گزارش فاز دوم پروژه مبانی هوش مصنوع کاربردی

1) عامل عکس العمل:

ابتدا متغیر های زیر را حساب میکنیم:

```
successorGameState = currentGameState.generatePacmanSuccessor(action)
newPos = successorGameState.getPacmanPosition()
newFood = successorGameState.getFood()
newGhostStates = successorGameState.getGhostStates()
newScaredTimes = [ghostState.scaredTimer for ghostState in newGhostStates]
```

در قدم بعدی چک میکنیم که آیا پکمن فاصله امنی با روح ها دارد یا خیر، برای چک کردن این مطلب فاصله منهتن موقعیت فعلی پکمن و روح ها را محاسبه کرده و مینیموم آن را بدست می آوریم، حداقل فاصله پکمن با روح ها باید از حد امن بیشتر باشد

```
minGhostDistance = min([manhattanDistance(newPos, state.getPosition()) for
state in newGhostStates])
```

چک میکنیم آیا این عمل موجب افزایش امتیاز شده است یا خیر، برای چک کردن این مطلب امتیاز موقعیت وارث را از موقعیت فعلی کم میکنیم

```
scoreDiff = successorGameState.getScore() - currentGameState.getScore()
```

چک میکنیم آیا این عمل موجب نزدیک تر شدن به غذای های نزدیک پکمن شده است یا نه

```
pos = currentGameState.getPacmanPosition()
nearestFoodDistance = min([manhattanDistance(pos, food) for food in
currentGameState.getFood().asList()])
newFoodsDistances = [manhattanDistance(newPos, food) for food in
newFood.asList()]
newNearestFoodDistance = 0 if not newFoodsDistances else
min(newFoodsDistances)
isFoodNearer = nearestFoodDistance - newNearestFoodDistance
```

موقعیت فعلی را ذخیره میکنیم

```
direction = currentGameState.getPacmanState().getDirection()
```

حالا ارزیابی را آغاز میکنیم، اگر پکمن فاصله کافی با روح هارا نداشته باشد اصلا عمل خوبی نداشته ایم پس:

```
if minGhostDistance <= 1 or action == Directions.STOP:
    return 0
```

اگر عمل پکمن موجب بیشتر شدن امتیاز آن شده باشد بهترین عملکرد را طبیعتا داشته ایم:

```
if scoreDiff > 0:
    return 8
```

اگر به غذاهای نزدیک، نزدیکتر شده ایم:

```
elif isFoodNearer > 0:
    return 4
```

اگر عمل ما در راستای جهت فعلی بوده :

```
elif action == direction:
    return 2
```

و درغیراین صورت عمل ما به سود خاصی نرسیده پس :

```
else:
    return 1
```

سوال: توضیح دهید که از هر کدام از پارامترهای دخیل در تابع ارزیابی چگونه استفاده کرده‌اید و هر کدام چگونه برروی خروجی تاثیر می‌گذارند (تاثیر کدام یک از پارامترها بیشتر است؟).

در توضیح کد کامل توضیح داده شده و تاثیر افزایش امتیاز منطقا بیشتر است زیرا به دنبال همین مطلب در بازی هستیم.

سوال: چگونه می‌توان پارامترهایی که مقادیرشان در یک راستا نمی‌باشند را با یکدیگر برای تابع ارزیابی ترکیب کرد؟ (مانند فاصله تا غذا و روح که ارزش آن‌ها بر خلاف یکدیگر می‌باشد)

ما پس از محاسبه این مقادیر دانه به دانه اینهارا چک میکنیم و درصورتی که در ارزیابی مفید واقع شوند(چه کم و چه زیاد بودنشان) به آنها نمره میدهیم

