گزارشکار پروژه **۲**

۱) عامل عکس العمل

```
54    def evaluationFunction(self, currentGameState, action):
55        currentFoods = currentGameState.getFood().asList()
56        successorGameState = currentGameState.generatePacmanSuccessor(action)
57        newPos = successorGameState.getPacmanPosition()
58        newGhostStates = successorGameState.getGhostStates()
59        evaluation = 0
60        # calculate distance of the nearest food (among current foods )from new position
61        nearestFood = float('inf')
62        for food in currentFoods:
63            temp = manhattanDistance(newPos, food)
64            if temp < nearestFood:
65                nearestFood = temp
66        evaluation -= nearestFood
67        # calculate distance of the nearest(among active ghosts)ghost from new position
68        nearestGhost = float('inf')
69        for ghost in newGhostStates:
70            if not ghost.scaredTimer > 0:
71                temp = manhattanDistance(newPos, ghost.getPosition())
72                if temp < nearestGhost:
73                    nearestGhost = temp
74        # checks that this position can lead to game over or not
75        if nearestGhost <= 1:
76            return -float('inf')
77        return evaluation
```

توضیحات : دراین متد، ابتدا فاصله نزدیک ترین غذا ازموقعیت جدید را محاسبه می کنیم(از بین غذاهای دردسترس درموقعیت حال و نه موقعیت جدید). این پارامتربه صورت قرینه برمقدارevaluation تاثیرمی گذارد زیرا هرچقدرفاصله تا نزدیک ترین غذا کم تر باشد مقدارevaluation باید بیشتر باشد. سپس فاصله نزدیک ترین روح فعال تا موقعیت جدید را بدست می اوریم که اگر این مقدارکمترمساوی ۱ باشد ینی عامل درمعرض باخت قراردارد و از رفتن به این موقعیت باید اجتناب کند (دراین شرایط مقدارمنفی بی نهایت به عنوان evaluation برگردانده می شود). درنهایت مقدارevaluation برگردانده می شود.

پاسخ سوال ۱ :

currentFoods و**newPos** :درمحاسبه فاصله نزدیک ترین غذا ازموقعیت جدید استفاده شد وهمان طور که در بخش توضیحات گفته شده این فاصله رابطه قرینه با مقدار evaluation دارد.

newGhostStates و **scaredTimer**: درمحاسبه فاصله نزدیک ترین روح فعال استفاده شدند که اگر این مقدار کمتر مساوی ۱ باشد، منفی بینهایت را به عنوان evaluation برمی گردانیم.

پاسخ سوال ۲ :

پارامترهایی که با کم شدن آن ها مقدار evaluation باید زیاد شود را با علامت منفی و پارامترهای دیگر را با علامت مثبت درنظرمی گیریم . به عنوان مثال چون با کم شدن فاصله تا نزدیک ترین غذا مقدارevaluation باید زیاد شود، فاصله تا نزدیک ترین غذا را با علامت منفی با evaluation جمع می کنیم.

```
Question q1
===========

Pacman emerges victorious! Score: 1238
Pacman emerges victorious! Score: 1244
Pacman emerges victorious! Score: 1239
Pacman emerges victorious! Score: 1240
Pacman emerges victorious! Score: 1239
Pacman emerges victorious! Score: 1237
Pacman emerges victorious! Score: 1238
Pacman emerges victorious! Score: 1247
Pacman emerges victorious! Score: 1238
Pacman emerges victorious! Score: 1242
Average Score: 1240.2
Scores:        1238.0, 1244.0, 1239.0, 1240.0, 1239.0, 1237.0, 1238.0, 1247.0, 1238.0, 1242.0
Win Rate:      10/10 (1.00)
```

۲) مینیماکس

```python
def getAction(self, gameState):
    def minimizer(state, index, depth):
        legalActions = state.getLegalActions(index)
        if not legalActions:
            return self.evaluationFunction(state)
        if index == state.getNumAgents() - 1:
            return min(maximizer(state.generateSuccessor(index, action), depth) for action in legalActions)
        else:
            return min(minimizer(state.generateSuccessor(index, action), index + 1, depth) for action in
                       legalActions)

    def maximizer(state, depth):
        legalActions = state.getLegalActions(0)
        if not legalActions or depth == self.depth:
            return self.evaluationFunction(state)
        return max(minimizer(state.generateSuccessor(0, action), 1, depth + 1) for action in legalActions)

    legals = gameState.getLegalActions(0)
    values = []
    for i in range(len(legals)):
        values.append(minimizer(gameState.generateSuccessor(0, legals[i]), 1, 1))
    maxValue = max(values)
    bestIndices = [index for index in range(len(values)) if values[index] == maxValue]
    return legals[random.choice(bestIndices)]
```

توضیحات: درابتدا به توضیح عملکرد متدهای minimizer وmaximizer می پردازیم.

