CSC 225 - Summer 2019

Traversals III

Department of Computer Science
University of Victoria

July 19, 2019

Flood Fill (1)

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

Flood Fill (2)

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

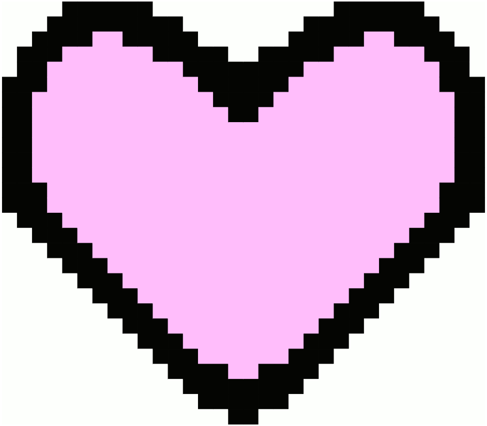
Flood Fill (3)

This problem can be solved using a graph traversal by modeling the image with an appropriate graph.

University
of Victoria

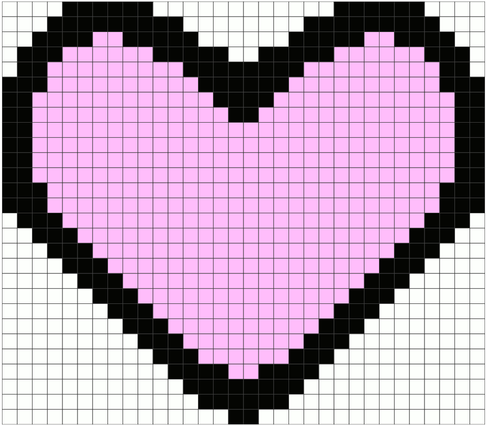
The image above is too high resolution to illustrate the graph needed, though.

Flood Fill (5)



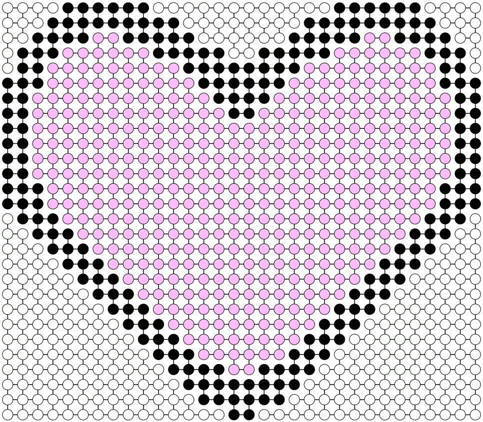
Consider the **lovely heart** above.

Flood Fill (6)



A **pixel graph** can be created from an image by creating a vertex for each pixel and adding edges between neighbouring pixels if they have the same colour.

Flood Fill (7)

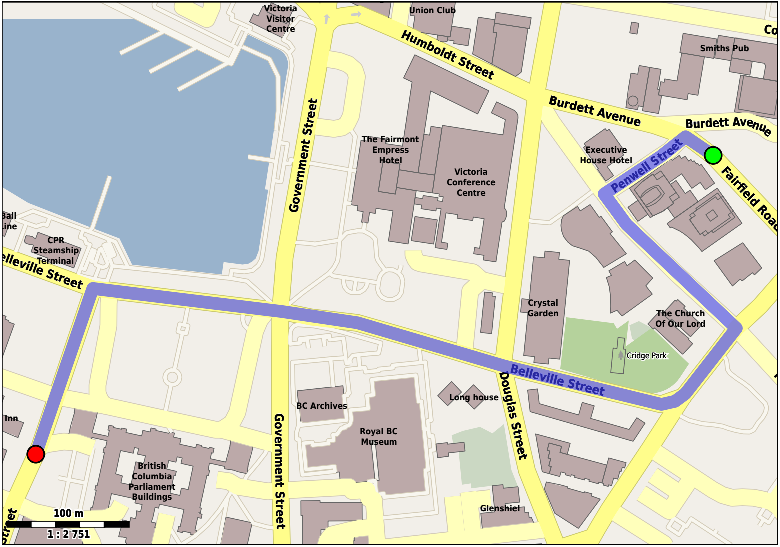


To perform the flood fill operation, a traversal (DFS or BFS) can be used to find all of the vertices in the connected component of the starting pixel and change their colour.

Path Finding (1)

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

Path Finding (2)

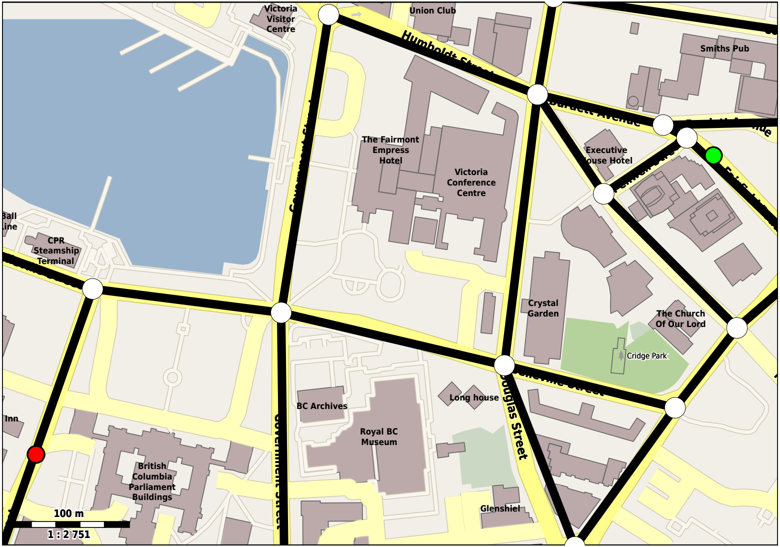


...find a path from the start point to the end point.

