Practical Task 4.3

(Credit Task)

Submission deadline: Wednesday, April 16
Discussion deadline: Friday, May 9

Instructions

Did you see Mario's black cat? He lost the cat on a construction site at Deakin University. The construction site is a kind of rectangular maze divided into R rows by C columns of cells. Both rows and columns are numbered starting from 0. Each cell is labelled via a binary number: 0 indicates an empty cell, and 1 represents a section of wooden wall. This maze (and mess) is surrounded by additional walls that are not taken into consideration. Last time the cat was seen in the cell [R-1, C-1].

Help Mario to rescue the cat by developing and programming an algorithmic solution to the problem. You must start in the cell [0, 0] and your goal is to reach the cell [R-1, C-1]. In each step, you can move from your current cell to one of the horizontally or vertically adjacent cells. Of course, you cannot walk into a wall. Achieving your goal is currently impossible. However, to reach the cat in the cell [R-1, C-1], you are allowed to break one of the wooden walls that is horizontally or vertically adjacent to your current cell. Note that you may ***break exactly one wall*** on the construction site.

As the answer to this task, find all walls such that if you can break ***exactly one of them***, it will be possible to reach the cell [R-1, C-1]. Return ***the number of such walls***.

1. Use the source code attached to this task. Create a new project and import the three files. Your newly built project should compile without errors. Note that the Tester.cs file provides you with a Main method as the starting point of your program and is important for the purpose of debugging and testing. Another file, TestGenerator.cs, serves the Tester class in Tester.cs with hard-coded test instances. Finally, FindTheCat.cs contains a template for the algorithm that you need to develop and implement as part of the FindTheCat class.
2. Explore the existing FindTheCat class to find the static method named Solve. This method must implement your algorithm. It accepts only a two-dimensional array of integers, with each cell of the array being a binary number, i.e., either 0 or 1. The two cells of the array, [0,0] and [R-1, C-1] that represent the start

and the destination, respectively, will be zero. Both R and C will be between 2 and 100 inclusively. It will not be possible to walk from [0, 0] to [R-1, C-1] straightaway.

3. Consider the following examples:
   - For a given input matrix

$$\begin{matrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{matrix}$$

     the algorithm must return 3. Here, Mario can break any of the three walls.
   - For a given input matrix

$$\begin{matrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{matrix}$$

     the algorithm must return 1. This time, there is just one possible choice: the wall in row 2, column 3.
   - For a given input matrix

$$\begin{matrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{matrix}$$

     the algorithm must return 0.
   - For a given input matrix

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 \\ \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

     the algorithm must return 6.
4. Remember that you are free to write your program code within the FindTheCat.cs as you wish. The only requirement is to meet the expected signature of the Solve method. Therefore, you may add any extra private methods and attributes if necessary. Remember that computer memory is another important resource that you have access to, and its use can reduce algorithm's complexity and simplify its implementation.
5. As you progress with the implementation of your algorithm, you should start using the Tester class in order to test your code for potential logical issues and runtime errors. This (testing) part of the task is as important as coding.
6. Finally, note that cheating is not allowed; that is, it is unacceptable to simply return the expected answers of a test instance you are given. Such illegal solutions will be marked as failed.

Further Notes

- What should help you to develop an algorithm to this problem is Lecture 4, particularly its "Blob Check" example.

Submission Instructions and Marking Process

To get your task completed, you must finish the following steps strictly on time.

- Make sure your programs implement the required functionality. They must compile, have no runtime errors, and pass all test cases of the task. Programs causing compilation or runtime errors will not be accepted as a solution. You need to test your programs thoroughly before submission. Think about potential errors where your programs might fail.
- **Submit** the expected code files as a solution to the task via OnTrack submission system.
- Once your solution is accepted by your tutor, you will be invited to **continue its discussion and answer relevant theoretical questions through a face-to-face interview**. Specifically, you will need to meet with the tutor to demonstrate and discuss the solution in one of the dedicated practical sessions (run online via MS Teams for online students and on-campus for students who selected to join classes at Burwood\Geelong). Please, come prepared so that the class time is used efficiently and fairly for all students in it. Be on time with respect to the specified discussion deadline.
  You will also need to **answer all additional questions** that your tutor may ask you. Questions will cover the lecture notes; so, attending (or watching) the lectures should help you with this **compulsory** discussion part. You should start the discussion as soon as possible as if your answers are wrong, you may have to pass another round, still before the deadline. Use available attempts properly.

