COSC201 Assignment 2: The shortest paths from Kaitaia

Due: 11:59 p.m. Thursday, May 15, 2025

Introduction

Jareth was very impressed with your work on assignment 1, but now he wants to take over New Zealand. To do that, he needs to be able to get around NZ as quickly as possible. Therefore, he has contracted Daedalus to implement Dijkstra's algorithm to find the shortest path between any two locations in NZ. Daedalus has asked you to implement Dijkstra's algorithm for him.

You have been sub-contracted by Daedalus to implement Dijkstra's algorithm and report on the efficiency of the data structure to be used in the algorithm. They have provided three implementations of a priority queue that you must compare. The priority queue implementations are:

- An unsorted array-based priority queue.
- A sorted array-based priority queue.
- A binary heap-based priority queue.

The implementation of the priority queue structures and the graph data structure are already provided and you must use them. The existing code is available at `https://altitude.otago.ac.nz/cosc201public/assignment2`.

Code submission (5 points)

The code you submit will be a completion of the `ShortestPath.java` file. Details of what code you need to complete are given in the file. You may use any resources to implement Dijkstra's algorithm, but it must use the provided priority queue implementations and satisfy all relevant interfaces. If you use any resources (e.g. you base your code on an online description of Dijkstra's algorithm), then you must refer to that resource both in your code and in your report. You should not copy code directly from another source.

Your code should pass the test given in `TestYourCode.java` which is given in the A2 repo.

Written submission (5 points)

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.
- Algorithms and Experiments: a description of Dijkstra's algorithm, the priority queue implementations, the graph representation 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 the results were not what you expected, you should discuss why this might be the case. You should also discuss any limitations of your experiments and suggest possible improvements.

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.

Bonus experiment (2 points)

You are not convinced that any of the provided priority queue implementations are as good as they could be. You know that the Java libraries also provide a priority queue implementation, so you decide to compare the performance of the provided implementations with the Java implementation. You will need to implement a new class called `PQJava` that implements the `PQueue` interface but makes use of the java `PriorityQueue` implementation. PQJava is called an Adapter (a type of programming pattern) because it adapts from an implementation to an interface. You will then need to run the same experiments as you did for the other implementations and report on the results.

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.