Algorithm Chosen

The algorithm chosen for this assignment was A\*. One of the reasons for this choice is that it is relatively simple to understand and implement. It was chosen over Dijkstra's Algorithm as, while it will definitely find the shortest path, A\* will only explore for one path to one specified goal, making it more efficient for this assignment. A\* will find the shortest path provided the heuristic is admissible.

It was chosen over Greedy Breadth First search as Greedy Breadth First search will not be guaranteed to find the shortest path.

Algorithm Explanation

//heuristic chosen - admissible and not manhattan

//

Data Structures Used

The Graph class is constructed with the provided dot file (renamed to "graphdata.dot"). The parseGraph method is not very extendable to other dot files as the first three lines will need to stay the same. That said, it should allow for varying node and position amounts. It parses the dot file using ifstream and stringstream.

Nodes are structs instead of simple integers as it allows a way for the priority queue return the most promising node to be investigated. The less than and greater than operators were overloaded to allow the priority queue to determine this.

Unordered maps in the Graph class were used instead of vectors as the Edge and Node structs did not hold costs and Positions, respectively. It was easier to retrieve specific edge costs and Positions than to iterate over vectors.

Hashers were created to allow the Edge and Node structs to be used as keys in unordered maps.

Unordered maps in the algorithm to store how the algorithm traverses through the graph and to store the cost the algorithm took to get to the nodes that were visited.

came\_from[Node, Node] means that the algorithm came from the value Node to get to the key Node.

Unordered maps also allowed using the [] operators to easily add a new entry for a key with a default value.

A priority queue is used for the frontier as it has a method to quickly return the most promising node. Nodes have a priority based on the total cost to get to that node and the estimated distance to the goal.