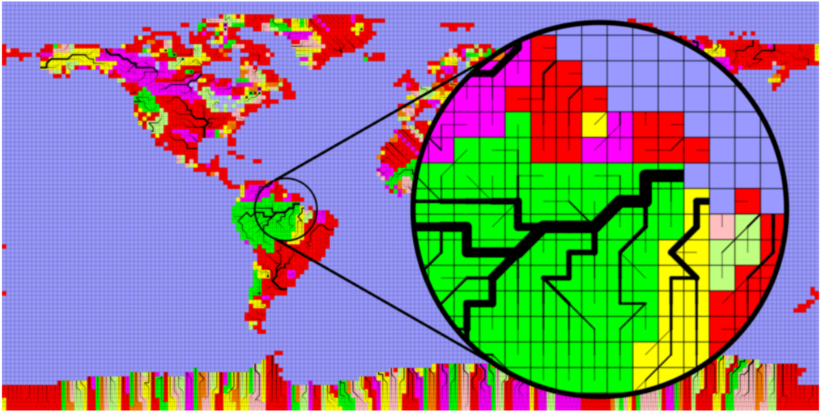
Problem: Given a regular grid covering the surface of the earth, in which every land cell contains the direction of water flow in the corresponding region...

River Routing (2)



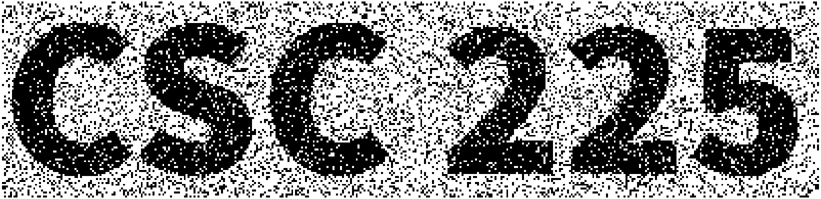
Problem: Given a regular grid covering the surface of the earth, in which every land cell contains the direction of water flow in the corresponding region...

River Routing (3)



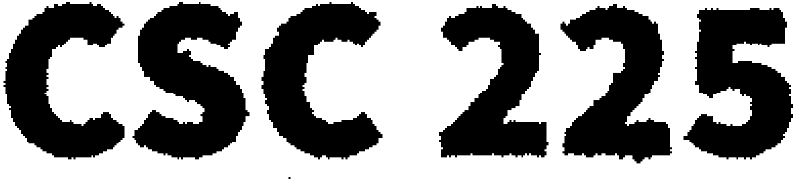
...create a map of the world's rivers.

OCR Noise Reduction (1)



Problem: Given a black and white image of text with noise corruption (for example, resulting from a low quality scan)...

OCR Noise Reduction (2)



CSC 225

...remove noise and identify clusters in the image which may correspond to characters.

CSC225 Overview

Mathematical Foundations

Proof techniques
Asymptotic notation
Mathematical modeling
Recurrence relations
Graph theory

Algorithms

Algorithm design and analysis
Efficiency and complexity
Sorting
Searching
Graph algorithms
Optimization

Data Structures

Stacks, Queues, Lists, Arrays
Trees
Priority Queues, Heaps
Hash Tables
Graphs

Course Prerequisites

CSC 115 or CSC 116

- Intermediate programming with an object-oriented language
- Fundamental data structures (Lists, Arrays, Stacks, Queues, Binary Search Trees, Heaps)
- Basic algorithm design and problem solving

MATH 122

- Logic and Set Theory
- Proof techniques (especially **Induction**)
- Discrete structures (trees, graphs, etc.)

If you did not take the UVic versions of the above courses, some parts of this course might be more challenging (but not impossible). You are encouraged to talk to the instructor as soon as possible to determine what material you might be missing.

Advising Information

Undergraduate Advisor

- Irene Statham (`cscadvisor@uvic.ca`)
- Office: ECS 512
- Undergraduate Advising Hours
  - Monday/Wednesday/Friday 9:30am - 11:30am
  - Tuesday/Thursday 12:30pm - 2:30pm

Administrative Announcements

- If you are taking this course for the third (or greater) time, you must request, **in writing**, permission from the Chair of the Department and the Dean of the Faculty.
- If you have not met **all** prerequisites for this course, you must receive department permission to stay in this class. If you do not receive permission, you will be **automatically dropped from the course** and a prerequisite drop will be recorded on your transcript.
- In both of the above cases, you should visit the undergraduate advisor for more information.

