Report B10705054 劉知微

Show your autograder results and describe each algorithm:

Q1. Reflex Agent (2%)

```
Question q1
===========

Pacman emerges victorious! Score: 1171
Pacman emerges victorious! Score: 1255
Pacman emerges victorious! Score: 1243
Pacman emerges victorious! Score: 1256
Pacman emerges victorious! Score: 1248
Pacman emerges victorious! Score: 1247
Pacman emerges victorious! Score: 1251
Pacman emerges victorious! Score: 1255
Pacman emerges victorious! Score: 1252
Pacman emerges victorious! Score: 1228
Average Score: 1240.6
Scores:        1171.0, 1255.0, 1243.0, 1256.0, 1248.0, 1247.0, 1251.0, 1255.0, 1252.0, 1228.0
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)
***     1240.6 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 ###
```

Find the clostest manhattan distances between newPos and the food in newFood and store them in nearestFoodDistance

For each ghost_state, scared_time pair in zip(newGhostStates, newScaredTimes):

max_distance_to_ghost += the manhattan distance between newPos and ghost_state.getPosition()

If distance< = 1 and scared_time <= 2:

near_ghost += 1

If distance <= 1 and scared_time > 5:

max_distance_to_ghost += 100

return successorGameState.getScore() + 1 / nearestFoodDistance - num_ghosts / max_distance_to_ghost - near_ghost

Q2. Minimax (2%)

```
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 ###
```

minimax(state, depth, agentIndex): # Start minimax algorithm from the root node

If depth ==0 or state is a winning state or state is a losing state:

Return the evaluationFunction of the state and None

If it is Pacman's turn(agentIndex == 0)

It is the Max layer search, each legal action generates a successor state. Recursive Minimax search is then performed on these successor states. The maximum evaluation value from these search results is selected as the evaluation value of the best action, which is returned along with the corresponding action.

Else it is the Ghost's turn (agentIndex > 0)

It is the Min layer search. For each legal action, it generates the corresponding successor state and recursively performs Minimax search on these successor states. Then, it selects the minimum evaluation value from these search results as the evaluation value of the current layer.

Q3. Alpha-Beta Pruning (2%)

```
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 ###
```

AlphaBeta(state, depth, alpha, beta, agentIndex)

Start by AlphaBeta(gameState, self.depth, -inf , inf, 0)

if depth == 0 or state.isWin() or state.isLose():

return evaluationFunction(state), None

If it is Pacman's turn (agentIndex == 0)

It is the maximized-layer, For each action, it generates the corresponding successor state and recursively calls the AlphaBeta function to evaluate it. It updates the value to the maximum of its current value and the value obtained from the successor state. If the updated value exceeds beta, indicating that it is greater than the best value achievable by the opponent, the function prunes the search and returns the current value along with the corresponding action. Otherwise, it updates alpha to the maximum of its current value and the value and continues the search.

Else it is the Ghost's turn (agentIndex > 0)

It is the Minimization layer . For each action, it generates the corresponding successor state and recursively calls the AlphaBeta function to evaluate it. If the current ghost is the last one in the agent sequence, it decrements the depth by 1 before the recursive call. It updates the value to the minimum of its current value and the value obtained from the successor state. If the updated value is less than alpha, indicating that it is less than the best value achievable by Pacman, the function prunes the search and returns the current value along with None action. Otherwise, it updates beta to the minimum of its current value and the value and continues the search.

Describe the idea of your design about evaluation function in Q1. (2%)

對於 pacman 而言肯定希望離食物越近且離鬼距離越遠，因此我設計的重點是去計算出新與食物的最近距離，另外去計算離鬼的平均距離，除此之外值得注意的是，scrare_time 也是十分值得注意的項目，當 scare_time 的數字很接近 0，我們靠近鬼，就有極大的分風險，但當 scare_time 時間很充裕，靠近鬼，就有機會吃掉它，獲得額外的分數，我將其加 100 到平均距離。

