

HW2

R12922016 葉丞勛

(一)

1. Reflex Agent：根據當下的情況，選擇最好的 action 去執行。在 evaluationFunction 裡，會先去判斷 pacman 下一步的位置是否會遇到鬼或是離鬼很近，如果發生上述情況的話就回傳 -99999，表示不要去選擇這個 action；若是不會遇到鬼，並且 pacman 下一步的位置上有食物的話，就回傳 99999，讓 pacman 去吃食物。如果上述情況都沒發生，就讓 pacman 去吃離下一步最近的 food。

```
### Question q1: 30/30 ###
```

```
Finished at 16:13:01
```

```
Provisional grades
```

```
=====
```

```
Question q1: 30/30
```

```
-----
```

```
Total: 30/30
```

2. Minimax：首先設計兩個 function：min()、max()：
 - (1) min：先判斷遊戲是否結束，結束了的話就回傳 self.evaluationFunction(gameState)。若是還沒結束的話，就去判斷下一層是否為鬼，如果是的話，就先將當前這層鬼合法的 actions 取出，算出這層鬼每個可能的 nextStage，然後把每個 nextStage 丟進 min() 裡面（因為下一層也是鬼），回傳 min() 的最小值作為這層鬼的分數；如果下一層不是鬼的話（表示下一層是 pacman），就一樣先將當前這層鬼合法的 actions 取出，算出這層鬼每個可能的 nextStage，然後把每個 nextStage 丟進 max() 裡面（因為下一層是 pacman），回傳 max() 的最小值作為這層鬼的分數。
 - (2) max：先判斷遊戲是否結束，結束了的話就回傳 self.evaluationFunction(gameState)。若是還沒結束的話，就將當前

這層 pacman 合法的 actions 取出，算出每個可能的 nextStage，然後把每個 nextStage 丟進 min() 裡面（因為下一層是鬼），回傳 min() 的最大值作為這層 pacman 的分數。

接著，我們先取得 pacman 從起點開始，所有可能的下一步，算出每個可能的 nextStage，然後把 nextStage 丟進 min() 裡面（因為下一層是鬼），找出 min() 的最大值，並回傳會導致 min() 有最大值的那個 action，作為當前 pacman 的下一步。

```
### Question q2: 30/30 ###  
  
Finished at 17:07:00  
  
Provisional grades  
=====br/>Question q2: 30/30  
-----br/>Total: 30/30
```

3. Alpha-Beta Pruning：首先將兩個 function：min()、max() 做一些小修改：

- (1) **min**：在把每個 nextStage 丟進 min() 或是 max() 裡面求最小值的時候，如果發現當前算出來的值小於 alpha，就直接回傳當前的值；如果沒有小於 alpha 的話，就把當前算出的最小的值設為 beta。
- (2) **max**：在把每個 nextStage 丟進 min() 裡面求最大值的時候，如果發現當前算出來的值大於 beta，就直接回傳當前的值；如果沒有大於 beta 的話，就把當前算出的最大的值設為 alpha。

接著，我們一樣先取得 pacman 從起點開始，所有可能的下一步，算出每個可能的 nextStage，然後把 nextStage 丟進 min() 裡面（因為下一層是鬼），找出 min() 的最大值，同時將算出來的最大值作為 alpha，並回傳會導致 min() 有最大值的那個 action，作為當前 pacman 的下一步。

```
### Question q3: 30/30 ###
```

```
Finished at 17:29:08
```

```
Provisional grades
```

```
=====
```

```
Question q3: 30/30
```

```
-----
```

```
Total: 30/30
```

(二)

：同（一）(1)，會根據當下的情況，選擇最好的 action 去執行。在 evaluationFunction 裡，會先去判斷 pacman 下一步的位置是否會遇到鬼或是離鬼很近，如果發生上述情況的話就回傳 -99999，表示不要去選擇這個 action；若是不會遇到鬼，並且 pacman 下一步的位置上有食物的話，就回傳 99999，讓 pacman 去吃食物。如果上述情況都沒發生，就找出 pacman 與最近的 food 的距離，回傳 (1000 - 此距離) 作為分數（因為分數越大表示越要讓 pacman 往那邊走）。

(三)

：以下為 pacman 分別採取 MinimaxAgent 以及 AlphaBetaAgent 獲勝的情況，可以看出採用 Alpha-Beta Pruning 的方式，可以讓 pacman 更快獲勝。

```
Pacman emerges victorious! Score: 853
Average Score: 853.0
Scores:      853.0
Win Rate:    1/1 (1.00)
Record:      Win
python3 pacman.py -p MinimaxAgent -l smallClassic 7.19s user 0.34s system 30% cpu 24.933 total
```

▲使用 Minimax 獲勝的情況：7.19s

```
Pacman emerges victorious! Score: 1313
Average Score: 1313.0
Scores:      1313.0
Win Rate:    1/1 (1.00)
Record:      Win
python3 pacman.py -p AlphaBetaAgent -l smallClassic 5.15s user 0.26s system 30% cpu 17.655 total
```

▲使用 Alpha-Beta pruning 獲勝的情況：5.15s