Instructor Information

Lectures

- Bill Bird (`bbird@uvic.ca`)
- Lectures: TWF 9:30 - 10:20am ECS 125
- Office: ECS 324
  (Office hours are in ECS 255)

Labs

- Lily Bryant (`lbryant@uvic.ca`)
  Quinton Yong (`quintonyong@uvic.ca`)
- Lab enrollment is mandatory - Register Now

Office Hours

- Bill (ECS 255): Tuesday, Wednesday, Thursday 2:30 - 4:00pm

Bill's Actual Student Evaluations

"He's really bad at erasing a chalkboard."	- CSC 225 Student (Summer 2015)
"Bill Bird is fucking hilarious and once tried to time travel in class."	- CSC 111 Student (Fall 2015)
"Talk more about FORTRAN."	- CSC 111 Student (Fall 2015)
"Bill has quite long hair, but always wears it the same way. Sometimes I wish he would braid it or something."	- CSC 116 Student (Fall 2015)
"Bill was doing really well until the last day when he tried to pronounce the word focaccia."	- SENG 265 Student (Summer 2016)
"At least 60% of his jokes are funny..."	- CSC 106 Student (Fall 2016)
"Great hair, although it would look better in a bun, or a braid."	- CSC 111 Student (Fall 2016)
"Essentially shoved knowledge into our ears."	- CSC 225 Student (Summer 2017)
"I definitely no longer believe in magic."	- CSC 230 Student (Summer 2017)
"Spend less time talking about FORTRAN."	- CSC 230 Student (Summer 2017)
"He looks like he just finished a shift at a Denny's"	- CSC 106 Student (Fall 2017)
"On the first day of class, this weird, scraggly looking man ran up to the front of the room, set his stuff down and proceeded to yell at the class. For the first half hour, I expected the professor to run in late and kick this guy out. Then I realized that this guy *was* the professor."	- CSC 106 Student (Fall 2017)
"Great prof. Funny. Can't draw circles."	- CSC 111 Student (Spring 2018)
"He is worryingly narcissistic, but in a good way..."	- CSC 370 Student (Spring 2018)
"Has an unusual hatred for Pineapple."	- CSC 370 Student (Spring 2018)
"He yelled the material until it made sense."	- CSC 225 Student (Summer 2018)
"Bill is the quietest professor I've ever had..."	- CSC 230 Student (Summer 2018)
"Obsessed with toast and how to make it."	- CSC 116 Student (Fall 2018)
"His love for pineapple is truly remarkable."	- CSC 111 Student (Spring 2019)

conneX Information

Course materials (including lecture slides), assignments and grades will be posted on conneX.

https://connex.csc.uvic.ca/

The posted slides are not a substitute for lecture attendance. **You are responsible for all material covered in lectures and labs, including material which is not part of the posted slides.**

Books

This course is somewhat unique to UVic, and there is no single book which covers all of the relevant material. However, the **optional** textbook below might be useful as an extra resource for many of the topics covered. The book is available for free through the UVic Library's licensing agreement (a link to a full PDF has been posted on conneX). You should view this book as a supplement to the course, since it will be possible to successfully complete the course without it. However, it should be your first resort if you need an extra resource.

The Algorithm Design Manual, 2nd ed.
Steven Skiena
Springer-Verlag, 2008/2012

Other resources and study materials will be posted as needed.

Evaluation Scheme (1)

Written Assignments

Assignment 1 (May 22, 2019)	3%
Assignment 2 (June 5, 2019)	3%
Assignment 3 (July 3, 2019)	3%
Assignment 4 (July 26, 2019)	3%

Programming Assignments

Assignment 1 (June 16, 2019)	9%
Assignment 2 (July 7, 2019)	9%
Assignment 3 (August 4, 2019)	6%

In-class Exercises

Occasional in-class exercises	7%

Exams

Exam 1 (June 7, 2019)	21%
Exam 2 (July 12, 2019)	21%
Exam 3 (August 2, 2019)	15%

Evaluation Scheme (2)

Assignments
There will be four theoretical written assignments and three programming assignments. Some of the programming assignments will be evaluated by mandatory in-person demonstrations.