2) مینماکس

برای پیاده سازی این بخش ابتدا در تابع getAction دو زیر تابع تعریف میکنیم:

```
def minValue(state, agentIndex, depth):
    legalActions = state.getLegalActions(agentIndex)
    if not legalActions:  # NO legalActions means game finished(win or lose)
        return self.evaluationFunction(state)

    # When all ghosts moved, it's pacman's turn
    if agentIndex == state.getNumAgents() - 1:
        return min(maxValue(state.generateSuccessor(agentIndex, action),
depth) for action in legalActions)
    else:
        return min(minValue(state.generateSuccessor(agentIndex, action),
agentIndex + 1, depth) for action in
                   legalActions)
```

ابتدا لیست legal actions را برای یک اجنت میگیریم، اگر legal action ای موجود نبود یعنی بازی تمام شده چه برد و چه باخت داده باشیم. گفته بودیم مقدار ایندکس پکمن 0 است پس چک میکنیم اگر تعداد اجنت ها منهای 1 برابر ایندکس باشد یعنی نوبت پکمن است و دیگر روحی در بازی نیست پس حداقل مقداری که تابع maxValue برمیگرداند را برای آن میخواهیم وگرنه یعنی روحی در بازی هست پس حداقل مقدار را برای آن برمیگردانیم

```
def maxValue(state, depth):
    legalActions = state.getLegalActions(0)
    if not legalActions or depth == self.depth:
        return self.evaluationFunction(state)

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

bestAction = max(gameState.getLegalActions(0),
                 key=lambda action: minValue(gameState.generateSuccessor(0,
action), 1, 1))
return bestAction
```

در تابع maxvalue، legal actionsرا برای پکمن پیدا میکنیم روند مشابهی با تابع قبلی پیش میرود اما اینجا حداکثر مقداری که تابع minValue برای حرکت پکمن پیدا میکند را برمیگردانیم

نهایتا بهترین عمل برگردانده میشود.

سوال: وقتی پکمن به این نتیجه برسد که مردن آن اجتناب ناپذیر است، تلاش می‌کند تا به منظور جلوگیری از کم شدن امتیاز، زودتر ببازد. این موضوع را می‌توانید با اجرای دستور زیر مشاهده کنید:

```
python pacman.py -p MinimaxAgent -l trappedClassic -a depth=3
```

بررسی کنید چرا پکمن در این حالت به دنبال باخت سریع‌تر است؟

طبیعتا اگر راه سودآوری نباشد صفر بهتر از ضرر کردن است بعبارتی با ادامه دادن پکمن تنها ضرر میکند و نتیجتا ضرر کمتر را ترجیح خواهد داد

3) حرس آلفا و بتا

در این قسمت بنابر کد موجود در دستور کار پیش میرویم، ابتدا تابع minValue را تعریف میکنیم:

```
def minValue(state, agentIndex, depth, a, b):
    legalActions = state.getLegalActions(agentIndex)
    if not legalActions:
        return self.evaluationFunction(state)

    v = Infinity
    for action in legalActions:
        newState = state.generateSuccessor(agentIndex, action)
        # Is it the last ghost?
        if agentIndex == state.getNumAgents() - 1:
            newV = maxValue(newState, depth, a, b)
        else:
            newV = minValue(newState, agentIndex + 1, depth, a, b)

        v = min(v, newV)
        if v < a:
            return v
        b = min(b, v)
    return v
```

