Homework 2

Autograder result

```
Starting on 3-30 at 16:29:56

Question q1
===========

Pacman emerges victorious! Score: 1238
Pacman emerges victorious! Score: 1244
Pacman emerges victorious! Score: 1239
Pacman emerges victorious! Score: 1235
Pacman emerges victorious! Score: 1233
Pacman emerges victorious! Score: 1241
Pacman emerges victorious! Score: 1246
Pacman emerges victorious! Score: 1242
Pacman emerges victorious! Score: 1239
Pacman emerges victorious! Score: 1242
Average Score: 1239.9
Scores:        1238.0, 1244.0, 1239.0, 1235.0, 1233.0, 1241.0, 1246.0, 1242.0, 1239.0, 1242.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** PASS: test_cases\q1\grade-agent.test (30.0 of 30.0 points)
***     1239.9 average score (2 of 2 points)
***         Grading scheme:
***          < 500:  0 points
***         >= 500:  1 points
***         >= 1000:  2 points
***     10 games not timed out (0 of 0 points)
***         Grading scheme:
***          < 10:  fail
***         >= 10:  0 points
***     10 wins (2 of 2 points)
***         Grading scheme:
***          < 1:  fail
***         >= 1:  0 points
***         >= 5:  1 points
***         >= 10:  2 points

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


Question q2
===========

*** PASS: test_cases\q2\0-eval-function-lose-states-1.test
*** PASS: test_cases\q2\0-eval-function-lose-states-2.test
*** PASS: test_cases\q2\0-eval-function-win-states-1.test
*** PASS: test_cases\q2\0-eval-function-win-states-2.test
*** PASS: test_cases\q2\0-lecture-6-tree.test
*** PASS: test_cases\q2\0-small-tree.test
*** PASS: test_cases\q2\1-1-minmax.test
*** PASS: test_cases\q2\1-2-minmax.test
*** PASS: test_cases\q2\1-3-minmax.test
*** PASS: test_cases\q2\1-4-minmax.test
*** PASS: test_cases\q2\1-5-minmax.test
*** PASS: test_cases\q2\1-6-minmax.test
*** PASS: test_cases\q2\1-7-minmax.test
*** PASS: test_cases\q2\1-8-minmax.test
*** PASS: test_cases\q2\2-1a-vary-depth.test
*** PASS: test_cases\q2\2-1b-vary-depth.test
*** PASS: test_cases\q2\2-2a-vary-depth.test
*** PASS: test_cases\q2\2-2b-vary-depth.test
*** PASS: test_cases\q2\2-3a-vary-depth.test
*** PASS: test_cases\q2\2-3b-vary-depth.test
*** PASS: test_cases\q2\2-4a-vary-depth.test
*** PASS: test_cases\q2\2-4b-vary-depth.test
*** PASS: test_cases\q2\2-one-ghost-3level.test
*** PASS: test_cases\q2\3-one-ghost-4level.test
*** PASS: test_cases\q2\4-two-ghosts-3level.test
*** PASS: test_cases\q2\5-two-ghosts-4level.test
*** PASS: test_cases\q2\6-tied-root.test
*** PASS: test_cases\q2\7-1a-check-depth-one-ghost.test
*** PASS: test_cases\q2\7-1b-check-depth-one-ghost.test
*** PASS: test_cases\q2\7-1c-check-depth-one-ghost.test
*** PASS: test_cases\q2\7-2a-check-depth-two-ghosts.test
*** PASS: test_cases\q2\7-2b-check-depth-two-ghosts.test
*** PASS: test_cases\q2\7-2c-check-depth-two-ghosts.test
*** Running MinimaxAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running MinimaxAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\q2\8-pacman-game.test

### Question q2: 30/30 ###


Question q3
===========

*** PASS: test_cases\q3\0-eval-function-lose-states-1.test
*** PASS: test_cases\q3\0-eval-function-lose-states-2.test
*** PASS: test_cases\q3\0-eval-function-win-states-1.test
*** PASS: test_cases\q3\0-eval-function-win-states-2.test
*** PASS: test_cases\q3\0-lecture-6-tree.test
*** PASS: test_cases\q3\0-small-tree.test
*** PASS: test_cases\q3\1-1-minmax.test
*** PASS: test_cases\q3\1-2-minmax.test
*** PASS: test_cases\q3\1-3-minmax.test
*** PASS: test_cases\q3\1-4-minmax.test
*** PASS: test_cases\q3\1-5-minmax.test
*** PASS: test_cases\q3\1-6-minmax.test
*** PASS: test_cases\q3\1-7-minmax.test
*** PASS: test_cases\q3\1-8-minmax.test
*** PASS: test_cases\q3\2-1a-vary-depth.test
*** PASS: test_cases\q3\2-1b-vary-depth.test
*** PASS: test_cases\q3\2-2a-vary-depth.test
*** PASS: test_cases\q3\2-2b-vary-depth.test
*** PASS: test_cases\q3\2-3a-vary-depth.test
*** PASS: test_cases\q3\2-3b-vary-depth.test
*** PASS: test_cases\q3\2-4a-vary-depth.test
*** PASS: test_cases\q3\2-4b-vary-depth.test
*** PASS: test_cases\q3\2-one-ghost-3level.test
*** PASS: test_cases\q3\3-one-ghost-4level.test
*** PASS: test_cases\q3\4-two-ghosts-3level.test
*** PASS: test_cases\q3\5-two-ghosts-4level.test
*** PASS: test_cases\q3\6-tied-root.test
*** PASS: test_cases\q3\7-1a-check-depth-one-ghost.test
*** PASS: test_cases\q3\7-1b-check-depth-one-ghost.test
*** PASS: test_cases\q3\7-1c-check-depth-one-ghost.test
*** PASS: test_cases\q3\7-2a-check-depth-two-ghosts.test
*** PASS: test_cases\q3\7-2b-check-depth-two-ghosts.test
*** PASS: test_cases\q3\7-2c-check-depth-two-ghosts.test
*** Running AlphaBetaAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running AlphaBetaAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\q3\8-pacman-game.test

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


Finished at 16:29:57

Provisional grades
==================
Question q1: 30/30
Question q2: 30/30
Question q3: 30/30
------------------
Total: 90/90
```

