40323952@live.napier.ac.uk
Bogoslava Dyankova

Artificial Intelligence

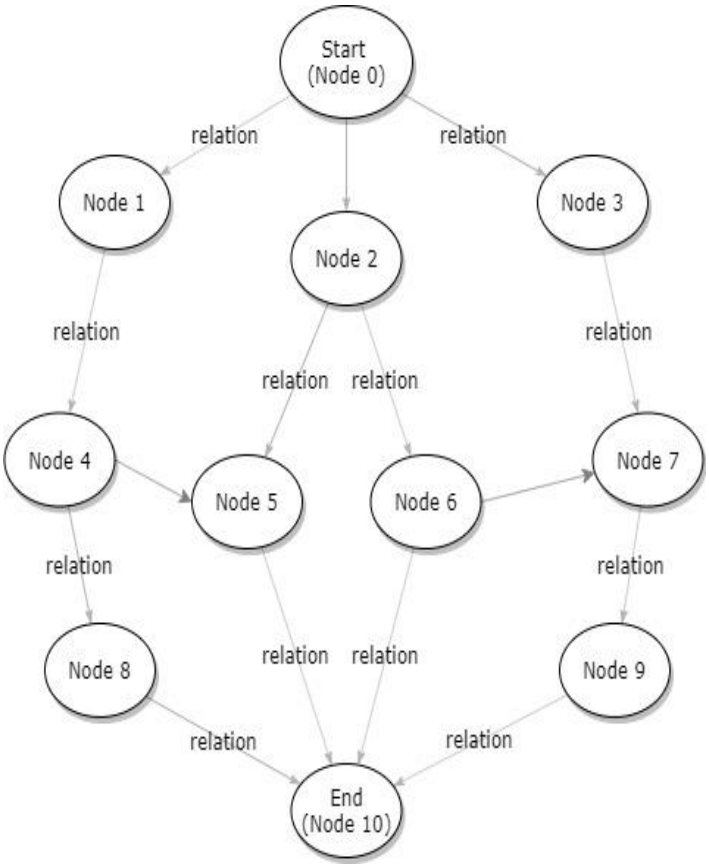
Pathfinding

Question 1:

Dijkstra's search algorithm is an algorithm for finding the shortest paths between nodes in a graph. It was developed by computer scientist Edsger W. Dijkstra in 1956 and published in 1959.

As an example for using this algorithm is having number of caves where we need to find the shortest path from the first cave to the last. The algorithm makes a list of all the unvisited caves and assigns them a value. It starts from the first cave, accepting it as current node, and checks if it has unvisited neighbour nodes. Afterwards, the algorithm gets the Euclidian distances of all the neighbour nodes of the current node, then does the same for their neighbours. The Euclidian distance is calculated by using the equation for a 2D cave, which in this case would be square root of $(x_2 - x_1)^2 + (y_2 - y_1)^2$. Dijkstra's algorithm then moves to the next cave (node), making it current node and putting the previous current node out of the list of unvisited nodes. While this happens, it compares its distance and every other node and compare it to the end node. If that node is not the end node, the algorithm goes back to the list of unvisited nodes and checks them in a sequence. If after this it still has not reached the end node, the algorithm starts checking all the neighbour nodes of the caves which were connected to the first cave.

To explain the algorithm step by step, I used a diagram:



1. Create an empty array/list, representing all the distances from first to last node (0 to 10).
2. Create a list for all unvisited nodes.
3. Set the local and global distance of all unvisited nodes to infinity. This means that for every node with a value of infinity, no path has been found yet.
4. Set start node as current node.
5. Calculate the Euclidian distance for all nodes and check which nodes are related to the current node.
6. Remove Start node from the list of unvisited nodes and choose the neighbour node with shortest distance calculated to end (i.e. Node 1)
7. If the current node is not the last node, remove it from the unvisited nodes list and go to the next node.
8. When the algorithm has reached the last node, it checks if the unvisited nodes list is empty. If not, it selects the node from the list with the smallest Euclidian distance calculated and does the same process until the list is empty.
9. After the list is empty, check which the fastest route is and pick it.

A positive aspect of Dijkstra's algorithm is that it is an uninformed algorithm, meaning it does not need to know the target node beforehand. Because of that, it is unsurpassed in conditions where there isn't given any previous information of the map, therefore the distance between each node and the target cannot be determined.

40323952@live.napier.ac.uk
Bogoslava Dyankova
The algorithm is very useful when there are multiple target nodes as it covers the map in considerable range. This is particularly handy when the closest nodes to the targets are unknown.

A negative aspect of the Dijkstra's search algorithm is the time it takes to process since it examines all possible solutions at the same time. Using the big-O notation, the algorithm takes the time of $O(|V|^2)$ which means that a file of 10 000 nodes, the algorithm would go through a total of 100 million possible connections. Therefore, the algorithm will perform poorly since it would use a huge amount of resources if the search parameters are too big.

Question 2:

Another good algorithm to use for path finding is the A-Star search algorithm. It was created as a project with the objective of building a mobile robot which could plan its own actions.

The A-star algorithm is very useful when it comes to path finding as it can be morphed into another search algorithm just by changing the heuristics it works with and how it checks the nodes. The algorithm is complete, which implies that it will always discover a path if it exists. Writing my program using this algorithm also made me see that it makes a node its current only if it seems promising. The algorithm's goal is to reach the last node (or the target node) from the current node as quickly as possible, unlike Dijkstra's algorithm which tries to reach every node.

I decided to use this algorithm in my program because it is straight-forward, easy to apply and very fast to process when there is a single target. Because of the way it can be implemented, the algorithm requires a very small amount of memory and time, therefore it is more efficient. Compared to Dijkstra's algorithm, A-star will search through 10 thousand connections and not 100 million. The algorithm's worst case scenario is having the same efficiency as Dijkstra's.

However, the A-star algorithm is not perfect as well. If there are many target nodes, the algorithm will stop after reaching the first one. Therefore, it needs to be run several times in order to reach all of the target nodes.

If the specification of the task wanted to go through a particular tunnel in a particular direction, the A-star algorithm would be modified so that it has to go through certain nodes instead of looking for the fastest route possible. This would not be hard to implement as the algorithm is very flexible when it comes to modifying its heuristics and the method in which the nodes are checked.

Resources:

Description and usage of the Dijkstra algorithm:

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

https://www.instructables.com/id/Dijkstras-Algorithm-in-Simple-Steps/

History of A-star search algorithm:

https://en.wikipedia.org/wiki/A*_search_algorithm#History