Homework 3: Multi-Agent Search

110550107 傅莉妮

Part I. Implementation (5%):

The explanation of each code is written in the comment of codes.

Part 1: Minimax Search

```
    # Begin your code (Part 1)
    # raise NotImplementedError("To be implemented")

    # initial agent  = 0 (pacman) => find max value first
    # initial cDepth = 0 (current depth)
    score, Action = self.max_value(0, 0, gameState)
    return Action

def max_value(self, agentIndex, cDepth, gameState): # agentIndex = 0(pacman)
    if gameState.isWin() or gameState.isLose() or cDepth == self.depth:
        return self.evaluationFunction(gameState), None

    legalActions = gameState.getLegalActions(agentIndex)
    bestV = -10000
    bestMove = "x"
    for action in legalActions:
        # check every value possible after current state
        score, _ = self.min_value(1, cDepth, gameState.getNextState(0, action))
        if bestV < score:
            bestV = score
            bestMove = action

    return bestV, bestMove
```

```
def min_value(self, agentIndex, cDepth, gameState): # agentIndex != 0(ghost)
    if gameState.isWin() or gameState.isLose() or cDepth == self.depth:
        return self.evaluationFunction(gameState), None

    legalActions = gameState.getLegalActions(agentIndex)
    bestV = 10000
    bestMove = "x"
    if gameState.getNumAgents() == (agentIndex+1):
        nextAgent = 0
        cDepth += 1 # last ghost reached, current depth++
    else:
        nextAgent = agentIndex + 1
    for action in legalActions:
        # check every value possible after current state
        if nextAgent == 0: # the last ghost
            score, _ = self.max_value(0, cDepth, gameState.getNextState(agentIndex, action))
        else:
            score, _ = self.min_value(nextAgent, cDepth, gameState.getNextState(agentIndex, action))
        if bestV > score:
            bestV = score
            bestMove = action

    return bestV, bestMove

    # End your code (Part 1)
```

Part 2: Alpha-Beta Pruning

```
    # Begin your code (Part 2)
    # raise NotImplementedError("To be implemented")

    # initial agent  = 0 (pacman) => find max value first
    # initial cDepth = 0 (current depth)
    # initial alpha  = -10000, beta = 10000
    score, Action = self.max_value(0, 0, gameState, -10000, 10000)
    return Action

def max_value(self, agentIndex, cDepth, gameState, alpha, beta): # agentIndex = 0(pacman)
    if gameState.isWin() or gameState.isLose() or cDepth == self.depth:
        return self.evaluationFunction(gameState), None

    legalActions = gameState.getLegalActions(agentIndex)
    bestV = -10000
    bestMove = "x"
    for action in legalActions:
        score, _ = self.min_value(1, cDepth, gameState.getNextState(0, action), alpha, beta)
        if bestV < score:
            bestV = score
            bestMove = action

        # prune
        if score > beta:
            break
        if score > alpha:
            alpha = score

    return bestV, bestMove
```

```
def min_value(self, agentIndex, cDepth, gameState, alpha, beta): # agentIndex != 0(ghost)
    if gameState.isWin() or gameState.isLose() or cDepth == self.depth:
        return self.evaluationFunction(gameState), None

    legalActions = gameState.getLegalActions(agentIndex)
    bestV = 10000
    bestMove = "x"
    if gameState.getNumAgents() == (agentIndex+1):
        nextAgent = 0
        cDepth += 1 # last ghost reached, current depth++
    else:
        nextAgent = agentIndex + 1
    for action in legalActions:
        if nextAgent == 0:
            score, _ = self.max_value(0, cDepth, gameState.getNextState(agentIndex, action), alpha, beta)
        else:
            score, _ = self.min_value(nextAgent, cDepth, gameState.getNextState(agentIndex, action), alpha, beta)
        if bestV > score:
            bestV = score
            bestMove = action

        # prune
        if score < alpha:
            break
        if score < beta:
            beta = score

    return bestV, bestMove

    # End your code (Part 2)
```

