기초 인공지능 : Assignment02(Adversarial Search를 활용한 Pacman 구현하기)

전공: 컴퓨터공학 학년: 3학년 학번: 20201635 이름: 전찬

0. 목차

1. 구현 목표

이번 과제에서는, Adversarial Search를 활용하여, Pacman game을 수행해야 한다. 이를 위해 Minimax, AlphaBeta Pruning, Expectimax 을 바탕으로 Pacman의 다음 움직임을 결정하는 search 알고리즘을 구현해야 한다.

2. 각 알고리즘마다 구현한 방법

이번 과제에서 구현해야 하는 것은, Minimax, AlphaBeta, Expectimax class 내부에 있는 Action method 이다. 각각의 방법에 따라, 정의되어 있는 evaluation function 을 바탕으로 adversarial search 을 수행하여 Pacman의 다음 움직임을 결정하는 Action method을 구현해야 한다. 따라서 총 3가지 Action method을 구현해야 한다.

- Minimax Action

minimax action 은 각각 Pacman을 max, 첫 번째 유령을 min, 두 번째 유령을 min으로 생각하여 구현할 수 있다. 따라서 일반적인 max – min – max – min ,,,, 이 반복되는 1대1 구조와 달리, max(Pacman) – min(유령1) – min(유령2) – max – min – min ,,,, 형태로 구현해야 하며, depth에 따라 위가 반복되는 횟수를 설정할 수 있도록 구현해야 한다.

따라서 이를 구현할 때, max_value, min_value 함수를 추가로 정의했다. max_value 함수는 실제 minimax agent에서 max을 수행하는 부분이며, min_value는 min을 수행하는 부분이다. 전체적인 흐름은 아래와 같다.

1. max_value 함수를 통해 Pacman의 다음 가능한 움직임에 따른, min_value을 호출한다.

2. min_value 가 max_value에 의해 호출되면, 유령1의 가능한 움직임을 판단하고, min_value을 다시 호출한다.

3. 다시 호출된 min_value는 유령2의 가능한 움직임을 판단하고, 다시 max_value 을 호출한다.

위를 반복하며, 만약 max_value의 호출에서 현재 반복의 횟수(depth)가 미리 설정해놓은 self.depth와 동일해진다면, 재귀 함수의 종료 지점이 되며, 현재 value을 계산해서 return 한다. 이러한 과정을 통해 minimax action을 구현할 수 있다.

- AlphaBeta Action

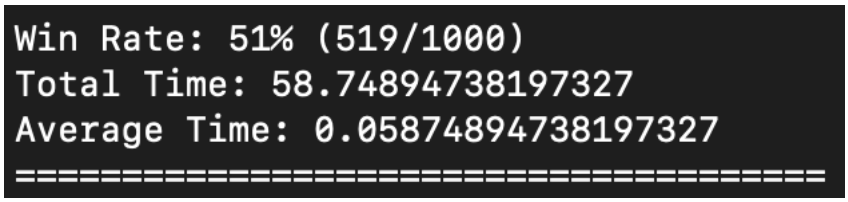
alphabeta action은 위의 minimax action에서 alpha, beta 값을 활용해 pruning을 수행해 실행 시간을 줄이는 형태이다. 따라서 각각의 max_value, min_value 함수에서 추가로 alpha, beta 값을 parameter로 전달받을 수 있도록 함수를 구현하며, max_value에서는 만약 beta보다 큰 값이 있다면,return 하며, min_value에서는 alpha보다 작은 값이 있다면 return 하는 코드를 추가해야 한다.

- Expectimax Action

expectimax action은 각각의 유령이 min을 수행하는 것이 아닌, chance을 수행하도록 변경해서 구현할 수 있다. 따라서 chance_value을 추가적으로 구현해야 하며, chance_value는 min_value에서 각각의 legal action의 evaluation 값 중 최소값을 저장하는 것이 아닌, 모든 값을 파악하며, 평균값을 return하는 형태로 구현해야 한다. 이때 평균값을 취하는 이유는, 원래 각 legal action에 대한 weight을 통해 계산하는 형태인데, weight 가 주어지지 않았기 때문에 평균값을 이용했다. 또한 min_value와 같이 유령1 인 경우에는 유령 2의 chance 값을 파악해야 하며, 유령 2의 경우에는 다음 Pacman의 max 값을 바탕으로 chance을 계산하며, 이 값을 return 해야 한다.

