# HW1

R12922016 葉丞勛

(-)

#### 1. DFS

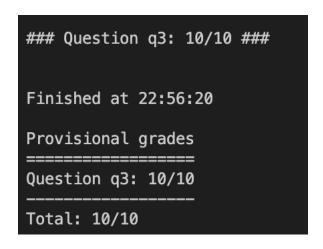
:使用 Stack 實作。一開始先將 Start node 加入 frontier 這個 Stack,接著當 frontier 不為空的情況下,先將 frontier 裡的 element pop 出來,作為當前的 node,接著檢查此 node 是否為 goal,是的話就表示找到了終點,回傳當前的點。若不是,則將當前的 node 加入 explored\_set 裡面,代表拜訪過此點,然後透過 getSuccessors() 去取得該 node 可能的下一步,並對每個 successors 去做檢查,如果這個 successor 不在 frontier 也不在 explored\_set 的話,就將這個 successor 加進 frontier。檢查完所有的 successor 後,則回到檢查 frontier 是否為空的步驟,直到 frontier 為空或是找到 goal 為止。

### 2. BFS

:與 DFS 大致上相同,差別在於 BFS 是使用 Queue 來實作,因此 pop 的時候會選到最先被加進去 frontier 的 node,而不是最新被加進去的。

### 3. UCS

:使用 PriorityQueue 實作,大致上的演算法與 BFS 相同,差別在於它會將 Start node 到當前的 node 的 cost 存到 PriorityQueue 裡面作為挑選下一個 node 的標準,並且當同一個 node 被拜訪到第二次的時候,如果這次的算出來的 cost 比上一次還要低,就更新 frontier 裡面這個 node 的 cost。



## 4. A\* Search(null Heuristic)

: 一樣使用 PriorityQueue 實作,大致上與 UCS 差不多,差別在於它是根據 Start code 到當前 node 的 cost 加上當前 node 的 heuristic 來作為判斷的標準。

### 5. BFS(Finding All Corners)

:我將每個 corners 是否被拜訪過用一個 tuple (0,0,0,0) 紀錄,分別代表左下、左上、右下、又上的 corner 是否有被拜訪過,有的話紀錄為 1,沒有的話紀錄為 0,並且將個紀錄用的 tuple 以及每個 corner 的位置一併藉由 getStartState 以及 getSuccessors 回傳給 search 演算法做使用。當紀錄用的 tuple 變為 (1,1,1,1) 時,表示有找到一條會經過每個 corner 的路徑。

## 6. A\* Search(Corner Problem:Heuristic)

:在 Heuristic() 當中,我先透過上題所說的紀錄用的 tuple 來判斷有哪些 corner 還沒有被拜訪過,並把每個沒被拜訪過的 corner 加入 unvisited\_list。 而當 unvisited\_list 不為空的時候,先把 list 中離當前 node 最近的 corner 以及他們之間的 manhattanDistance 求出來,找到後把求出來的 manhattanDistance 累加到 cost\_sum 裡,並且將剛剛找到的最近的 corner 從 list 中移除,最後把當前的 node 變成剛剛找到的最近的 corner,重複以上動作直到 unvisited\_list 為空為止。

# **(**□)

:UCS 與 A\* Search 的差別在於 UCS 只會透過 Start node 與當前 node 的 cost 去判斷要選誰作為下一個 node,而 A\* 除了會考慮 Start node 與當前 node 的 cost 之外,同時還會考慮到當前 node 的 Heuristic function,表示當前 node 與 goal 的關係。總結來說,UCS 只會考慮 Start node 與當前 node 的 cost,而 A\* 則是會考慮到 Start node 與當前 node 的 cost 再加上當前 node 的 Heuristic。

## (三)

:Admissibility Heuristic 表示當當前的 node 去呼叫他的時候,每次他回傳的值不會大於當前的 node 到 goal 的真實距離,以確保 node 可以往正確的方向前進。