Project 2 Multi-Agent

29.11.22 Honghong Fang

1: Reflex Agent

Focus on eating the nearest food 'minFoodist' when the ghost is not there, and avoid the ghost if it is close. The distance is calculated using the Manhattan distance, and the final score is getScore() + 1.0/minFoodist.

2: Minimax

Using the template provided by the slides, pacman, as a max player, needs to find the optimal correspondence under the optimal action of each ghost. The core maxLevel and minLevel functions call each other recursively. But there may be more than one ghost in this game. In the minLevel function, I use a recursive way to get all the state after each step of each ghost, and evaluate the maxLevel of each state, and select the minimum value. And pacman should not take the "stop" action.

3: Alpha-Beta Pruning

Alpha-Beta algorithm is an optimization of minimax algorithm. The minimax algorithm is an exhaustive algorithm that requires traversing all nodes, while the Alpha-Beta algorithm can improve the efficiency of the algorithm by pruning and subtracting unnecessary nodes. In this algorithm, α represents the maximum lower bound of all possible solutions and β represents the minimum upper bound of all possible solutions. In the process of solving, alpha and beta will gradually approach. If for some node, alpha &gt occurs; The case of beta, then, indicates that this point will not produce an optimal solution, so it is not necessary to extend it to complete the pruning.
Alpha-Beta algorithm code is similar to minimaxs algorithm, but the judgment condition is changed to judge the relationship between the current value and α and β.

4: Expectimax

This algorithm considers that not every time ghost can make the best action for ghost, so it is not to take the minimum value of all the state values after ghost action, but to expect them. According to the requirements of the topic, ghost take each action probability is the same, so only need to average the value of the state can be. Other parts are the same as those in Minimax.

5: Evaluation Function

In this case, we need to design better evaluation functions to make Pac-Man score

higher and more efficient.
As Q1 but I added a few multipliers to a few variables for the final score accordingly to their significance.
A better evaluation would have taken into account both the position of the dots and the monster's score, allowing Pac-Man to eat more dots while avoiding a collision with the monster, using the Manhattan distance.