DEPARTMENT OF COMPUTING AND INFORMATION SYSTEMS
SCHOOL OF ENGINEERING AND TECHNOLOGY

BSC (HONS) COMPUTER SCIENCE

ACADEMIC SESSION: APRIL 2024

CSC3206 ARTIFICIAL INTELLIGENCE

GROUP 11

ASSIGNMENT 2

GROUP MEMBERS:

No	Student ID	Student Name
1	20099107	ADAM LIM HAN JUNG
2	22018519	BONG HONG JUN
3	20072310	CHONG KAR YAN
4	20024626	NIKCO TAN MIN KHE

Table of Contents

1.0 Introduction

In this assignment, the goal is to navigate a virtual agent through a 6x10 hexagonal grid world from a start position to multiple treasure locations while managing energy levels and avoiding traps. This pathfinding and resource management problem is complex due to the various cell types in the grid, including empty spots, obstacles, traps, rewards, and treasure locations. The grid map is visualized as a two-dimensional list where each element specifies the type of cell. For example, `grid[0][0]` is the starting position, `grid[1][4]` is a treasure location, and `grid[2][8]` contains a trap. The unique characteristics of each cell type, such as traps increasing energy cost or rewards decreasing it, add to the complexity of the problem.

2.0 Algorithm Implementation

2.1 Problem Formulation and Implementation Details

2.1.1 State Representation

The virtual environment is structured as a grid where each cell can exhibit diverse types such as ‘obstacle’, ‘treasure’, ‘start’, ‘reward1’, etc. This representation provides a structured layout for navigation and interaction within the environment. Positions within the grid are denoted using (row, column) coordinates, facilitating precise tracking of the agent's current location (current_position) and the positions of treasures (treasures_positions) to be collected. The current state encompasses critical information including the agent's position and the objectives (treasures) yet to be collected, essential for monitoring progress towards achieving the primary goal.

One of the initial and crucial decisions was how to represent the state space of the problem. Given that the environment is a hexagonal grid, traditional grid-based algorithms needed modification to handle the unique movement dynamics. Each state was represented as a tuple (x, y), denoting the coordinates within the grid. This representation simplified the implementation of the transition model and allowed easy calculation of neighboring states.

2.1.2 Cost Function

Initially, the cost function assumes a uniform cost per step for movement between adjacent cells. This simplistic model assigns a fixed cost of 1 unit per step, forming the baseline for pathfinding calculations. While not fully integrated in the provided code, the cost function can potentially be adapted based on encountered traps and rewards. For example, traps might increase energy consumption per step, while rewards could lower movement costs or provide bonuses.

2.1.3 Transition Model

The transition model was designed to accommodate the unique characteristics of a hexagonal grid environment. Unlike traditional square grids, where each cell typically has four neighbors, hexagonal grids present six potential neighboring cells per cell. Navigating this grid requires movements to respect grid boundaries while avoiding obstacles and traps strategically placed throughout the virtual world.

To facilitate efficient pathfinding, we implemented the get_neighbors(pos) function. This function is integral to the A* algorithm's operations as it generates valid neighboring cells based on the six possible movement directions in the hexagonal grid. It ensures that any newly computed positions are within the grid boundaries and do not correspond to obstacles or traps. Traps and rewards influence movement dynamics by modifying movement costs or altering directional choices within the get_neighbors function. Depending on the trap type encountered, movement efficiency may be reduced (e.g., increased energy consumption) or redirected (e.g., forced movement in a specific direction), whereas rewards may enhance movement efficiency or provide strategic advantages. By filtering out invalid movements at this stage, our algorithm focuses solely on feasible transitions, optimizing the exploration and pathfinding processes within the virtual environment.

2.1.4 Heuristic

The heuristic function computes the Octile distance heuristic between two positions within the grid (Björnsson & Halldórsson, 2021). Specifically chosen for its suitability in hexagonal grid-based environments, this heuristic effectively estimates minimal movement costs from the current agent position to the goal (treasure_positions). By guiding the A* algorithm towards paths with lower estimated costs, it optimizes the search for the shortest route to each treasure.