متد minimizer : دراین متد اگر اخرین روح باشیم، مینیمم فرزندهایی که به صورت ماکس هستند را برمی گردانیم. اما اگر آخرین روح نباشیم، ازبین مقادیرروح بعدی درحالات مختلف مینیمم شان را برمی گردانیم .

متد maximizer : دراین متد اگربه اخرین عمق رسیده باشیم و یا حرکت مجاز بعدی وجود نداشته باشد(بازی با برد یا باخت تمام شده باشد)، مقدارevaluationFunction را برای این state برمی گردانیم واگراین طورنباشد، از بین مقادیر روح اول درحالات مختلف ماکسیمم شان را برمی گردانیم.

درمتد getAction ابتدا به ازای هرحرکت مجاز مقدار روح اول را با استفاده از متد minimizer بدست می آوریم . سپس از بین همه حرکات مجازی که منجربه بدست آمدن بیشترین مقدارشدند، یکی را به تصادف انتخاب می کنیم.

پاسخ سوال ۳ :

چون تصورعامل این است که روح ها کاملا تخاصمی عمل می کنند، سعی می کند زودتر ببازد زیرا اگراین کاررا نکند، روح ها آن را احاطه کرده و درنهایت بازهم می بازد اما چون دراین شرایط بیشترزنده مانده ، امتیازکمتری دریافت می کند.

```
*** 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 25 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\q2\8-pacman-game.test

### Question q2: 5/5 ###


Finished at 1:55:45

Provisional grades
==================
Question q2: 5/5
------------------
Total: 5/5
```