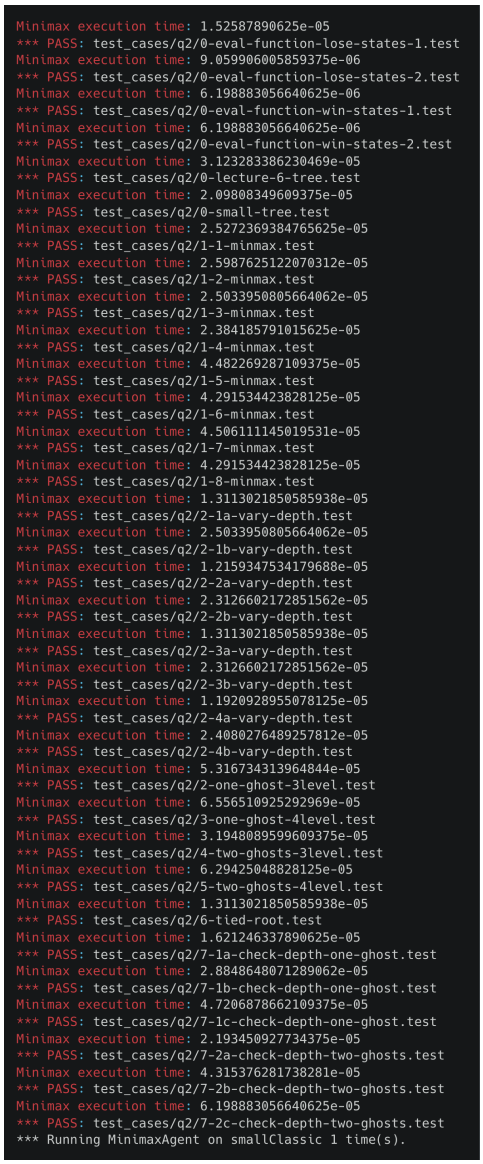
最後我在跟據以上資料計算出相應的分數，輸出。我本來的想法是

return successorGameState.getScore() ＋平均距離-最近食物距離- 靠近鬼＊100;