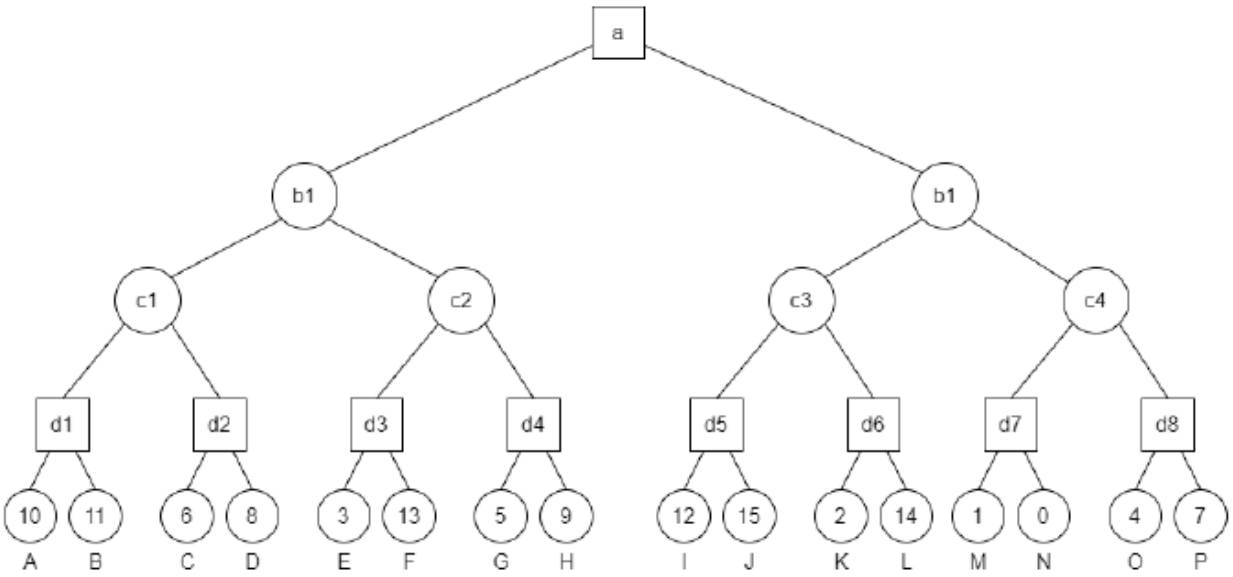
سپس تابع maxValue پیاده سازی میشود:

```
def maxValue(state, depth, a, b):
    legalActions = state.getLegalActions(0)
    if not legalActions or depth == self.depth:
        return self.evaluationFunction(state)

    v = -Infinity
    # For enable second play pruning
    if depth == 0:
        bestAction = legalActions[0]
    for action in legalActions:
        newState = state.generateSuccessor(0, action)
        newV = minValue(newState, 0 + 1, depth + 1, a, b)
        if newV > v:
            v = newV
            if depth == 0:
                bestAction = action
        if v > b:
            return v
        a = max(a, v)

    if depth == 0:
        return bestAction
    return v
```

سپس با فراخوانی تابع maxValue با توجه به اینکه پکمن میخواهد اجرا کند بهترین عمل را برمیگردانیم.

```
bestAction = maxValue(gameState, 0, -Infinity, Infinity)
return bestAction
```

سوال: فرض کنید درخت زیر یکی از تست‌های داده شده به الگوریتم آلفا-بتا شما است. گره‌های مربوط به پکمن با مربع و گره‌های هر روح با دایره نمایش داده شده است. در وضعیت فعلی پکمن دو حرکت مجاز دارد، یا می‌تواند به سمت راست حرکت کرده و وارد زیر درخت 2b شود و یا به سمت چپ حرکت کرده و وارد زیر درخت 1b شود. الگوریتم آلفا-بتا را تا عمق ۴ روی درخت زیر اجرا کرده و مشخص کنید کدام گره‌ها و به چه دلیل هرس می‌شوند. همچنین مشخص کنید در وضعیت فعلی، حرکت بعدی پکمن باید به سمت راست باشد یا چپ؟



از چپ ترین شاخه شروع میکنیم بین 10و11 یازده را انتخاب میکنیم (مقدار حداکثری برای پکمن) در گره والد روح میخواهد حداقل انتخاب کند 11 بزرگتر از هشت است و شرط حرس در چنین شرایطی بزرگتر بودن شاخه است پس حرسی انجام نشده هشت بالا می آید نهایتا روح c1 8 را انتخاب میکند. هشت همچنان بزرگتر از 3 است پس حرسی اینجا هم هنوز انجام نمیشود بین 3 و 13 سیزده بالا می آید. در مرحله بعدی هم روح داریم پس 13 که بزرگ تر از 5 است حرسی انجام نمیدهد بین 5 و 9 نه انتخاب شده و بالا می آید و روحc2 9 را انتخاب میکند نهایتا روح b1 با مقدار نه بالا میرود. حال سراغ دوازده میرویم که بزرگتر از

