CSE4185 기초 인공지능 과제 2 보고서

20190554 송재현

MinimaxAgent

Code

```
class MinimaxAgent(AdversialSearchAgent):
    """
    [문제 01] MiniMaxAgent의 Action을 구현하시오.
    (depth와 evaluation function은 위에서 정의한 self.depth and
self.evaluationFunction을 사용할 것.)
     """
    ######################## Write Your Code Here
################################
    def Action(self, gameState):

        def pacman_max(state, depth):
            if state.isWin() or state.isLose() or depth == self.depth:
                return self.evaluationFunction(state)
            value = float("-inf")

            for action in state.getLegalActions(0):
                new_state = state.generateSuccessor(0, action)
                value = max(value, ghost_min(new_state, depth,
state.getNumAgents()-1))

            return value

        def ghost_min(state, depth, agent_index):
            if state.isWin() or state.isLose() or depth == self.depth:
                return self.evaluationFunction(state)
            value = float("inf")

            for action in state.getLegalActions(agent_index):
                new_state = state.generateSuccessor(agent_index, action)

                if(agent_index > 1):
                    value = min(value, ghost_min(new_state, depth,
agent_index - 1))
                else:
                    value = min(value, pacman_max(new_state, depth + 1))

            return value

        result_action = None
        value = float("-inf")

        for action in gameState.getLegalActions(0):
            new_state = gameState.generateSuccessor(0, action)
            temp_value = ghost_min(new_state, 0,
gameState.getNumAgents()-1)
```

```
            if temp_value > value:
                value = temp_value
                result_action = action

        return result_action
```

```
#########################################################################
###
```

실행 결과

```
Win Rate: 64% (642/1000)
Total Time: 73.9837658405304
Average Time: 0.0739837658405304
======================================
```

python pacman.py -p MinimaxAgent -m minimaxmap -a depth=4 -n 1000 -q

위의 명령어 실행 시 승률이 64%로 50~70%를 만족함을 보인다.

구현 방법

기존의 MinMax 알고리즘과 유사한 형태로 구현하였다. 현재 state의 상태와 depth를 확인하여 win하였거나 lose 즉 terminal state이거나 혹은 depth가 처음에 설정한 depth에 도달했다면 점수를 반환하였다. 또한 팩맨에서의 max와 ghost들에서의 min값을 구하는 것을 다른 함수를 이용하여 구현하였는데 pacman_max함수에서는 현재 state에서부터의 successor_state들을 구해서 ghost_min 함수를 호출한 뒤 이 중 가장 점수가 높은 선택지를 선택하였고 ghost_min에서는 이와 반대로 구현하였다. 다만 ghost_min 함수에서는 ghost가 pacman처럼 하나만 있는 것이 아니기 때문에 agent_index를 통해서 ghost들을 구분하도록 하였고 index를 감소시켜 만약 index가 1일 경우에는 다시 pacman_max를 호출하여 계산하였다. 이후 이 함수들을 이용하여 점수가 더 높은 state가 나온다면 value값과 action값을 바꾸도록 하였고 이렇게 해서 구한 action값을 return하였다.

AlphaBetaAgent

Code

```
class AlphaBetaAgent(AdversialSearchAgent):
    """
    [문제 02] AlphaBetaAgent의 Action을 구현하시오.
    (depth와 evaluation function은 위에서 정의한 self.depth and
self.evaluationFunction을 사용할 것.)
     """
    def Action(self, gameState):
    ######################## Write Your Code Here
################################

        def pac_max(state, depth, a, b):
            if state.isWin() or state.isLose() or depth == self.depth:
                return self.evaluationFunction(state)
            value = float("-inf")

            for action in state.getLegalActions(0):
                new_state = state.generateSuccessor(0, action)

                value = max(value, ghost_min(new_state, depth,
state.getNumAgents()-1, a, b))
                if value >= b:
                    return value
                a = max(a, value)

            return value


        def ghost_min(state, depth, agent_index, a, b):
            if state.isWin() or state.isLose() or depth == self.depth:
                return self.evaluationFunction(state)
            value = float("inf")

            for action in state.getLegalActions(agent_index):
                new_state = state.generateSuccessor(agent_index, action)

                if agent_index > 1:
                    value = min(value, ghost_min(new_state, depth,
agent_index - 1, a, b))
                else:
                    value = min(value, pac_max(new_state, depth + 1, a,
b))
                if value <= a:
                    return value
                b = min(b, value)
            return value

        result_action = None
        value = float("-inf")
        a = float("-inf")
        b = float("inf")
```

```
        for action in gameState.getLegalActions(0):
            temp_value = ghost_min(gameState.generateSuccessor(0,
action), 0, gameState.getNumAgents()-1, a, b)
            if temp_value > value:
                value = temp_value
                result_action = action

        return result_action


##########################################################################
###
```

실행 결과

```
Win Rate: 15% (47/300)
Total Time: 290.9742147922516
Average Time: 0.9699140493075052
========================================
----- END  MiniMax (depth=3) For Medium Map

Win Rate: 11% (35/300)
Total Time: 158.11082887649536
Average Time: 0.5270360962549845
========================================
----- END  AlphaBeta (depth=3) For Medium Map



Win Rate: 35% (354/1000)
Total Time: 38.5490186214447
Average Time: 0.038549018621444706
========================================
----- END  MiniMax (depth=4) For Minimax Map


Win Rate: 36% (364/1000)
Total Time: 20.37481451034546
Average Time: 0.02037481451034546
========================================
----- END  AlphaBeta (depth=4) For Minimax Map
```

Depth = 3일 때 Medium Map에서의 결과와 Depth = 4일 때 Minimax Map에서의 결과 모두 MinimaxAgent에 비해서 AlphaBetaAgent가 실행시간이 더 빠른 것을 통해 후자가 더 효율적임을 확인할 수 있다.