Written assignments are normally due at **noon** on their due date and programming assignments are normally due at **11:55pm** on their due date. Late assignments will not be accepted.

In-class exercises
Certain lectures and lab sessions will involve exercises (usually written), which will be worth a cumulative total of 7% of your final grade. The dates of such exercises will not necessarily be announced in advance. You will receive a mark of zero if you miss such an exercise, unless an academic concession applies.

Evaluation Scheme (3)

Exams
All of the exams are 45 minutes long, in-class. All three exams are closed-book and no electronic devices are permitted, including calculators. This course has no final exam in the August exam period.

To pass the course, you must meet **all three** of the following conditions.

- Your overall grade is at least 50%.
- You receive a passing grade (at least 28/57) on the weighted average of the three exams.
- You receive a passing grade (at least 18/36) on the weighted average of the written and programming assignments.

Evaluation Scheme (4)

Missed Work
Exceptions will be made for missed work (including late assignments) **only** in cases where the standard conditions for academic concession (with documentation) apply. Links to the relevant university policies are available from the posted official course outline.

Academic Integrity
Plagiarism detection software will be used on assignment submissions where appropriate. Academic integrity violations will be reported to the department's academic integrity committee with recommendations for appropriate penalties. Links to the relevant university policies are available from the posted official course outline. Note that the university's guidelines clearly state that handing in an assignment which is mostly or entirely plagiarized should result in a grade of F being given for the course.

Evaluation Scheme (5)

Acceptable Collaboration
Computer Science and Mathematics are inherently collaborative disciplines, even if the stereotypes might say otherwise. You are encouraged to discuss all aspects of this course, including assignment questions, with your peers.

However, your actual assignment submissions must be your own work, and should be created independently (in your own words). Handing in the work of another student and claiming it as your own is plagiarism. Sharing any part of your finished submission with another student (or the internet), even if it is not directly copied by anyone else, is also plagiarism.

Rule of thumb: Talk to your peers about assignments and collaborate on conceptual solutions, but **do not** look at each other's code or finished solution (either over their shoulder or by sharing it electronically).

Evaluation Scheme (6)

Using Outside Resources
If proper citation is given, you are permitted to use material from other resources (including online sources) as part of your assignments, as long as the material was not originally from another student in this course. However, you will only be marked on the parts of your assignment that were your original work.

In fact, you are encouraged to research software resources (such as libraries) when implementing your programming assignment, since the ability to identify and harness existing resources is a valuable skill in software development.

Programming Languages

Programming assignments must be implemented in **Java**. You will not be required to use any Java features more advanced than those taught in CSC 115, although you are welcome to do so.

If you only know C++ or C#, you should be able to learn enough Java to complete the programming assignments. You should talk to your instructor soon if you do not have previous Java experience, though.

Other than the programming assignments, the course material is independent of any particular language.

Scheduling Classes (1)

Course	Prerequisites
CSC 106	(none)
CSC 110	(none)
CSC 115	CSC 110
CSC 225	CSC 115, MATH 122
CSC 226	CSC 225
CSC 230	CSC 115
CSC 320	CSC 226
CSC 360	CSC 226, CSC 230, SENG 265
CSC 370	CSC 226, SENG 265
MATH 122	(none)
SENG 265	CSC 115

One classic problem is **job scheduling**. Given a set of tasks to complete, where some tasks cannot be started until others are finished, find an feasible order in which to complete all tasks.

Scheduling Classes (2)

Course	**Prerequisites**
CSC 106	(none)
CSC 110	(none)
CSC 115	CSC 110
CSC 225	CSC 115, MATH 122
CSC 226	CSC 225
CSC 230	CSC 115
CSC 320	CSC 226
CSC 360	CSC 226, CSC 230, SENG 265
CSC 370	CSC 226, SENG 265
MATH 122	(none)
SENG 265	CSC 115

Scheduling courses for a Computer Science degree is an example of job scheduling: every course must be completed, but some courses have prerequisites that must be completed first.

Scheduling Classes (3)

Course	Prerequisites
CSC 106	(none)
CSC 110	(none)
CSC 115	CSC 110
CSC 225	CSC 115, MATH 122
CSC 226	CSC 225
CSC 230	CSC 115
CSC 320	CSC 226
CSC 360	CSC 226, CSC 230, SENG 265
CSC 370	CSC 226, SENG 265
MATH 122	(none)
SENG 265	CSC 115

Step 1: State the problem concisely.

