Project 3: Graph modeling and graph algorithms

COT 4400, Fall 2019

Due Dec 4, 2019

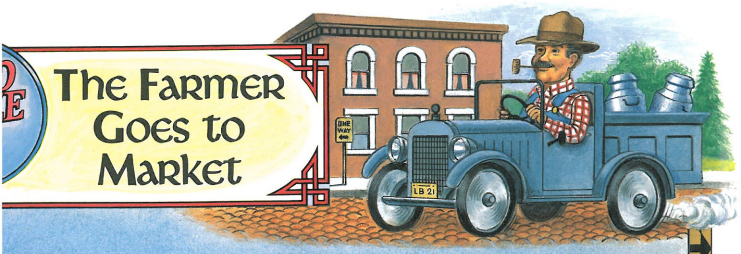
1 Overview

This project requires you to model the problem below as graph and then use a known graph algorithm to solve the problem. **You are not allowed to use the internet or consult any references. This is an individual project. This policy will be strictly enforced.**

This problem is based on the "Itsy-Bitsy Spider" maze problem (from "MAD MAZES: Intriguing Mind Twisters for Puzzle Buffs, Game Nuts and Other Smart People" by Robert Abbott). The text of the problem is quoted below. A diagram of the maze is provided on the following page.

> Fred was discussing a problem with his architect:
>
> "I have a terrible itsy bitsy spider infestation. Actually, it's only one itsy bitsy spider, but he's very persistent. He spins a web on the roof that clogs up the gutter. When it rains, he is washed down the water spout. But then along comes the sun and dries up all the rain and the itsy bitsy spider climbs up the spout again. And there's not a damn thing I can do about it."
>
> "Don't worry," said the architect. "He may be persistent but spiders aren't that bright. I suggest we put a maze at the bottom of the water spout; then the spider won't be able to find his way back up."
>
> The architect came back the next day with the diagram shown at the right. "What I propose is this: we run the water spout into a large metal box. The box will contain many small chambers on five levels. The water will come in at the top, on Level A, will travel from chamber to chamber, and will drain out at this opening on Level E. There has to be a path through the box, otherwise the water won't be able to drain through. But I've made the path very complicated. Once the spider gets washed out of the opening on Level E, he won't be able to figure out how to get back through the box and back into the water spout.
>
> "The solid black lines represent walls, which the spider cannot get through. I've used yellow to indicate floors. If the spider's on a yellow square, he cannot travel from that point down to the next level. If a square is not colored yellow, then he can travel down to the corresponding square in the level below. To figure out whether the spider can travel *up*, you have to check out the corresponding square in the level above. If that square has no floor, then the spider can travel up. If the square does have a floor (that is, it's colored yellow), then the spider cannot travel up."
>
> "This sounds like a perfectly reasonable plan to me," said Fred. "In fact, I can't figure out how to get through the box myself." So, they built the box and added it to the water spout. The first rainstorm did wash the spider through the box, but, unfortunately, after the sun dried up the rain the spider went back into the box, traveled through it and back up the water spout. Can you discover a path the spider could take to get through the box? You have to in the entrance on Level E and then work your way up to the water spout on Level A.

The Farmer Goes to Market



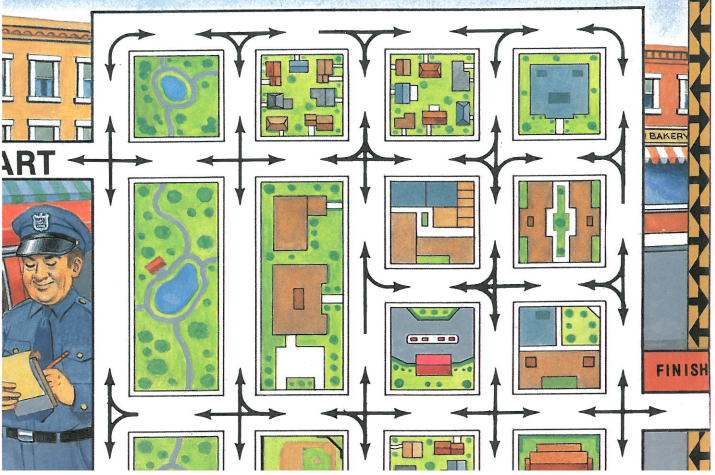
the town of Floyd's Knob, In-
ad only 37 registered automobiles,
or thought it would be safe to ap-
s cousin, Henry Stables, who was
cutup, as its traffic commissioner.
oon regretted his decision. When
awoke one morning, it found that
ion of signs had been erected es-
ng confusing restrictions on turns
intersection in town.
e citizens were all for tearing
ese signs until the police chief,
cousin of the mayor, made a
ng discovery. Motorists passing
town became so exasperated that
r later they made a prohibited turn.
ice chief found that the town was
even more money from these
ns than from its speed trap on an
country road.

Of course everyone was overjoyed, particularly because the next day was Saturday and Moses MacAdam, the county's richest farmer, was due to pass through town on his way to the county seat. They expected to extract a large fine from Moses, believing it to be impossible to drive through town without at least one traffic violation. But Moses had been secretly studying the signs. When Saturday morning came, he astonished the entire town by driving from his farm through town to the county seat without a single violation!