Part 3: Expectimax Search

```
    # Begin your code (Part 3)
    # raise NotImplementedError("To be implemented")

    # initial agent  = 0 (pacman) => find max value first
    # initial cDepth = 0 (current depth)
    score, Action = self.max_value(0, 0, gameState)
    return Action

#  this part is same as minimax
def max_value(self, agentIndex, cDepth, gameState): # agentIndex = 0(pacman)
    if gameState.isWin() or gameState.isLose() or cDepth == self.depth:
        return self.evaluationFunction(gameState), None
    legalActions = gameState.getLegalActions(agentIndex)
    bestV = -10000
    bestMove = "x"
    for action in legalActions:
        score = self.exp_value(1, cDepth, gameState.getNextState(0, action))
        if bestV < score:
            bestV = score
            bestMove = action

    return bestV, bestMove
```

```python
# this part has changed into calculating expected value
def exp_value(self, agentIndex, cDepth, gameState): # agentIndex != 0(ghost)
    if gameState.isWin() or gameState.isLose() or cDepth == self.depth:
        return self.evaluationFunction(gameState)
    legalActions = gameState.getLegalActions(agentIndex)

    # expect_score : add all score first, return (all_score/num_of_actions) in the end
    expect_score = 0
    if gameState.getNumAgents() == (agentIndex+1):
        nextAgent = 0
        cDepth += 1
    else:
        nextAgent = agentIndex + 1
    for action in legalActions:
        if nextAgent == 0:
            score, _ = self.max_value(0, cDepth, gameState.getNextState(agentIndex, action))
            expect_score += score

        else:
            score = self.exp_value(nextAgent, cDepth, gameState.getNextState(agentIndex, action))
            expect_score += score

    return expect_score / len(legalActions)

    # End your code (Part 3)
```

Part 4: Better Evaluation Function

```python
# Begin your code (Part 4)
# raise NotImplementedError("To be implemented")

# get infomation first
PacPos = currentGameState.getPacmanPosition()
legalActions = currentGameState.getLegalActions(0)

# food info
if currentGameState.getFood().asList():
    nearestFoodDistance = min([manhattanDistance(PacPos, food) for food in currentGameState.getFood().asList()])
else:
    nearestFoodDistance = 0

# capsule info
capsules = currentGameState.getCapsules()
nearestCapsuleDistance = 0
if len(capsules) != 0:
    capsulesDistance = [manhattanDistance(PacPos, capsule) for capsule in capsules]
    nearestCapsuleDistance = min(capsulesDistance)

# ghost info
GhostStates = currentGameState.getGhostStates()
minGhostDistance = 1000
for ghostState in GhostStates:
    dist = manhattanDistance(PacPos, ghostState.getPosition())
    if dist < minGhostDistance:
        minGhostDistance = dist
        nearestScaredTime = ghostState.scaredTimer
isScared = nearestScaredTime > 1

# initialize Score num
Score = currentGameState.getScore() * 10
```

```
"""
rules:
1. if near food then go to eat
2. if near capsule then go to eat
3. if near ghost and ghost is scared then go to eat ghost
4. if near ghost but not scared then run away
5. STOP is the least preferrable action
"""

# 1.
Score -= 10 * nearestFoodDistance

# 2.
if nearestCapsuleDistance <= 5:
    Score -= 20 * nearestCapsuleDistance

# 3.
if isScared:
    Score -= 2 * minGhostDistance
# 4.
else:
    Score += 10 * minGhostDistance

# 5.
for action in legalActions:
    if action == Directions.STOP:
        Score += 1000

return Score
# End your code (Part 4)
```

Part II. Results & Analysis (5%):

Result of autograder

```
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 1 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
==============

*** 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 1 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
==============

*** 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 1 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
==============

Pacman emerges victorious! Score: 1122
Pacman emerges victorious! Score: 408
Pacman emerges victorious! Score: 867
Pacman emerges victorious! Score: 1221
Pacman emerges victorious! Score: 1266
Pacman emerges victorious! Score: 1290
Pacman emerges victorious! Score: 855
Pacman emerges victorious! Score: 1192
Pacman emerges victorious! Score: 1120
Pacman emerges victorious! Score: 1087
Average Score: 1042.8
Scores:        1122.0, 408.0, 867.0, 1221.0, 1266.0, 1290.0, 855.0, 1192.0, 1120.0, 1087.0
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
***     1042.8 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 ###
```

```
Finished at 15:52:49

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     @@@@@@@@@@@@@@@@@@@@@
                          @@@@@@@@@@@@@@@@@@@
                        /\      @@@@@@@@@@@@@@@@@
                       /  \  @@@@@@@@@@@@@@@@@@@@
                     /\  /    @@@@@@@@@@@@@@@@@@@
                    /  \ @@@@@@@@@@@@@@@@@@@@@@@@
                   /    @@@@@@@@@@@@@@@@@@@@@@@@
                 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                   @@@@@@@@@@@@@@@@@@@@@@@@@@@
                     @@@@@@@@@@@@@@@@@@@@@@@
                        @@@@@@@@@@@@@@@@@
```