۳) هرس آلفا-بتا

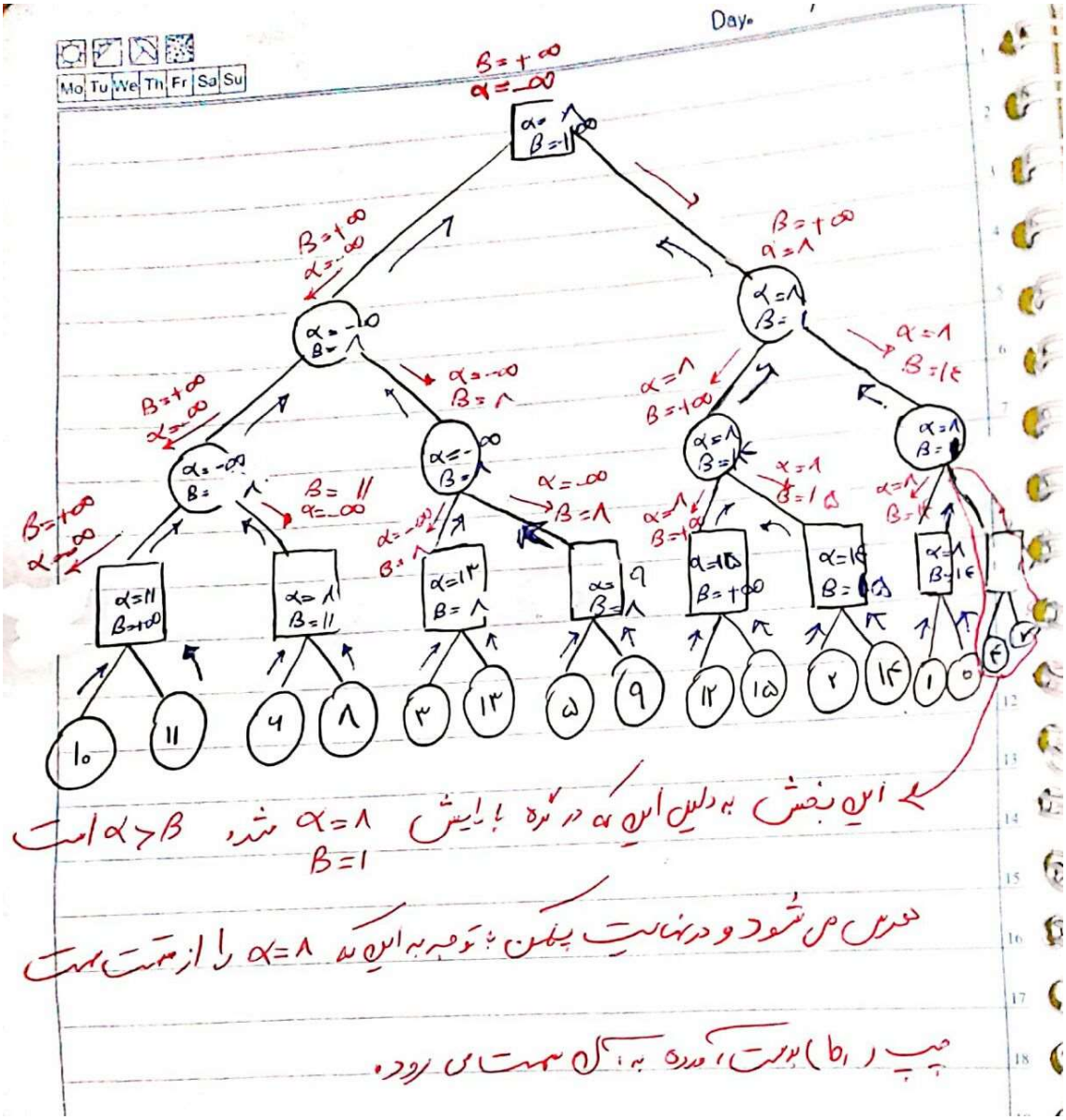
```
def getAction(self, gameState):
    def minimizer(state, index, depth, alpha, beta):
        legalActions = state.getLegalActions(index)
        if not legalActions:
            return self.evaluationFunction(state)
        minValue = float('inf')
        if index == state.getNumAgents() - 1:
            for action in legalActions:
                temp = maximizer(state.generateSuccessor(index, action), depth, alpha, beta)
                if temp < minValue:
                    minValue = temp
                beta = min(beta, minValue)
                if beta < alpha:
                    return minValue
        else:
            for action in legalActions:
                temp = minimizer(state.generateSuccessor(index, action), index + 1, depth, alpha, beta)
                if temp < minValue:
                    minValue = temp
                beta = min(beta, minValue)
                if beta < alpha:
                    return minValue

        return minValue

    def maximizer(state, depth, alpha, beta):
        legalActions = state.getLegalActions(0)
        if not legalActions or depth == self.depth:
            return self.evaluationFunction(state)
        maxValue = -float('inf')
        for action in legalActions:
            temp = minimizer(state.generateSuccessor(0, action), 1, depth + 1, alpha, beta)
            if temp > maxValue:
                maxValue = temp
            alpha = max(alpha, maxValue)
            if beta < alpha:
                return maxValue
        return maxValue
```

توضیحات: این متد هم مانند متد getAction در قسمت قبل(مینیماکس) پیاده سازی شده است با این تفاوت که در متدهای minimizer و maximizer شرط هرس بررسی شده است (خطوط ۱۵۹ و۱۶۷).

پاسخ سوال ۴ :



پاسخ سوال ۵ :

مقداربرگردانده شده برای ریشه در حالت بدون هرس وبا هرس تفاوتی ندارد اما برای گره های میانی ممکن است متفاوت باشد. به عنوان مثال درشکل بالا درقسمتی که هرس شد اگربه جای مقادیر ۴ و۷ ، ۰و۱- بود آن گاه مقدارگره با a=8 و B=1 در حالت بدون هرس ۱- و در حالت با هرس ۱ می شد.

```
*** 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 27 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\q3\8-pacman-game.test

### Question q3: 5/5 ###


Finished at 16:15:18

Provisional grades
==================
Question q3: 5/5
------------------
Total: 5/5
```