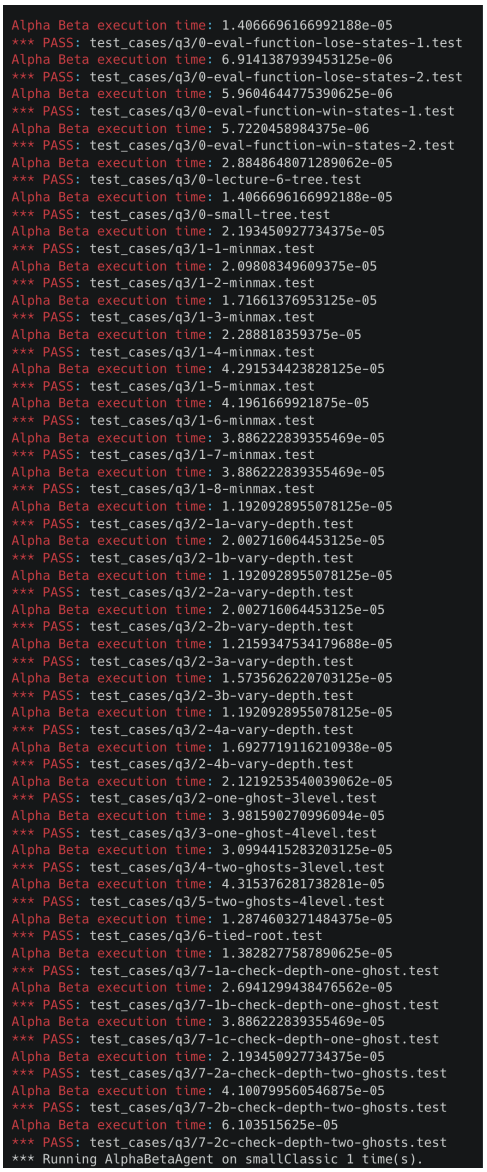
但最後數值過大，程式跑不動

因此轉為 successorGameState.getScore（）＋1/最近食物距離-1/平均距離-靠近鬼，也可以達到相同效果。

Demonstrate the speed up after the implementation of pruning.



```
Minimax execution time: 1.52587890625e-05
*** PASS: test_cases/q2/0-eval-function-lose-states-1.test
Minimax execution time: 9.059906005859375e-06
*** PASS: test_cases/q2/0-eval-function-lose-states-2.test
Minimax execution time: 6.198883056640625e-06
*** PASS: test_cases/q2/0-eval-function-win-states-1.test
Minimax execution time: 6.198883056640625e-06
*** PASS: test_cases/q2/0-eval-function-win-states-2.test
Minimax execution time: 3.1232833862304690e-05
*** PASS: test_cases/q2/0-lecture-6-tree.test
Minimax execution time: 2.09808349609375e-05
*** PASS: test_cases/q2/0-small-tree.test
Minimax execution time: 2.5272369384765625e-05
*** PASS: test_cases/q2/1-1-minmax.test
Minimax execution time: 2.5987625122070312e-05
*** PASS: test_cases/q2/1-2-minmax.test
Minimax execution time: 2.5033950805664062e-05
*** PASS: test_cases/q2/1-3-minmax.test
Minimax execution time: 2.384185791015625e-05
*** PASS: test_cases/q2/1-4-minmax.test
Minimax execution time: 4.482269287109375e-05
*** PASS: test_cases/q2/1-5-minmax.test
Minimax execution time: 4.291534423828125e-05
*** PASS: test_cases/q2/1-6-minmax.test
Minimax execution time: 4.506111145019531e-05
*** PASS: test_cases/q2/1-7-minmax.test
Minimax execution time: 4.291534423828125e-05
*** PASS: test_cases/q2/1-8-minmax.test
Minimax execution time: 1.3113021850585938e-05
*** PASS: test_cases/q2/2-1a-vary-depth.test
Minimax execution time: 2.5033950805664062e-05
*** PASS: test_cases/q2/2-1b-vary-depth.test
Minimax execution time: 1.2159347534179688e-05
*** PASS: test_cases/q2/2-2a-vary-depth.test
Minimax execution time: 2.3126602172851562e-05
*** PASS: test_cases/q2/2-2b-vary-depth.test
Minimax execution time: 1.3113021850585938e-05
*** PASS: test_cases/q2/2-3a-vary-depth.test
Minimax execution time: 2.3126602172851562e-05
*** PASS: test_cases/q2/2-3b-vary-depth.test
Minimax execution time: 1.1920928955078125e-05
*** PASS: test_cases/q2/2-4a-vary-depth.test
Minimax execution time: 2.4080276489257812e-05
*** PASS: test_cases/q2/2-4b-vary-depth.test
Minimax execution time: 5.316734313964844e-05
*** PASS: test_cases/q2/2-one-ghost-3level.test
Minimax execution time: 6.556510925292969e-05
*** PASS: test_cases/q2/3-one-ghost-4level.test
Minimax execution time: 3.1948089599609375e-05
*** PASS: test_cases/q2/4-two-ghosts-3level.test
Minimax execution time: 6.29425048828125e-05
*** PASS: test_cases/q2/5-two-ghosts-4level.test
Minimax execution time: 1.3113021850585938e-05
*** PASS: test_cases/q2/6-tied-root.test
Minimax execution time: 1.6212463378906250e-05
*** PASS: test_cases/q2/7-1a-check-depth-one-ghost.test
Minimax execution time: 2.8848648071289062e-05
*** PASS: test_cases/q2/7-1b-check-depth-one-ghost.test
Minimax execution time: 4.7206878662109375e-05
*** PASS: test_cases/q2/7-1c-check-depth-one-ghost.test
Minimax execution time: 2.193450927734375e-05
*** PASS: test_cases/q2/7-2a-check-depth-two-ghosts.test
Minimax execution time: 4.315376281738281e-05
*** PASS: test_cases/q2/7-2b-check-depth-two-ghosts.test
Minimax execution time: 6.198883056640625e-05
*** PASS: test_cases/q2/7-2c-check-depth-two-ghosts.test
*** Running MinimaxAgent on smallClassic 1 time(s).
```



```
