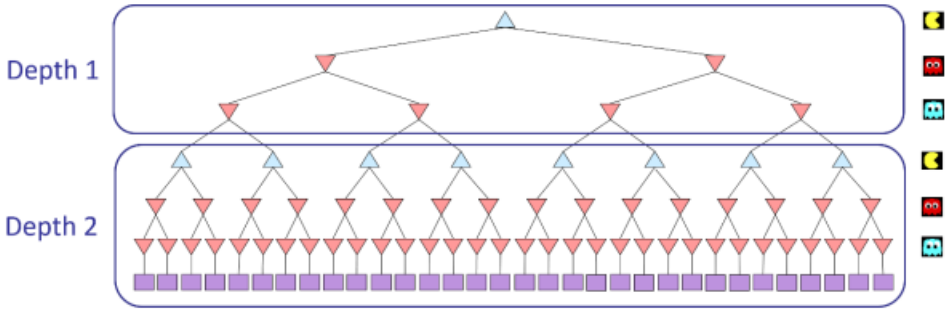
Multi-agent Pacman – Minimax



- 執行指令:

  python autograder.py -q q2

- Output:

```
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 19 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\q2\8-pacman-game.test

### Question q2: 5/5 ###


Finished at 22:01:00

Provisional grades
==================
Question q2: 5/5
------------------
Total: 5/5

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

➢ 理由：

我會安排以下程式的理由是因為 Minimax 演算法是一種先找出失敗的最大可能性，然後在這之中找到最小值的演算法，這演算法是俗稱悲觀演算法，如名字一樣，我認為要從鬼的動作步驟來判斷小精靈要走哪一步比較好，因此用 for 迴圈紀錄一下目前的鬼的下一步動作，因為這個演算法是由兩方輪流選擇自己要下哪一步，有一方要從可選的眾多選項中選會讓其優勢最大化的優勢選擇，而另一方是選使對手優勢最小化的方式，所以我們分別用 min_ value 函式計算下一個鬼的步驟動作並跟 num 比誰小後，紀錄最小值，還有呼叫 max_ value 函式計算 P1 的行為產生最大值來完成上述所說，最後得以完成這個演算法。

➢ 說明：

- 第一個部分是先在 getAction 函式中建立一個名字叫做 enemy _ team 的 List，然後用 for 迴圈紀錄一下目前的鬼的下一步動作，在這之後也會持續調用來計算其他鬼的下一步。

```
#對目前的鬼的下一步動作，會持續呼叫來計算其他鬼的下一步
enemy_team =[]
for i in range(1, gameState.getNumAgents()):
    enemy_team.append(i)
```

- 第二個部分是自訂義的 min _ value 函式，一開始先做判斷當 gameState. isWin() == True 有一方贏了，或是 gameState. isLose() == True 是輸掉，又或者是 depth == self. depth 搜尋的深度到達最深時 check 值就會加等於 1，如果不是以上情況則 check 值等於 0，當 check 大於等於 1 時，就回傳遊戲狀態結束遊戲，反之則繼續跑其他 code，之後用 for 迴圈來遞迴 min _value 函式，來持續計算鬼的下一步應該怎麼樣動，當 agent Index(鬼)不等於 0 並且 agent Index 等於 enemy_ team 最後一個元素時，就深度增加 1 層並且呼叫 max_ value 函式計算 P1 的行為產生最大值，反之則當前鬼增加 1 並且呼叫 min_ value 函式計算下一個鬼的步驟動作並跟 num 比誰小後，紀錄最小值。

```
def min_value(gameState, depth, agentIndex):
    #當有一方贏了或是輸掉又或者搜尋的深度到達最深，就回傳遊戲狀態結束遊戲
    num = 10000000000000000
    check = 0
    if gameState.isWin() == True:
        check+=1
    elif  gameState.isLose() == True:
        check+=1
    elif depth == self.depth:
        check+=1
    else:
```

```
else:
    check=0
    #用min_value函式來持續計算鬼的下一步怎麼動
    # #agentIndex是鬼
    for choice in gameState.getLegalActions(agentIndex):
        if agentIndex != 0:
            # agentIndex 等於 enemy_team 最後一個元素(現階段鬼已經是最後一個了)
            if enemy_team[-1] ==agentIndex:
                #計算P1的行為產生最大值(呼叫max_value函式)深度增加1層
                n1=max_value(gameState.generateSuccessor(agentIndex, choice ), depth + 1)
                n2=num
                if n1>n2:
                    num=n2
                elif n1<n2:
                    num=n1

            elif enemy_team[-1] !=agentIndex:#換計算下一個鬼的步驟動作並找出最小值
                n1=min_value(gameState.generateSuccessor(agentIndex, choice), depth, agentIndex + 1)
                n2=num
                if n1>n2:
                    num=n2
                elif n1<n2:
                    num=n1
    return num
```

```
                    num=n1
    return num

if check>=1:
    check=0
    return self.evaluationFunction(gameState)
```

- 第三個部分是 max_ value 函式，一開始也先判斷，當有一方贏了或是輸掉又或者搜尋的深度到達最深，就回傳遊戲狀態結束遊戲，沒有上述狀況則使用 for 迴圈調用 max_ value 來得到 P1 下一步所有的操作並跟 num 比誰大後，紀錄最大值。

```
def max_value(gameState, depth):
    #當有一方贏了或是輸掉又或者搜尋的深度到達最深，就回傳遊戲狀態結束遊戲
    num = -10000000000000000
    check = 0
    if gameState.isWin() == True:
        check+=1
    elif  gameState.isLose() == True:
        check+=1
    elif depth == self.depth:
        check+=1
    else:
        check=0
        #得到P1下一步所有的操作
        for choice in gameState.getLegalActions(0):
```

```
        #得到P1下一步所有的操作
        for choice in gameState.getLegalActions(0):
            n1=min_value(gameState.generateSuccessor(0, choice), depth, enemy_team[0])
            n2=num
            if n1>n2:
                num=n1
            elif n1<n2:
                num=n2
        return num

    if check>=1:
        check=0
        return self.evaluationFunction(gameState)
```

- 第四個部分是把動作產生的值放到 Ans 中，然後按值對數組由小到大進行排序(reverse=False)，最後返回最大值 Ans[-1][0]。

```
#綁定動作和動作產生的值，然後按值對數組進行排序，返回最大值。
Ans =[]
Ans = [(action, min_value(gameState.generateSuccessor(0, action), 0, enemy_team[0])) for action in gameState.getLegalActions(0)]
Ans.sort(reverse=False,key=lambda number: number[1])

return Ans[-1][0]
```