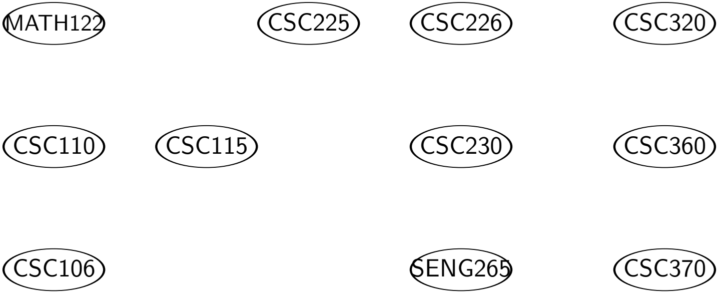
- **Input**: A set of courses, each with a list of prerequisites.
- **Output**: A sequence containing a valid ordering of courses.

Scheduling Classes (4)

Course	**Prerequisites**
CSC 106	(none)
CSC 110	(none)
CSC 115	CSC 110
CSC 225	CSC 115, MATH 122
CSC 226	CSC 225
CSC 230	CSC 115
CSC 320	CSC 226
CSC 360	CSC 226, CSC 230, SENG 265
CSC 370	CSC 226, SENG 265
MATH 122	(none)
SENG 265	CSC 115

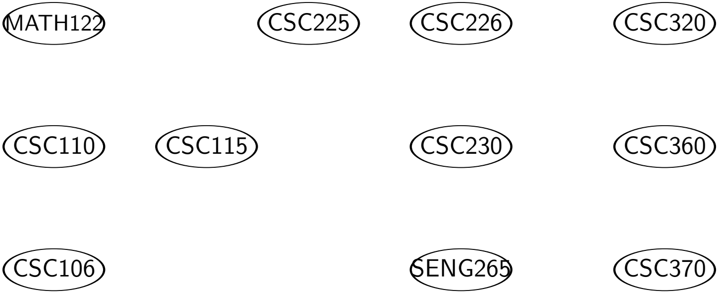
The meaning of terms such as ‘set’, ‘list’ and ‘sequence’ will become significant when an algorithm is implemented. However, to formulate a solution, we do not need to implement anything.

Scheduling Classes (5)



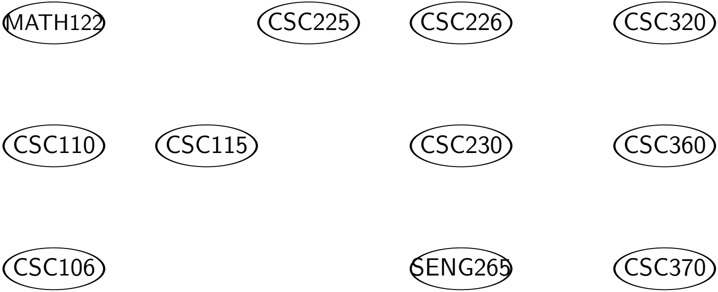
Step 2: Choose data structures to model the problem.

Scheduling Classes (6)



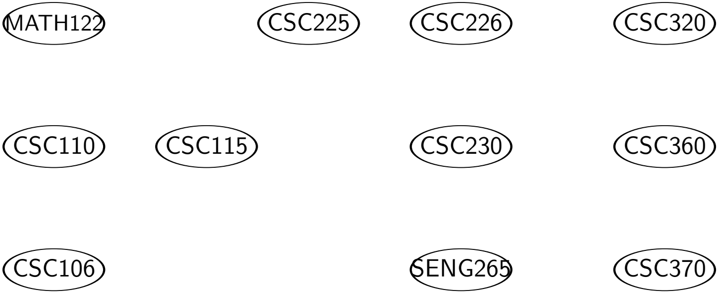
For this problem, a **graph** is a good choice.

Scheduling Classes (7)



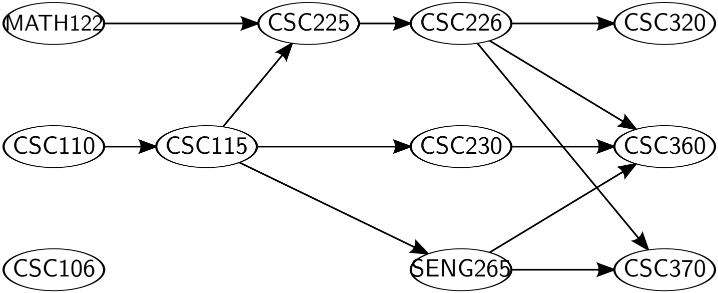
Choosing the right data structures can make a solution much easier to create and understand.