In the code development, the heuristic function, heuristic(a, b) was critical for the performance of the A* algorithm. In a hexagonal grid, the Octile Distance was selected because it effectively estimates the cost of diagonal and straight-line movements between two points (x1, y1) and (x2, y2), which is crucial for a hexagonal grid. The heuristic needed to be both admissible (never overestimate the true cost) and consistent (satisfy the triangle inequality) to ensure optimality and efficiency. The function calculates the maximum and minimum differences in the x and y coordinates to reflect the cost of moving in a hexagonal space accurately.

2.1.5 Goal

The primary objective of the implemented code is to guide an agent through a grid-based environment to collect scattered treasures, with each treasure acting as a distinct goal. The agent must navigate through obstacles, avoid activated traps, and carefully manage movement costs and energy consumption throughout its journey.

The code employs the A* algorithm with an Octile Distance heuristic to find the optimal path from the agent's current position to each designated treasure position sequentially. Within the code, the check to determine if a goal (treasure) has been reached occurs after each successful pathfinding operation using A*. Upon finding a valid path (if path:), the agent follows each step in the path, updating its position and adjusting energy and step costs as necessary based on the grid conditions (such as rewards or traps). Visual feedback through the Pygame display updates dynamically, showing the agent's progress and the grid state in real-time, ensuring the player can monitor the agent's movements and goal achievements throughout the gameplay session. This systematic approach ensures that the agent efficiently collects all treasures while adhering to the specified constraints and optimizing its pathfinding and resource management strategies.

2.3 Algorithm Choice: A* Search

Given the project's requirements, the A* search algorithm is chosen for its efficiency in finding the shortest path. A* combines the advantages of Dijkstra's algorithm and greedy best-first search by using a heuristic to estimate the cost to reach the goal (Russell & Norvig, 2016). The heuristic function employed is the Octile Distance, effective for hexagonal grids. It calculates the distance by considering diagonal movements, thus providing a more accurate estimation compared to Euclidean distance.

The astar(start, goal, grid) function implements the A* algorithm. It maintains an open list of nodes to be explored, a closed list of explored nodes, and tracks the cost to reach each node. The function begins at the start position and expands nodes by calculating the cost of moving to each neighbor, incorporating both the actual cost and the heuristic estimate. The open list is managed using a priority queue to always expand the node with the lowest estimated total cost first. Once the goal is reached, the function reconstructs the path by backtracking from the goal to the start.

2.3 User Interface

To enhance the understanding and validation of the algorithm's performance, the program was integrated with Pygame for visualization. This graphical representation allows us to see the agent's path and interactions with the environment. The draw_grid function is responsible for drawing the hexagonal grid and coloring each cell based on its type. Empty spots are colored white, obstacles black, traps red, rewards green, treasures yellow, and the start position blue. On the other hand, the draw_path function visualizes the path found by the A* algorithm by coloring the cells in the path in orange, making it easy to see the route the agent takes from the start to each treasure. These functions provided a clear and interactive way to observe the algorithm in action, enhancing the understanding and validation of its performance.

2.4 Implementation Summary

In summary, the key considerations in this project were the choice of state representation, the selection of the A* algorithm, the design of an appropriate heuristic function, the implementation of a transition model that accurately reflected the hexagonal grid's properties, and the integration of a dynamic cost function. Additionally, visualizing the results using Pygame was crucial for validating and debugging the algorithm. These decisions collectively ensured that the implementation was robust, efficient, and met the project's requirements comprehensively.

3.0 Results

3.1 Output Format

The simulation presents a dynamic visualization of the virtual treasure hunt, as shown in *Figure 1* and *Figure 2*. The simulation is done by using Pygame, which is a Python library to help easily create 2D games. In this simulation, Pygame is used to draw the hexagonal grid map, path, information table and the buttons. The presentation of the simulation was designed based on the

sample map provided from the coursework question. Hexagonal grids indicate the area where steps can be moved to, purple grid indicates traps, green grids are rewards, grey grids are obstacles, yellows are where the treasure is, and lastly red grid indicates the starting point.

