1 Problem 1

1.1 Problem Analysis

The reflex agent uses the evaluation function to select the best action. The evaluation function returns a higher score for states that are closer to food pellets compares to those which are far, Our task was to implement the evaluation function.

1.2 Solution

The evaluation function takes three parameters including current state and action. After generating the successor, we retrieve the new position of the agent, food positions, state of the ghosts and scared time of the ghosts. The function calculates the distance to the closest food pellet from agent's new position. This serves as a metric and allows Pacman to move towards nearby food pellets. The function checks if any ghost's position collide with the agent's new position. If so, a large negative value as score is returned to discourage such actions and penalize it. If no collision with ghosts is detected, then the function returns a score inversely proportional to the distance of the closest food pellet. This encourages Pacman to move towards nearby food pellets.

1.3 Findings

After using this evaluation function, the results show that Pacman gives priority to behaviors which lead to closest food pellets while avoiding collision with ghosts who are not afraid. Pacman's decision making is strongly influenced by the proximity to food pellets.

2 Problem 2

2.1 Problem Analysis

By weighing possible ghost moves and choosing moves that will maximize Pacman's score, the Minimax algorithm helps Pacman make strategic choices.

2.2 Solution

MinimaxAgent had a number of methods.

Firstly, the value method, which represents a Minimax node in the game tree. It corresponds to a specific state of the game at a certain depth in the tree. At this node, the agent evaluates the current state to determine its value. If the node is a terminal node, the evaluation function is used to determine the node's value. Otherwise, the method recursively explores the successors of this node by calling minValue() or maxValue() depending on whether the agent is minimizing or maximizing.

minValue method corresponds to a Min node in the game tree, where a ghost agent is making a decision. The ghost aims to minimize Pacman's score. It explores each legal action available to the ghost then generates successor states for each action and recursively computes the value of each successor state using the value() method. The minimum value among these successor states is returned, along with the corresponding action.

maxValue method corresponds to a Max node in the game tree, where Pacman is making a decision. At this node, Pacman aims to maximize its score. It explores each legal action available and generates successor states for each action, recursively computes the value of each successor state using the value() method. The maximum value among these successor states is returned, along with the corresponding action.

2.3 Findings

The Minimax algorithm indicates that Pacman navigates through the game tree considering both its own actions and those of the ghost agents. The Pacman maximize its own score and minimize the ghost's scores. Additionally, the Minimax algorithm successfully handles both-terminal states and intermediate states.

3 Problem 3

3.1 Problem Analysis

The Minimax technique is extended by the Alpha-Beta Pruning method. It reduces the number of nodes explored by pruning the branches that are unable to affect the outcome.

3.2 Solution

Similar to MiniMax agent, AlphaBeta agent also has a value method which serves as the entry point for the search tree. It decides whether to call minValue() or maxValue() based on the current agent index. It handles terminal states by checking if the depth limit is reached or if the game is won or lost. If so, it returns the evaluation score of the state and a placeholder action.

The minValue method explores each legal action available and generates successor states for each action. For each successor state, it calls value(). If the obtained value is lower than the current minimum score, it updates the minimum score and corresponding action.Here alpha-beta pruning is applied by checking if the current minimum score is less than the alpha value. If so, the method returns the current minimum score and action without further exploration. This prevents unnecessary exploration of nodes that will not affect the final decision.

The maxValue method is similar to the minValue method. But here it represents the decision-making process of maximizing agents. If the obtained value of successor is higher than the current maximum score, it updates the maximum score and corresponding action. Alpha-beta pruning is applied by checking if the current maximum score is greater than the beta value. If so, the method returns the current maximum score and action.

3.3 Findings

The alpha-beta pruning leads to improvements compared to traditional Minimax search. By pruning branches of the search tree that will not affect the final decision, the algorithm reduces the number of nodes explored. This leads to faster computation and more optimal decision-making

4 Problem 4

4.1 Problem Analysis

A deterministic Minimax or Alpha-Beta agent might not fairly depict an opponent's behavior in situations where the adversaries are probabilistic, like Pacman with random ghosts. The ExpectimaxAgent adjusts to this unpredictable environment by making decisions on the probability of different outcomes.

4.2 Solution

The expValue method corresponds to the Chance nodes in the Expectimax tree, where uncertainty exists due to random events. This method iterates over each legal action available to the chance node and generates successor states for each action. Then computes the value of each successor state using the value() method. The expected score in the Expectimax algorithm is calculated by taking the weighted average of the scores of all possible successor states. The method returns the expected score and a randomly selected action from the available legal actions. It represents the probabilistic nature of the ghost's decision-making.

Thie maxValue method represents the Max node in the game tree, where Pacman is making a decision. It selects the action that leads to the maximum value among possible successor states. For each legal action available to Pacman, it generates successor states and recursively computes the value of each successor state using the value() method. It selects the action that results to the maximum value and returns it along with the corresponding value.

4.3 Findings

The implementation of the Expectimax algorithm introduces a probabilistic element to the decision-making process by calculating the expected value based on the probabilities of different actions.

5 Problem 5

5.1 Problem Analysis

A better Evaluation Function allows Pacman to collect food pellets while avoiding ghosts. It incorporates factors such as Pacman's score, the proximity to food, and the positions of ghosts.

5.2 Solution

The method get Pacman's current position and calculate the Manhattan distances between Pacman's position and each food pellet. It determines the closest distance to a food pellet and update the score by adding a value inversely proportional to the distance to the closest food pellet. This encourage Pacman to move towards nearby food.

Then identify non-scared ghosts and calculate their distances from Pacman and update the score by subtracting a value inversely proportional to the distance to the closest non-scared ghost, discouraging Pacman from approaching them. Similarly, identify scared ghosts and update the score by adding a value inversely proportional to the distance to the closest scared ghost, encouraging Pacman to approach them.

Then check if Pacman's position overlaps with a non-scared ghost. If a collision is detected, return a large negative value to discourage Pacman from colliding with it.

Finally return the calculated score.

5.3 Findings

The evaluation function takes into account various factors such as Pacman's current score, the proximity to food pellets, and the distances to both non-scared and scared ghosts.By assigning positive scores for being close to food pellets and scared ghosts, and negative scores for being close to non-scared ghosts, the function encourages Pacman to prioritize actions that lead to acquiring food while avoiding collisions with ghosts.

6 Challenges

It took me some time to understand completely the Task2. As Task3 and Task4 were quite similarly to Task2 in terms of implementation, unable to interpret it timely had caused me to lag behind in the lab. However, I could understand it after some time and move forward with other tasks.