۴) مینیماکس احتمالی

```python
class ExpectimaxAgent(MultiAgentSearchAgent):
    """
      Your expectimax agent (question 4)
    """

    def getAction(self, gameState):
        def maximizer(state, depth):
            legalActions = state.getLegalActions(0)
            if not legalActions or depth == self.depth:
                return self.evaluationFunction(state)

            return max(expect(state.generateSuccessor(0, action), 1, depth + 1) for action in legalActions)

        def expect(state, index, depth):
            legalActions = state.getLegalActions(index)
            if not legalActions:
                return self.evaluationFunction(state)
            value = 0
            if index == state.getNumAgents() - 1:
                for action in legalActions:
                    value += maximizer(state.generateSuccessor(index, action), depth) * (1.0 / len(legalActions))
            else:
                for action in legalActions:
                    value += expect(state.generateSuccessor(index, action), index + 1, depth) * (
                            1.0 / len(legalActions))
            return value

        legals = gameState.getLegalActions(0)
        values = []
        for i in range(len(legals)):
            values.append(expect(gameState.generateSuccessor(0, legals[i]), 1, 1))
        maxValue = max(values)
        bestIndices = [index for index in range(len(values)) if values[index] == maxValue]
        return legals[random.choice(bestIndices)]
```

توضیحات:

این متد هم مانند متد getAction در قسمت مینیماکس پیاده سازی شده است با این تفاوت که اینجا متد expectimax جایگزین متد minimizer شده است و درآن به جای مینیمم گیری ، میانگین وزن دارمقادیربرگردانده می شود.
((تعداد اکشن های مجازدرآن وضعیت)/۱ = وزن هر مقدار)

پاسخ سوال ۶ :

```
C:\Users\Ashkan\Downloads\Compressed\AI-P2\AI-P2\AI-P2>python pacman.py -p AlphaBetaAgent -l trappedClassic -a depth=3
q -n 10
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Pacman died! Score: -501
Average Score: -501.0
Scores:        -501.0, -501.0, -501.0, -501.0, -501.0, -501.0, -501.0, -501.0, -501.0, -501.0
Win Rate:      0/10 (0.00)
Record:        Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss
```

```
C:\Users\Ashkan\Downloads\Compressed\AI-P2\AI-P2\AI-P2>python pacman.py -p ExpectimaxAgent -l trappedClassic -a depth=3
-q -n 10
Pacman died! Score: -502
Pacman died! Score: -502
Pacman died! Score: -502
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 531
Pacman died! Score: -502
Pacman emerges victorious! Score: 531
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 531
Pacman died! Score: -502
Average Score: 14.7
Scores:        -502.0, -502.0, -502.0, 532.0, 531.0, -502.0, 531.0, 532.0, 531.0, -502.0
Win Rate:      5/10 (0.50)
Record:        Loss, Loss, Loss, Win, Win, Loss, Win, Win, Win, Loss
```

درحالت اول چون تصورعامل این است که روح ها کاملا تخاصمی عمل می کنند، سعی می کند زودتر ببازد زیرا اگراین کاررا نکند، روح ها آن را احاطه کرده و درنهایت بازهم می بازد اما چون دراین شرایط بیشترزنده مانده ، امتیازکمتری دریافت می کند اما درحالت دوم چون می داند روح ها به صورت تصادفی عمل می کند دیگرسعی نمی کند زودتر ببازد و با این کاربه طورمتوسط در ۵۰ درصد مواقع برنده می شود.

پاسخ سوال ۷ :

اگربخواهیم درحالت مینیماکس احتمالی از الگوریتم رولت ویل استفاده کنیم ، در متد expect ابتدا evaluation وضعیتی از بازی که پس از هر اکشن به وجود می آید را

بدست می آوریم و با توجه به آن مقدار برای هراکشن یه احتمال مشخصی را درنظر می گیریم ((مجموع evaluation ها) /(evaluation(i)) = pi) ودرنهایت یکی را به صورت رندوم انتخاب کرده و به ازای آن اکشن مقادیر را بدست می آوریم.

