기초 인공지능

#Assignment02 – Pacman

Adversarial Search 를 활용한 Pacman 구현하기

1. 각 알고리즘마다 구현한 방법에 대한 설명

1.1) MinimaxAgent

먼저 action 함수에서 첫 팩맨의 legalactions에 대해 반복문을 통해min_value 함수를 불러서 각 최솟값들에 대한 최댓값을 찾는다. 이때 maximum 변수는 처음 -무한대 값으로, 각 최솟값들이 maximum보다 값이 클 때 maximum 값들을 해당 값으로 바꾸고 그때의 action을 result_move에 저장한다. 반복문을 탈출했다면 result_move를 return 한다.

Min_Value값은 처음 해당 state가 종단 state인지 isLose와 isWin 함수를 통해 검사하고 종단 state라면 evaluationFuctoin을 통해 값을 return한다. 종단 state가 아니라면 해당 ghost의 legal action에 대해 반복문을 진행하고 만약 현재 ghost의 인덱스가 마지막 ghost 인덱스라면(getNumAgent를 통해 몇 개의 ghost가 있는 알 수 있다) 다음은 pacman의 차례이기 때문에 Max_Value의 값과 v 값 중 작은 값을 v에 저장하고, 아니라면 다시 한번 ghost 차례이기 때문에 Min_Value의 값과 v 값 중 작은 값을 v에 저장한다. 반복문을 탈출하면 v 값을 return 한다. 이때 pacman의 Max_Value함수를 부를때는 depth에 1을 더해주고 index는 0으로 한다. Depth의 경우 pacman과 모든 ghost의 move가 끝났을 때 1이 늘어나고, pacman의 인덱스는 무조건 0이기 때문이다. Ghost의 Min_Value함수를 부른다면 depth는 그대로, index에 1을 더해서 부른다.

Max_Value값은 마찬가지로 처음 해당 state가 종단 state인지 isLose와 isWin 함수 그리고 주어진 depth값과 현재 depth값의 비교를 통해 검사하고 종단 state라면 evaluationFuctoin을 통해 값을 return한다. 종단 state가 아니라면 해당 pacman의 legal action에 대해 반복문을 진행하고 v값과 Min_Value 값 중 큰 값을 v에 저장한다. 이때 Min_Value는 ghost의 move에 대한 값으로 depth는 그대로, index값을 1 더해서 부른다. Depth의 경우 pacman과 모든 ghost의 move가 끝났을 때만 1이 늘어나기 때문이다. 반복문을 탈출했다면 v값을 리턴한다.

1.2) AlphaBeta

먼저 action 함수에서 alpha 값을 -무한대, beta 값을 무한대, maximum변수 값을 -무한대로 초기화 한다. 이후 첫 팩맨의 legal actions에 대해 반복문을 통해 ab_min_value 함수를 불러서 각 최솟값들에 대한 최댓값을 찾는다. 각 최솟값들이 maximum보다 값이클 때 maximum 값들을 해당 값으로 바꾸고 그때의 action을 result_move에 저장한다. 또한 alpha 값을 v로 바꾼다. 즉 Max의 alpha를 설정하는 것이다. 반복문을 탈출했다면 result_move를 return 한다.

AB_Min_Value값은 처음 해당 state가 종단 state인지 isLose와 isWin 함수를 통해 검사하고 종단 state라면 evaluationFuctoin을 통해 값을 return한다. 종단 state가 아니라면 해당 ghost의 legal action에 대해 반복문을 진행하고 만약 현재 ghost의 인덱스가 마지막

ghost 인덱스라면(getNumAgent를 통해 몇 개의 ghost가 있는 알 수 있다) 다음은 pacman의 차례이기 때문에 vv 변수에 AB_Max_Value의 값을 저장하고 아니라면 다시한번 ghost 차례이기 때문에 AB_Min_Value의 값을 vv에 저장한다. 이후 vv가 v보다 작다면 v값을 wv로 바꾸고, v가 alpha보다 작다면 v를 리턴하고 함수를 종료한다. 즉 pruning이 되는 시점이다. Min에서 나올 수 있는 값이 max의 alpha보다 작다면 min에서 나오는 값이 의미가 없기 때문에 거기서 가지치기를 해주는 것이다. beta에는 beta와 v값 중 작은 값을 저장한다. 반복문을 탈출했다면 v를 리턴한다. 이때 pacman의 Max_Value함수를 부를때는 depth에 1을 더해주고 index는 0으로 한다. Depth의 경우 pacman과 모든 ghost의 move가 끝났을 때 1이 늘어나고, pacman의 인덱스는 무조건 이이기 때문이다. Ghost의 Min_Value함수를 부른다면 depth는 그대로, index에 1을 더해서 부른다.

AB_Max_Value값은 마찬가지로 처음 해당 state가 종단 state인지 isLose와 isWin 함수 그리고 주어진 depth값과 현재 depth값의 비교를 통해 검사하고 종단 state라면 evaluationFuctoin을 통해 값을 return한다. 종단 state가 아니라면 해당 pacman의 legal action에 대해 반복문을 진행하고 w 변수에 AB_Min_Value 값을 저장한 후 v가 w보다 작을 때 v값을 vv로 바꿔준다. v가 beta보다 크다면 v를 리턴하고 함수를 종료한다. 즉 pruning이 되는 시점이다. max에서 나올 수 있는 값이 min의 beta보다 크다면 max에서 나오는 값이 의미가 없기 때문에 거기서 가지치기를 해주는 것이다. alpha에는 alpha와 v값 중 큰 값을 저장한다. 이때 Min_Value는 ghost의 move에 대한 값으로 depth는 그대로, index값을 1 더해서 부른다. Depth의 경우 pacman과 모든 ghost의 move가 끝났을 때만 1이 늘어나기 때문이다. 반복문을 탈출했다면 v값을 리턴한다.