Note that we will not accept your solution after **the submission deadline** and will not discuss it after **the discussion deadline**. If you fail one of the deadlines, you fail the task, and this reduces the chance to pass the unit. Unless extended for all students, the deadlines are strict to guarantee smooth and on-time work throughout the unit.

Remember that this is your responsibility to keep track of your progress in the unit that includes checking which tasks have been marked as completed in the OnTrack system by your marking tutor, and which are still to be finalised. When grading your achievements at the end of the unit, we will solely rely on the records of the OnTrack system and feedback provided by your tutor about your overall progress and the quality of your solutions.

Appendix A: Expected Printout

This section displays the printout produced by the attached Tester class, specifically by its Main method. It is based on our solution. The printout is provided here to help with testing your code for potential runtime errors and logical mistakes in your algorithm.

```
Attempting test instance 0 with the expected answer 3 for matrix
0,0,1
0,1,0
1,0,0
 :: SUCCESS (Time elapsed 00:00:00.0035783)

Attempting test instance 1 with the expected answer 1 for matrix
0,0,1,1,0,0
0,0,1,1,0,0
0,0,0,1,0,0
0,0,1,1,0,0
 :: SUCCESS (Time elapsed 00:00:00.0000546)

Attempting test instance 2 with the expected answer 0 for matrix
0,0,1,1,0,0
0,0,1,1,0,0
0,0,1,1,0,0
0,0,1,1,0,0
 :: SUCCESS (Time elapsed 00:00:00.0000264)
```

```
Attempting test instance 3 with the expected answer 6 for matrix
0,0,0,0,0,1,0
0,1,1,1,0,1,0
0,1,0,1,0,0,0
0,1,1,1,0,1,1
0,1,0,1,0,1,0
0,1,0,1,1,1,0
1,1,0,0,0,0,0
 :: SUCCESS (Time elapsed 00:00:00.0000338)

Attempting test instance 4 with the expected answer 2 for matrix
0,1
1,0
 :: SUCCESS (Time elapsed 00:00:00.0002496)

Attempting test instance 5 with the expected answer 5 for matrix
0,0,0,0,0
1,1,1,1,0
0,0,1,0,0
0,1,1,1,1
0,0,0,0,0
 :: SUCCESS (Time elapsed 00:00:00.0007736)

Attempting test instance 6 with the expected answer 6 for matrix
0,0,0,0,0,1,0
0,1,1,1,0,1,0
0,1,0,1,0,0,0
0,1,1,1,0,1,1
0,1,0,1,0,1,0
0,1,0,1,1,1,0
1,1,0,0,0,0,0
 :: SUCCESS (Time elapsed 00:00:00.0001678)

Attempting test instance 7 with the expected answer 42 for matrix
…
 :: SUCCESS (Time elapsed 00:00:00.0047194)

Attempting test instance 8 with the expected answer 4 for matrix
…
 :: SUCCESS (Time elapsed 00:00:00.0011385)

Attempting test instance 9 with the expected answer 11 for matrix
…
 :: SUCCESS (Time elapsed 00:00:00.0021470)

Attempting test instance 10 with the expected answer 7 for matrix
```

```
…
 :: SUCCESS (Time elapsed 00:00:00.0002732)

Attempting test instance 11 with the expected answer 4 for matrix
…
 :: SUCCESS (Time elapsed 00:00:00.0001558)

Summary: 12 tests out of 12 passed
Tests passed (0 to 12): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
Tests failed (0 to 12): none
```