HW2 report

B05901084 電機四 劉容均

A. 簡介

我的 betterEvaluationFunction 和 ReflexAgent 的 evaluationFunction 的使用了相似的演算法，差別在於 evaluationFunction 只能計算到下一回合的可能性，和動態的局面變化，better 是計算靜態 state 的局勢優劣。Score 的計算方式如下：

$$\text{score} = \text{w}_{food}\times\text{e}_{food} + \text{w}_{ghost}\times\text{e}_{ghost} + \text{w}_{capsule}\times\text{e}_{capsule}$$

將 score 拆解成三個子項目，w 是權重 weight，調整權重可以改變各項目的優先度。e 則是各項目的 evaluation，之中也有不同的權重，接下來會一一說明計算方式，

B. food

場上食物越少，分數越高；若食物數量相同，則離最近食物的距離越近，分數越高。實作上採用扣掉場上食物數量，同時扣掉最近食物的距離。兩種分數也設有權重，兩權重比值必須夠大，避免發生下一個食物太遠時，pacman 為了最近食物距離的分數而不吃下已在面前的食物。例如設 100：1，當下一個食物距離大於 100 步時，不吃的分數會比吃的分數高。

C. ghost

ghost 分為 active 和 scared，由於 ghost 只有在靠近 pacman 時需要考慮逃跑或追逐，太遠的 active ghost 不需要考慮，太遠的 scared ghost 也不用花心力去追，因此兩者採用 exponential 的方式計算 evaluation，距離 active ghost 愈近分數急遽下降，距離 scared ghost 愈近分數急遽上升。其中 active ghost 的 decayFactor 較大，因為 pacman 和 ghost 的速度相同，即使在非常近的距離才開始逃跑也不易被追上，所以距離 3 以後影響分數的幅度極小。當吃下 capsule 轉為 scared ghost，追逐 scared ghost 優先度變成第一，因此設置最大的權重。

這邊會遇到一個難題，當 pacman 即將吃下 ghost 時，因為 ghost 會變回 active ghost，並回到重生點，此時靠近 scared ghost 而上升的分數會突然消失，導致 pacman 停下不吃 ghost。我想到的方式是以 active ghost 數量調整分數，補償消失的 scared ghost 分數，但此時又會遇到因為吃

capsule 會讓 active ghost 減少，active ghost 數量分數消失而不吃 capsule，最後以 capsule 的數量調整分數，補償 active ghost 數量分數，吃 ghost 的機制才正常運作。此外，由於與 ghost 的互動攸關存亡，這部分額外設置了幾種判斷機制，避免找死的情況發生。例如不能在重生點吃 ghost，不然 active ghost 會重生在旁邊；當 ghostState.scaredTimer 倒數快到時，放棄追逐。

D. capsule

capsule 在遊戲中非必須吃光的東西，但要衝高分數，在 ghost 接近時吃 capsule 反追非常重要，因此使用以下方式計算 capsule 分數：

$$(\mathrm{minDist}_c < \mathrm{minDist}_g) \times \frac{1}{\mathrm{minDist}_c} \times \frac{1}{\frac{\mathrm{minDist}_g{}^2}{8} + 0.875}$$

其中$(\mathrm{minDist}_c < \mathrm{minDist}_g)$判斷是否有機會先一步走到，$\frac{1}{\mathrm{minDist}_c}$判斷距離， $\mathrm{minDist}_g$的參數可以判斷 ghost 遠近，在適合的時機吃下 capsule。此分數在同時靠近 ghost 和 capsule 時影響幅度大過 active ghost 接近減去的分數，使反殺的優先度高過逃跑。

E. minMazeDistance

由於 mazeDistance 是計算兩點距離，在尋找最近食物時，需要執行相當於場上食物數量的 bfs，相當耗時，因此我重寫 FoodSearchProblem，將 goal state 改成第一個遇到的食物，因為 bfs 由淺到深的搜尋，可以確保這是最近的食物。MinMazeDistance 呼叫這個 problem，並回傳路徑長度。

F. 各項目權重

根據優先度$\mathrm{w}_{capsule\&ghost} > \mathrm{w}_{ghost} \gg \mathrm{w}_{food} > \mathrm{w}_{capsule}$，當 ghost 極遠時，$\mathrm{e}_{food}$是影響分數的主要因素，且 capsule 也不一定要吃。當 ghost 靠近時，逃跑變為優先，而同時也靠近 capsule 時，吃 capsule 為優先，變成 scared ghost 後追逐變為最優先。

G. 討論

這次作業的優化重點放在生存，因為死亡的懲罰太大，其次才是利益最大化。經過一連串改良和優化，最終 pacman 的勝率近 97%（-n 30 -f），場上

有兩隻 ghost 時，平均分數達 1500 以上，觀察失敗案例，幾乎都是夾殺，原因是 depth 不夠深，無法算到另一端的 ghost。也有一些未實作進作業中的想法，例如在吃完 scared ghost 前不吃下一個 capsule、減少跑進死路的機會（這次作業自己沒有使用到 wall 的功能），讓分數能達到最高。此外還有一些難題尚未克服，例如因為計算深度關係，pacman 會提早迴避 ghost，這是 reflex agent 不會有的情況，因為 reflex agent 只算到下一個 action。

H. 結果

```
(base) liurongjunde-MacBook-Pro:multiagent yore$ python2 pacman.py -l smallClas
sic -p ExpectimaxAgent -a evalFn=better -n 10 -f --frameTime 0 -q
Pacman emerges victorious! Score: 1743
Pacman emerges victorious! Score: 1755
Pacman emerges victorious! Score: 1746
Pacman emerges victorious! Score: 1349
Pacman emerges victorious! Score: 1177
Pacman emerges victorious! Score: 1734
Pacman emerges victorious! Score: 1746
Pacman emerges victorious! Score: 1750
Pacman emerges victorious! Score: 1504
Pacman emerges victorious! Score: 1739
Average Score: 1624.3
Scores:        1743.0, 1755.0, 1746.0, 1349.0, 1177.0, 1734.0, 1746.0, 1750.0,
1504.0, 1739.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
```