9 بوده و همچنان حرسی انجام نمیشود، بین 12 و 15 پانزده بالا می آید و چون از 2 بزرگتر است باز هم حرصی نداریم 14 بالا رفته و روح 14c3 را بالا میبرد که بزرگتر از 1 است همچنان حرسی نداریم بین 1 و 0 یک بالا میرود 1 اما از 4 بزرگتر نیست پس شاخه مربوط به انتخاب d8حرس میشود و انتخاب c4 یک خواهد بود و b1 هم یک را بالا میبرد حال پکمن a بین 9 و 1 نه را انتخاب میکند.

پس انتخاب پکمن سمت چپ خواهد بود

سوال: آیا در حالت کلی هرس آلفا-بتا قادر است که مقداری متفاوت با مقدار به دست آمده بدون هرس را در ریشه درخت تولید کند؟ در گره‌های میانی چطور؟ به طور خلاصه دلیل خودتان را توضیح دهید.

خیر، امکان ندارد که مقادیر مختلفی بدست آید اما درمورد گره های میانی این مورد برقرار نیست و جواب میتواند متفاوت باشد چون هدف ما پیدا کردن بهترین پاسخ نیست و دنبال این هستیم تا درصورت امکان حتما پاسخ داشته باشیم و مواردی را درنظر میگیریم که در روش هرس آلفا بتا از آنها صرف نظر کرده بودیم پس دخیل شدن این موارد تاثیر گذار بوده و امکان دارد به پاسخ مشابهی نرسیم.

4) مینماکس احتمالی

کد این بخش مشابه کدیست که برای الگوریتم expectimax تعریف کرده بودیم.

سوال: همانطور که در سوال دوم اشاره شد روش مینیماکس در موقعیتی که در دام قرار گرفته باشد خودش اقدام به باختن و پایان سریع‌تر بازی می‌کند ولی در صورت استفاده از مینیماکس احتمالی در ۵۰ درصد از موارد برنده می‌شود. این سناریو را با هر دو دستور زیر امتحان کنید و درستی این گزاره را نشان دهید. همچنین دلیل این تفاوت در عملکرد مینیماکس و مینیماکس احتمالی را توضیح دهید.

```
python pacman.py -p AlphaBetaAgent -l trappedClassic -a depth=3 -q -n 10

python pacman.py -p ExpectimaxAgent -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
```

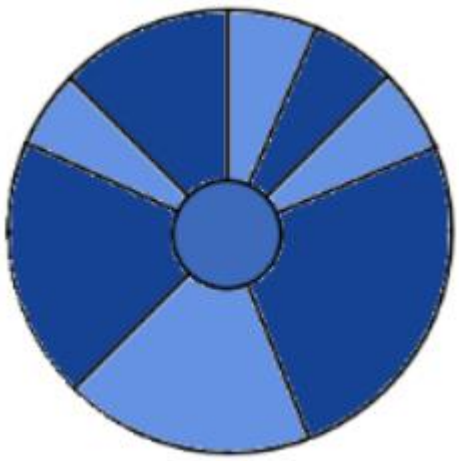
```
Pacman died! Score: -502
Pacman died! Score: -502
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 532
Pacman died! Score: -502
Pacman died! Score: -502
Pacman died! Score: -502
Pacman died! Score: -502
Pacman died! Score: -502
Pacman emerges victorious! Score: 532
Average Score: -191.8
Scores:        -502.0, -502.0, 532.0, 532.0, -502.0, -502.0, -502.0, -502.0, -502.0, 532.0
Win Rate:      3/10 (0.30)
Record:        Loss, Loss, Win, Win, Loss, Loss, Loss, Loss, Loss, Win
```

