



A Study of Comparing and Implementing AI Pathfinding Methods

By Jake Russell

Introduction

Pathfinding is a technique used all around the world, used in Satellite Navigation, video games, self driving cars, etc. There are many different pathfinding methods each better suited at solving one problem over the other. This project attempts to implement and compare pathfinding methods solving a maze.

Aims and Objectives

Implement pathfinding algorithms to solve a maze and assess their capability

The following objectives were placed:

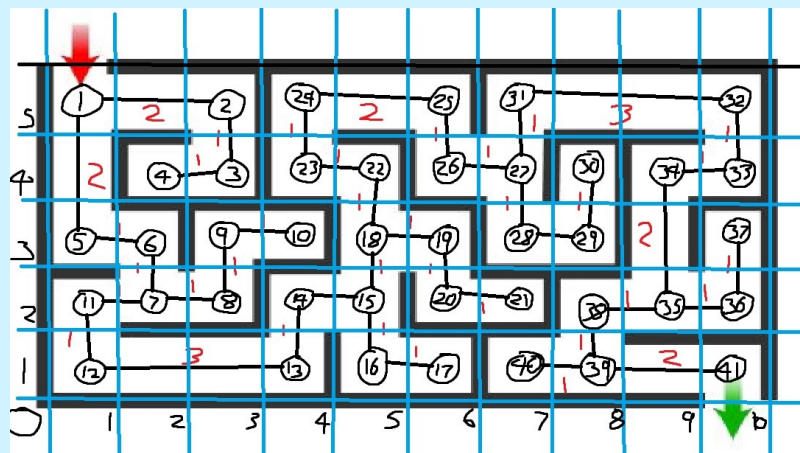
- Learn how to convert a Maze into a format which can be used within code
- Learn about types of graph traversal and search algorithms
- Learn how Dijkstra's and A* star search algorithm function and implement them into code to solve the maze
- Compare how the two algorithms operate when tasked with solving the maze

Results

The Dijkstra's algorithm was successfully implemented and when ran successfully solved the maze, it was capable of returning the shortest path and its distance for every node created in the maze/graph. The A* search implementation however was unsuccessful therefore not enabling a comparison.

Implementation

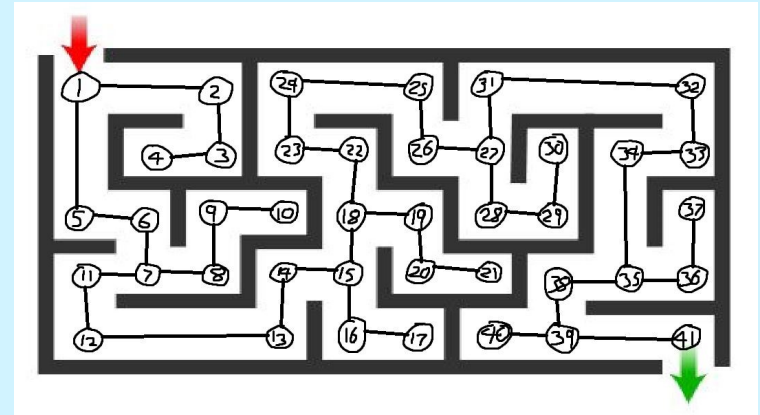
A maze was acquired and converted into a graph, from there it was converted into an adjacency matrix which was able to be used within a C++ coded program. This was enough to implement Dijkstra's algorithm but for A* search, a heuristic was needed. The heuristic chosen was Manhattan distance which requires coordinates, therefore the maze/graph was turned into a grid to give the nodes coordinates.



- Maze/graph turned into a grid

Conclusions

Dijkstra's algorithm was more than capable of solving the maze. It was also capable of solving the shortest path for all nodes, however that does mean that these paths and searched nodes remain in memory till the program is executed, this means Dijkstra's space complexity is not optimal if you only care for a single pair solution. This is where A* would have the advantage. Due to it being informed it attempts to ignore less promising routes in order to reach its goal. A* however does still hold the same time and space complexity in the worst case scenario as Dijkstra's. This is due to it being an extension of Dijkstra's.



- Maze turned into a graph

Recommendations

- Create an automatic maze generation which enable specific sizes to be set, this way the algorithms can be more accurately tested with how to cope with smaller and larger node amounts
- Create a graphical interface that shows the maze being solved in real time, highlight nodes that were visited and then highlight the nodes of the solution in a separate colour, this way there is an easy to see representation on where the algorithm travels for comparison and to showcase to those not already versed in how the algorithms function.

University of Wales Trinity Saint David
Swansea

Email: 1602835@student.uwstd.ac.uk