HW3: Multi-Agent Search

110550093 蔡師睿

Part I. Implementation

Part 1: Minimax Search

```python
# Begin your code (Part 1)
lastAgent = gameState.getNumAgents()-1  # record the number of the last agent

"""
Minimax Search
1. Initialize an empty list called 'resVal' to record the score of each step in this recursion.
2. The recursion stop when the depth equals to the given depth, or the the game is either a winning
   state or losing state, and return the score calculated by self.evaluationFunction.
3. Run for loop to do recursion to evaluate the score of each legal action and store the results
   in the 'resVal'.
4. After getting all the evaluated scores, determine if the 'agentIdx' represents ghosts, return
   the minimum value of 'resVal'; otherwise, return the maximum value if the recursion not in the
   zero depth, and if it is in the zero depth, then return the action which has the highest score.
"""
def minimax(state, depth, agentIdx):
    if depth == self.depth or state.isWin() or state.isLose():
        return self.evaluationFunction(state)
    resVal = []
    actions = state.getLegalActions(agentIdx)
    if Directions.STOP in actions:
        actions.remove(Directions.STOP)  # we don't want the pacman to stop in some cases.
    for action in actions:
        nextState = state.getNextState(agentIdx, action)
        if agentIdx != lastAgent:
            resVal.append(minimax(nextState, depth, agentIdx+1))
        else:
            resVal.append(minimax(nextState, depth+1, 0))
    if agentIdx:
        return min(resVal)
    else:
        return max(resVal) if depth else actions[resVal.index(max(resVal))]

return minimax(gameState, 0, 0)
# End your code (Part 1)
```

Part 2: Alpha-Beta Pruning

```python
# Begin your code (Part 2)
lastAgent = gameState.getNumAgents()-1  # record the number of the last agent

"""
Alpha-Beta Pruning
1. Initialize 'initVal' to positive infinity if the agentIdx represents ghosts, or to negative
   infinity if the agentIdx represents pacman.
2. The recursion stop when the depth equals to the given depth, or the the game is either a winning
   state or losing state, and return the score calculated by self.evaluationFunction.
3. If the depth and agentIdx both aren't zero, run for loop to do recursion to evaluate the score
   of each legal action. If agentIdx is zero, 'initVal' equals to the maximum value between 'initVal'
   and the returned score, and update the 'alpha' if 'initVal' is larger than 'alpha'; else, 'initVal'
   equals to the minimum value between 'initVal' and the returned score, and update the 'beta' if
   'initVal' is less than 'beta'.
4. If 'alpha' is larger than 'beta', break the for loop and return the 'initVal'; if not, do the next
   for loop and return 'initVal' after finishing.
5. If the depth and agentIdx are both zero, initialize an empty list 'resVal' to record the evaluated
   score, and run for loop to get scores and store in the 'resVal' and update the 'initVal' and 'alpha'.
6. If 'alpha' is larger than 'beta', break the for loop; if not, do the next for loop. After finishing,
   return the action which value is equal to 'alpha'.
"""
```

```python
        def alphabeta(state, depth, agentIdx, alpha=float('-inf'), beta=float('inf')):
            if depth == self.depth or state.isWin() or state.isLose():
                return self.evaluationFunction(state)
            initVal = float('inf') if agentIdx else float('-inf')
            actions = state.getLegalActions(agentIdx)
            if Directions.STOP in actions:
                actions.remove(Directions.STOP)  # we don't want the pacman to stop in some cases.
            if depth or agentIdx:
                for action in actions:
                    nextState = state.getNextState(agentIdx, action)
                    if agentIdx == 0:
                        initVal = max(initVal, alphabeta(nextState, depth, 1, alpha, beta))
                        alpha = max(alpha, initVal)
                    else:
                        if agentIdx == lastAgent:
                            initVal = min(initVal, alphabeta(nextState, depth+1, 0, alpha, beta))
                        else:
                            initVal = min(initVal, alphabeta(
                                nextState, depth, agentIdx+1, alpha, beta))
                        beta = min(beta, initVal)
                    if alpha > beta:
                        break
                return initVal
            else:
                resVal = []
                for action in actions:
                    resVal.append(alphabeta(state.getNextState(0, action), depth, 1, alpha, beta))
                    initVal = max(initVal, resVal[-1])
                    alpha = max(alpha, initVal)
                    if alpha > beta:
                        break
                return actions[resVal.index(alpha)]

        return alphabeta(gameState, 0, 0)
```