Path Finding (3)

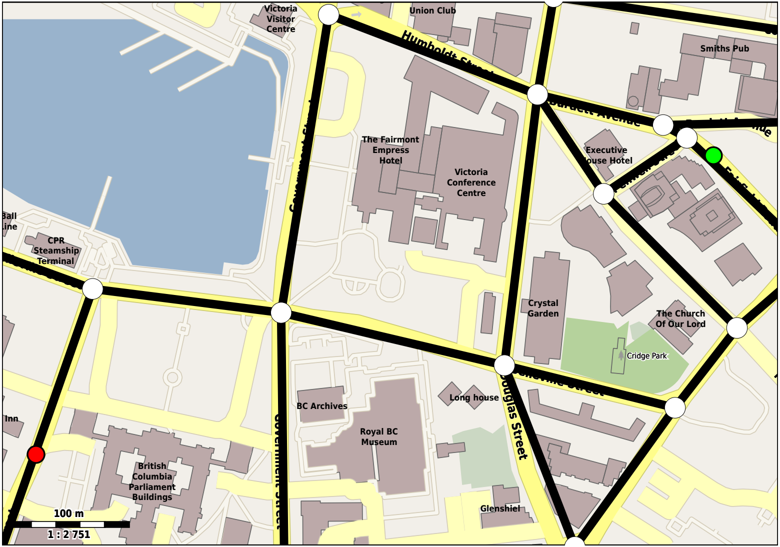
The map can be represented by a graph with a vertex for each intersection.

Path Finding (4)



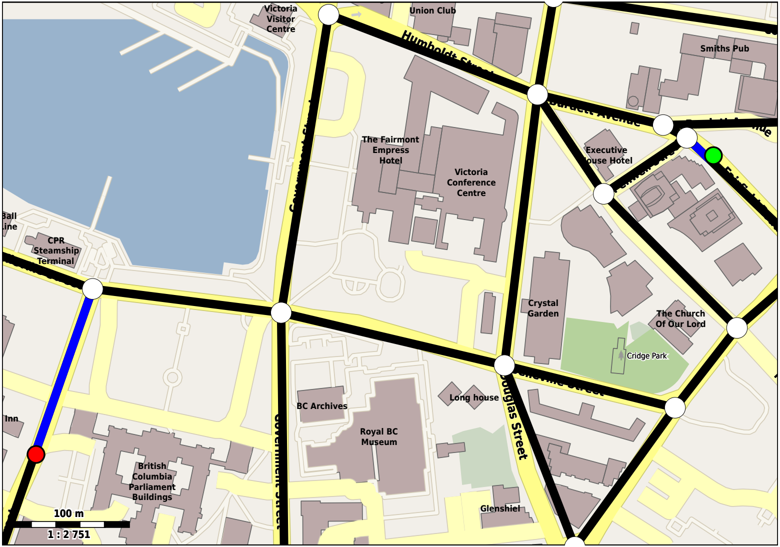
The roads themselves are represented by edges that connect two intersections.

Path Finding (5)



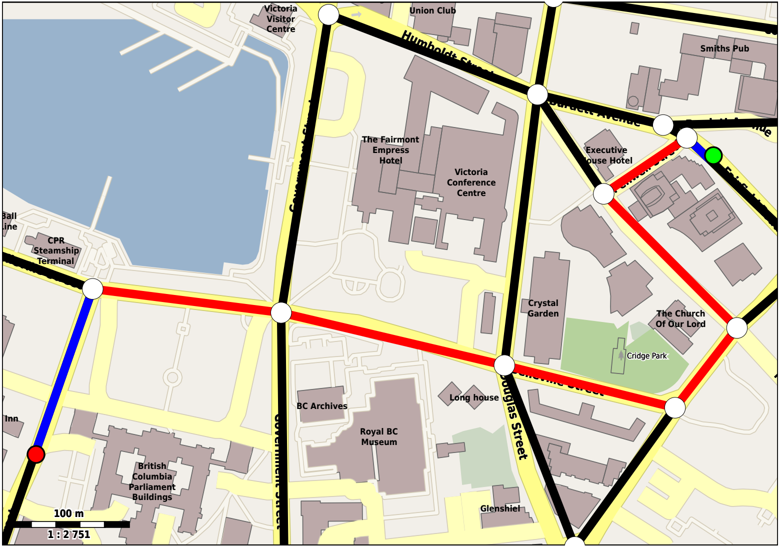
(This example ignores cases like dead-end roads).

