Name: Ali Nawaz Maan
Student ID: 44688813 Email: a.maan@uqconnect.com.au
Course code: COMP7702

Navigation App Report

(AI Assignment 1)

1 What is the formal definition of the navigational agent in this assignment?

In this assignment, environment is a navigational map with sequence of houses, roads and junctions connecting the roads. The agent is a navigational agent looking for shortest path from a starting house to a goal house. The environment is translated into house groups. Houses on either side of the road are considered one group and treated as a node. Distance between neighbouring houses and junctions are defined in terms of edges and length of each edge.

1.1 Action Space

The agent can move to its neighbouring nodes from a given node connected by its edges. In this particular problem, house group nodes can only have two adjacent houses or neighbours or junctions. A junction can have multiple neighbours and agent can move towards any of the attached houses and roads.

1.2 Percept Space

A string representation of state space. It contains Road name, starting junction, ending junction, road length and number of houses or plots on the road – in that order.

1.3 State Space

The states of nodes in terms of distance from starting point and if it is visited or not. Also, all the neighbouring nodes and their edge length is also included in the state space.

1.4 World Dynamics

Change of distance of house nodes from starting point as a new distance is calculated for every visiting neighbour house nodes combined with edge length of the neighbouring houses. Every time the agent moves to the next house, its neighbours' distance from source gets updated and the current house is marked as visited. This allows us to keep track of the minimum cost and the nodes which are already visited.

1.5 Perception Function

All the neighbouring houses connected to the current house node is included in the percept function as the agent can visit all the connected nodes at any given point – obviously based on some criteria defined in the algorithm.

1.6 Utility Function

Success when the agent reaches the goal house and finding the shortest path and failure if the agent is unable to reach the goal house from the starting house – meaning in an unconnected map/graph.

Name: Ali Nawaz Maan
Student ID: 44688813 Email: a.maan@uqconnect.com.au
Course code: COMP7702

2 What type of agent is the navigational agent as described in the previous question?

2.1 Discrete vs Continuous

The environment is discrete as the map is divided into house nodes and edges connecting them. So, the percept, state and action spaces are all countable sets. Hence, it is discrete. This is also true for the fact that the agent can only travel from one house node to another or to a junction. It cannot stay in the middle of an edge. This characteristic makes the agent discrete.

2.2 Fully vs partially observable

The problem is fully observable as the agent has all the information for the world itself and there is no ambiguity in the meaning of house nodes, distance from source and edge lengths.

2.3 Deterministic vs Non-deterministic

The problem here is deterministic as the navigational agent always know the state after performing a certain action. For example, if the agent is visiting a house group node, it will know exactly its next state based on the information from its neighbours and edge lengths.

2.4 Static vs Dynamic

The problem is static as once a map is generated, it will not change automatically while the agent is thinking. For example, if an agent is visiting a house group node, all the other nodes and edges will stay as they are.

3 Please describe the method that the agent uses to solve the problem.

The agent uses Uniform cost search to solve the problem. As the map is a weighted graph in terms of road lengths and distance between two houses, the best way to navigate to shortest path to goal is by accounting costs.
The step by step process for implemented method is described below.

3.1 Step 1 - Reading the environment file

When the environment file is passed into the program, it reads that and generate all the edges connecting house group nodes and junctions. Houses are grouped together in a way that houses on either side of the road where traveling cost is zero are grouped together as one node. Moreover, the house group nodes will be indexed as the first junction on first line of environment file as 0 and house group wherever it is as last index. So, each edge will be initialized as to and from node names and indices and the edge length.

3.2 Step 2 – Making all the house group nodes and assigning edges to each node

After initializing the environment file and making all the edges, these edges are passed into the main Graph class. The constructor generates all the nodes based on the maximum index of the nodes. Afterwards, each edge is added to 2 nodes – origin node and destination node.

Name: Ali Nawaz Maan
Student ID: 44688813 Email: a.maan@uqconnect.com.au
Course code: COMP7702

3.3 Step 3 – Reading the query file

After initializing the edges and nodes, query file is read to process each query. As the house group nodes are a combination of 2 houses, the "getShortestPath" method will find the starting and goal nodes by looking at the node houses and comparing those with queries.