Part 3: Expectimax Search

```python
        # Begin your code (Part 3)
        lastAgent = gameState.getNumAgents()-1  # record the number of the last agent

        """
        Expectimax Search
        1. The steps are the same as Minimax Search, but return the mean value of 'resVal' rather than minimal
           value if the agentIdx represents ghosts.
        """
        def expectimax(state, depth, agentIdx):
            if depth == self.depth or state.isWin() or state.isLose():
                return self.evaluationFunction(state)
            resVal = []
            actions = state.getLegalActions(agentIdx)
            if Directions.STOP in actions:
                actions.remove(Directions.STOP)  # we don't want the pacman to stop in some cases.
            for action in actions:
                nextState = state.getNextState(agentIdx, action)
                if agentIdx != lastAgent:
                    resVal.append(expectimax(nextState, depth, agentIdx+1))
                else:
                    resVal.append(expectimax(nextState, depth+1, 0))
            if agentIdx:
                return sum(resVal)/len(resVal)
            else:
                if depth:
                    return max(resVal)
                else:
                    return actions[resVal.index(max(resVal))]

        return expectimax(gameState, 0, 0)
        # End your code (Part 3)
```

Part 4: Evaluation Function

```python
# Begin your code (Part 4)
"""
Initialize variables and get the current game state we want.
"""
score = currentGameState.getScore()
pos = currentGameState.getPacmanPosition()
foodList = currentGameState.getFood().asList()
capsuleList = currentGameState.getCapsules()
ghostStates = currentGameState.getGhostStates()
minFoodDist = float('inf')
minCapsuleDist = float('inf')
scaredGhostDist = float('inf')

"""
Calculate the minimal position of food, capsule, and scred ghosts.
"""
for food in foodList:
    minFoodDist = min(minFoodDist, manhattanDistance(pos, food))
for capsule in capsuleList:
    minCapsuleDist = min(minCapsuleDist, manhattanDistance(pos, capsule))
for ghost in ghostStates:
    if ghost.scaredTimer > 0:
        scaredGhostDist = min(scaredGhostDist, manhattanDistance(pos, ghost.getPosition()))

"""
My evaluation function consider the current score, minimal food distance, minimal capsule distance, and
minimal scared ghost distance.
"""
return score+(10/(minFoodDist))+(20/(minCapsuleDist))+(200/(scaredGhostDist))
# End your code (Part 4)
```

Part II. Results & Analysis

Question Part1

```
D:\NYCU CS\111_Spring\Intro. to AI\HW3\AI_HW3>python autograder.py
D:\NYCU CS\111_Spring\Intro. to AI\HW3\AI_HW3\autograder.py:2: DeprecationWarning: the imp module is deprecated in favour of importlib and slated for remova
l in Python 3.12; see the module's documentation for alternative uses
  import imp
Starting on 4-19 at 19:57:58

Question part1
==============

*** PASS: test_cases\part1\0-eval-function-lose-states-1.test
*** PASS: test_cases\part1\0-eval-function-lose-states-2.test
*** PASS: test_cases\part1\0-eval-function-win-states-1.test
*** PASS: test_cases\part1\0-eval-function-win-states-2.test
*** PASS: test_cases\part1\0-lecture-6-tree.test
*** PASS: test_cases\part1\0-small-tree.test
*** PASS: test_cases\part1\1-1-minmax.test
*** PASS: test_cases\part1\1-2-minmax.test
*** PASS: test_cases\part1\1-3-minmax.test
*** PASS: test_cases\part1\1-4-minmax.test
*** PASS: test_cases\part1\1-5-minmax.test
*** PASS: test_cases\part1\1-6-minmax.test
*** PASS: test_cases\part1\1-7-minmax.test
*** PASS: test_cases\part1\1-8-minmax.test
*** PASS: test_cases\part1\2-1a-vary-depth.test
*** PASS: test_cases\part1\2-1b-vary-depth.test
*** PASS: test_cases\part1\2-2a-vary-depth.test
*** PASS: test_cases\part1\2-2b-vary-depth.test
*** PASS: test_cases\part1\2-3a-vary-depth.test
*** PASS: test_cases\part1\2-3b-vary-depth.test
*** PASS: test_cases\part1\2-4a-vary-depth.test
*** PASS: test_cases\part1\2-4b-vary-depth.test
*** PASS: test_cases\part1\2-one-ghost-3level.test
*** PASS: test_cases\part1\3-one-ghost-4level.test
*** PASS: test_cases\part1\4-two-ghosts-3level.test
*** PASS: test_cases\part1\5-two-ghosts-4level.test
*** PASS: test_cases\part1\6-tied-root.test
*** PASS: test_cases\part1\7-1a-check-depth-one-ghost.test
*** PASS: test_cases\part1\7-1b-check-depth-one-ghost.test
*** PASS: test_cases\part1\7-1c-check-depth-one-ghost.test
*** PASS: test_cases\part1\7-2a-check-depth-two-ghosts.test
*** PASS: test_cases\part1\7-2b-check-depth-two-ghosts.test
```

```
*** PASS: test_cases\part1\7-2c-check-depth-two-ghosts.test
*** Running MinimaxAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running MinimaxAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\part1\8-pacman-game.test

### Question part1: 20/20 ###
```