Better Evaluation Function

My rules for Evaluation Function is:

1. Go to nearest food
2. Go to nearest capsule if near to any one of capsule
3. If ghost is scared then go to eat it
4. If ghost is not scared then run away
5. Make stop the least preferrable action to choose

I tried to change the weight of food distance (shown in the figure below), and I found out that it performs better when set as 10. Therefore I set constant for food to 10, and constant for capsule to 20.

```
# 1.
Score -= 10 * nearestFoodDistance
```

change this constant

```
❯ python3 pacman.py -p ExpectimaxAgent -l smallClassic -a depth=2 -q -n 10
Pacman died! Score: -61
Pacman emerges victorious! Score: 885
Pacman died! Score: -383
Pacman emerges victorious! Score: 790
Pacman died! Score: -372
Pacman died! Score: -158
Pacman emerges victorious! Score: 398
Pacman died! Score: -92
Pacman emerges victorious! Score: 744
Pacman died! Score: -149
Average Score: 160.2
Scores:        -61.0, 885.0, -383.0, 790.0, -372.0, -158.0, 398.0, -92.0, 744.0, -149.0
Win Rate:      4/10 (0.40)
Record:        Loss, Win, Loss, Win, Loss, Loss, Win, Loss, Win, Loss
```

set as 1

```
❯ python3 pacman.py -p ExpectimaxAgent -l smallClassic -a depth=2 -q -n 10
Pacman emerges victorious! Score: 907
Pacman emerges victorious! Score: 931
Pacman emerges victorious! Score: 896
Pacman emerges victorious! Score: 840
Pacman emerges victorious! Score: 784
Pacman emerges victorious! Score: 937
Pacman emerges victorious! Score: 1113
Pacman emerges victorious! Score: 945
Pacman emerges victorious! Score: 950
Pacman emerges victorious! Score: 1360
Average Score: 966.3
Scores:        907.0, 931.0, 896.0, 840.0, 784.0, 937.0, 1113.0, 945.0, 950.0, 1360.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
```

set as 10

I set constant for isScared to 10 originally, but it didn't perform well, so I set it to 2 instead.

```
# 3.
if isScared:
    Score -= 2 * minGhostDistance
```

change this constant

```
Pacman emerges victorious! Score: 1122
Pacman emerges victorious! Score: 408
Pacman emerges victorious! Score: 867
Pacman emerges victorious! Score: 1221
Pacman emerges victorious! Score: 1266
Pacman emerges victorious! Score: 1290
Pacman emerges victorious! Score: 855
Pacman emerges victorious! Score: 1192
Pacman emerges victorious! Score: 1120
Pacman emerges victorious! Score: 1087
Average Score: 1042.8
Scores:        1122.0, 408.0, 867.0, 1221.0, 1266.0, 1290.0, 855.0, 1192.0, 1120.0, 1087.0
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
***     1042.8 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 ###
```

result for constant = 2

```
Pacman emerges victorious! Score: 944
Pacman emerges victorious! Score: 397
Pacman emerges victorious! Score: 894
Pacman emerges victorious! Score: 804
Pacman emerges victorious! Score: 1230
Pacman emerges victorious! Score: 1101
Pacman emerges victorious! Score: 1283
Pacman died! Score: 91
Pacman emerges victorious! Score: 1196
Pacman emerges victorious! Score: 847
Average Score: 878.7
Scores:        944.0, 397.0, 894.0, 804.0, 1230.0, 1101.0, 1283.0, 91.0, 1196.0, 847.0
Win Rate:      9/10 (0.90)
Record:        Win, Win, Win, Win, Win, Win, Win, Loss, Win, Win
*** FAIL: test_cases/part4/grade-agent.test (7 of 8 points)
***     878.7 average score (2 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
***     9 wins (3 of 4 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  1 points
***         >= 4:  2 points
***         >= 7:  3 points
***         >= 10:  4 points

### Question part4: 7/10 ###
```

result for constant = 10