Scheduling Classes (8)



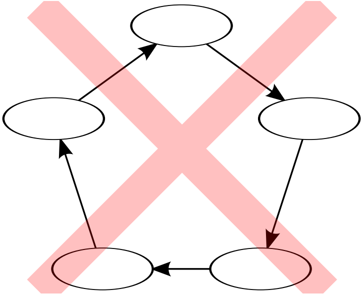
Having a broad understanding of data structures is important to make the right choices.

Scheduling Classes (9)



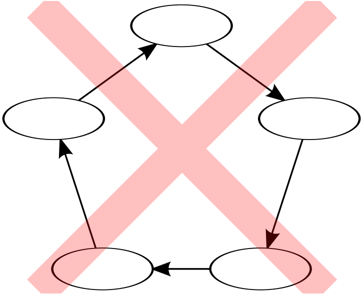
Create a node for each course, and add an arrow from each course C to every course that requires C as a prerequisite.

Scheduling Classes (10)



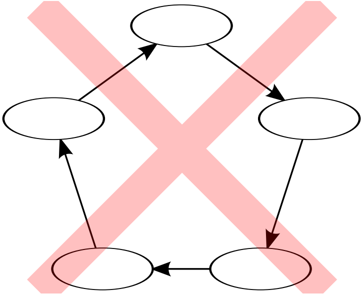
Step 3: Determine the constraints of the problem.

Scheduling Classes (11)



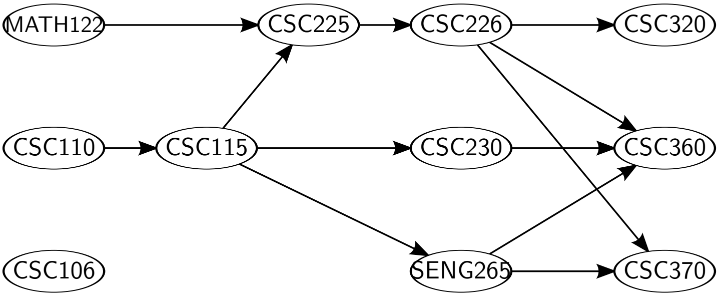
No course can be a prerequisite of itself, directly or indirectly, since otherwise it would be impossible to take the course.

Scheduling Classes (12)



Therefore, configurations like the one above (cycles) will never occur.

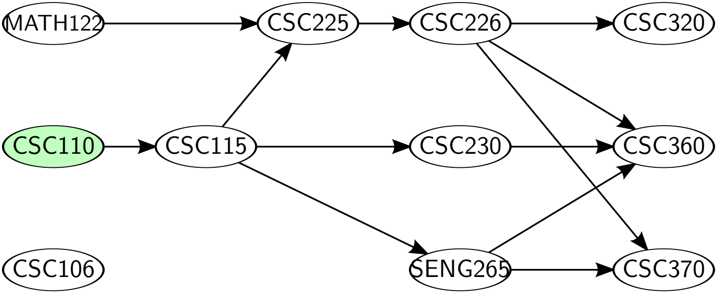
Scheduling Classes (13)



Course Order:

Step 4: Design an algorithm.

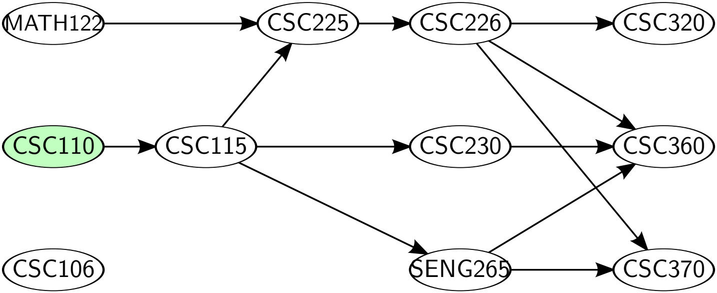
Scheduling Classes (14)



Course Order: CSC 110

Observation: It is always permissible to take a course with no prerequisites.