Question Part2

```
Question part2
==============

*** PASS: test_cases\part2\0-eval-function-lose-states-1.test
*** PASS: test_cases\part2\0-eval-function-lose-states-2.test
*** PASS: test_cases\part2\0-eval-function-win-states-1.test
*** PASS: test_cases\part2\0-eval-function-win-states-2.test
*** PASS: test_cases\part2\0-lecture-6-tree.test
*** PASS: test_cases\part2\0-small-tree.test
*** PASS: test_cases\part2\1-1-minmax.test
*** PASS: test_cases\part2\1-2-minmax.test
*** PASS: test_cases\part2\1-3-minmax.test
*** PASS: test_cases\part2\1-4-minmax.test
*** PASS: test_cases\part2\1-5-minmax.test
*** PASS: test_cases\part2\1-6-minmax.test
*** PASS: test_cases\part2\1-7-minmax.test
*** PASS: test_cases\part2\1-8-minmax.test
*** PASS: test_cases\part2\2-1a-vary-depth.test
*** PASS: test_cases\part2\2-1b-vary-depth.test
*** PASS: test_cases\part2\2-2a-vary-depth.test
*** PASS: test_cases\part2\2-2b-vary-depth.test
*** PASS: test_cases\part2\2-3a-vary-depth.test
*** PASS: test_cases\part2\2-3b-vary-depth.test
*** PASS: test_cases\part2\2-4a-vary-depth.test
*** PASS: test_cases\part2\2-4b-vary-depth.test
*** PASS: test_cases\part2\2-one-ghost-3level.test
*** PASS: test_cases\part2\3-one-ghost-4level.test
*** PASS: test_cases\part2\4-two-ghosts-3level.test
*** PASS: test_cases\part2\5-two-ghosts-4level.test
*** PASS: test_cases\part2\6-tied-root.test
*** PASS: test_cases\part2\7-1a-check-depth-one-ghost.test
*** PASS: test_cases\part2\7-1b-check-depth-one-ghost.test
*** PASS: test_cases\part2\7-1c-check-depth-one-ghost.test
*** PASS: test_cases\part2\7-2a-check-depth-two-ghosts.test
*** PASS: test_cases\part2\7-2b-check-depth-two-ghosts.test
*** PASS: test_cases\part2\7-2c-check-depth-two-ghosts.test
*** Running AlphaBetaAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running AlphaBetaAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\part2\8-pacman-game.test

### Question part2: 25/25 ###
```

