A* Search Algorithm

1 Introduction

Finding the shortest path from a starting point to an end point within a weighted graph in the most optimal in reasonable time way has been a tricky problem to solve since the early 1950's.

Intuitively, for humans, we can find a concise path between to points by merely looking at it in a matter of a few seconds. However, when more obstacles or so called 'barriers' are added in between, it might take us longer to find the most optimal path. This is where, we use the efficiency of computers to help us.

2 History of Path-finding Algorithms

Pathfinding or Pathing is the search of an 'optimal' or shortest path between two nodes in a weighted graph. The problem was first pioneered by Edsger W. Djikstra, who developed Dijkstra's algorithm in 1956. New developments in the field were made in 1968 when A* was developed as part of The Shakey Project. A* was a variant of Dijkstra's commonly used in games which addresses some of the setbacks of Dijkstra's algorithm.

3 Algorithms

3.1 Dijkstra's Algorithm

The algorithm begins with a start node and an open set of candidate nodes, i.e nodes that can be added to the open set/visited by the algorithm. At each step, the node with the least distance from the start is examined and marked 'closed' and all nodes adjacent to it are added to the open set to be examined. This process repeats till a shortest distance is found. Since the lowest distance nodes are examined first, the first time the distance is found, is deemed the shortest path.

3.2 A* Search Algorithm

Similar to Dijkstra's, A* begins with a start node and an open set. It can be easily understood by assigning a cost to each path, and finding a path of nodes from the start to the end with the minimum cost. This is achieved by assigning

each node an 'H cost' $h(n)$ through a heuristic distance function and a 'G cost' $g(n)$ which is the cost of travel to from the current node to the next node.

3.2.1 H Cost

The H cost is defined by the minimum distance between the current node and the end node. It is a guess and not the actual distance between the two nodes. Any distance function can be used here such as Euclidean distance and Manhattan distance.

3.2.2 G cost

The G cost is the distance from the current node to the next node.

3.2.3 Working

The algorithm starts by checking for all open nodes and adding them to the open set or priority queue if they are "open" i.e not barriers. For each node, its neighbouring nodes or children nodes are checked and assigned an F cost, which a sum of the G and H cost.

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

The node with the least F cost is noted and the cycle continues. If the F cost for two nodes is the same, then the algorithm checks for the G cost. This method is implemented as A* is an informed search algorithm, in essence the algorithm does not randomly check all paths until it finds the most optimal one, each time it checks the path which may be closer to the end node.

Once the path to the end node is found. The Algorithm reconstructs it using another function. The first time that the shortest path is found is said to be the most optimal hence it reconstructs it.

4 Acknowledgements

All the code has been inspired by 'Tech With Tim' on YouTube. All the content for this explanation has been gathered by through various websites including but not limited to Wikipedia, Brilliant.org, W3Schools, GitHub and The Official Python Documentation.