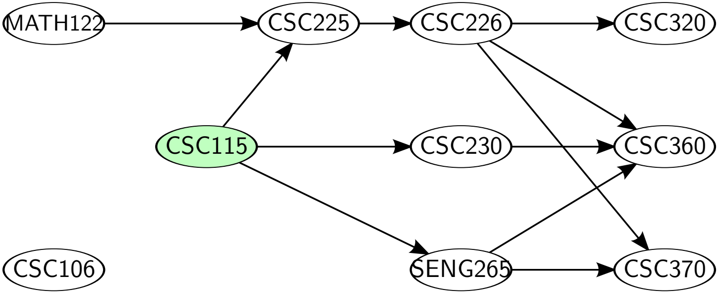
Scheduling Classes (15)



Course Order: CSC 110

Therefore, a node with no incoming arrows can always be added to the course list.

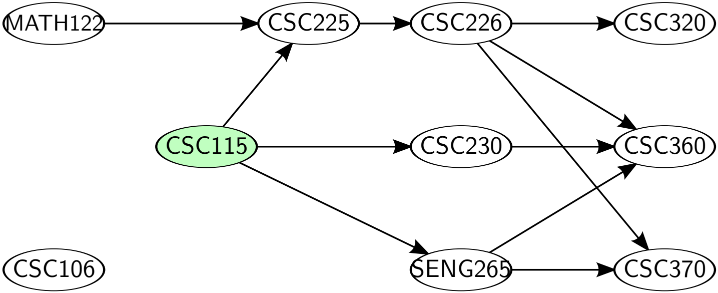
Scheduling Classes (16)



Course Order: CSC 110

Since CSC 110 has now been added to the final ordering, it can be removed from the graph, since it will be taken before any courses which require it.

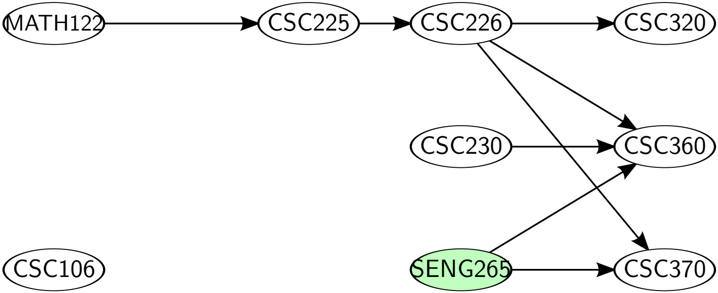
Scheduling Classes (17)



Course Order: CSC 110, CSC 115

After removing CSC 110 from the graph, another course with no incoming arrows can be chosen and added to the ordering.

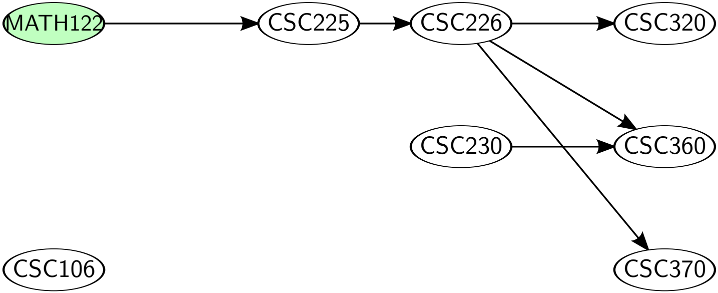
Scheduling Classes (18)



Course Order: CSC 110, CSC 115, SENG 265

As each course is added to the list, it is deleted from the graph. Deleting a course will free up other courses to be added.

Scheduling Classes (19)

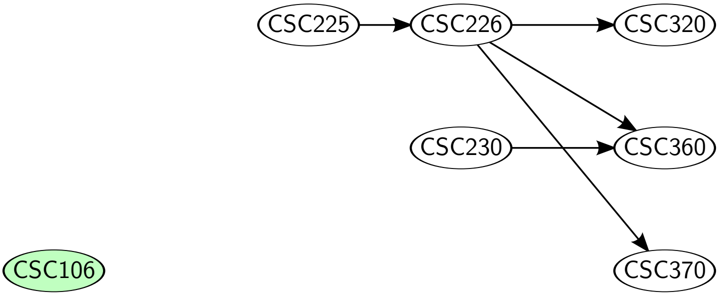


Course Order: CSC 110, CSC 115, SENG 265, MATH 122

Algorithm

```
while the graph is non-empty do
    Choose a course $C$ with no incoming arrows.
    Add $C$ to the course list and delete it from the graph.
end while
```