3-1. Minimax Agent 명령어의 승률 출력 화면

명령어 실행 결과는 아래와 같다.



<Minimax Agent 명령어 출력 결과>

따라서 50%-70% 사이의 승률을 만족함을 파악할 수 있다.

3-2. time_check.py에서 출력된 실행시간 캡쳐 화면

명령어 실행 결과는 아래와 같다.

```
Win Rate: 8% (24/300)
Total Time: 384.2093470096588
Average Time: 1.2806978233655293
=====================================
----- END  MiniMax (depth=3) For Medium Map
```

<Minimax Agent 실행 결과1>

```
Win Rate: 5% (17/300)
Total Time: 202.77003264427185
Average Time: 0.6759001088142395
=====================================
----- END  AlphaBeta (depth=3) For Medium Map
```

<AlphaBeta Pruning Agent 실행 결과1>

```
Win Rate: 62% (621/1000)
Total Time: 71.54901266098022
Average Time: 0.07154901266098022
=====================================
----- END  MiniMax (depth=4) For Minimax Map
```

<Minimax Agent 실행 결과2>

```
Win Rate: 61% (618/1000)
Total Time: 41.99380087852478
Average Time: 0.04199380087852478
=====================================
----- END  AlphaBeta (depth=4) For Minimax Map
```

<AlphaBeta Pruning Agent 실행 결과2>

따라서 Minimax 보다, AlphaBeta Pruning이 더 실행시간이 빠르며, 효율적임을 파악할 수 있다. 또한 계산상 tree에서 약 40%의 가지치기를 수행할 수 있음을 파악할 수 있다.

3-3. Expectimax Agent 명령어의 승률 및 Score 출력 화면

명령어 실행 결과는 아래와 같다.

```
----------Game Results-------------
Average Score: -26.36
Score Results: -502, -502, -502, 532, 532, -502, -502, 532, 532, -502, 532, 532, 53
2, 532, 532, -502, -502, 532, 532, 532, -502, -502, 532, -502, -502, -502, -502, -5
02, 532, -502, -502, -502, 532, 532, -502, -502, -502, -502, -502, 532, -502, -502,
 -502, 532, 532, 532, 532, 532, 532, -502, -502, -502, -502, -502, 532, 532, -502,
-502, 532, -502, -502, 532, 532, 532, 532, -502, -502, -502, -502, 532, 532, 532, 5
32, -502, -502, 532, -502, 532, -502, 532, 532, -502, -502, -502, -502, -502, 532,
-502, -502, -502, 532, 532, -502, 532, -502, 532, 532, 532, -502, 532
Record: Lose, Lose, Lose, Win, Win, Lose, Lose, Win, Win, Lose, Win, Win, Win, Win,
 Win, Lose, Lose, Win, Win, Win, Lose, Lose, Win, Lose, Lose, Lose, Lose, Lose, Win
, Lose, Lose, Lose, Win, Win, Lose, Lose, Lose, Lose, Lose, Win, Lose, Lose, Lose,
Win, Win, Win, Win, Win, Win, Lose, Lose, Lose, Lose, Lose, Win, Win, Lose, Lose, W
in, Lose, Lose, Win, Win, Win, Win, Lose, Lose, Lose, Lose, Win, Win, Win, Win, Los
e, Lose, Win, Lose, Win, Lose, Win, Win, Lose, Lose, Lose, Lose, Lose, Win, Lose, L
ose, Lose, Win, Win, Lose, Win, Lose, Win, Win, Win, Lose, Win

Win Rate: 46% (46/100)
Total Time: 0.4840993881225586
Average Time: 0.004840993881225586
========================================
```

<Expectimax Agent 명령어 실행 결과>

이를 통해 Expectimax 방법이 약 50%의 승률을 가짐을 파악할 수 있으며, 이긴 경우 532, 진 경우 -502의 score을 가짐을 파악할 수 있다.