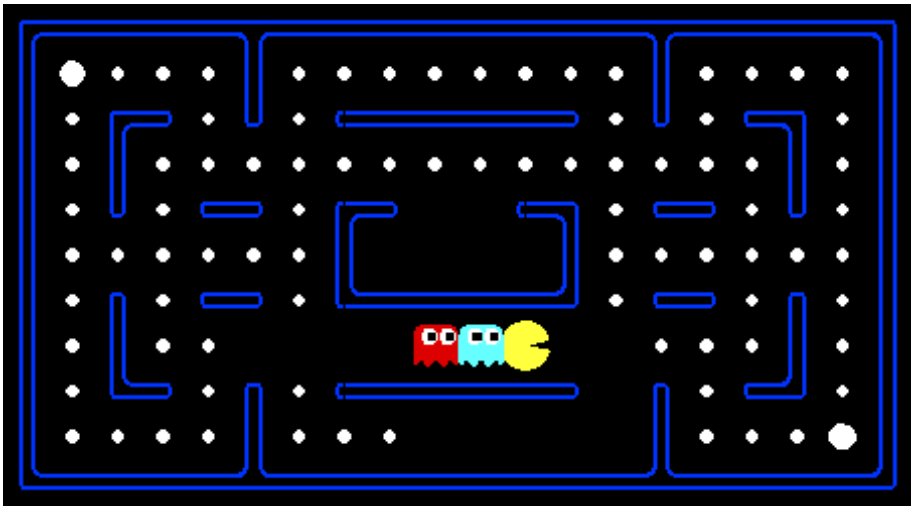
Pacman - Multi agent



سیدمحمدحسین هاشمی ۴۰۲۲۳۶۳۱۴۳

دی ۱۴۰۲

فهرست مطالب

۱ توابع هیوریستیک (**heuristic**) ۲
۱.۱ heuristicFood ۲
۲.۱ heuristicLastFood ۲
۳.۱ heuristicGhost ۳
۴.۱ heuristicScaredGhost ۳

۲ پردازش **Minimax** ۴
۱.۲ Handler ۴
۲.۲ calcValue ۴

۳ **getAction** ۵

۴ **output** ۵

۱ توابع هیوریستیک (heuristic)

از این توابع برای امتیاز دهی دقیق‌تر حالت ها استفاده می‌کنیم که هم حرکت‌های دقیق‌تری داشته باشیم و هم اینکه به لیل عمق محدود درخت بازی عامل در صورت عدم حضور عامل تاثیرگذار در امتیاز به دلیل بکسان بودن حرکت ها عمل Stop را بصورت مداوم انتخاب می‌کند تا وقتی عامل تاثیر گذاری وارد محدوده بررسی عامل شود و در این صورت عامل بهترین بازی ممکن را انجام نمی‌دهد که برای هدایت عامل از این توابع استفاده می‌کنیم.

۱.۱ heuristicFood

multiAgent.py

```python
def heuristicFood(self, gameState: GameState):
    position = gameState.getPacmanPosition()
    distance = float('inf')
    for i in gameState.getFood().asList():
        distance = min(distance, manhattanDistance(position, i))
    return 50 - distance
```

این تابع بر اساس نزدیکی Pacman به نزدیک ترین خانه غذا امتیاز می‌دهد.

۲.۱ heuristicLastFood

multiAgent.py

```python
def heuristicLastFood(self, gameState: GameState):
    foods = gameState.getFood().asList()
    if len(foods) == 1:
        return 100
    else:
        return 1
```

این تابع در صورتی که تنها یک غذا باقی مانده باشد عدد 100 و در غیر اینصورت عدد 1 را برمیگرداند که از آن به عنوان ضریبی برای ۱.۱ استفاده می‌کنیم.

heuristicGhost ۳.۱

multiAgent.py

```
def heuristicGhost(self, gameState: GameState):
    position = gameState.getPacmanPosition()
    distance = []
    for i in gameState.getGhostStates():
        if i.scaredTimer > 4:
            distance.append(-manhattanDistance(position, i.getPosition()))
        else:
            distance.append(manhattanDistance(position, i.getPosition()))
    return 20 + min(distance)
```

این تابع بر اساس نزدیکی به روح‌های بازی امتیاز بندی می‌کند و دو حالت دارد:

- نزدیک بودن به روح ترسیده امتیاز بالاتری دارد.
- نزدیک بودن به روح در حالت معمول امتیاز کمتری دارد.

heuristicScaredGhost ۴.۱

multiAgent.py

```
def heuristicScaredGhost(self, gameState: GameState):
    position = gameState.getPacmanPosition()
    distance = float('inf')
    status = False
    for i in gameState.getGhostStates():
        if i.scaredTimer > 4:
            status = True
            distance = min(distance, manhattanDistance(position, i.getPosition()))
    return 20 - distance if status else 0
```

این تابع برای نزدیکی به نزدیک ترین روح ترسیده امتیاز دهی می‌کند و در غیر اینصورت امتیاز 0 را برمی‌گرداند.

