Assignment 1 - Search

Matthieu Vilain
SFSU ID: 920010985
Number of hours spend on this assignment : between 6 and 10 hours

Questions 1 to 4 : Search Algorithm

For these questions I created a data structure 'State' to simplify the readability. This 'State' object has :

- attribute 'state' : the current game state
- an attribute 'previous' : the previous 'State' from which we come
- an attribute 'action' : the action that was performed
- an attribute 'cost' : the cost of this action (for A*)
- an methode 'getPath' : that return the list of actions we took to be in this State
- other utility methode to print and compare 'State' object

I used this data structure for each search algorithm

Question 5 and 6 : Corners Problem

For these questions I chose the following state representation:

- (starting position, (list of remaining corners))

for exemple the initial state of the CornersProblem look like :

- (self.startingPosition, (self.corners[0], self.corners[1], self.corners[2], self.corners[3]))

I had to chose tuple and not list because list are not hashable, so the comparison of 'State' object wont work.

For the heuristic of the Corners Problem I created a function getClosestGoal(currentPosition, corners) that return the closest of the remaining corners and the distance to this corner.
My heuristic is the sum of manhattanDistance to the closest corner until there is remaining corner. So the heuristic is the length of the shortest path as the crow flies between the 4 corners.

This heuristic is very efficient, it has 692 search nodes expanded in 0.0sec

Questions 7 : Eating All The Dots

For this problem I found 4 heuristics:

Heuristic	Search nodes expended	Search time
mean length of the shortest path as the crow flies between dots	14049	8.1s
number of remaining dots	12517	5.5s
manhattan distance between the current position and the farthest dot	9551	3.8s
maze distance between the current position and the farthest dot	4137	26.2s

We can see that the last heuristic is very efficient but also very time expensive. The best trade off may be the 3rd heuristic which has a good number of exoended nodes and much more faster.

Question 8 :

For this qeustion I simply reused the function getClosestGoal(currentPosition, corners) mentioned earlier.
This strategy to eat always the closest dot is very fast but leads to strange and sub optimal behavior.