**Pathfinding - Assignment 1**

**Sean Dornan - B00298673**

**Why I chose to work with A\***

The A\* algorithm is commonly used for graph traversal and pathfinding. The assignment we were given is tasking us to find a path between two nodes. It hasn’t specified whether to find the shortest path, so I’ve chosen A\* to tackle this problem. The Dijkstra’s algorithm is always guaranteed to find the shortest path; however, it is more beneficial to use A\* algorithm over this since it is more efficient and accurate. This is due to the heuristic function it uses to guide the path search –

f (n) = g (n) + h (n)

where:

F is the total cost of the node

G is the distance between the current node and the start node

H is the heuristic – estimated distance from the current node to the end node

It may not always find the shortest path but in my opinion, A\* is the best algorithm to follow as it is more focused on reaching its goal node, causing it to be more efficient than all the other algorithms.

With the Dijkstra algorithm, without testing it, I can already tell that this algorithm will run slower than A\* as it has no focus on reaching its goal. Instead, it chooses to traverse through all the possible paths or grid and eventually decides which is the shortest path once the whole grid has been discovered. Dijkstra essentially does more work than A\* which is why it generally runs slower. A\* with its heuristics, makes it more advanced compared to the other algorithms as it prioritizes its goal, which in result cuts out any paths which do not need to be discovered, making it run faster overall.

**Code review**

The base code structure has been taken from the A\* star cities sample code supplied in Lab 1.

There are several heuristic formulas that can be used for path finding. The base structure already has the Euclidean heuristic formula set up. The Euclidean distance essentially calculates the distance between two points in a straight line. I’ve also decided to add the Manhattan heuristic which has a different return value compared to the Euclidean. Manhattan calculates the distance by summing up the difference between the x and y coordinates of two nodes. Both methods are the most commonly used heuristics for grid path finding. They operate differently, although, they have similarities.

The Euclidean heuristic will usually result in giving you shorter paths compared to Manhattan. However, you may find that it would run slower compared to the Manhattan since the cost function may not always match the heuristic function. Euclidean will generally run slower as it has a more complex calculation due to the function involving more calculations than the Manhattan version. But with the cost of time, the Euclidean generally gives a more accurate result.