اگر نیاز بود تا با کمک الگوریتم رولت ویل بیشتر از یک حالت انتخاب شود کافی است درمتد getAction به ازای هر legalAction دوبارمتد expect فراخوانی شود وبین مقادیری که این دوفراخوانی برمی گردانند ماکسیمم آن ها را به عنوان ماکسیمم مقداری که با انجام آن legalAction بدست می آید درنظربگیریم.

```
*** 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 27 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\q4\7-pacman-game.test

### Question q4: 5/5 ###


Finished at 16:38:48

Provisional grades
==================
Question q4: 5/5
------------------
Total: 5/5
```

۵) تابع ارزیابی

```python
def betterEvaluationFunction(currentGameState):
    """
    Your extreme ghost-hunting, pellet-nabbing, food-gobbling, unstoppable
    evaluation function (question 5).
      """
    position = currentGameState.getPacmanPosition()
    foods = currentGameState.getFood().asList()
    consumedFoods = len(currentGameState.getFood().asList())
    ghosts = currentGameState.getGhostStates()
    scaredTimes = [ghost.scaredTimer for ghost in ghosts]
    evaluation = consumedFoods + currentGameState.getScore()
    length = len(foods)
    if length > 0:
        # calculate sum of distances to foods
        sumOfDistancesToFoods = 0
        for food in foods:
            sumOfDistancesToFoods += manhattanDistance(position, food)
        evaluation += (1.0 / sumOfDistancesToFoods)
    # calculate  sum of distances to ghosts
    sumOfDistancesToGhosts = 0
    for ghost in ghosts:
        sumOfDistancesToGhosts += manhattanDistance(position, ghost.getPosition())
    # calculate sum of scared times
    sumOfScaredTimes = sum(scaredTimes)
    if sumOfScaredTimes > 0:
        evaluation += sumOfScaredTimes - len(currentGameState.getCapsules()) - sumOfDistancesToGhosts
    else:
        evaluation += len(currentGameState.getCapsules()) + sumOfDistancesToGhosts
    return evaluation
```

توضیحات : دراین متد، ابتدا مجموع فاصله های عامل تا هرغذای دردسترس را بدست می آوریم . سپس اگر این مجموع صفر نباشد آن را به صورت قرینه با evaluation جمع می کنیم(evaluation درابتدا برابر مجموع امتیاز دراین وضعیت و غذاهای خورده شده تا الان است). پس ازآن مجموع فاصله های عامل تا روح ها را بدست می آوریم . درنهایت اگرمجموع scaredTime های روح ها مثبت بود (ینی حداقل یک روح دروضعیت white است)، قرینه مجموع فاصله تا روح ها وقرینه تعداد powerPalletها ومجموع scaredTime ها را با evaluation جمع می کنیم (وقتی حداقل یکی از روح ها دروضعیت white است هرچه مجموع فاصله تا روح ها کم ترباشد بهتر است زیرا می توانیم روح سفید را بخوریم و آن روح به فاصله دوری از ما برود و هرچه تعداد powerPallet ها کم تر باشد بهتراست زیرا با خوردن powerPallet جدید دراین وضعیت از scaredTime بالقوه روح ها به صورت بهینه استفاده نکرده ایم) و اگر مجموع scaredTime ها صفربود، مجموع فاصله تا روح ها و تعدادpowerPallet ها را با evaluation جمع می کنیم.

پاسخ سوال ۸ :

دراین قسمت ارزیابی به شکل بهتری نسبت به قسمت اول انجام می شود زیرا به جزئیات بیشتری توجه می شود به عنوان مثال همان طورکه دربخش توضیحات گفته شد اگرمجموعscaredTime روح ها مثبت باشد مقادیربه طور متفاوتی نسبت به وقتی که مجموعscaredTime ها صفر است با evaluation جمع می شوند.

```
Average Score: 1217.4
Scores:        1132.0, 951.0, 1333.0, 1350.0, 1118.0, 1117.0, 1283.0, 1231.0, 1345.0, 1314.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** PASS: test_cases\q5\grade-agent.test (6 of 6 points)
***     1217.4 average score (2 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (1 of 1 points)
***         Grading scheme:
***          < 0:  fail
***         >= 0:  0 points
***         >= 10:  1 points
***     10 wins (3 of 3 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  1 points
***         >= 5:  2 points
***         >= 10:  3 points

### Question q5: 6/6 ###


Finished at 17:48:40
```