AI\_Pacman(b)

404410107 資工三 蔡東霖

1. A\* search

這個演算法結構跟BFS類似，不過使用PriorityQueue，因為是要比較目前最小值，而A\*又是屬於heuristic的(f(n)=c(n)+h(n))搜尋方式，其中：

• c(n): real cost (so far) to reach the node

• h(n): estimated cost to get from the node to the goal

• f(n): estimated total cost of path through n to goal

傳入PriorityQueue中的內容是f(n)，此題目規定h(n)的估測方式是使用曼哈頓距離的估測法(2座標x y值的絕對值之和)

每節點中存父節點,移動,位置,c(n),h(n)等資訊，接著從PriorityQueue中取節點，並每次檢查是否到終點，走到終點後，一個一個將結點走的方位吐出來形成路徑，不過是從終點吐所以reverse()。

Find corners

找角落的算法需要更改到searchAgents.py中的Class CornersProblem，

實作中，我只要是參考上面PositionSearchProblem的內容，在\_\_init\_\_中增加了幾個之後會使用到的array，這些也是仿照PositionSearchProblem，其中不同的是多新增corner的初始值，回傳開始位置時也同時回傳corner狀況，isGoalState則是看是否達到目標，state中存的是corner狀態，因此若是都走訪了，則達成目標。

getSuccessors部分也是幾乎仿照PositionSearchProblem，同樣也是要處理corner的部分(若是下一步走到corner則在陣列中標記)。

getCostOfActions無需修改。

與PositionSearchProblem最大不同之處就是cornersHeuristic，這也是第2題與第3題不同之處：

在cornersHeuristic中，首先也是宣告角落狀態的陣列，上面都只用到走訪的角落，但是這邊需要用heuristic估測距離，因此需要未走訪的角落陣列，接著就是heuristic的使用，在上面的程式碼中，有manhattanHeuristic及euclideanHeuristic兩種方式，第一題使用了manhattanHeuristic，這次就試試用euclideanHeuristic，因此自定一個def處理估測函式，將每個未走訪的角落丟到副程式中，在所有輸出的估測距離裡，選擇最小的去走，走完就將點去除，直到所有點走完為止。