همانطور که مشاهده میشود در استفاده از الگوریتم expectimax دربرخی موارد به جواب میرسیم زیرا در این الگوریتم مینماکس ما دنبال راه حلی بهینه هستیم اما در این الگوریتم بهینگی صرفا هدف ما نیست و امکان پیدا کردن پاسخ را داریم چون جواب های غیربهینه را هم

به شکلی دخیل کرده ایم که ممکن است نهایتا به پیدا کردن راه حلی منجر شود درصورتی که به دنبال بهینگی بودن ممکن است نهایتا راه حلی پیدا نکند بعبارتی شاید با ضرر یا با شرایطی غیرایده ال به جواب برسیم اما اگر صرفا دنبال بهینگی باشیم شاید جوابی در نهایت پیدا نکنیم که ایده ال باشد.

سوال: الگوریتم رولت ویل را بررسی کنید و بیان کنید که انتخاب هر کروموزوم در این الگوریتم بر چه اساسی است؟ اگر در بازی پکمن خودمان از آن استفاده کنیم، چه معیاری برای انتخاب هر **action** مناسب است؟ بر فرض اگر نیاز بود تا با کمک الگوریتم رولت ویل بیش‌تر از یک حالت انتخاب شود (با کمک مقدار تابع ارزیابی برای هر حالت) و درخت با توجه به این دو حالت گسترش پیدا کند و حالت‌های بعدی آن‌ها هم بررسی شوند (تا بتوانیم برای حالت بعدی انتخاب بهتری داشته باشیم)، چه راهی به نظر شما منطقی می‌باشد؟

در این الگوریتم برای هر کروموزوم یک آرایه باینری که نشان دهنده صفات و ویژگی های آن است تعریف میشود سپس با توجه به چگونگی عملکرد آن تابعی مثل evaluation function امتیازی را به آن تخصیص میدهد تحت عنوان f، حالا دنبال انتخاب این کروموزم ها به عنوان واحد هستیم ولی اولویت انتخاب آنها به امتیازشان نیست بلکه به نسبت امتیازی که گرفته اند نسبت به باقی کروموزم هاست. اگر در بازی خودمان از ان استفاده کنیم خروجی تابع evaluation function را بدست اورده سپس نسبت این خروجی به همه خروجی ها حساب کرده و معیار قرار میدهیم تا عمل کنیم. برای چند انتخاب میتوان نسبت های بدست امده را سورت کرد و همانند عملکرد رولت ویل مواردی را انتخاب کرد که بزرگترین مقدارها را دارند



5) تابع ارزیابی

```
6) position = currentGameState.getPacmanPosition()
   foods = currentGameState.getFood().asList()
   closestFoodDis = min(manhattanDistance(position, food) for food in
   foods) if foods else 0.5
   score = currentGameState.getScore()

   evaluation = 1.0 / closestFoodDis + score
   return evaluation
```

سوال: تفاوت‌های تابع ارزیابی پیاده شده در این بخش را با تابع ارزیابی بخش اول بیان کنید و دلیل عملکرد بهتر این تابع ارزیابی را بررسی کنید.

در بخش اول با بررسی موارد مختلفی نمره دهی میکردیم اما این نمره دهی نسبی نبود، بعبارتی برحسب تصور ما بود و گسسته و مستقل بود که این مشکلی ایجاد میکند از قبیل اینکه اکشن stop اولویت برابری با باقی اکشن ها دارد و درموقعیتی که امکان حرکت را دارد ساکن میماند، برای نمره دهی بهتر شاید لازم باشد که ترکیبی از المان های مورد تصورمان را داشته باشیم و معیار مقایسه قراردهیم.