Path Finding (6)



To find a path between two arbitrary points on edges in the graph, first find a path (blue) to the nearest intersection.

Path Finding (7)



Then, use a graph traversal algorithm to find a path between the two intersections.

The 16-Puzzle (1)

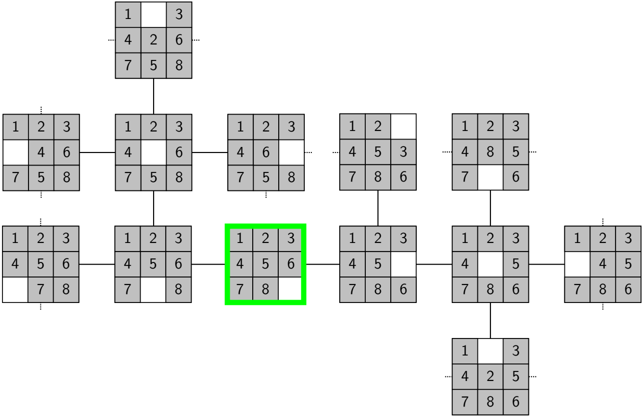
2	15	4	8
13	3	14	11
6		1	12
7	9	5	10

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)

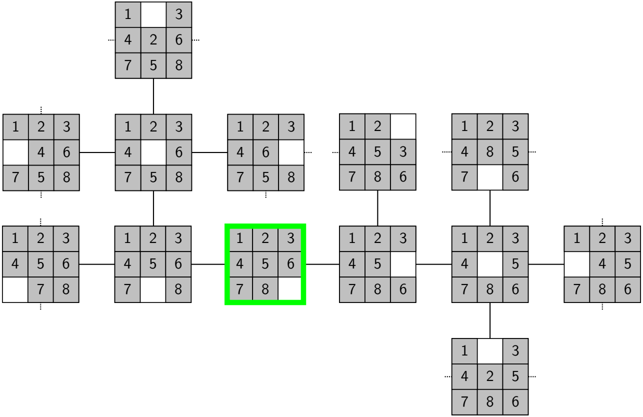
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

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



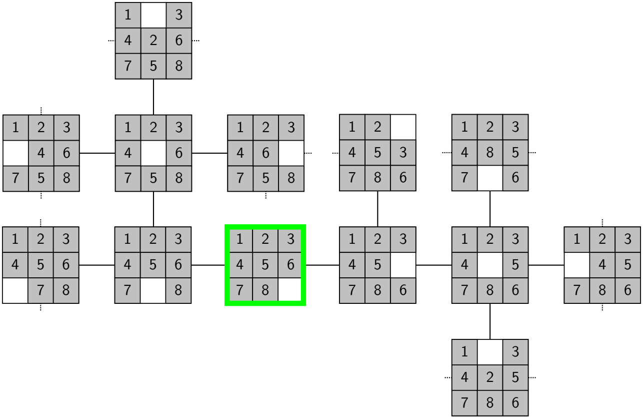
Sliding tile puzzles (such as the 9-puzzle above) can be solved by building a graph with all possible states of the board as vertices.

The 16-Puzzle (4)



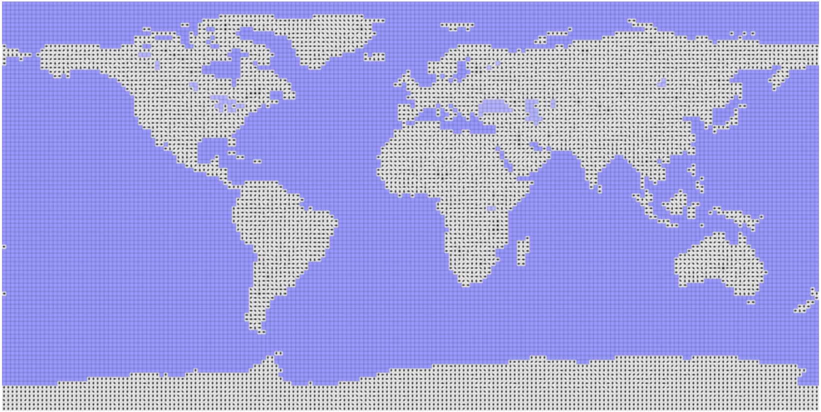
Edges connect two states which can be obtained from each other by moving a single tile.

The 16-Puzzle (5)



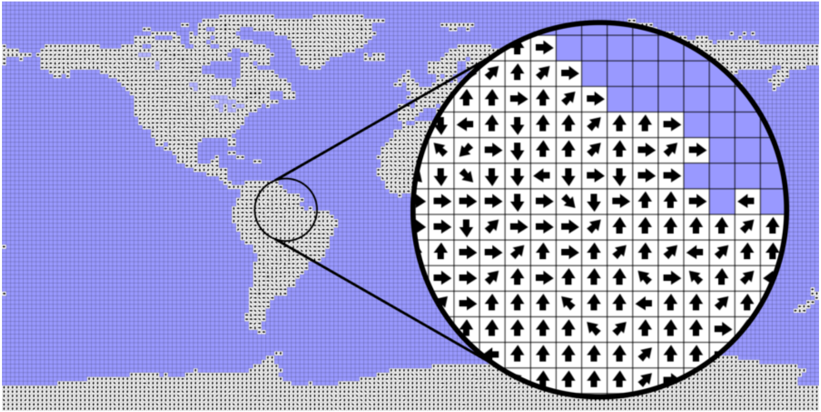
To find a sequence of moves that solves a given configuration, a traversal algorithm can be used to find a path to the goal state (framed in green).

