IN3063 Programming and Mathematics for AI Coursework Task 1

By Tayo Ajibode-Folkes

Table of Contents

*Implementation*

*Basic Heuristic algorithm*

*Dijkstra’s algorithm*

Analysis

References

Implementation

The aim of task one was to create a grid game based on a grid of random numbers between 0 and 9. The objective was to get from the top left of the grid and find a path to the bottom right of the grid, the length of this path was to be calculated using a basic heuristic algorithm that would be better than random movement, and then using Dijkstra’s shortest path algorithm to find the shortest length. In order to achieve the flexibility asked of me I allowed the player to change aspects of the grid such as height and width as well as the numbers that are randomised within. Changing these altered the parameters that the algorithms worked with.

Basic Heuristic Algorithm

For the basic heuristic algorithm I made a simple algorithm which moved the cell downwards and then to the right, it would do this while under the conditions that:

1. The cell was not located on the bottom row

2. The cell was not located on the right most row

If 1 was false then the algorithm would no longer move the cell downwards and would only move right

If 2 was false then the algorithm would no longer move the cell to the right and would only move downwards

If 1 and 2 were false then that would mean that the cell would be in the target destination and the loop could stop, ending the algorithm.

The path would have not backtracking, as the movement was only down and right with no left or up movement.

The end result is that the algorithm would perform better than a random path.

Dijkstra’s Algorithm

For the Dijkstra algorithm, I knew this would be the more challenging section of task one so I made sure to give myself time for research and execution. I approached the task by transforming the grid into a node weighted graph making each cell would represent a vertex. I then built the Dijkstra algorithm by looping through all the cells of the grid and taking note of the neighbours of each cell and adding them to a queue. Then within the range of the vertices I would compare vertex points and plot the shortest path and calculate the distance, this was achieved with loops, conditional statements (if statements etc.), and queues. Because of some issues with my laptop testing took a long time but through different methods I was able to get it done.

Analysis

The shortest path length increases if the dimensions of the grid increase, this is because the algorithm will have more cells to cover overall. Even if the path from the top left to the bottom right was a straight path if the grid was 500x500 it would still have a larger path length than a 5x5 whose path is very curved. If the size of each of the cell numbers generated was greater, the sum of all the cell numbers in the shortest path would be greater exponentially.

References

I used the lecture videos and notes about Dijkstra’s algorithm to help me plan my methodology and build my Dijkstra algorithm code, I also watched the following videos to give me a better understanding of how to implement and structure my code:

<https://www.youtube.com/watch?v=OrJ004Wid4o>

<https://medium.com/p/a514a9d31076>

I also learned more about using python classes, functions, syntax, etc from the following websites:

<https://www.programiz.com/python-programming/>

https://www.programiz.com/python-programming/function

https://www.programiz.com/python-programming/class-object

<https://www.w3schools.com/python/>

https://www.w3schools.com/python/python\_classes.asp

<https://www.w3schools.com/python/numpy/default.asp>

<https://www.w3schools.com/python/python_functions.asp>

https://www.geeksforgeeks.org/queue-in-python/