COSC201 Assignment 1: aMazing Grace

Due: 11:59 p.m. Friday, April 4, 2025

Introduction

Daedalus Pty Ltd® is a company that specialises in the design and construction of large-scale mazes. They have been contracted by the goblin king, Jareth, to build a maze for his castle. You have been employed by Daedalus to complete a simulation of possible maze designs. The mazes will be a square grid of cells with walls between them and an opening at the top left and bottom right corners. The basic task is to construct a grid of arbitrary size, $n \times n$, with walls between the cells decided randomly with a given probability, p, and to determine what proportion of such randomly generated mazes have a path from the top left to the bottom right corner. Whether a path exists can be determined using union-find. Daedelus also wants a report on the efficiency of each of the union-find algorithms that are provided.

Daedalus has provided you with a number of Java classes completed by a previous employee[1] that they insist that you use. These include:

- All the union-find implementations we've looked at.
- A class `Maze.java` that represents mazes of the type we're interested in. This class includes a number of convenience methods for various tasks. For instance:
  - index: the top-left corner has index 0, and the bottom-right corner has index $n \times n - 1$, with indexes increasing along the rows first, and then down the rows. For example, the first cell in row 0 has index 0, the second cell in row 0 has index 1, and the first cell in row 1 has index n etc.
  - identifying all the neighbours of a given cell (by index).
  - checking if a cell has a wall in a given direction, and adding or removing walls.

[1] The previous employee, Icarus, overworked himself and burnt out, and has moved to Queenstown to pursue a career in hang-gliding.

- A template for a `MazeSimulator` class. The constructor for a `MazeSimulator` requires a `Maze` instance. All of its methods refer to this underlying `Maze`. You do not need to change anything in `Maze`, but you can add helper methods if you like. When constructed, `MazeSimulator` generates walls randomly[2] based on the input probability p.
- A class for visualising mazes (for debugging purposes).

Using these classes, you are asked to produce some more code and to conduct and report on some experiments. The main purpose of the assignment is to give you experience with the union-find data structure and to give you practice in conducting experiments to determine the efficiency of specific implementations of data structures.

Code submission (5 points)

Code is to be submitted via altitude (our gitlab server). We will use the code last checked in prior to the deadline as your submission.

The code you submit will be a completion of the `MazeSimulator` template. The purpose of this class is to determine whether there is a path from the top left to the bottom right corner of a given maze. It should do this by using an associated `UnionFind` instance whose underlying elements correspond to the cells of the maze and where two elements belong to the same group if there is a path between them. The methods that you need to complete are marked in the example code given as part of the assignment.

Written submission (5 points)

You report is to be submitted via Blackboard.

There are no formal requirements for the format of your submission, but presentation, spelling and grammar are all important elements which will account for at least 40% of the marks for this part of the assignment. Note that there are strict page limits for the report - exceeding the page limit is an automatic 30% penalty for the report. For text, a font size of at least 11 points for the main body is required.

If you are unsure how to structure your report, you might consider the following outline:

- Introduction: a brief overview of the problem and the purpose of the report.

[2] This is not how mazes are usually constructed, because it can generate disconnected walls and completely enclosed spaces, but this is how we're going to do it.

- Algorithms and Experiments: a description of the union-find algorithms and the experiments you conducted to determine their efficiency). You should also include what you expect the results to be based on the theory of the algorithms (this is your hypothesis).
- Results: a summary of the results of the experiments presented in an appropriate form, such as tables or graphs.
- Discussion and Conclusion: a discussion of the results and a conclusion based on the results. Here you might want to compare the results of the experiments with your hypothesis.

If you refer to anything in your report that is not your own work, you should provide a citation using a standard citation format. This includes webpages, lecture notes, textbooks and any other sources you might use.

For more writing tips, see the following link: `https://www.otago.ac.nz/hedc/students/digital#writing-and-language`.

The page limit for a report without the bonus experiment is 3 sides of A4 including all tables, figures, and references. If you choose to do the bonus experiment, the page limit is 4 sides of A4.

The two experiments you are asked to conduct are described in more detail below as well as a bonus experiment.

Union-find efficiency

In this part of the report you are being asked to determine how the execution time (as given by `System.nanoTime`) varies for the simulation of a certain size of the maze and the probability of a wall. You should conduct experiments to address the issues across a range of maze sizes and probabilities and the available union-find algorithms. The data you collect should be rich enough to determine whether, in this context, the various union-find algorithms seem to scale as expected and which one is most effective in practice.

Maze properties

Using the most efficient algorithm you find from part 1, you should conduct experiments to determine the relationship between the size of the maze and the probability of walls and the likelihood of a path existing between the top left and bottom right corners. Based on your results you should be able to make make a qualitative statement about the relationship between maze size, wall probability, and the likelihood of a path existing. Given such a statement, make a recommendation to Daedalus regarding what size of maze and what wall probability would give a good chance of generating a maze with a solution.

Bonus experiment (2 points)

Implement a method or class for constructing a maze with a solution with as many walls as possible, whilst still being random. As a hint, you might start with a maze with walls everywhere and then remove walls randomly. In this case, keep removing walls until a solution is found. Or you might start with walls nowhere and add walls randomly. In this case, keep adding walls until you can no longer add a wall without disconnecting the start and end cells. Which of these methods is the most efficient? You should report on the method you use, the efficiency of the method, and how many walls are in the maze compared to the other maze generators included in `MazeSimulator`.

Note that the bonus experiment is optional and it means the best mark you can get is 12/10.

Academic Integrity

Everything you submit must be your own work. You may discuss the assignment in general terms with other students, but you must not share code or written work. You can use generative AI to help with proofreading, but you must not use it to generate any part of your submission. You must not use any other sources in your submission without citing them (both code and written work). If you are unsure about what constitutes academic integrity, please ask. See https://cosc201.cspages.otago.ac.nz/academic-integrity/ for more information.