River Routing (1)



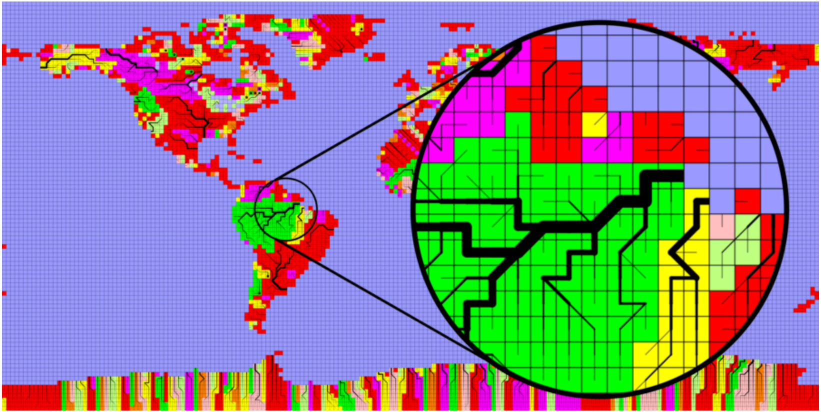
Problem: Given a regular grid covering the surface of the earth, in which every land cell contains the direction of water flow in the corresponding region...

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.

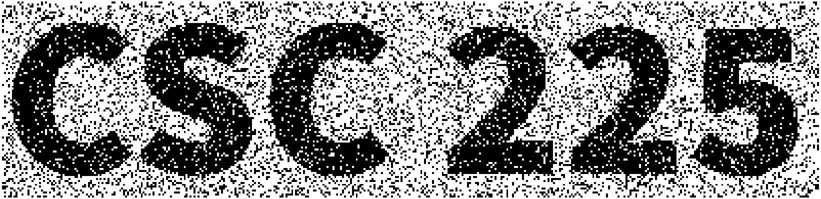
River Routing (4)

The river routing problem is a surprisingly straightforward application of graph algorithms.

The map of river directions can be represented by a graph with a vertex for each grid cell and an edge between each cell and the cell it drains into. The graph will be a forest, with each component representing a single river basin.

After constructing the graph, each river basin can be determined by performing a traversal starting at the river endpoint (which will be a cell that drains into the ocean). The volume of the river at each point can be estimated by considering the number of upstream vertices.

OCR Noise Reduction (1)



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

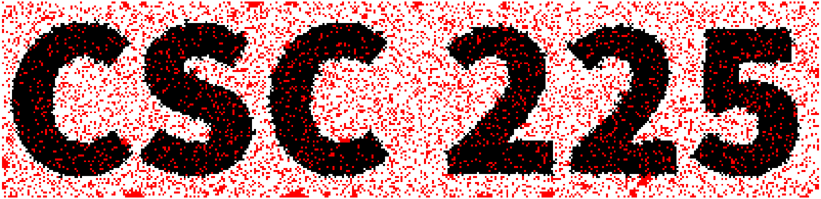
OCR Noise Reduction (2)

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

OCR Noise Reduction (3)

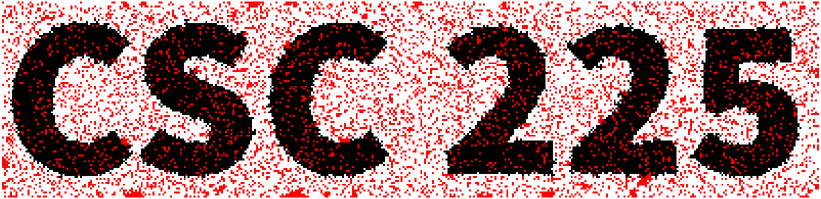
This problem can be solved by using a pixel graph structure.

OCR Noise Reduction (4)



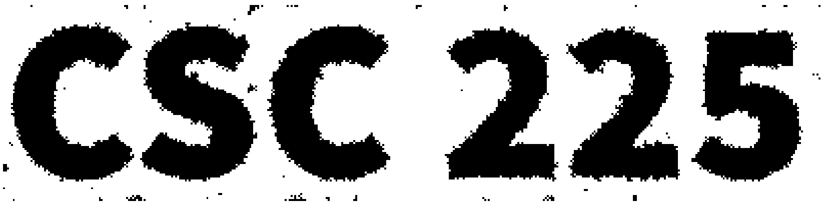
First, find all connected components of the graph.