Can you discover a route that Moses could have taken? Enter town on the road at the left and exit on the road at the right. At each intersection you must follow one of the arrows. That is, you may turn in a given direction only when there is a curved line in that direction, and you may go straight only when there is a straight line to follow. You may leave an intersection only at the head of an arrow. U-turns or backing up is not allowed.

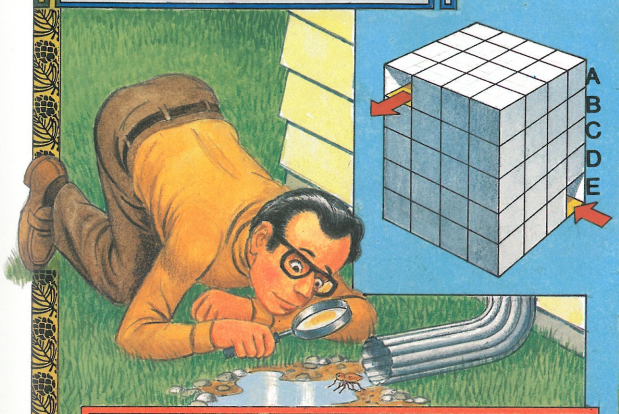
As you can see, at the first intersection you can only go straight. At the second intersection you encounter you can again only go straight. At the third intersection you can go straight or turn north. Suppose you turn north. At the next intersection you can only turn east. True, there is a line that curves to the west, but there's no arrowhead pointing west, so you can't leave that intersection in a westerly direction.

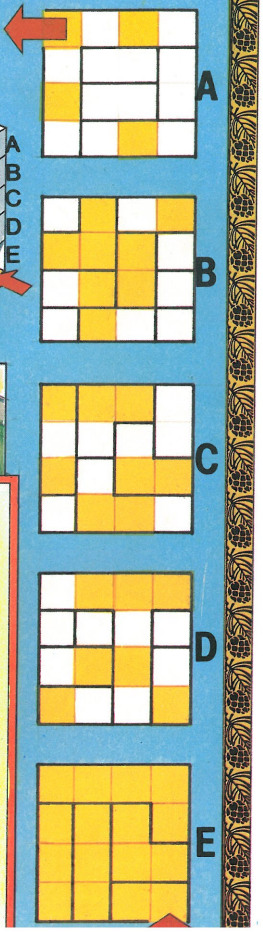
In this map, and in the other city maps in the book, there are driveways drawn inside the blocks. The driveways are only decorative; you can't use them as part of your route through a maze.



MAD MAZE 4

The Itsy Bitsy Spider





Fred was discussing a problem with his architect:

"I have a terrible itsy bitsy spider infestation. Actually, it's only one itsy bitsy spider, but he's very persistent. He spins a web on the roof that clogs up the gutter. When it rains, he is washed down the water spout. But then along comes the sun and dries up all the rain and the itsy bitsy spider climbs up the spout again. And there's not a damn thing I can do about it."

"Don't worry," said the architect. "He may be persistent but spiders aren't that bright. I suggest we put a maze at the bottom of the water spout; then the spider won't be able to find his way back up."

The architect came back the next day with the diagram shown at the right. "What I propose is this: we run the water spout into a large metal box. The box will contain many small chambers on five levels. The water will come in at the top, on Level A, will travel from chamber to chamber, and will drain out at this opening on Level E. There has to be a path through the box, otherwise the water won't be able to drain figure out how to get back through the box and back into the water spout.

"The solid black lines represent walls, which the spider cannot get through. I've used yellow to indicate floors. If the spider's on a yellow square, he cannot travel from that point down to the next level. If a square is not colored yellow, then he can travel down to the corresponding square in the level below. To figure out whether the spider can travel *up*, you have to check out the corresponding square in the level above. If that square has no floor, then the spider can travel up. If the square does have a floor (that is, it's colored yellow), then the spider cannot travel up."

"This sounds like a perfectly reasonable plan to me," said Fred. "In fact, I can't figure out how to get through the box myself." So, they built the box and added it to the water spout. The first rainstorm did wash the spider through the box, but, unfortunately, after the sun dried up the rain the spider went back into the box, traveled through it and back up the water spout. Can you discover a path the spider could take to get through the box? You have to go in the

2 Modeling the problem

Before you write a program to solve this problem, you will first write a report describing (in English and pseudocode) how you will solve this problem. This report should answer two basic questions:

1. What type of graph would you use to model the problem input (detailed in the Section 3.1), and how would you construct this graph? (I.e., what do the vertices, edges, etc., correspond to?) Be specific here; we discussed a number of different types of graphs in class.
2. What algorithm will you use to solve the problem? Be sure to describe not just the general algorithm you will use, but how you will identify the sequence of moves the spider must take in order to reach the goal.

3 Coding your solution

In addition to the report, you should implement your algorithm in C++ or Java so that it can solve "Itsy Bitsy Spider" mazes. Your code may be in C++ or Java, but it must compile and run on the C4 Linux Lab machines.

