FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING

ENCS3340

ARTIFICIAL INTELLIGENCE

Project#1

Search

Name: Firas Romaneh. ID#: 1181088.

Name: Bara Al-jabale. ID#: 1180816.

Instructor: Adnan Yahya.

Section: 2.

❖ Depth First:

Our design was to get a start node and set of goals and get the first goal found.

First, we make a loop of the 50 which is the limit if we don’t find any goal to prevent it to enter a cycle.

Inside each loop, we do this.

First, we get the nodes that connected to the start point and add them as children to the first parent (start goal) and push the start node to the solution stack then check if the node in the stack is one of the goal if it is, it will break out the loop and print the solution path else it will continue the loop.

Second, we make a loop for the parent-children (the parent that we add into the stack) and set the parent for all child nodes. The parent is the current node we use. Why? To get the solution path in the end and then check if the node has a parent and does not have the parent id to add it, we sort the array list of the children by id also (alphabetically) to connect the nodes with the graph.

However, there are problems with this algorithm which that this algorithm is not complete, so if the start has no way to the goal or enter a loop, it will print that there is no path found and set the path cost to zero, and then we continue to the remaining goals.

❖ A* algorithm:

Our design was to get a start node and set of goals and get the shortest path from the set of goals, but in this case, the cost is based on the actual travel distance and the areal distance (heuristic).

First, we make a loop to run over all the goals selected to get the shortest path between them.

Inside each loop, we make the heuristic function which means we set the Arial distance for all cities based on the goal (the Arial distance must be between each city and the goal city each time). We define priority queue because one of its properties that the first node we poll from the priority queue will be that node with the shortest path (accurate distance + Arial distance). Then we make a visited priority queue to make sure that we do not add a node that we already add before. Then, we get to make a loop for the parent-children and set the parent for all child nodes, the parent which is the current node. We add it to the queue we repeat this process until we reach the goal.

Then we move into the next goal city and repeat what we have done for the first city, and after going for all the cities, we get the lowest cost from the goals and print it as a solution.

❖ Optimal 2 algorithm:

Our design was to get a start node and set of goals and get the shortest path with the lowest time consumed to go through all the goal points.

First, we define alpha as a congestion factor that will either increase the time consumed or decrease it, which depends on how much there is congestion on the roads.

Our idea is to get the closest goal from the start point, then make the road into it, then find the next closest goal from the first goal found and repeat this operation until we reach the last goal.

First, we make a loop that will run overall goal nodes. We then define a priority queue to get the closest value to the start when we poll the node the same work in the A* algorithm but depending on the travel time and without the heuristic function.

❖ How it works:

First, you choose which algorithm you want to set
Then press at set start city and choose the starting point and save it if you want to save this start or cancel this city
Then press at set goal city and choose either one point or multiple points then either you save this choice or not
After that, press the run button to display the road (if you choose optimal 2 algorithm another scene will display to choose the time of the driving time to choose alpha or to choose it randomly)

Things we add to the Project:

1. We add a “listen to the road” button in Arabic and English that tells you the road you have to go to reach the goal. We enable this thing using the “jlayer” jar that we add, enabling us to use the device's output like the sound. We create two functions, one for Arabic and one for English. Each of them takes the event from the button when pressed and out the path as a voice, which we made it using a loop to get all the cities in the path. Note: you have to install the “jlayer” jar to enable this function.
2. A line that will appear from the start city to through the whole path reaching the goal we enable this thing by using a function that gets the city and the next city in the path and get the (X, Y) Coordinates of the radio button and make a line between them using the line in JavaFX.
3. A numbering system that will tell you which city you have to go first and second until reaching the goal we enable this thing by printing the number which increase each time we move to another city by taking the (X, Y) Coordinates of the radio button and print it as a label.

4. Additional algorithms: we add three additional algorithms

❖ Breadth First:

Our design was to get a start node and set of goals and get the goal depending on the Breadth algorithm.

First, we make a loop that will run as long as the queue is not empty.

Inside each loop, we do this.

First, we check if the node we visit or not if it is we continue else if the node is the goal node we exit the loop and print the path else we get the nodes that connected to the start point and add them as children to the first parent (start goal) and add the start node to the solution queue then check if the node in the queue is the goal if it is, it will break out the loop and print the solution path else it will continue the loop.

Second, we make a loop for the parent-children (the parent that we add into the queue) and set the parent for all child nodes. The parent is the current node we use. Why? To get the solution path in the end and then check if the node has a parent and does not have the parent id to add it, we sort the array list of the children by id also (alphabetically) to connect the nodes with the graph.

Then all the nodes that we have visit we add them into a second queue to enable them to not add any visited node to avoid any loops that may happen if we add a visited node to the graph again.

❖ Uniform algorithm:

Our design was to get a start node and set of goals and get the shortest path from the set of goals, but in this case, the cost is based on the actual travel distance.

First, we make a loop to run over all the goals selected to get the shortest path between them.

Inside each loop, we define priority queue because one of its properties that the first node we poll from the priority queue will be that node with the shortest path (accurate distance). Then we make a visited priority queue to make sure that we do not add a node that we already add before. Then, we get to make a loop for the parent-children and set the parent for all child nodes, the parent which is the current node. We add it to the queue we repeat this process until we reach the goal.

Then we move into the next goal city and repeat what we have done for the first city, and after going for all the cities, we get the lowest cost from the goals and print it as a solution.

❖ Optimal 1 algorithm:

Our design was to get a start node and set of goals and get the shortest path with the lowest path cost to go through all the goal points.

Our idea is to get the closest goal from the start point, then make the road into it, then find the next closest goal from the first goal found and repeat this operation until we reach the last goal.

First, we make a loop that will run overall goal nodes. We then define a priority queue to get the closest value to the start when we poll the node the same work in the uniform algorithm.