OCR Noise Reduction (5)



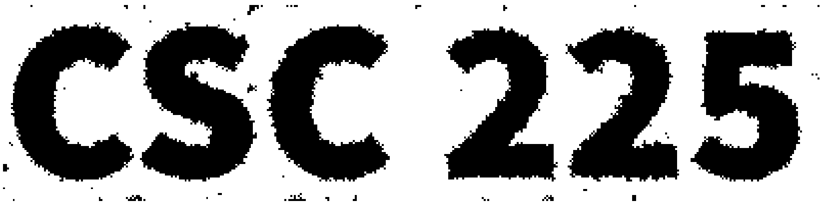
Noise pixels should occur in relatively small and sparse clusters, which correspond to connected components with a small number of vertices (e.g. less than 50).

OCR Noise Reduction (6)



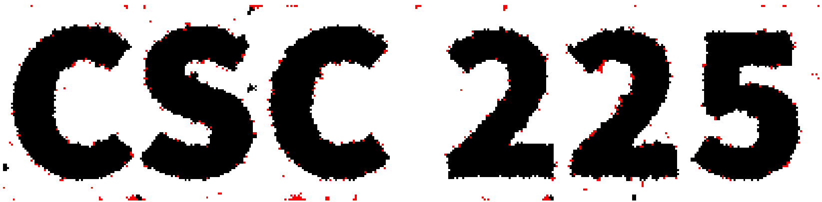
Since the image has only two colours (black and white), it is easy to 'fix' any noise pixels once they are found.

OCR Noise Reduction (7)

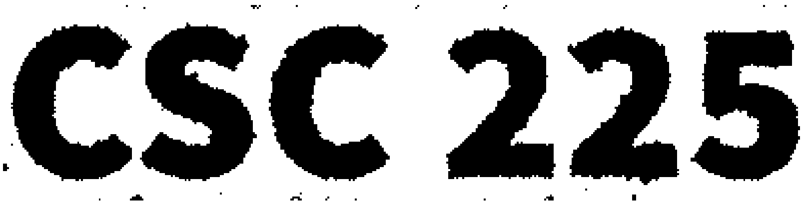


Flipping the colour value of all pixels in small components cleans up most of the noise in the image.

OCR Noise Reduction (8)

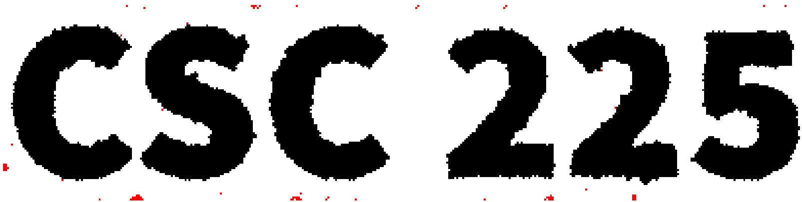


The ragged appearance of the result image is the result of large numbers of pixels lying on the boundaries between regions. These pixels correspond to pixel graph vertices with low degree (less than 3).



A graph traversal can be used to smooth the boundaries by flipping the colours of all low-degree vertices and recursively flipping the colours of any newly-created low-degree vertices (which must be adjacent to a previously flipped vertex).

OCR Noise Reduction (10)



To further improve the image, the entire noise reduction process can be repeated.

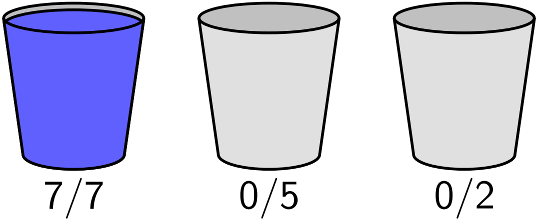
OCR Noise Reduction (11)

To further improve the image, the entire noise reduction process can be repeated.

OCR Noise Reduction (12)

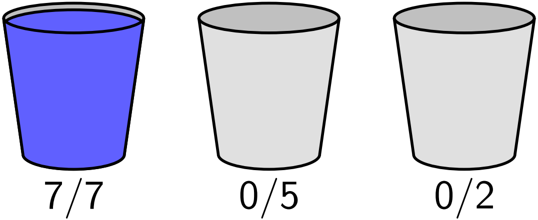
This algorithm is a form of the DBSCAN algorithm used in data mining applications, and can be implemented with two graph traversals (either DFS or BFS).

Measuring With Buckets (1)



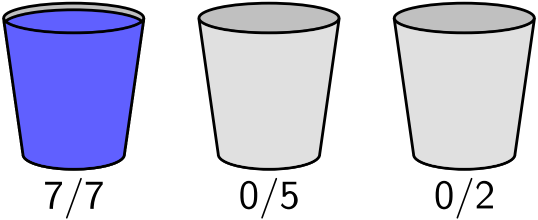
Problem: You are given three buckets with capacities $7, 5$ and 2. The largest bucket is full. Find a way to measure exactly 4 units of water.

Measuring With Buckets (2)



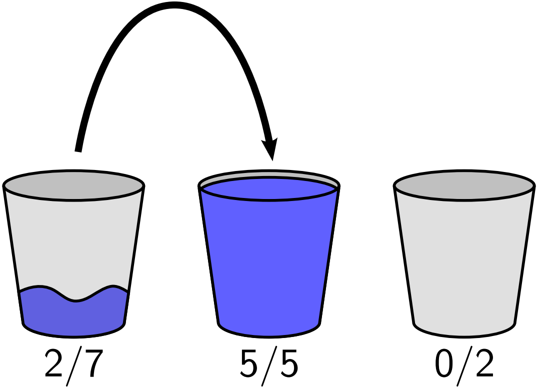
Water can be poured between buckets, but no water can be added or spilled. When you pour between two buckets, you must pour until either the source bucket is empty or the destination bucket is full.