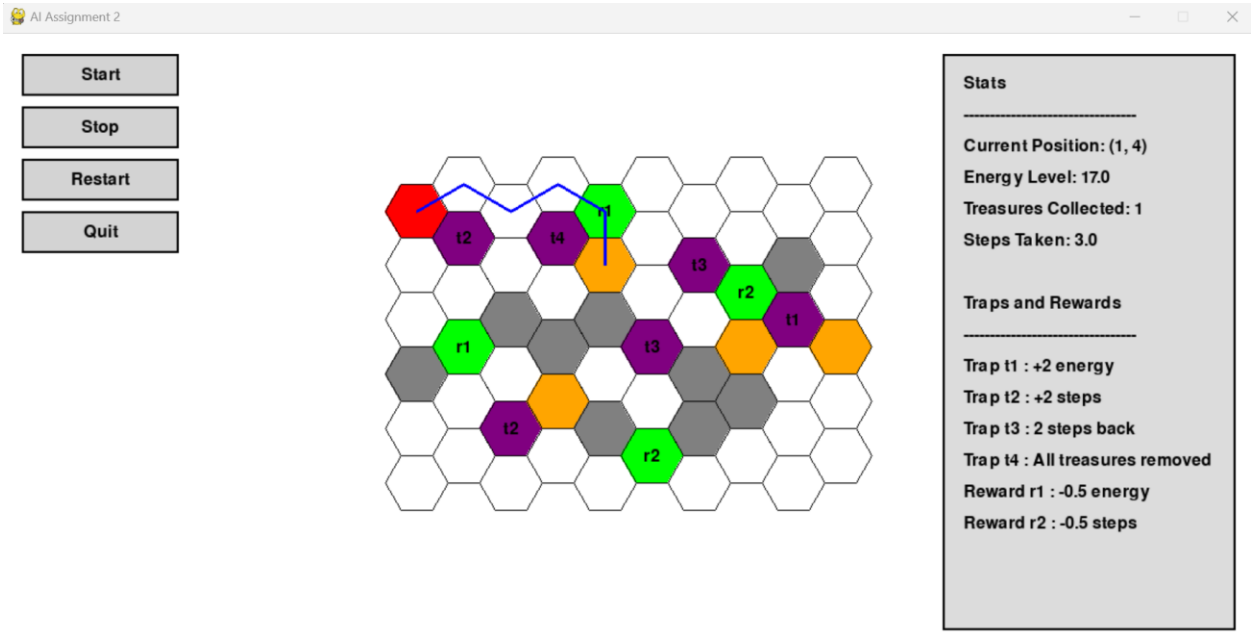

Figure 1 Interface and statistics shown as the first treasure is collected

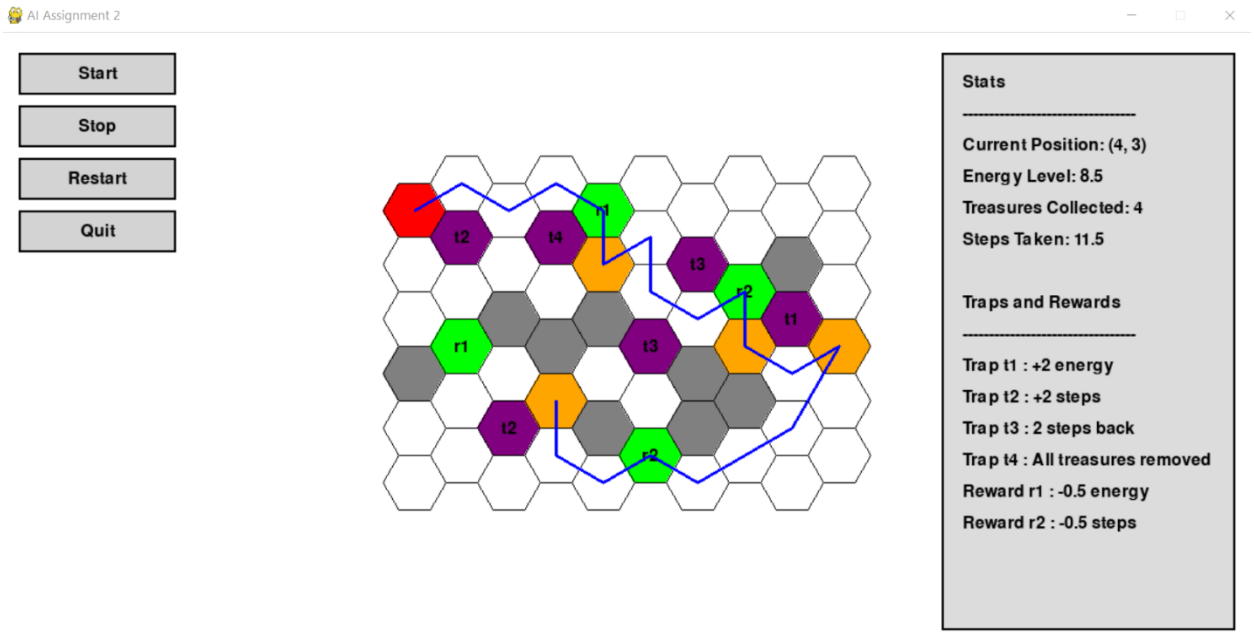

Figure 2 Output visualization - final path taken by agent upon collection of all treasures

The agent's path is depicted with a dynamic line, showing real-time navigation through the grid. The path updates as the agent collects treasures, providing a clear visual representation of the traversal and decision-making process. The steps moved are shown in a blue line indicating the

current position of the step. When the simulation finds a treasure, it highlights the path line in green leading to the current treasure grid.

The right side of the map displays an information table explaining each trap and rewards details and shows the current position of the step, energy level, numbers of treasures collected, and the steps taken to reach the current grid. This table is clear and well-labeled to provide a comprehensive overview of the agent's status and progress. Labels for traps and rewards are also displayed here, helping users understand the impact of these elements on the agent's journey. The labels are clearly marked to enhance the comprehensibility of the traps' and rewards' effects.

The left side of the map shows a few buttons for different features. "Start" is used to begin the simulation for treasure hunting. "Stop" is used to pause the simulation from continuing the treasure hunt, and step will be stop from the current position. "Restart" is used to allow the simulation to start all over again. "Quit" is used to exit the simulation. If "Start" button is pressed the simulation will start the treasure hunt by choosing the best path to find the treasures and will not stop until all the treasure are hunted.