Your code may be split into any number of files. In addition, you are allowed to make use of any built-in library, and C++ users may use the Boost library in their implementations. Boost is a free, open-source library with a rich collection of mathematical functions, including several that deal with graphs. You may read more about Boost at `www.boost.org`.

3.1 Input format

Your program should read its input from the file `input.txt`, in the following format. The input begins with a one positive integer on a line representing the number of mazes in the file. Mazes are separated by blank lines in the input.

Each maze starts with 9 integers on 3 lines. The first line describes the number of levels, rows, and columns in the maze (ℓ, r, and c), respectively, while the second and third lines describe the locations of the start and goal. Note that the coordinates of the start and goal are given using a base of 0.

The next hr lines contain the directional information for each cell in the maze. The bottom level is described first, and all rows for one level of the maze are described before the next level appears. Each line has c 6-digit binary values, where each bit represents whether the spider can travel in that direction (1) or not (0). The direction bits are given in the order north, east, south, west, up, and down.

For the original "Itsy Bitsy Spider" maze, the input is:

```
  5       4       4
  0       3       3
  4       0       0
010010  010100  010100  001100
001010  001010  001010  100010
101010  101000  110000  000110
100000  100010  010010  000100
010001  010100  011100  000110
000011  000011  101011  000011
010001  000110  101000  001001
010010  000101  100001  100010
001010  011000  000110  001001
101001  100001  001001  100011
100010  001001  110000  000110
000011  110000  010110  000111
000001  001010  010001  000110
010010  100110  001010  000011
010001  000110  100010  000011
000011  010010  000101  000011
011000  010101  010100  000101
101001  010001  000101  001001
101000  010001  000101  100001
100001  010001  010100  000101
```

Note that the levels are given bottom-to-top, rather than top-to-bottom, as they appear in the original. The starting point for the maze is the bottom-left corner of the bottom level (coordinate (0, 3, 3), last bit string on the fourth line), while the goal is in the upper-left of the top level (coordinate (4, 0, 0)).

As an example, when the spider is in the upper-left corner of the bottom level of the maze (the first bit string in the input), it can move east or up. You may assume that the input is well-formed: the spider will not be able to leave the maze from the top, bottom, left, right, front, or back, and there are no "one-way" walls. You may also assume that the maze will not contain more than 10 million cells total.

3.2 Output format

Your program should write its output to the file `output.txt`, in the following format. The output should consist of a path from the start to the goal, for each maze. Each solution should be a single line consisting of a sequence of moves, separated by spaces, where each move is a single letter representing the direction of that move: N, E, S, W, U, or D (north, east, south, west, up, or down). Of course, the sequence of moves must solve the corresponding maze described in the input file.

For example, if your first four moves take you 2 spaces right and down on the bottom level, then rise to second-lowest level, your output should begin with `E E S U`.

You are welcome to try figuring out the solution to the "Itsy Bitsy Spider" puzzle on your own, but that won't get you any points. Your assignment is to model the maze as a graph and to solve the problem using an appropriate graph algorithm.

4 Example

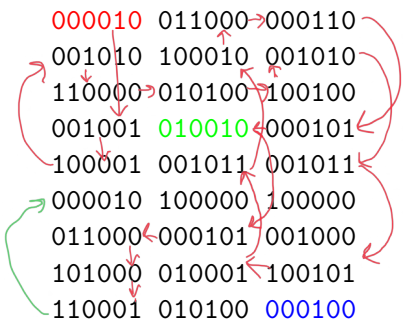
Consider the $3 \times 3 \times 3$ maze whose layout is represented by the ASCII art below (bottom level first):

```
+-+---+
|#|# #|
+-+ +-+
|#|#|#|
| +-+ |
|# # #|
+-----+

+-+---+
| |#  |
| +-+-+
| | | |
+-+ | |
|#|#|#|
+-+-+-+

+---+-+
|#  |#|
| +-+ |
|#|   |
| +---+
|  # #|
+-----+
```

In this representation, the walls are shown as `-`, `|`, and `+`, while the floors are represented as `#`. Assuming the starting point is the upper-left corner of the bottom (first) level, and the goal is the bottom-right of the top (final) level, the input describing this maze is given by:



The solution to this maze is:

```
U S D S E E N U U W D D N E U W U W S S E E
```

5 Submission

You must submit a zip archive containing 1) your report (described in Section 2) as a PDF document, 2) your code (described in Section 3), and 3) a README file describing how to compile and run your code to Canvas. If your code requires more than a simple command to compile and run then you must also provide a Makefile and/or shell script. A simple command might be something like:

```
g++ *.cpp -o maze
```

If you are using Boost in your solution, you must provide a Makefile and/or shell script that uses the environment variable `$BOOST_HOME` (pointing to the Boost installation directory) to compile your code.

As this is an *individual* project, your project report and code will be checked for plagiarism.

6 Grading

Report	**50 points**
Graph model	30
Algorithm description	20
Code	**50 points**
README file	5
Follows input and output specs	10
Compiles and is correct	30
Good coding style	5

Note: if your code is unreasonably slow, you will lose points for both your algorithm design and your correct output grade.