Measuring With Buckets (3)



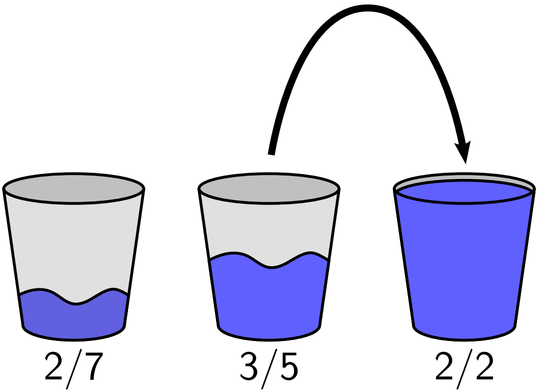
This example can be solved in four steps.

Measuring With Buckets (4)



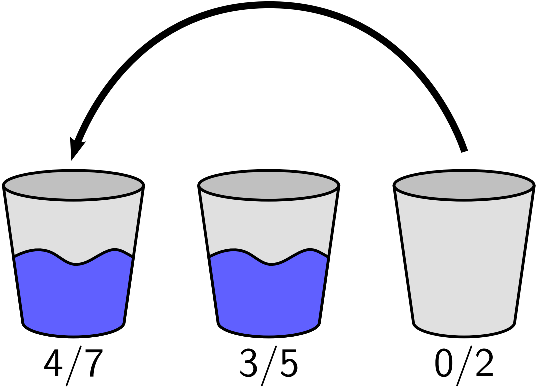
First, pour five units from bucket 1 to bucket 2.

Measuring With Buckets (5)



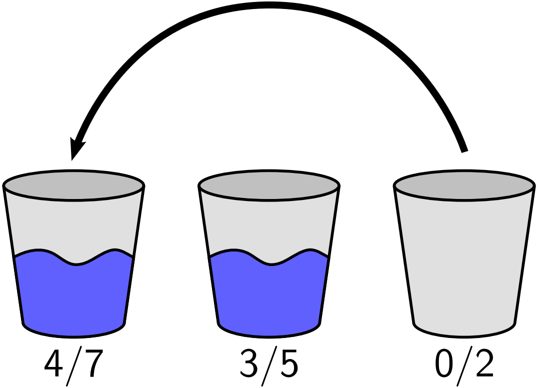
Then, pour two units from bucket 2 to bucket 3.

Measuring With Buckets (6)



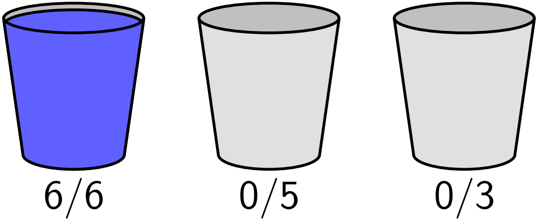
Finally, empty bucket 3 into bucket 1.

Measuring With Buckets (7)



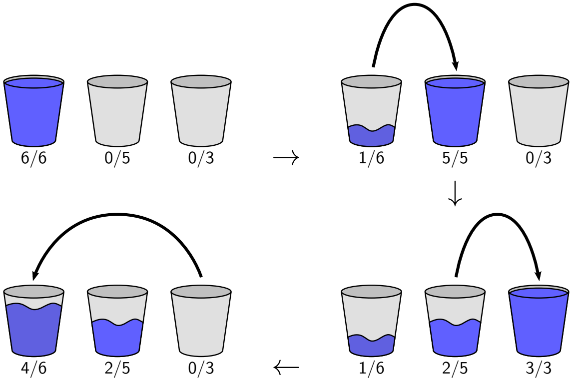
The problem is solved when any bucket contains exactly 4 units.

Measuring With Buckets (8)



Exercise: Starting from the buckets above, find a sequence of pouring operations that measures 4 units of water.

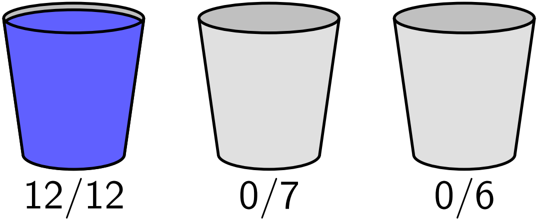
Measuring With Buckets (9)



This version is also solvable in 4 steps.
We can represent the solution by a sequence of triples:

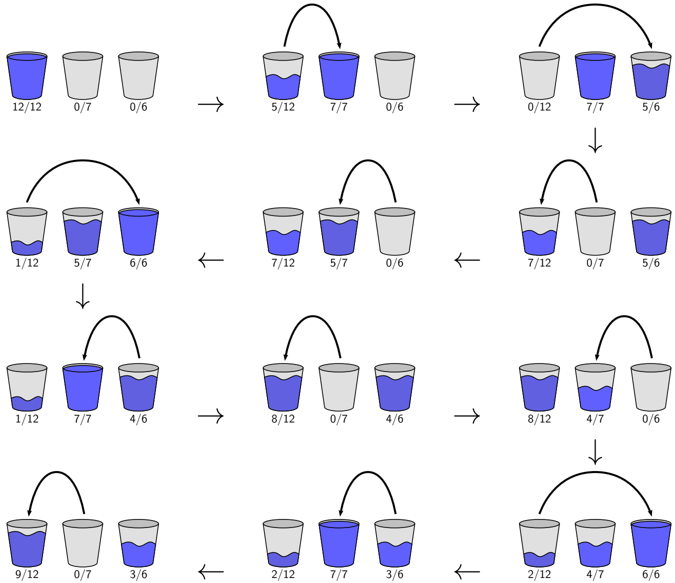
$$(6\ 0\ 0) \rightarrow (1\ 5\ 0) \rightarrow (1\ 2\ 3) \rightarrow (4\ 2\ 0)$$

Measuring With Buckets (10)

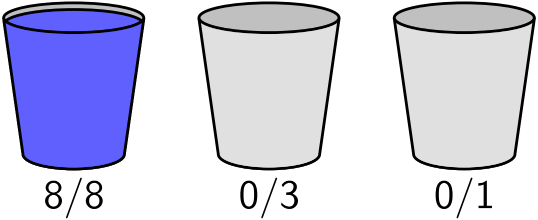


Exercise: Find a sequence of pouring operations that measures 9 units of water starting from the initial conditions above.
This problem requires at least 12 steps (see next slide).

Measuring With Buckets (11)

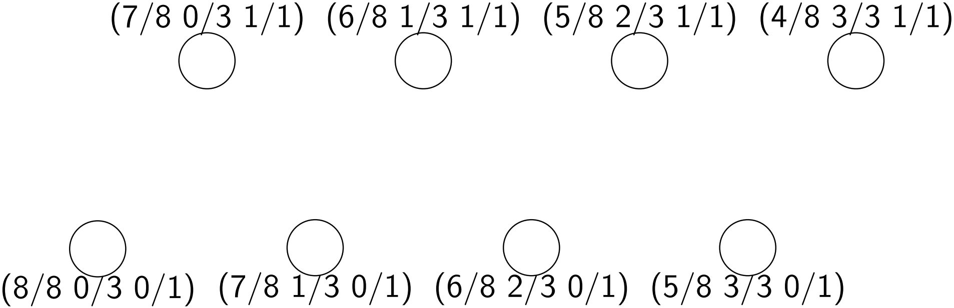


Filling Buckets Using Graphs (1)



Problem: Design an algorithm which takes an initial configuration (such as (8 0 0) in the diagram above) and a goal measurement (such as 6) and a sequences of pouring operations to obtain the measurement (if a solution exists).

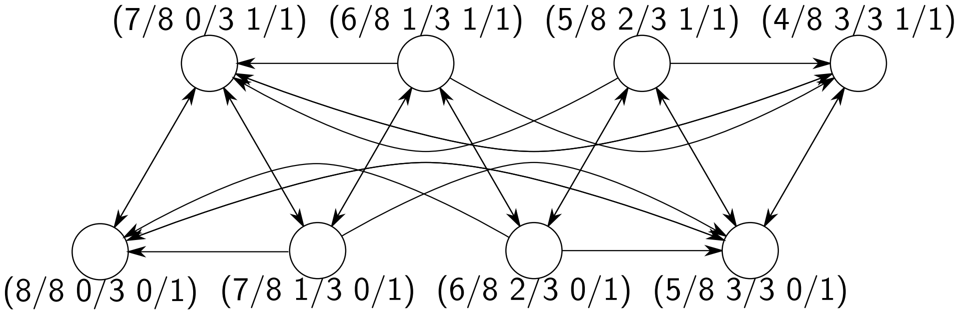
Filling Buckets Using Graphs (2)



- The problem can be modelled with a graph, using a vertex for each possible state of the buckets.
- If the maximum capacities are C_1, C_2 and C_3 (and are integers), at most $C_1 C_2 C_3$ vertices are needed.
- In the example on the previous slide, the capacities were

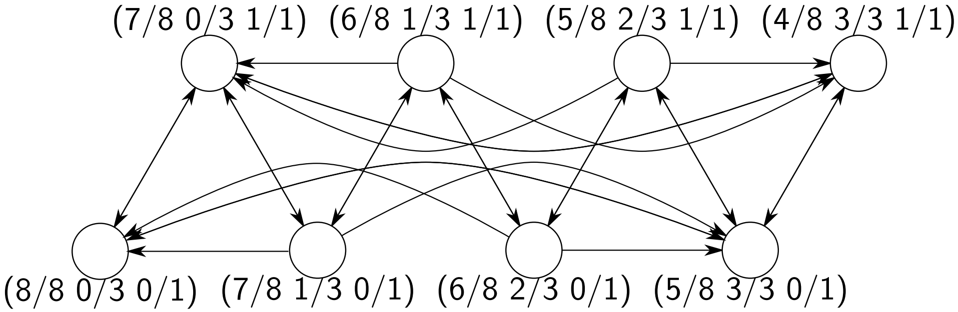
$$C_1 = 8, \quad C_2 = 3, \quad C_3 = 1$$