3.4 Step 4 – Finding the shortest path

The idea is to implement Uniform Cost Search algorithm for finding the shortest path from a starting house to goal house in this weighted navigational map. The agent starts at the goal house group node. Neighbouring house group nodes are handled through a priority queue. A comparator is used to arrange the elements in the priority queue. The comparator is based on each node's distance from starting point.

At start, each house group node has distance from source at infinity or Integer maximum value in Java. When a node is expanded, it is marked as visited and all of its neighbours explored. For each neighbour connected by an edge, a new distance is calculated as the sum of its edge length and expanded node's distance from starting point. If the neighbour's distance from starting point is greater than the new distance, neighbour's distance from starting point gets updated as new distance.

Each neighbour is then added to the priority queue if it is not previously visited and the visited node's previous node to path is updated as its previous shortest distanced node to keep track of the shortest path. Priority queue handles the ordering and gives the shortest distanced node on the next iteration of the loop. When a node is expanded, it is checked if it is the goal house and the loop breaks updating goal house node the distance from starting point as minimum.

3.5 Step 5 – Getting the result

Shortest path method does not return anything. It just traverses the map and updates the expanded node with previous node in path and the distance from starting point. Another method related to get results picks up the goal node and gets its distance from the starting point. It then traverses back to each node using previous node in path and pushes the node name into a stack.

String builder is used to build a resultant string by putting the path length first and popping and appending remaining path names.

Fileprocessor method is called to write the result in the output file as specified.

4 Does the method use heuristics? Explain why or why not the heuristics have been used in this method.

Heuristics are used in A* search to guide the agent's search decisions and help navigate the agent towards optimal path without compromising the efficiency. In the given problem, we have to navigate the agent from a starting house point to the goal house through roads. It is an undirected weighted graph and the weights depend on road length and number of houses on that road. Now in order to calculate heuristics values, we need to know some relationship between start house and goal house. This relationship can be in the form of road or junctions connecting them. In this particular problem, if we want to assign heuristics, we must know this relationship beforehand.

Unfortunately, it is only feasible to calculate the heuristics and assign to each node if the edge weights vary significantly and there is some direct connection between start and goal

node. We could assign lower heuristic score to starting point neighbours on the same road and higher uniform values to all other nodes. But this approach will provide no greater benefit than Uniform Cost search as assigning the heuristic value will contribute towards a CPU operation and Java's garbage collector might take load. Of course, there are ways to make this efficient but the overall value addition is not greater than uniform cost search algorithm. The operation cost of Heuristics assignment is allocated in making the actual algorithm fast and parsing the file in an efficient way. My implementation of uniform cost search is efficient enough to outweigh the heuristics search or A* search.
The heaviest operation in the entire program is parsing the file and making house edge objects. In my analysis, heuristics will make the program more processor intensive and will slow down the performance.

5 Identifying unsolvable query

If a query is unsolvable, it means that the graph is unconnected. This ultimately result in dividing graph into different portions. One portion would contain starting house node and the other will contain the goal house node. My implementation of the uniform cost search will only search starting house portion and will neglect any disconnected part altogether. The algorithm will visit each and every node in order to find the goal house node in the portion containing starting house node and will result in unsolvable after finishing the search. Although the search is being performed on a portion of a graph, the time complexity is still O(nlogn). It is not linear because for every visiting node, the algorithm has to see its neighbouring nodes and decide the new distance for each.
Algorithm checks if the node's distance from staring point has been modified or not. If the distance is not modified during the search, this means that the agent has not been able to reach the goal node and the graph is disconnected. This accounts for linear time complexity of algorithm.

The algorithm itself for finding the shortest path has space complexity of O(n) where n is the number of houses and junctions. However, reading the file and generating objects in memory has exponential space complexity. This is because my implementation relies heavily on list of stored objects and in order to do that, I had to generate edge objects in a loop. This problem can be avoided by reducing the number of generated objects. This can be achieved if we can only rely on junction nodes and edges connecting those. This will return in the list of objects equal to junctions in the environment and edges connecting those. The logic of algorithm has to be changed to identify correct junction and traversing through its neighbours and identifying road name to identify houses and distance between those.

6 Executing the query

The zip file is provided with compiled jar file and source code necessary to compile with ant. Build.xml is included in the source files. Please compile using ant and then run the jar file using " java –jar a1-7702-44688813.jar <environment_file.txt> <query_file.txt> <output_file.txt>"