Alpha Beta execution time: 1.4066696166992188e-05
*** PASS: test_cases/q3/0-eval-function-lose-states-1.test
Alpha Beta execution time: 6.9141387939453125e-06
*** PASS: test_cases/q3/0-eval-function-lose-states-2.test
Alpha Beta execution time: 5.9604644775390625e-06
*** PASS: test_cases/q3/0-eval-function-win-states-1.test
Alpha Beta execution time: 5.7220458984375e-06
*** PASS: test_cases/q3/0-eval-function-win-states-2.test
Alpha Beta execution time: 2.8848648071289062e-05
*** PASS: test_cases/q3/0-lecture-6-tree.test
Alpha Beta execution time: 1.4066696166992188e-05
*** PASS: test_cases/q3/0-small-tree.test
Alpha Beta execution time: 2.193450927734375e-05
*** PASS: test_cases/q3/1-1-minmax.test
Alpha Beta execution time: 2.09808349609375e-05
*** PASS: test_cases/q3/1-2-minmax.test
Alpha Beta execution time: 1.71661376953125e-05
*** PASS: test_cases/q3/1-3-minmax.test
Alpha Beta execution time: 2.288818359375e-05
*** PASS: test_cases/q3/1-4-minmax.test
Alpha Beta execution time: 4.291534423828125e-05
*** PASS: test_cases/q3/1-5-minmax.test
Alpha Beta execution time: 4.1961669921875e-05
*** PASS: test_cases/q3/1-6-minmax.test
Alpha Beta execution time: 3.886222839355469e-05
*** PASS: test_cases/q3/1-7-minmax.test
Alpha Beta execution time: 3.886222839355469e-05
*** PASS: test_cases/q3/1-8-minmax.test
Alpha Beta execution time: 1.1920928955078125e-05
*** PASS: test_cases/q3/2-1a-vary-depth.test
Alpha Beta execution time: 2.002716064453125e-05
*** PASS: test_cases/q3/2-1b-vary-depth.test
Alpha Beta execution time: 1.1920928955078125e-05
*** PASS: test_cases/q3/2-2a-vary-depth.test
Alpha Beta execution time: 2.002716064453125e-05
*** PASS: test_cases/q3/2-2b-vary-depth.test
Alpha Beta execution time: 1.2159347534179688e-05
*** PASS: test_cases/q3/2-3a-vary-depth.test
Alpha Beta execution time: 1.5735626220703125e-05
*** PASS: test_cases/q3/2-3b-vary-depth.test
Alpha Beta execution time: 1.1920928955078125e-05
*** PASS: test_cases/q3/2-4a-vary-depth.test
Alpha Beta execution time: 1.6927719116210938e-05
*** PASS: test_cases/q3/2-4b-vary-depth.test
Alpha Beta execution time: 2.1219253540039062e-05
*** PASS: test_cases/q3/2-one-ghost-3level.test
Alpha Beta execution time: 3.981590270996094e-05
*** PASS: test_cases/q3/3-one-ghost-4level.test
Alpha Beta execution time: 3.0994415283203125e-05
*** PASS: test_cases/q3/4-two-ghosts-3level.test
Alpha Beta execution time: 4.315376281738281e-05
*** PASS: test_cases/q3/5-two-ghosts-4level.test
Alpha Beta execution time: 1.2874603271484375e-05
*** PASS: test_cases/q3/6-tied-root.test
Alpha Beta execution time: 1.3828277587890625e-05
*** PASS: test_cases/q3/7-1a-check-depth-one-ghost.test
Alpha Beta execution time: 2.6941299438476562e-05
*** PASS: test_cases/q3/7-1b-check-depth-one-ghost.test
Alpha Beta execution time: 3.886222839355469e-05
*** PASS: test_cases/q3/7-1c-check-depth-one-ghost.test
Alpha Beta execution time: 2.193450927734375e-05
*** PASS: test_cases/q3/7-2a-check-depth-two-ghosts.test
Alpha Beta execution time: 4.100799560546875e-05
*** PASS: test_cases/q3/7-2b-check-depth-two-ghosts.test
Alpha Beta execution time: 6.103515625e-05
*** PASS: test_cases/q3/7-2c-check-depth-two-ghosts.test
*** Running AlphaBetaAgent on smallClassic 1 time(s).
```

我在 multiagent 中 import time，去計算每一次呼叫 minimax 或是 alph-abeta 後時間的消耗，由於 alpha-beta 的搜尋，減少針對不影響結果的 successive state 的 traverse 時間，可以發現 alpha beta 在時間效率上有顯著提升。