Overall, this simulation demonstrates how effective it is to utilize the A* algorithm in virtual treasure hunts to navigate the hexagonal grid efficiently by finding the shortest path to collect each treasure. The visual presentation of the simulations helps to understand the algorithm's decision-making process and the effect of different functions grid (rewards and traps) on its performance.

3.2 Interpretation

The results of the treasure collection process are displayed in *Table 1*, which provides a detailed account of the agent's performance at various stages. The agen t starts at the Initial State with a position at (0, 0), an energy level of 20.0, zero treasures collected, and no steps taken. This serves as the baseline for evaluating the agent's subsequent movements and actions.

State	**Initial State**	**After T1**	**After T2**	**After T3**	**After T4**
Current Position	(0,0)	(1,4)	(3,7)	(3,9)	(4,3)
Energy Level	20.0	17.0	13.5	12.5	8.5
Treasures Collected	0	1	2	3	4
Steps Taken	0.0	3.0	6.5	7.5	11.5

Table 1 Statistics throughout treasure collection process

Upon collecting the First Treasure (T1), the agent reaches the position (1, 4). At this point, the energy level decreases to 16.5, indicating an energy expenditure of 3.5 units. This reduction can be attributed to the steps taken, as well as reward R1 encountered along the path, which specifically reduces the energy consumption of every step further by 0.5. The agent successfully collects its first treasure and has taken 3.5 steps in total. This early stage demonstrates the agent's ability to navigate the grid, efficiently identify and reach the treasure while managing its energy resources.

