Homework 1

Autograder result

```
Starting on 3-16 at 16:45:08

Question q1
===========
*** PASS: test_cases\q1\pacman_1.test
***     pacman layout:          mediumMaze
***     solution length: 130
***     nodes expanded:         146

### Question q1: 5/5 ###


Question q2
===========
*** PASS: test_cases\q2\pacman_1.test
***     pacman layout:          mediumMaze
***     solution length: 68
***     nodes expanded:         269

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


Question q3
===========
*** PASS: test_cases\q3\ucs_4_testSearch.test
***     pacman layout:          testSearch
***     solution length: 7
***     nodes expanded:         14
*** PASS: test_cases\q3\ucs_5_goalAtDequeue.test
***     solution:               ['1:A->B', '0:B->C', '0:C->G'
***     expanded_states:        ['A', 'B', 'C']

### Question q3: 10/10 ###
```

```
Question q4
===========
*** PASS: test_cases\q4\astar_0.test
***     solution:               ['Right', 'Down', 'Down']
***     expanded_states:        ['A', 'B', 'D', 'C', 'G']

### Question q4: 15/15 ###


Question q5
===========
*** PASS: test_cases\q5\corner_tiny_corner.test
***     pacman layout:          tinyCorner
***     solution length:                28

### Question q5: 5/5 ###


Question q6
===========
*** PASS: heuristic value less than true cost at start state
path: ['North', 'East', 'East', 'East', 'East', 'North', 'Nor
North', 'North', 'East', 'East', 'North', 'North', 'North', '
h', 'South', 'South', 'East', 'East', 'East', 'East', 'East',
h', 'North', 'East', 'East', 'East', 'East', 'South', 'South'
path length: 106
*** PASS: Heuristic resulted in expansion of 1136 nodes

### Question q6: 9/9 ###


Finished at 16:45:08

Provisional grades
==================
Question q1: 5/5
```

```
Question q2: 5/5
Question q3: 10/10
Question q4: 15/15
Question q5: 5/5
Question q6: 9/9
------------------
Total: 49/49

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

Algorithm description

Q1. Depth First Search

深度優先演算法，透過 stack 實作；Stack 是 FILO (First-in Last-out)，先輸入的資料後輸出；DFS 會持續在 branch 上優先往深度走直到無法向下再回溯到上一個 node 繼續向下深入。演算法說明：

1. 初始化 stack 與已訪問過的 visited set
2. 將 startState push 進 stack
3. 當 stack 不為空
    1. 判斷是否是 goal，則回傳所有 action
    2. 如果尚未被訪問過，紀錄此 state，並且將後續的 state 及 action 都 push 進 stack
4. 重複執行步驟 3 至完成 goal

Q2. Breadth First Search

廣度優先演算法，透過 queue 實作；Queue 是 FIFO (First-in First-out)，先輸入的資料先輸出；BFS 會優先將同一層的 node 都先完成後再向下執行。演算法流程同 Q1，僅差別使用的陣列不同。

Q3. Uniform Cost Search

UCS 與 DFS 或 BFS 不同的點在於 UCS 雖使用類似 queue 的 FIFO，但使用的是 Priority queue，在 push node 的時候會計算每個 node 的成本進行優先順序的判斷進行排序，並在 pop 的時候丟出最低成本的 node。演算法同 Q1，僅使用 Priority queue 作為陣列在每次迴圈處理時 pop 出最低成本的節點向下進行。

Q4. A* Search (null Heuristic)

與 UCS 類似，同樣會計算路徑的成本，但會透過一個 Heuristic function (啟發函式) 進行成本計算。當 Heuristic function 設為 null，表示每個節點的成本相同，即可視為與 UCS 演算法相同。

Q5. Breadth First Search (Finding all the Corners)

透過 BFS 找到所有 Corner。

- getStartState: 首先提示中有說明要考慮的第一點是起始位置以及四個 corner 的位置。調整 getStartState，將 start state 的回傳定義為初始位置 (start position) 以及四個 corner 是否被訪問過 (visited)。
- isGoalState: 如何判斷是否達成 Goal 則是判斷四個 corner 是否都被訪問過，若都訪問過則完成 goal。
- getSuccessors: 主要進行兩個判斷，第一個是判斷是否為 wall，若不是 wall 則判斷是否為 corner，如果是 corner 則更新 visited corner 為 true，並更新 successors (從提示 2 提到要設定 cost 為 1)

Q6. A* Search (Corners Problem: Heuristic)

- cornerHeuristic: 透過啟發函示估算 current 到 corner 的最短路徑。首先判斷 corner 是否都被訪問過 (若都被訪問過則回傳 0 trivial solution)。接著計算 current 到每個未訪問的 corner 距離 (透過 util.manhattanDistance)，並且取得最遠的距離作為啟發函式的回傳參數。在此假設如果能以最少的 step 到達最遠的未訪問 corner，則可使到達其他未到達 corner step 更小。

Describe the difference between Uniform Cost Search and A* Contours

UCS 在搜尋過程中只考慮實際距離作為成本，不透過 heuristic function 作為成本估算方式。而 A* 除估算距離成本外，額外透過 heuristic function 來提供額外資訊來優化搜尋結果。

Describe the idea of Admissibility Heuristic

如同 Q6 所提及，在評估一個可被接受的啟發函式，主要是希望判斷提供的估計值不會超過實際成本。藉由對於每個 corner 所計算的曼哈頓距離可視為一種基本的啟發函式，再增加一點變形取得最遠的 corner 當作啟發函式的成本以試圖表達只要完成優化最遠距離的 step，則其他未訪問的 corner 則應該不會大於此 step 數量。因而可以作為 Admissibility Heuristic function。