Scheduling Classes (20)

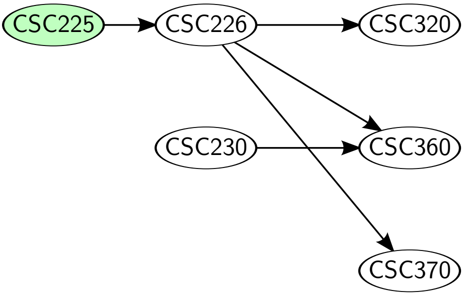


Course Order: CSC 110, CSC 115, SENG 265, MATH 122, CSC 106

Algorithm

```
while the graph is non-empty do
    Choose a course $C$ with no incoming arrows.
    Add $C$ to the course list and delete it from the graph.
end while
```

Scheduling Classes (21)

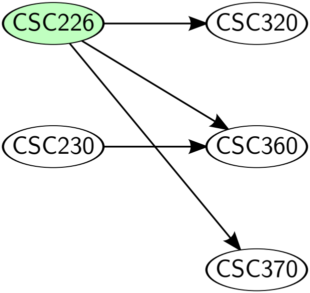


Course Order: CSC 110, CSC 115, SENG 265, MATH 122, CSC 106
CSC 225

Algorithm

```
while the graph is non-empty do
    Choose a course C with no incoming arrows.
    Add C to the course list and delete it from the graph.
end while
```

Scheduling Classes (22)

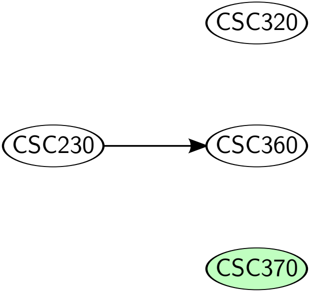


Course Order: CSC 110, CSC 115, SENG 265, MATH 122, CSC 106
CSC 225, CSC 226

Algorithm

```
while the graph is non-empty do
    Choose a course $C$ with no incoming arrows.
    Add $C$ to the course list and delete it from the graph.
end while
```

Scheduling Classes (23)

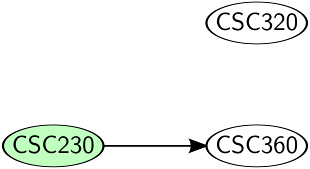


Course Order: CSC 110, CSC 115, SENG 265, MATH 122, CSC 106
CSC 225, CSC 226, CSC 370

Algorithm

```
while the graph is non-empty do
    Choose a course $C$ with no incoming arrows.
    Add $C$ to the course list and delete it from the graph.
end while
```

Scheduling Classes (24)

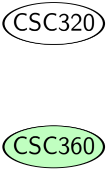


Course Order: CSC 110, CSC 115, SENG 265, MATH 122, CSC 106
CSC 225, CSC 226, CSC 370, CSC 230

Algorithm

```
while the graph is non-empty do
    Choose a course $C$ with no incoming arrows.
    Add $C$ to the course list and delete it from the graph.
end while
```

Scheduling Classes (25)



Course Order: CSC 110, CSC 115, SENG 265, MATH 122, CSC 106
CSC 225, CSC 226, CSC 370, CSC 230, CSC 360

Algorithm

```
while the graph is non-empty do
    Choose a course $C$ with no incoming arrows.
    Add $C$ to the course list and delete it from the graph.
end while
```

Scheduling Classes (26)

Course Order: CSC 110, CSC 115, SENG 265, MATH 122, CSC 106
CSC 225, CSC 226, CSC 370, CSC 230, CSC 360, CSC 320

Algorithm

```
while the graph is non-empty do
    Choose a course $C$ with no incoming arrows.
    Add $C$ to the course list and delete it from the graph.
end while
```

Sources

- Slides by B. Bird, 2015 - 2019.
- The road maps on slides 7 and 8 were created from OpenStreetMap data (© OpenStreetMap contributors).
- The data used to construct the river maps on slides 13 - 15 was derived from the TRIP dataset (T. Oki and Y. C. Sud, 1998).