Name: shuya shou

Perm: 7867963

Email: shuyashou@ucsb.edu

In this machine problem, I try to make my Pacman cleverer and earn points as many as possible without being caught by the ghosts. To achieve this goal, I implement minimax search algorithm for the Pacman and design my own evaluation function to optimize its choice of possible movements.

The minimax decision maximizes the utility under the assumption that the opponent seeks to minimize it. I write a recursive function with 3 input parameters: game state, index of agent and depth. The base/end case is when game state equals to win or lose or reaching the depth limit, return the utility of this movement.

If agent is Pacman, it is going to find a way to maximize utility. From the given input game state, I obtain all its possible legal actions and iterate through all actions to generate their successive game state and recursively call minimax search function with changing agent index to find the one with the maximum utility.

If agent is ghost, it is going to find a way to minimize utility. I also iterate through all possible successive states with given input game state. Then I calculate the agent index of next agent. If the next agent index is 0, I call minimax function with an increasing depth. For all the actions, I return the one with minimum utility.

To determine the utility, I design my own evaluation function. I consider constructing it from 3 perspectives: food, capsule, and proximity of ghost.

1. For food:

   Since we're aiming at consuming all foods, so if there's food left, we should always credit score to the total utility. I measure the distance between the Pacman and the most distant food spot. If this distance is very long, meaning that we should probably choose another way with a smaller distance, so I decide minus 1/distance from total score.

   I also keep track of total food counts. If there're huge amount of food left after a move I will kind of lower down total score to tell the Pacman that this option will not consume many foods, so please choose other moves.

2. For capsule:

   Capsule worth more points if there're more food left, so I want the Pacman to consume capsule as early as it could. I keep track on amount of capsule and lower down points significantly if the Pacman's move didn't consume them.

3. For ghosts:

   I don't want my Pacman to be caught by ghost because it will get 500 points deducted and end of game, so I want the Pacman to be alert when there're ghosts nearby. I define two variables: danger signal and danger index. Danger signal is a Boolean variable to tell whether the Pacman is surround by danger or not. I put relatively high weight of ghost proximity, but if ghosts are far distant away, I don't want the Pacman to concern about it. So, when danger signal is false, danger index will be zero, meaning that no points will be deducted from total utility and the evaluation function should measure utility entirely based on food and capsule consumption.

   If danger signal equals true, I will deduct value of danger index from total utility. Danger index increases as distance decreases. Index is 0 if distance is greater than 3, meaning that ghosts are far away It becomes very large if distance equals 0, meaning that if next move will cause the Pacman to be caught, this move should have extremely low utility.

Weakness and possible improvements:

Minimax agent will have no idea what to choose when 2 options have same weight. Expectimax may solve that problem. Also, for my evaluation function, I just estimate the impacts from each determinant instead of qualitatively verify them. That's probably the reason why sometimes especially when number of food left is small, the Pacman will move back and forth repeatedly and not move on to consume remaining food until a ghost coming closer to it.