۲ پردازش Minimax

۱.۲ Handler

multiAgent.py

```
def Handler(self, gameState, agentIndex, depth, startTime, alpha, beta, currentGameState):
    oldFoods = gameState.getNumFood()
     if gameState.isWin():
        return 100000, None
    elif gameState.isLose():
         return -100000, None
     elif depth == self.depth or (time.time() - startTime) >= self.time_limit:
         score = gameState.getScore()
         return score + (score / (self.depth)) * (oldFoods - gameState.getNumFood()) + (
                self.heuristicLastFood(currentGameState) * self.heuristicFood(gameState)) + (0.3 * self.heuristicGhost(gameState)), None
    else:
        if agentIndex == 0:
            lState = self.calcValue(gameState, True, depth, agentIndex, startTime, alpha, beta, currentGameState)
            return lState[0] - 1, lState[1]
        else:
            lState = self.calcValue(gameState, False, depth, agentIndex, startTime, alpha, beta, currentGameState)
            return lState[0], lState[1]
```

این تابع بصورت بازگشتی اجرا شده و مانند درخت Minimax حالت‌های ممکن را بررسی می‌کند و بهترین اکشن را با استفاده از بخش ۲.۲ محاسبه و برمی‌گرداند. همچنین در این تابع دو محدودیت زمان پردازش و عمق نیز قرار داده‌ایم. این درخت برای اخرین مقادیر ارزش ها را براساس حالت بازی و همچنین ارزش وضعیت‌ها و مقادیر بخش ۱ تولید می‌کند.

۲.۲ calcValue

multiAgent.py

```
def calcValue(self, gameState: GameState, isMax: bool, depth, agentIndex, startTime, alpha, beta, currentGameState):
    score = float('-inf') if isMax else float('inf')
    actions = []
    legalAction = gameState.getLegalActions(agentIndex)
    for action in legalAction:
        successor = gameState.generateSuccessor(agentIndex, action)
        nextAgent = (agentIndex + 1) % gameState.getNumAgents()
        nextAgentDepth = depth + 1 if (agentIndex + 1) == gameState.getNumAgents() else depth
        nextAgentsScore = self.Handler(successor, nextAgent, nextAgentDepth, startTime, alpha, beta, currentGameState)[0]
        if isMax:
            if score < nextAgentsScore:
                score = nextAgentsScore
                actions = [action]
            elif score == nextAgentsScore:
                 actions = actions + [action]
         else:
             if score > nextAgentsScore:
                score = nextAgentsScore
                actions = [action]
             elif score == nextAgentsScore:
                  actions = actions + [action]
         if isMax:  # AlphaBetaPruning
             if score > beta:
                return score, random.choice(actions)
        else:
             if score < alpha:
                return score, random.choice(actions)
    return score, random.choice(actions)
```

این تابع همان بخش اجرایی برای پردازش درخت Minimax است که توسط بخش ۱.۲ فراخوانی می‌شود. در این تابع علاوه بر پردازش گفته شده حرص آلفابتا (AlphaBeta Pruning) نیز انجام می‌شود.

۳ getAction

multiAgent.py

```
def getAction(self, gameState: GameState):
    score, action = self.Handler(gameState, 0, 0, time.time(), float("-inf"), float("inf"), gameState)

    nextState = gameState.generateSuccessor(0, action)

    if nextState.isWin() or nextState.isLose():  # output
        self.output(nextState)

    return action
```

این تابع وضعیت فعلی را به بخش ۱.۲ داده و حرکتی فعلی Pacman را دریافت و انرا برمی‌گرداند.

۴ output

multiAgent.py

```
def output(self, nextState: GameState):
    output = "Pacman Game\n"
    if nextState.isWin():
         output += "Result: Win\n"
    elif nextState.isLose():
         output += "Result: Lose\n"
         output += "Foods: " + str(nextState.getNumFood())
    output += "Score: " + str(nextState.getScore()) + "\n"
    output += "Agents: " + str(nextState.getNumAgents()) + "\n"
    output += "Depth: " + str(self.depth) + "\n"
    output += "Ghosts: " + str(nextState.getNumAgents() - 1) + "\n"
     output += "----------------\n"
     f = open("../Outputs/The_Phoenix-UIAI4021-PR4.txt", 'a')
     f.write(output)
     f.close()
```

در نهایت این تابع خروجی به فرم زیر تولید و ذخیره می‌کند.

```
Pacman Game
Result: Win
Score: 1730.0
Agents: 2
Depth: 4
Ghosts: 1
----------------
```