Algorithm description

Q1. Reflex Agent

藉由當下已知的資訊進行決策，不考慮行動後的可能影響。透過實作一個 evaluation score 的評估函數來決定要進行什麼行動。藉由相關已知資訊 (例如 ghost 的距離、good 的距離、Pacman 的無敵狀態等等) 來計算並 output 一個分數進行決策， Score 越高表示此行動的當下成效最好，反之則越差。詳細演算法流程整合至 description idea 中說明。

Q2. Minimax

Minmax 透過最小化最壞情況來進行損失評估，也就是最大化最小收益（嘗試將自身的最小收益達到最大可能，同時將對方的最大收益達到最小，亦即損失最大）。

1. 檢查基本條件：遊戲如果已經結束（勝利或失敗）或達到 depth == 0，就直接用評估函數 (evaluationFunction) 回傳現在狀態的分數
2. 取得合法行動 (legalActions)
3. 計算下一個 Agent 與深度：取得下一個 agent 的 Index，並判斷是不是回到 Pacman 來調整深度 (depth -= 1)。
4. 遞迴呼叫 Minimax：對所有可能的行動生成 successor (透過 gameState.generateSuccessor)，並遞迴計算每個狀態的 Minimax 分數。
5. 選擇最佳分數：Pacman 試圖最大化分數，選擇分數最高的行動；鬼魂則試圖最小化，選擇分數最低的行動。

6. 回傳最佳行動

Q3. Alpha-Beta Pruning

Alpha-Beta Pruning 是 Minimax 的優化版本，透過 Alpha 及 Beta 兩個參數來嘗試減少 Minimax 評估過程中不必要的節點數量來減少 (Pruning) 搜尋空間，進而提升效率，但也不影響 Minimax 的結果。Alpha 表示為最大化玩家的最小分數，Beta 則是最小化玩家的最大分數 (如同 Minimax 中的自我收益及對手收益)

1. 依照投影片的範例概念，實作 max_value 及 min_value，分別取得 Pacman 的最大化及 ghost 的最小化。
    - max_value (for pacman): 若發現 value > beta 則 prune，並更新 alpha 為 max value
    - min_value (for ghost): 若發現 value < alpha 則 prune，並更新 beta 為 min value
    - max_value, min_value recursively call AlphaBeta function
2. AlphaBeta: 根據 agent index 判斷是 pacman 或是 ghost，分別呼叫 max_value 及 min_value，並確保是否符合遊戲結束條件。
3. 選擇 max 作為 bestAction

Describe the idea of your design about evaluation function in Q1.

主要概念在取得 Pacman 當下的已知資訊：位置、食物、Ghost 的狀態以及 Ghost 害怕時間後進行相關評估。步驟如下：

1. 取得 concurrent 資訊
    - 取得與最近的食物距離 (透過 manhattanDistance)，評估最快可以獲取食物的機會
    - 取得每一個 ghost 的距離 (透過 manhattanDistance)，評估是否當下處於安全狀態
    - 取得是否有 scared ghost 存在，存在表示目前是否處於無敵狀態
2. 透過 successorGameState.getScore 取得預設分數
3. 依據與最近食物距離增加 score

    ```
    # 與食物距離成反比，食物越近，分數越高
    # 分母加一確保分母不為 0
    # weight = 1.0
    score += weight / (minFoodDistance + 1)
    ```

4. 如果目前存在 scared ghost，則提高 score，否則略過此步驟
5. 依據與最近的 ghost 調整分數，在此處設定若距離 < 2，則將 score 定為 float('-inf')
6. 回傳 evaluation score

Demonstrate the speed up after the implementation of pruning.

在 multiagentTestClass.py 中的 run method, 觀察到 stats 有紀錄 time (執行時間)，分別 print 結果後獲得以下：

- MinimaxAgent: 0.43781018257141113 seconds.
- AlphaBetaAgent: 0.3667459487915039 seconds.

由此可知 pruning 提升約 16% 的 performance