As the agent proceeds to collect the Second Treasure (T2), it moves to position (3, 7). Here, the energy level drops further to 13.5, reflecting an additional energy cost of 3.0 units from the previous state. The agent now has two treasures collected and has taken a total of 6.5 steps, after encountering reward R2 that decreases the steps count by 0.5. This stage shows the agent's continued ability to find treasures, although it suggests that the paths taken may be relatively energy-intensive, potentially due to the traps or the specific routes chosen.

In the pursuit of the Third Treasure (T3), the agent reaches position (3, 9), with the energy level reducing to 12.5. This slight decrease of 1.0 unit from the previous stage is due to a relatively short and less costly path in terms of energy. The agent has now collected three treasures and taken 7.5 steps. This stage is critical as it reflects the agent's improved efficiency in finding and reaching treasures, possibly learning from earlier movements or benefiting from favorable grid conditions.

Finally, upon collecting the Fourth Treasure (T4), the agent arrives at position (4, 3). The energy level at this point is 8.5, showing a more significant energy expenditure of 4.0 units compared to the previous state. This is due to the final path to the last treasure being more energy-intensive, after considering collection of reward R2 which reduces the steps count by another 0.5. Overall, the energy consumed to collect all treasures is 11.5. The agent successfully collects all four treasures and has taken a total of 11.5 steps. This final stage demonstrates the agent's overall success in the treasure hunt, balancing energy consumption and steps taken to achieve the goal.

4.0 Evaluation

4.1 Performance Evaluation

Performance evaluation is done by evaluating based on the algorithm’s ability to successfully collect all treasure by choosing the most suitable path. The A* algorithm was able to navigate all 4 treasures on the map while adjusting the presence of traps and rewards as well as minimizing the step costs. As shown in the simulations, the path taken is the fastest and the most efficient to collect all the treasures. The algorithm made accurate calculation by avoiding all traps to save energy and steps costs while considering which rewards can be useful to get the treasure. The results indicate effective pathfinding and goal achievement as expected. Key performance statistics include:

- According to *Figure 1* and *Figure 2*, the energy level drops from the initial level of 20 to 17 after collecting the first treasure, and finally 8.5 as all treasures are collected, reflecting the cost of movement and interaction with rewards, with a 11.5 final energy consumption.
- The total steps taken to collect all treasures was 11.5 as shown in *Figure 2*, which is indicative of the algorithm's efficiency. This metric shows that the algorithm was able to find a path that minimizes unnecessary movements.

Overall, these metrics indicate that the algorithm meets its primary objective of collecting all treasures while effectively managing movement costs and energy consumption. The results are as

expected, demonstrating the algorithm's robustness and reliability. However, a deeper analysis of these metrics could provide insights into areas for potential optimization.

4.2 Analysis of Results

The simulation which uses A* algorithms successfully complete the virtual treasure hunt by avoiding obstacles, considering the rewards and navigating the steps to take. The dynamic presentation of the simulation provides a clear understanding of the grid chosen which shows the accuracy of using the A* algorithm. The dynamic presentation of the simulation and the step-by-step visualization provided insights into the algorithm's accuracy and efficiency. The algorithm's ability to avoid traps and strategically interact with rewards indicates its effectiveness in real-time decision-making. Nevertheless, this might not always be the case in more dynamic environments where conditions change frequently.

In terms of trap and reward management, the Octile Distance heuristic played a crucial role in guiding the agent efficiently across the hexagonal grid. This heuristic considers both diagonal and straight-line distances, which helps in minimizing travel costs and time. However, although the Octile Distance heuristic was effective in guiding the agent, but its performance might vary with different grid configurations or more complex scenarios. The heuristic's balance between travel cost and distance is crucial for efficient pathfinding.

