人工智慧　Artificial Intelligence

Program 1(b) – Search in Pac-Man

一、 方法描述

1. A* Search : 我的 A* Search 主要是利用 util.py 裡所定義的 PriorityQueue 來實作的，主要精神就是當傳入資訊的 Priority 越低，就會越早被 pop 出來。我的 Code 首先將初始位置的 Priority 設做 0，並且設一個 While 迴圈，只要 PriorityQueue 內仍有物件的話，就 pop 一個新的出來給(pos, direction, cost)接收。如果 pop 出來的點(pos)位於我負責記錄走過的點的 record 裡的話，就直接 pop 下一個值。接下來的 Code 就和 BFS、DFS 差不多了，但比較不一樣的是，當下一個點的 position 不在我所記錄的 record 裡面的話，我就會將到目前所在位置所花的距離加上走到下一步所需的 cost 再加上 heuristic function 所估計出來離終點的距離，當作我丟到 PriorityQueue 的權重(Priority)。最後再將資訊都已儲存好的 Successor push 到我的 PriorityQueue 中。
2. CornersProblem : CornersProblem 主要實作分為四個部分，分別是 init、getStartState、isGoalState 和 getSuccessors。在 init 中，大部分的都和預設的一樣，唯一不同的是，這次終止條件為走完四個 Corner，所以在 self.startState 的設定上除了前面要記錄目前的位置之外，還要再多設一個 list 在後方，紀錄已經走過的 Corners。而 getStartState 和 isGoalState 則

分別回傳 self.startState 和 True(若走過了四個 Corner)。getSuccessors 就稍微複雜了一點，他主要在記錄的是 state[0](目前位置)的下一個座標們，和 state[1](已走過的角落)有無記錄在 cornersVisited 裡面，若無就增加進去。最後再將 successors 利用 append (FIFO)的方式，傳回去給原本呼叫的函式。

3. cornersHeuristic：此為利用 A* Search 來跑 CornersProblem 所需的 heuristic function。大致上和之前的做法一樣，都是利用 Manhattan 來計算目前位置離終點的 x, y 的差距絕對值相加。只是這次的差別是終點被定義為角落們，不同角落的組合可能會影響到 expand nodes 的數量，因此若是只找最近的角落就走的話，出來 expand nodes 的數量，不一定會是最好的結果。所以我利用在 unCornerVisited 中，將所有尚未被拜訪過的 Corners 與目前所在位置的距離作總合當作 cost，並且回傳。最終 A* Search 就會走我剩餘沒拜訪過的 Corners 當中，距離總和最小的那一個。

二、 作業結果

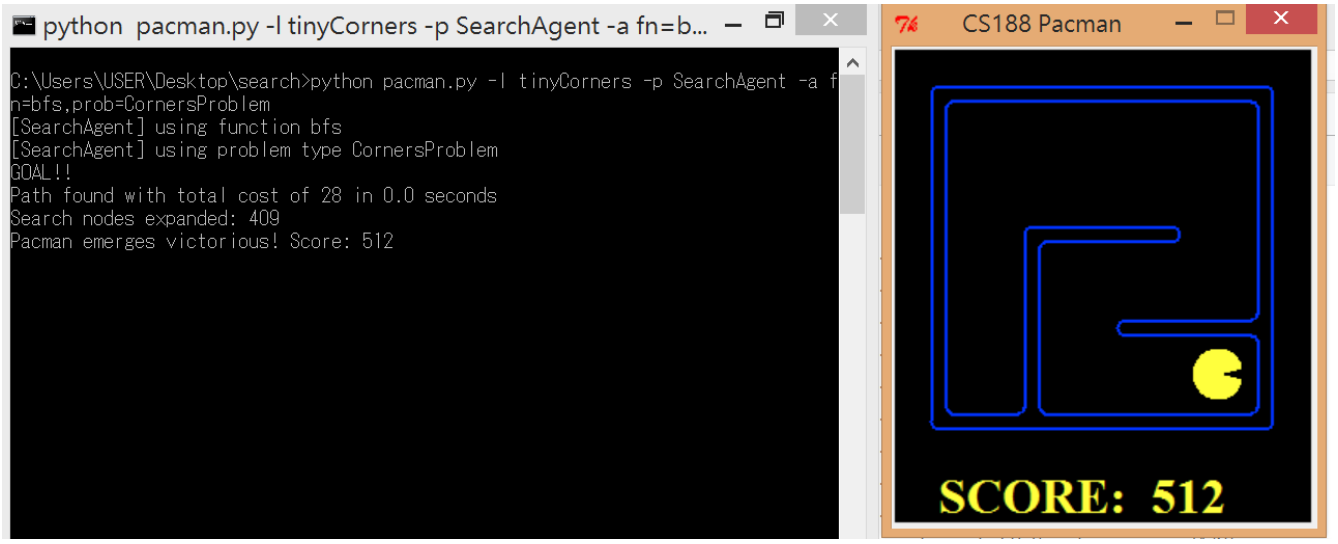
1. A* Search

指令: python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic

2. Finding All the Corners (BFS, Tiny Maze)

指令: python pacman.py -l tinyCorners -p SearchAgent -a

fn=bfs,prob=CornersProblem



2. Finding All the Corners (BFS, Medium Maze)

指令: python pacman.py -l MediumCorners -p SearchAgent -a

fn=bfs,prob=CornersProblem

2. Finding All the Corners (A* Search, Medium Maze)

指令: python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5