COEN 166 Artificial Intelligence

Lab Assignment #4

Name:Krating Khemkhon,Carlo Bilbao, Sanjana Parekh ID: 00001579161, 00001608742, 00001478244

Task: Minimax Agent

```
class MinimaxAgent(MultiAgentSearchAgent):
```

Function 2:getAction

```
   def getAction(self, gameState):

       minAgent = gameState.getNumAgents() - 1 #pacman is not a min agent
       best_action = self.maxValue_fun(gameState, 1, minAgent)#pass agent index =1 as
first ghost
       return best_action
```

Comment:

minAgent is the number of ghosts in the game, This number stays constant so it is figured out before the recursive call.

MaxValue_fun will return an action after the recursive call which is the best action that is returned. Depth is passed in as 1 as that is the first pacman

Function 2:maxValue_fun

```
   def maxValue_fun(self, gameState, depth, minAgent):
        if gameState.isWin() or gameState.isLose():#terminal test
          return self.evaluationFunction(gameState)

        v = float("-inf")

        for action in gameState.getLegalActions(0):
          successor = gameState.generateSuccessor(0, action)#generate successor for each action

          v2 = self.minValue_fun(successor, depth, 1, minAgent)#recurse
          if v2 > v:#iterate among all succesors to find the maxval and aquire the best action that correspons to max val
            v = v2
            if depth == 1:
                best_action = action

        if depth ==1:#root node action
            return best_action
        return v#non root node action(pacman for depth 2,3,4)
```

Comment:

-It takes in the game state, depth and the total number of ghosts.
-It checks if the gamestate is win or lost as the terminal test.
	It then returns the minimax action from the gameState using the self.evaluation function
We initialized v as the smallest number possible, which is -infinity
We loop through all the actions and generate successors. We call gameState.getLegalActions and pass in the value 0, which corresponds to the agent index number, where 0 represents pacman. For every action that pac man can make, we create the successor, which should correspond to the first ghost. We call minValue_fun and set it to v2. This is the recursive call, which will try to find the value v2 of the first ghost for that action, in which we will want to compare this value with our current max value v.
We iterate to find the biggest value of V2
At the same time if depth is equal to 1 when also get the best_action that corresponds to the largest value of V

If depth is equal to 1 we return best action as it is the root node else we return a value v which gets passed on to a ghost.

Function3: minValue_fun

```
   def minValue_fun(self, gameState, depth, playerIndex, minAgent):
       if gameState.isWin() or gameState.isLose(): #terminal test
         return self.evaluationFunction(gameState)

       v = float("inf")

       actions = gameState.getLegalActions(playerIndex)
       successors = [gameState.generateSuccessor(playerIndex, action) for action in
actions]#don't have to worry abt returning action

       if playerIndex == minAgent:#3 two things can happen here either the next thing
is pacman or a leaf
         if depth < self.depth:
           for successor in successors:
             v = min(v, self.maxValue_fun(successor, depth + 1, minAgent))#next agent
is pacman(maximizer)
         else:#depth = 4 reached as we are on the 3rd ghost but the depth is the same
           for successor in successors:
             v = min(v, self.evaluationFunction(successor))#leaf

       else:#0,1,2 repeat
         playerIndex+=1
         for successor in successors:
           v = min(v, self.minValue_fun(successor, depth, playerIndex, minAgent))
       return v
```

Comment:

First we run a terminal test.
We then set V to infinity.
We call gameState.getLegalActions and pass in the playerIndex, which corresponds to one of the three ghosts, in order to generate the possible actions for this current ghost.
We then generate an array of successors for all possible actions.
The first if condition checks if the player index is on the third ghosts.

If the player index is equal to the number of ghosts only 2 things can happen either the next agent is pacman or a leaf.
The next condition evaluates this by comparing self.depth with depth. If self.depth is greater than depth then that means the next agent is pacman. So we call maxValue_fun with depth+1 and choose the min from all the values generated for the successors.
Else we if depth is the same as self.depth it means we have reached the leafs so we just evaluate to find the min among the leafs.
If the player index is not equal to the number of ghosts it means we are on ghost 1 or 2 where the following agent is still a ghost. So we call minValue_fun again and evaluate the min values from the value returned

Lab 4 Contributions of each group member (in percentage):

Krating:40%
Carlo:30%
Sanjana:30%