ARTIFICIAL INTELLIGENCE ASSIGNMENT #2

2016320187 홍성윤

1. The result of autograder.py

```
Finished at 19:38:26

Provisional grades
==================
Question q1: 4/4
Question q2: 5/5
Question q3: 5/5
Question q4: 0/5
Question q5: 0/6
------------------
Total: 14/25
```

2. Discussions

★ How can alpha-beta pruning make it more efficient to explore the minimax tree?

-> MINIMAX 알고리즘은 깊이우선 탐색으로서 모든 노드들을 다 계산해야 한다. 그렇기 때문에 깊이가 깊어질수록 MINIMAX TREE에서 탐색해야 하는 노드의 수는 기하급수적으로 증가한다. 체스 정도의 복잡도를 가진 게임만 되어도 TREE의 모든 STATE 들을 탐색하는 것이 불가능하다. 알파 베타 가지치기는 더 이상 탐색해봐야 의미 없는 노드의 가지들을 잘라내어 검색속도를 훨씬 빠르게 만들어 준다. 노드들이 최적에 가까운 순서대로 정렬되어 있다면 기존의 MINIMAX TREE의 깊이의 1/2이 되도록 최적화할 수 있다. 따라서 알파 베타 가지치기를 통해, 같은 시간안에 더 깊은 노드까지 더 효율적으로 탐색할 수 있다.

★ What is the worst case and best case of Alpha-beta pruning?

-> 알파 베타 가지치기의 성능은 노드들이 어떻게 정렬되어 있는지에 따라 크게 달라진다. 노드들의 정렬된 상태에 따라서 어떤 경우에는 알파 베타 가지치기 알고리즘을 이용해도 아무런 가지치기가 이루어지지 않을 수 있다. 이 경우, 즉 최악의 경우에는 알파, 베타라는 변수 사용으로 인한 추가적인 시간이 소요될 것이고 일반적인 MINIMAX 알고리즘과 동일하게 실행될 것이다. 어떠한 게임에서 b를 가능한 행동의 경우의 수, m을 탐색 알고리즘의 평균적인 깊이라고 한다면, 최악의 경우 알고리즘의 시간 복잡도는 O(b^m) 일 것이다.

이상적인 정렬 상황에서는 알파 베타 가지치기를 통해 가지치기가 매우 여러 번 발생할 것이다. 위에서 이야기한 바와 같이 이 경우 같은 시간안에 일반적인 MINIMAX 알고리즘보다 두 배 더 깊은 노드까지 탐색 가능하다. 따라서 이 경우 알고리즘의 시간 복잡도는 O(b^m/2)이다.