Artificial Intelligence (AIR711s) Assignment 1

Question 1(60%)

Imagine you are tasked with developing the navigation system for a home cleaning robot. The robot operates in a rectangular home environment with furniture and other obstacles scattered around. The robot needs to clean the entire floor efficiently, navigating around obstacles and reaching all areas.

Your Assignment:

1. Modelling the Environment (20%)

- Design a suitable representation for the robot's environment. Consider using a grid-based map where each cell represents a small area of the floor.
- Define states as specific grid cells that the robot can occupy.
- Identify possible actions the robot can take, such as moving forward, backward, turning left, or turning right (considering the robot's movement limitations).

2. Cost Function and Heuristic (20%)

- Define a cost function that assigns a cost value to each action the robot takes. This could be a simple unit cost for each movement or consider factors like the type of terrain (carpet vs. hardwood) or energy expenditure for turns.
- Design a heuristic function h(x) that estimates the remaining distance for the robot to reach the cleaning target (represented by another specific grid cell). A simple option might be the Manhattan distance between the robot's current cell and the target cell, ignoring obstacles. However, explore a more sophisticated heuristic that considers the actual obstacles in the environment, penalising paths that would require the robot to navigate around them extensively.

A Implementation and Testing (50%)

- Implement the A* search algorithm to find the optimal path for the robot to navigate from its starting position to clean every cell in the environment and finally reach the charging station (another designated cell).
- Test your implementation on various environments with different obstacle layouts and target locations.

- Analyse the impact of the chosen heuristic function on the efficiency of the robot's path. Compare the path lengths and total costs with the simple Manhattan distance heuristic vs. the more sophisticated obstacle-aware heuristic.

Visualisation (10%)

- Develop a visualisation tool (text-based or graphical) to display the robot's environment, the planned path using A*, and the robot's cleaning progress.
- Discuss the effectiveness of the A* algorithm for robot pathfinding in this scenario.
- Include considerations for real-world limitations like sensor noise or dynamic obstacles that may require modifications to the base A* approach.

Question 2 (40%)

Tuafeni (You), the travelling salesman, stared at the map of Windhoek sprawled across his desk. Pins marked vibrant places, each a promise of exotic trinkets and potential customers. But a frown creased his brow. How could he visit all these places in the most efficient way possible? Fuel wasn't cheap, and time was money.

He/She needs to visit a set of places in Windhoek exactly once and return to the starting point, minimising the total travel distance. Hill climbing is a search algorithm that can be used to find an acceptable solution for the Travelling Salesman Problem (TSP), even though it might not be the absolute shortest route.

Assignment Objective:

This assignment explores the application of the hill climbing algorithm to solve the TSP. You will implement the algorithm, analyse its performance, and compare it to an optimal solution.

Tasks:

Problem Representation (20%)

- Design a data structure to represent the places and distances between them. This could be an adjacency matrix or a list of city objects with distance attributes.
- Implement a function to calculate the total distance of a given route (visiting order) for all the places.

Here's an example of a distance matrix representing distances between five places (Dorado Park, Khomasdal, Katutura, Eros, Klein Windhoek) measured in Kilometres:

Places	Dorado Park	Khomasdal	Katutura	Eros	Klein Windhoek
Dorado Park	0	7	20	15	12
Khomasdal	10	0	6	14	18
Katutura	20	6	0	15	30
Eros	15	14	25	0	2
Klein Windhoek	12	18	30	2	0

Hill Climbing Algorithm (50%)

- Implement the hill climbing algorithm for the TSP.
- Define a function that generates a random initial route visiting all places.
- Define a function to explore neighbouring solutions. This might involve swapping the order of two randomly chosen places in the current route.
- Implement the core logic of hill climbing: evaluate neighbouring routes, move to a better route if found, and repeat until reaching a local optimum (no improving neighbours).

Analysis and Comparison (20%)

- Analyse the time complexity of your hill climbing implementation.
- Run your hill climbing algorithm on various test cases with different numbers of places.
- Compare the total distance found by your algorithm with the optimal distance. You can calculate the optimal solution exhaustively for a small number of cities or use an existing optimal TSP solver for larger datasets (libraries or online tools).
- Analyse the impact of the number of iterations (number of times you explore neighbours) on the quality of the solution found by hill climbing (distance compared to optimal).

Visualisation (10%)

- Develop a visualisation tool (text-based or graphical) to display a sample city set and the routes generated by your hill climbing algorithm (initial, intermediate, and final).
- Reflect on potential improvements or alternative algorithms for finding better solutions to the TSP. (Explain this during your presentation).

Instructions

Weighting:

- Question 1: 60%
- Question 2: 40%

Group Work:

- This assignment is to be completed in groups of 4-5 students.

Important Dates:

- Closing Date: Sunday, April 7, 2024
- Submission Platform: GitHub or GitLab (individual commits required)

Plagiarism:

- Identical assignments will be considered plagiarism and result in a score of 0 for all involved students.

Submission Instructions:

- Your group's code repository must be hosted on GitHub or GitLab.
- Each group member must make individual commits to the repository.
- A Google Form link will be provided for you to register your group name and members. (Come up with Cool Names).

Collaboration and Communication:

- Use Python or Julia as programming Languages

- This assignment is designed to be completed collaboratively within your group.
- Make sure everyone in your group contributes and understands the concepts involved.
- Feel free to consult online resources or textbooks for further guidance on the implementation and concepts.
- Work effectively with your group members to divide tasks, share code, and discuss results.
- Use comments and clear variable names in your code to enhance readability and collaboration.
- Utilise online communication tools (e.g., video conferencing, project management platforms) to stay on track and facilitate group discussions.

Remember:

- Demonstrate your understanding of AI concepts and algorithms.
- Apply problem-solving skills to design and implement solutions.
- Showcase effective communication and collaboration within your group.

Wish you the best, Good luck!