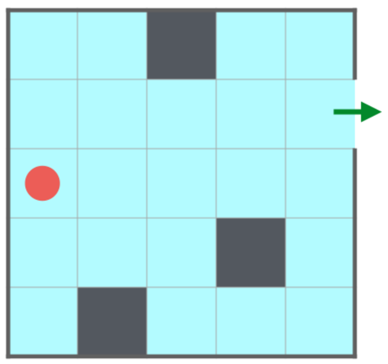
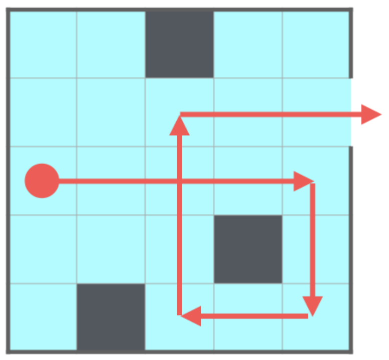
Computational Problem Solving CSCI-603
Get Off the Ice! Lab 9

1 Introduction

It's winter in Rochester, and you have been trapped on a frozen pond! You would like to escape, but since the ice is so slippery, you can only go in straight lines until you run in to a rock or the edge of the pond. To make things worse, there is only one exit from the pond. Of course, there are many different ponds out there that you might get trapped in, so you need a general strategy to escape.

Here (on the left) is one such pond that you might find yourself in, and (on the right) how you could escape — the darkened squares represent rocks in the pond against which you can stop, and the gap in the border of the pond is your escape.

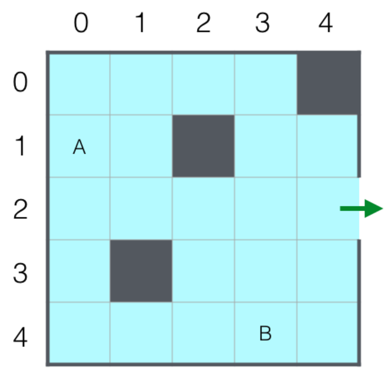




In this assignment, you will be given a pond and must figure out the quickest way to escape the pond from any possible starting location (you never can be too careful. . .), and discover how many places you could start from where you cannot escape at all!

2 Problem Solving

Consider the following frozen pond (ignore the numbers on the outside until question 5):



1. If you start on the square marked A, can you get off the ice? If so, how?

2. If you start on the square marked B, can you get off the ice? If so, how?

3. In this question, we will consider all possible moves starting from the square marked A. On one of the provided copies of this pond, write a 1 in all of the squares that you can stop in after one motion, a 2 in all of the remaining squares that you can reach (and stop in) after two motions, and a 3 in all of the remaining squares that you can reach (and stop in) after three motions. Here a motion means a single move in a straight line from the starting square until running in to a rock or the edge of the pond.

4. On the other provided copy of this pond, write a 1 in each square from which you can escape in one move, and a 2 in each square from which you can escape in two moves.

5. Draw a graph of the squares that you can reach, starting with the square marked A. For this question, represent each square as a vertex labeled with its (x,y) coordinates, and each **complete** move as an edge. Include all edges, being sure to note whether they are directed or undirected. If you need to include the escape point, this should be an appropriately labeled vertex.

3 Implementation

For the implementation portion of this assignment, you will write code that reads in a maze from a file, constructs a graph, finds out from which squares you can escape and for those squares from which you can escape, how quickly you can.

3.1 File format

The pond will be given to you in a file with the following format. The first line consists of three numbers: the height of the pond, the width of the pond, and the row in which the pond can be escaped. The escape point will always be along the right side of the pond, and the row number given will be 0 for the topmost row and increase from there downward. After that, each row of the pond will be given with one character per square — `.` for ice and `*` for rock — separated by spaces. For example, for the first pond given in problem solving, the file would look like this:

```
5 5 1
. . * . .
. . . . .
. . . . .
. . . * .
. * . . .
```

3.2 Internal representation

You should read in the pond and construct a graph (so that it can more efficiently searched many times). Don't forget to include a node for the escape point. You are free (and encouraged) to use the graph code from lecture, with proper citation. It is on the course website under the graph week (labelled with the link `Lecture Code`). You can modify any of the code provided to suit your needs.

3.3 Output

Once you have built your data, we would like to know which ice squares can escape, and for each one, how quickly. Specifically, you should print this information out in order by the number of steps in the shortest escape. For example, the output for the pond given in the file above would look like this:

```
1: [(0, 1), (1, 1), (2, 1), (3, 1), (4, 1)]
2: [(2, 2), (2, 3), (2, 4)]
3: [(0, 3), (1, 3), (3, 4), (4, 4)]
4: [(1, 0), (4, 0), (1, 2), (4, 2), (4, 3)]
5: [(0, 0), (3, 0), (0, 2), (3, 2)]
6: [(0, 4)]
```

If there are starting squares for which you cannot escape the pond, those should appear after the successful squares, in a single line like those given above, except that it starts with `No path:`

3.4 Testing

You must construct at least 3 non-trivial test cases and store them in files named `test1.txt`, `test2.txt` and `test3.txt`. Also include a file `test-description.txt` that explains what each of your test cases is supposed to test.

4 Grading

This assignment will be graded using the following rubric:

- (20%) Problem Solving
- (55%) Functionality:
  - (25%) Graph is built correctly.
  - (10%) Searches are performed correctly.
  - (20%) Output is given in proper format (grouped and ordered by escape-path length)
- (10%) Testing: Provided a good range of test cases with descriptions.
- (5%) Style: Proper commenting of modules, classes and methods (e.g. using doc-string's).

5 Submission

One team member should upload all the source files to your CS account and run the following try command (note this is a single command line):

```
try grd-603 lab9-1 escape.py other-py-files test1.txt
   test2.txt test3.txt test-description.txt
```

6 Grading

6.1 Problem Solving

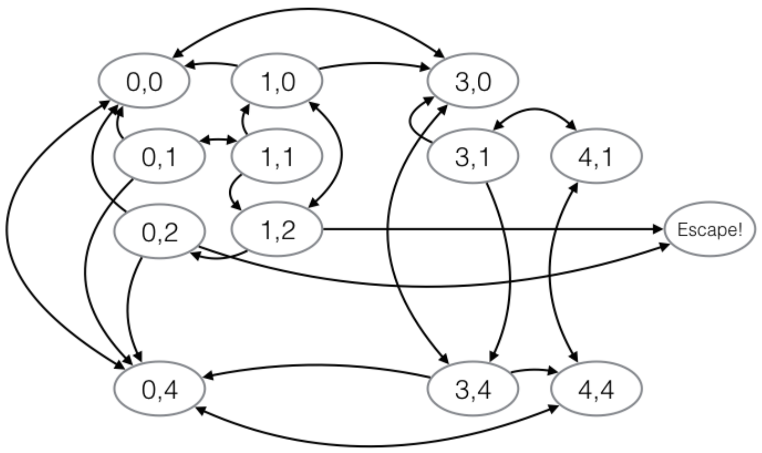
1. Yes - Right, down, right.

2. No

3.
```
1 2 . 2 *
A 1 * . 3
3 2 . . .
. * . . .
1 . . 3 2
```

4.
```
. 2 . . *
. 2 * . .
1 1 1 1 1
. * 2 . .
. . 2 . .
```

5. This is pretty annoying, but...



You can check that each vertex has sufficient outgoing edges, that should verify that it is complete. . .