1.3) Expecti

먼저 action 함수에서 첫 팩맨의 legalactions에 대해 반복문을 통해 EX_Chance_πalue 함수를 불러서 각 최솟값들에 대한 최댓값을 찾는다. 이때 maximum 변수는 처음 -무한 대 값으로, 각 최솟값들이 maximum보다 값이 클 때 maximum 값들을 해당 값으로 바꾸고 그때의 action을 result_move에 저장한다. 반복문을 탈출했다면 result_move를 return 한다.

EX_Chance_Value 값은 위의 두 알고리즘의 Min_Value와 유사하게 돌아간다. 다만 값에 확률을 구하여 더한 것이다. 처음 해당 state가 종단 state인지 isLose와 isWin 함수를 통해 검사하고 종단 state라면 evaluationFuctoin을 통해 값을 return한다. v값으로 0으로 초기화 한 후 해당 ghost의 legal action에 대해 반복문을 진행하고 만약 현재 ghost의 인덱스가 마지막 ghost 인덱스라면(getNumAgent를 통해 몇 개의 ghost가 있는 알 수 있다) 다음은 pacman의 차례이기 때문에 EX_Max_Value의 값을 v에 더해주고 아니라면 다시 한번 ghost 차례이기 때문에 EX_Chance_Value의 값을 v에 더해준다. 반복문을 탈출하면 result에 값을 더해왔던 v를 legalActions의 개수로 나눈 값을 저장하고 리턴한다.

이때 pacman의 EX_Max_Value함수를 부를때는 depth에 1을 더해주고 index는 0으로 한다. Depth의 경우 pacman과 모든 ghost의 move가 끝났을 때 1이 늘어나고, pacman의 인덱스는 무조건 0이기 때문이다. Ghost의 EX_Chance_Value 함수를 부른다면 depth는 그대로, index에 1을 더해서 부른다.

Max_Value값은 마찬가지로 처음 해당 state가 종단 state인지 isLose와 isWin 함수 그리고 주어진 depth값과 현재 depth값의 비교를 통해 검사하고 종단 state라면 evaluationFuctoin을 통해 값을 return한다. 종단 state가 아니라면 해당 pacman의 legal action에 대해 반복문을 진행하고 v값과 EX_Chance_Value 값 중 큰 값을 v에 저장한다. 이때 EX_Chance_Value 는 ghost의 move에 대한 값으로 depth는 그대로, index값을 1 더해서 부른다. Depth의 경우 pacman과 모든 ghost의 move가 끝났을 때만 1이 늘어나기 때문이다. 반복문을 탈출했다면 v값을 리턴한다.

2. 실행 캡처 화면

[1] 위에서 제시한 Minimax Agent 명령어의 승률 출력 화면



[2] time_check.py에서 출력된 실행시간 캡처 화면

(small map (minimax, alphabeta), medium map (minimax, alphabeta), minimax map (minimax, alphabeta))

```
명령 프롬프트
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            0
                                                                                                    77
-365
97
-338
-376
-365
-123
-52
97
-349
-290
-365
     in Rate: 0% (2/300)
otal Time: 40.09074544906616
verage Time: 0.13363581816355388
                    END MiniMax (depth=2) For Small Map
                   START PART AlphaBeta (depth=2) For Small Map
██ 선택 명령 프롬프트
                                                                           . 365, -135, -149, -167, 367, -368, -365, -149, -338, -173, -365, -332, -356, -297, 93, /6, -370, -360, -365, -368, -382, -342, -122, -376, -149, -338, -173, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365, -365
                                                                                                                                                                                                                                                                                                                                                                        330, 130, 131, 37, 38, 395, 395, 335, 348, 31, 320, 1

0, 103, 54, 294, -293, -71, -365, -365, -376, -377, -55

-296, 78, -338, 231, -332, -342, -122, -376, 100, -301, .

, -150, -338, 50, -329, 32, 323, -365, 69, 264, 320, -78

-154, -329, -376, -332, -338, 79, -365, 377, -284, 49,
      in Rate: 0% (1/300)
otal Time: 32,43632793426514
verage Time: 0,10812109311421712
                     END AlphaBeta (depth=2) For Small Map
                                                                   Score: 639
Score: 439
Score: -240
                      icman Lose...
icman Lose...
```

```
█ 명령 프롬프트
  in Rate: 3% (10/300)
btal Time: 90.52981781959534
verage Time: 0.30176605939865114
       END MiniMax (depth=2) For Medium Map
       START PART AlphaBeta (depth=2) For Medium Map:
                                                                                                                                                                                                                - 6
  in Rate: 4% (12/300)
otal Time: 88,31338763237
verage Time: 0,29437795877456663
       END AlphaBeta (depth=2) For Medium Map
       START PART MiniMax (depth=4) For Minimax Map:
```

049 - ### 18 -

1. 1882 1483 1480 - 404 - 405 - 408 - 409 - 401 - 408 - 401 - 408 - 408 - 408 - 408 - 511 - 515 - 408 - 513 - 408