Question Part3

```
Question part3
==============

*** PASS: test_cases\part3\0-eval-function-lose-states-1.test
*** PASS: test_cases\part3\0-eval-function-lose-states-2.test
*** PASS: test_cases\part3\0-eval-function-win-states-1.test
*** PASS: test_cases\part3\0-eval-function-win-states-2.test
*** PASS: test_cases\part3\0-expectimax1.test
*** PASS: test_cases\part3\1-expectimax2.test
*** PASS: test_cases\part3\2-one-ghost-3level.test
*** PASS: test_cases\part3\3-one-ghost-4level.test
*** PASS: test_cases\part3\4-two-ghosts-3level.test
*** PASS: test_cases\part3\5-two-ghosts-4level.test
*** PASS: test_cases\part3\6-1a-check-depth-one-ghost.test
*** PASS: test_cases\part3\6-1b-check-depth-one-ghost.test
*** PASS: test_cases\part3\6-1c-check-depth-one-ghost.test
*** PASS: test_cases\part3\6-2a-check-depth-two-ghosts.test
*** PASS: test_cases\part3\6-2b-check-depth-two-ghosts.test
*** PASS: test_cases\part3\6-2c-check-depth-two-ghosts.test
*** Running ExpectimaxAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running ExpectimaxAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\part3\7-pacman-game.test

### Question part3: 25/25 ###
```

Question Part4

```
Question part4
==============

Pacman emerges victorious! Score: 1352
Pacman emerges victorious! Score: 1339
Pacman emerges victorious! Score: 1260
Pacman emerges victorious! Score: 1367
Pacman emerges victorious! Score: 1372
Pacman emerges victorious! Score: 1358
Pacman emerges victorious! Score: 1367
Pacman emerges victorious! Score: 1180
Pacman emerges victorious! Score: 1360
Pacman emerges victorious! Score: 1363
Average Score: 1331.8
Scores:        1352.0, 1339.0, 1260.0, 1367.0, 1372.0, 1358.0, 1367.0, 1180.0, 1360.0, 1363.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** PASS: test_cases\part4\grade-agent.test (8 of 8 points)
*** EXTRA CREDIT: 2 points
***     1331.8 average score (4 of 4 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  2 points
***         >= 1000:  4 points
***     10 games not timed out (2 of 2 points)
***         Grading scheme:
***          < 0:  fail
***         >= 0:  0 points
***         >= 5:  1 points
***         >= 10:  2 points
***     10 wins (4 of 4 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  1 points
***         >= 4:  2 points
***         >= 7:  3 points
***         >= 10:  4 points

### Question part4: 10/10 ###
```

Final

```
Finished at 19:58:00

Provisional grades
==================
Question part1: 20/20
Question part2: 25/25
Question part3: 25/25
Question part4: 10/10
------------------
Total: 80/80


                     ALL HAIL GRANDPAC.
              LONG LIVE THE GHOSTBUSTING KING.

                  ---      ----      ---
                  |  \    /  + \    /  |
                  | + \--/      \--/ + |
                  |   +     +          |
                  | +     +        +   |
                @@@@@@@@@@@@@@@@@@@@@@@@@@
              @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            \    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
             \ /  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
              V   \   @@@@@@@@@@@@@@@@@@@@@@@@@@@@
                   \ /  @@@@@@@@@@@@@@@@@@@@@@@@@@
                    V     @@@@@@@@@@@@@@@@@@@@@@@@
                            @@@@@@@@@@@@@@@@@@@@@@
                    /\      @@@@@@@@@@@@@@@@@@@@@@
                   /  \  @@@@@@@@@@@@@@@@@@@@@@@@@
              /\  /    @@@@@@@@@@@@@@@@@@@@@@@@@@@
             /  \ @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            /    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
              @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                @@@@@@@@@@@@@@@@@@@@@@@@@@
                    @@@@@@@@@@@@@@@@@@
```

Discussion

In pacman game, we can get 10 pts by eating food, get 50 pts by eating capsules, and get 200 pts by eating scared ghosts. Thus, I assume that the formula is

$$CurrentScore + (\frac{x}{minFoodDist}) + (\frac{y}{minCapsuleDist}) + (\frac{z}{scaredGhostDist})$$

In the beginning, I make x, y, and z be 10, 50, 200, respectively. However, I discovered that I will get different results if I change the parameter x, y, or z, and interestingly, if I only raise the value z, the result score will become higher, but sometimes the higher value I use, the worst score I will get.
Therefore, I've tried many reasonable values of x, y, and z, and finally, I consider

$$CurrentScore + (\frac{10}{minFoodDist}) + (\frac{20}{minCapsuleDist}) + (\frac{200}{scaredGhostDist})$$

is the most suitable formula for the evaluation function.