Filling Buckets Using Graphs (3)



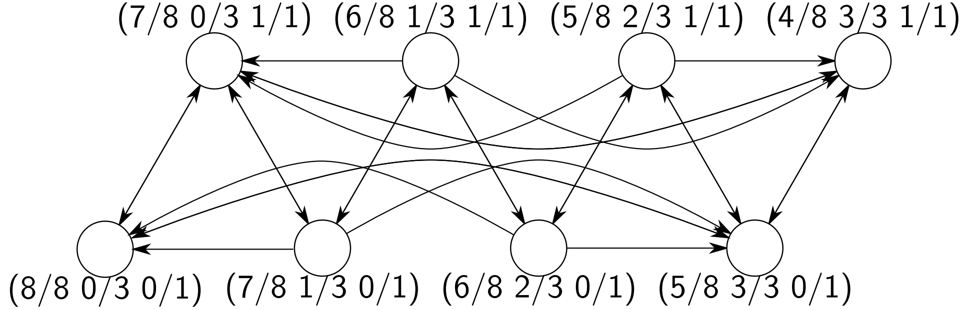
- Edges will correspond to the pouring operation.
- An edge is added from *u* to *w* if *w* is the result of performing some pouring operation on the configuration at vertex *u*.

Filling Buckets Using Graphs (4)



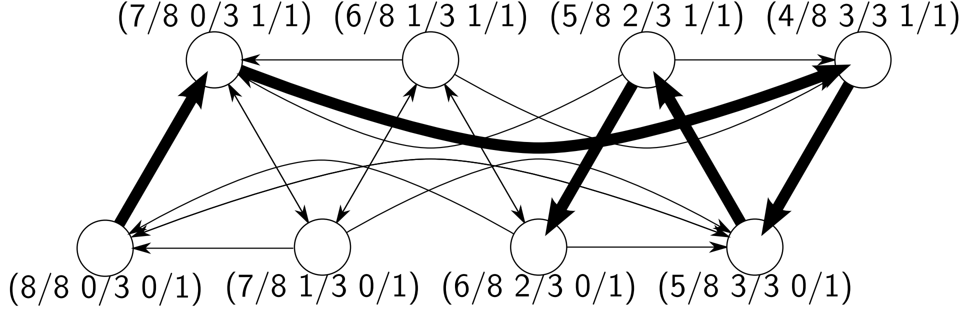
- Note that the graph must be directed, since some pouring operations are not reversible.

Filling Buckets Using Graphs (5)



The graph can be constructed with completely 'local' knowledge of each state. Given some state $u = (x\ y\ z)$, the set of other states w that can be obtained by pouring buckets is easy to compute.

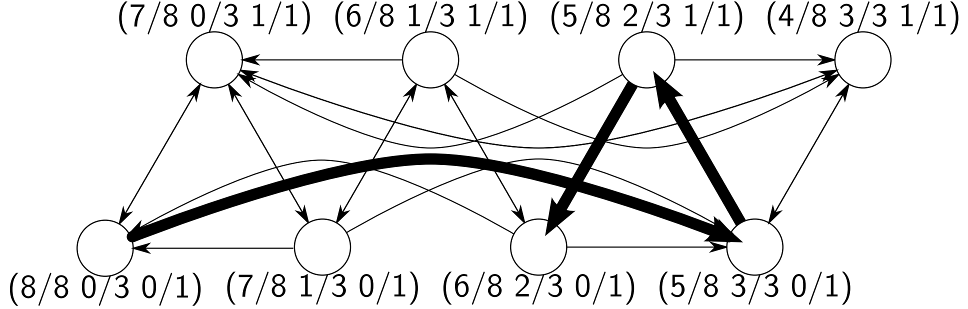
Filling Buckets Using Graphs (6)



After the graph is constructed, a DFS traversal can be used to find a path from the initial state (8 0 0) to a state with the goal of 6 units measured. In the example above, the path is

$$(8\ 0\ 0) \rightarrow (7\ 0\ 1) \rightarrow (4\ 3\ 1) \rightarrow (5\ 3\ 0) \rightarrow (5\ 2\ 1) \rightarrow (6\ 2\ 0)$$

Filling Buckets Using Graphs (7)



Using a BFS traversal guarantees the shortest possible path. In the example above, the path is

$$(8\ 0\ 0) \rightarrow (5\ 3\ 0) \rightarrow (5\ 2\ 1) \rightarrow (6\ 2\ 0)$$

Sources

- Slides by B. Bird, 2015 - 2019.
- The road maps on slides 9 - 15 were created from OpenStreetMap data (© OpenStreetMap contributors).
- The data used to construct the river maps on slides 21 - 23 was derived from the TRIP dataset (T. Oki and Y. C. Sud, 1998).
- The version of the three bucket problem used in these slides (and one of the examples) is based on a description by Stan Wagon at `http://mathforum.org/wagon/2015/p1204.html`.