The energy level reduction from 20 to 8.5 is significant. The algorithm demonstrated a balanced approach between conserving energy and minimizing steps. Each step taken had a calculated impact on the energy level, reflecting efficient pathfinding. The total steps taken (11.5) is another a key indicator of the algorithm's efficiency. The steps taken show that the algorithm was able to find a path that minimizes unnecessary movements and optimizes resource usage.

The algorithm also interacts with traps and rewards as expected. The avoidance of all traps including Trap t4, which removes all treasures, indicates that the algorithm successfully navigates the grid to achieve the desired outcomes without unnecessary setbacks.

4.3 Optimization and Desired Outcomes

The algorithm's use of the Octile Distance heuristic is a key factor in its achieving the desired outcomes. This heuristic balances the travel cost and distance, ensuring the agent follows the most efficient path to each treasure (Björnsson & Halldórsson, 2021). The careful management of energy levels and steps, even while encountering traps and rewards, shows the algorithm's ability to optimize resource usage. The energy level reduction from 20 to 8.5, while significant, is still within an acceptable range given the number of traps and rewards encountered.

Optimization is witnessed in the energy and steps metrics. The final energy level of 8.5 and the total steps taken (11.5) to collect 4 treasures provide clear evidence of the algorithm's efficiency. These metrics indicate that the algorithm effectively minimizes unnecessary movements and optimizes the path to conserve energy. Besides, the avoidance of the critical Trap t4 and the

strategic interaction with other traps and rewards illustrate the algorithm's capability to make optimal decisions in real-time.

In terms of real-time adaptation, the algorithm's fixed rules for pathfinding and obstacle management might limit its ability to adapt to unforeseen changes in the environment. Integrating machine learning techniques could enhance its adaptability and performance in dynamic scenarios.

4.4 Limitations and Improvements

4.4.1 Limitations

One of the limitations of the current algorithm is scalability. The A* algorithm's complexity increases with larger grid sizes, leading to potential delays in simulation and reduced performance in real-time visualization. This limitation is significant, especially in environments with frequent movement changes.

Besides, the algorithm's reliance on fixed rules for pathfinding and obstacle management might hinder its ability to adapt to dynamic changes in the environment. This limitation affects the algorithm's robustness in more complex scenarios.

4.4.2 Improvements

Implementing hierarchical pathfinding can help overcome scalability issues by dividing the grid into smaller sub-grids and using A* algorithms to perform pathfinding within these sub-grids. This approach reduces computational load and improves performance.

To enable simulation to adapt unexpected challenges, more machine learning techniques need to be integrated to the system. For example, develop heuristics that refers the past performance and environmental changes to handle unforeseen challenges more effectively.

To make the simple simulation more amusing, dynamic grid elements that change during the simulations can be implemented. This makes the simulation more challenging and interesting for the user to try on. For example, by adding moving obstacles and upgrading the rewards to help to hunt the treasure more easily. This enhancement not only improves the user experience but also provides a better understanding of the algorithm's performance in real-world scenarios.

4.5 Evaluation Summary

Overall, the evaluation of the A* algorithm highlights its strengths in efficient pathfinding, energy conservation, and strategic decision-making. The use of the Octile Distance guides the agent effectively. However, the algorithm's scalability and adaptability to dynamic environments are areas that require improvement. Implementing hierarchical pathfinding and integrating machine learning techniques can address these limitations and enhance the algorithm's performance. Additionally, adding dynamic grid elements can improve user engagement and provide a more comprehensive evaluation of the algorithm's capabilities.

References

Björnsson, Y., & Halldórsson, K. (2021). Improved heuristics for optimal pathfinding on game maps. *Proceedings*, *2*(1), 9–14. https://doi.org/10.1609/aiide.v2i1.18740

Russell, S., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach, eBook, Global Edition. -. https://elibrary.pearson.de/book/99.150005/9781292153971

Appendix 1: Program Flowchart