AI HW1

洪郡辰
R11944050

March 25, 2024

Q1

```
PS C:\Users\OeO\Desktop\Master\smester2-2\AI\hw1\AI2024-hw1>
 * History restored
                                                    python autograder.py -q q1
Starting on 3-25 at 0:05:51

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

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


Finished at 0:05:51

Provisional grades
==================
Question q1: 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.
```

Depth First Search(DFS) 是深度優先搜尋，它會沿著樹的深度搜尋直到最深的節點，然後再回到上一個分支節點，直到搜尋完整個樹。在實作上我透過 stack 實作 frontier 保存 successor，利用 FILO 的特性讓取得 successor 的順序符合 DFS 的需求。

Q2

```
PS C:\Users\OeO\Desktop\Master\smester2-2\AI\hw1\AI2024-hw1> python autograder.py -q q2
Starting on 3-25 at 0:07:22

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

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


Finished at 0:07:22

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.
```

Breadth First Search(BFS) 是廣度優先搜尋，它會沿著樹的廣度搜尋，逐層地向下搜索，直到搜尋完整個樹。在實作上我透過 queue 實作 frontier 保存 successor，利用 FIFO 的特性讓取得 successor 的順序符合 BFS 的需求。

Q3

```
PS C:\Users\OeO\Desktop\Master\smester2-2\AI\hw1\AI2024-hw1> python autograder.py -q q3
Starting on 3-25 at 0:07:52

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 ###


Finished at 0:07:52

Provisional grades
==================
Question q3: 10/10
------------------
Total: 10/10

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

Uniform Cost Search(UCS) 是根據路徑成本進行搜尋，他會選擇成本最小的路徑進行搜尋，確保目前找到的路徑成本最低，也代表延伸的節點在 successor 具有最優先的順位。在實作上我透過 PriorityQueue 實作 frontier 保存 successor，利用 PriorityQueue 可以根據 priority 調整 successor 順位的特性讓取得的 successor 具有最低的成本。

Q4

```
PS C:\Users\OeO\Desktop\Master\smester2-2\AI\hw1\AI2024-hw1> python autograder.py -q q4
Starting on 3-25 at 0:08:19

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

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


Finished at 0:08:19

Provisional grades
==================
Question q4: 15/15
------------------
Total: 15/15

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

A* Search (null Heuristic) 是根據目前路徑成本以及預計成本進行搜尋，他會選擇起始節點至目的節點之路徑成本及目的節點至目標節點之預計成本總和最低的節點進行延伸，確保延伸後路徑的預計成本最低，延伸的節點在 successor 中具有最優先的順位。在實作上我透過 PriorityQueue 實作 frontier 保存 successor，利用 PriorityQueue 可以根據 priority 調整 successor 順位的特性讓取得的 successor 具有最低的預計成本。null Heuristic 則代表目前的預計成本設定為 0。

Q5

```
PS C:\Users\OeO\Desktop\Master\smester2-2\AI\hw1\AI2024-hw1> python autograder.py -q q5
Note: due to dependencies, the following tests will be run: q2 q5
Starting on 3-25 at 0:08:42

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

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


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

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


Finished at 0:08:42

Provisional grades
==================
Question q2: 5/5
Question q5: 5/5
------------------
Total: 10/10

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

在這個問題我需要透過 BFS 去解決 four corners problem，我需要去定義 state,start state,successor,goal state。我定義 state 格式為 ((x,y),(True/False,True/False,True/False,True/False))，(x,y) 代表目前的位置，False 代表 corner 還沒被找到，True 則代表 corner 已經被找到，因為有四個 corner 所以有四個 True or False 欄位。start state 為 ((x,y),(False,False,False,False)),goal state 為 (x,y),(True,True,True,True))。Successor 則定義成 (state,action,cost)，state 代表走到這個 node 的狀態，如果遇到 corner 則要將相對應的節點標示為 True。action 走到這個 node 需要的動作，cost 則代表從當前 state 走到此節點的成本。

Q6

```
Question q6
===========
*** PASS: heuristic value less than true cost at start state
path: ['North', 'East', 'East', 'East', 'East', 'North', 'North', 'West', 'West', 'West', 'West', 'West', '
North', 'North', 'North', 'North', 'West', 'West', 'West', 'West', 'South', 'South', 'East', 'East', 'East'
'East', 'East', 'East', 'North', 'North', 'East', 'East', 'North', 'North', 'East', 'East', 'North', 'North
, 'South', 'South', 'North', 'North', 'North', 'North', 'North', 'North', 'North', 'West', 'West', 'North',
path length: 106
*** PASS: Heuristic resulted in expansion of 1136 nodes

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


Finished at 0:09:14

Provisional grades
==================
Question q4: 15/15
Question q6: 9/9
------------------
Total: 24/24

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

在這個問題我需要透過 A* Search 去解決 four corners problem，在 Q4 及 Q5 已經定義好 A* Search 及 four corners problem，因此在這邊只需要定義 heuristic function 進行預估路徑成本，我將 heuristic function 定義為 manhattan Distance。manhattan Distance = |x1-x2|+|y1-y2|，因此符合 nonnegative 的需求。同時透過取 x 軸 y 軸絕對距離的算法也使得 manhattan Distance 能夠保持 consistent，符合題目之需求。

Q7

兩個演算法的差異在於計算路徑成本進行排序的部分，Uniform Cost Search 是根據起始節點至目的節點的成本進行排序，而 A* 則是根據起始節點至目的節點的成本加上目的節點至目標節點之預計成本進行排序。

Q8

Admissibility Heuristic 的概念在於合理地假設成本，設 h() 代表 heuristic function，確保 0<=h(n)<=real cost，如果 h(n)>real cost 會失去假設成本的好處，我們希望的是透過假設成本來了解哪些選擇更接近目標，如果假設成本大於真實成本，這很可能會使得根據 heuristic 做出的選擇離目標越來越遠。