구현 방법

위에서 구한 MiniMax 알고리즘과 유사한 형태로 구현하였다. pac_max와 ghost_min 함수가 동작하는 것은 대부분 유사하지만 pac_max 함수에서 new_state에서 구한 값이 value에 할당된 후 이 값이 Beta 값보다 크거나 같다면 그 즉시 value 값을 return 하고 함수를 종료하도록 하였다. 여기서 alpha 값을 계속 가장 큰 value 값으로 업데이트 시켜준다. ghost_min 함수에서도 마찬가지로 new_state에서 구한 값이 value에 할당된 후 이 값이 alpha 값보다 작거나 같다면 그 즉시 value 값을 return 하고 함수를 종료하도록 하였다. 여기서 beta 값을 계속 가장 작은 value값으로 업데이트 시켜준다. 이런 방식을 통해 대체적으로 minimax 알고리즘과 동일하게 작동하지만 시간이 매우 줄어드는 것을 확인할 수 있다.

ExpectimaxAgent

Code

```
class ExpectimaxAgent(AdversialSearchAgent):
    """
    [문제 03] ExpectimaxAgent의 Action을 구현하시오.
    (depth와 evaluation function은 위에서 정의한 self.depth and
self.evaluationFunction을 사용할 것.)
    """
    def Action(self, gameState):
    ######################## Write Your Code Here
################################
        def pac_max(state,depth):
            if state.isWin() or state.isLose() or depth == self.depth:
                return self.evaluationFunction(state)
            value = float("-inf")

            for action in state.getLegalActions(0):
                new_state = state.generateSuccessor(0, action)
                value = max(value, ghost_min(new_state, depth,
state.getNumAgents()-1))

            return value

        def ghost_min(state, depth, agent_index):
            if state.isWin() or state.isLose() or depth == self.depth:
                return self.evaluationFunction(state)
            value_sum = 0
            cnt = 0

            for action in state.getLegalActions(agent_index):
                new_state = state.generateSuccessor(agent_index, action)
                cnt += 1

                if agent_index > 1:
                    value_sum += ghost_min(new_state, depth, agent_index
- 1)
                else:
```

```
                    value_sum += pac_max(new_state, depth + 1)

            return float(value_sum) / cnt

        result_action = None
        value = float("-inf")

        for action in gameState.getLegalActions(0):
            new_state = gameState.generateSuccessor(0, action)
            temp_value = ghost_min(new_state, 0,
gameState.getNumAgents()-1)

            if temp_value > value:
                value = temp_value
                result_action = action

        return result_action
```

```
#########################################################################
###
```

실행 결과

```
----------Game Results--------------
Average Score: 4.66
Score Results: -502, 532, 532, -502, -502, 532, -502, -502, -502, -502, -502, 532, -502, 532, -502, 53
2, -502, -502, -502, 532, 532, -502, -502, -502, -502, -502, -502, 532, -502, 532, -502, -502, -502, -
502, 532, 532, 532, 532, -502, 532, -502, 532, 532, -502, -502, -502, 532, 532, -502, 532, 532, -502,
532, -502, 532, -502, 532, 532, 532, 532, 532, -502, -502, 532, -502, 532, -502, -502, 532, -502, -502
, 532, 532, -502, 532, -502, -502, -502, 532, -502, -502, 532, 532, 532, 532, 532, 532, -502, -502, 53
2, -502, 532, 532, 532, 532, 532, -502, -502, 532, 532
Record: Lose, Win, Win, Lose, Lose, Win, Lose, Lose, Lose, Lose, Lose, Win, Lose, Win, Lose, Win, Lose
, Lose, Lose, Win, Win, Lose, Lose, Lose, Lose, Lose, Lose, Win, Lose, Win, Lose, Lose, Lose, Lose, Wi
n, Win, Win, Win, Lose, Win, Lose, Win, Win, Lose, Lose, Lose, Win, Win, Lose, Win, Win, Lose, Win, Lo
se, Win, Lose, Win, Win, Win, Win, Win, Lose, Lose, Win, Lose, Win, Lose, Lose, Win, Lose, Lose, Win,
Win, Lose, Win, Lose, Lose, Lose, Win, Lose, Lose, Win, Win, Win, Win, Win, Win, Lose, Lose, Win, Lose
, Win, Win, Win, Win, Win, Lose, Lose, Win, Win

Win Rate: 49% (49/100)
Total Time: 0.2989845275878906
Average Time: 0.002989845275878906
```

Expectimax Agent가 50% 내외의 승률인 49%의 승률을 보이고 이긴 경우 532의 score를 패배한 경우 -502의 score를 출력하는 것을 확인할 수 있다.

구현 방법

위에서의 MiniMax 알고리즘과 유사한 형태로 구현하였다. pac_max에서는 항상 value를 max값으로 할당해주는 것은 동일하나 이 알고리즘은 상대가 항상 optimal한 선택을 하지는 않는다고 생각하기 때문에 ghost_min 함수에서의 수정이 일부분 있다. ghost_min 함수에서 값을 return 할 때 기존의 방식과는 다르게 후보 node들의 value 값의 평균 값을 return 한다. 그렇게 된다면 선택할 때에도 이들의 평균값으로 선택하는 것이기 때문에 optimal한 solution이 도출되지는 않지만 상대가 실수를 할 경우 더 큰 기댓값을 얻는다고 할 수 있다.