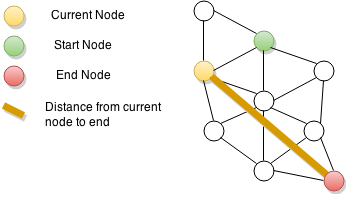
Exercise – A\*

In the Dijkstra’s Shortest Path exercise we added a gScore value and had the open list sorted by gScore. A\* (a-star) is a quick and simple modification to Dijkstra’s algorithm. When sorting the open list we will prioritise nodes that are assumed to potentially be the shortest path by using a heuristic combined with the current gScore, ordering the open list by the combined score rather than just the gScore.

Unlike Dijkstra’s algorithm, A\* can only pathfind to a single known end location. Using this end location we can calculate a heuristic (hScore) from the current node being processed while executing the search.

Heuristics are just an estimate, and there are many ways to calculate this estimate. One common way is usually the squared Cartesian distance from the current node to the target node.



The open list should be sorted by the "fScore" which is another variable stored within the node. The fScore stands for "Final Score" which is calculated by adding the node’s "gScore" and "hScore" together.

You'll need to update the Node structure to store the hScore and fScore.

struct Node {

Vector2 position;

float gScore;

**float hScore;**

**float fScore;**

Node\* previous;

std::vector< Edge > connections;

};

Let's take a look at the altered pseudo code. You'll notice the highlighted lines are those that have changed from the Dijkstras algorithm.

|  |
| --- |
| Procedure aStarSearch(startNode, endNode)  Let openList be a List of Nodes  Let closedList be a List of Nodes  Set startNode.previous to null  Add startNode to openList  While openList is not empty  Sort openList by Node.fScore  Let currentNode = first item in openList  // Process the node, do what you want with it. EG:  if currentNode is endNode  break out of loop  remove currentNode from openList  Add currentNode to closedList  for all Edges e in currentNode.connections  Let n = e.target  Add n to openList if not in closedList  n.gScore = currentNode.gScore + e.cost  n.hScore = distance from n to endNode  n.fScore = n.gScore + n.hScore  n.previous = currentNode  // Calculate Path  Let path be a List of Vector2  Let currentNode = endNode;  While currentNode is not null  Add currentNode.position to path  currentNode = currentNode.parent  Return path |

Implement the above pseudo code with the changes for the added hScore and fScore.

# Additional Exercise – Heuristics

Modify your aStarSearch to enable the use of custom heuristics. This could be achieved by passing in a function pointer to the function, or a similar method.

This custom heuristic function would take in 2 nodes and return the estimated heuristic.