HAROKOPIO UNIVERSITY
DEPARTMENT OF INFORMATION & TELEMATICS

Artificial Intelligence

1st Assignment

Manousos Linardakis, it22064

Question 1:

A node in my solution is represented by a tuple, which consists of the successor of the state I'm expanding at that moment and the actions to get to it from the initial state.

Question 2:

Running “python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic”, we notice that it took 549 nodes for a*, while running “python pacman.py -l bigMaze -z . 5 -p SearchAgent \
- a fn=ucs,heuristic=manhattanHeuristic”, took 620 for ucs.
Correspondingly for openMaze, the nodes needed are 535 for a* and 682 for ucs.

Question 3:

The logic of the heuristic function implemented is to find the manhattan distance of the agent from the corners it has not yet visited. The manhattan distance provides an acceptable and consistent solution to pacman, since it (pacman) can only be moved vertically or horizontally. Thus, manhattan would be the optimal route if the track were free of obstacles, and if there are obstacles, the heuristic will always be less than the actual cost (the same is true for the difference of the two point heuristics, which is less than or equal to from the actual cost between them).

Also, by choosing max manhattan distances, we reduce the number of nodes that are expanded (maintaining the consistency and acceptability of the solution). In this way, the above heuristic function expands 1136 nodes (according to the grader as well).

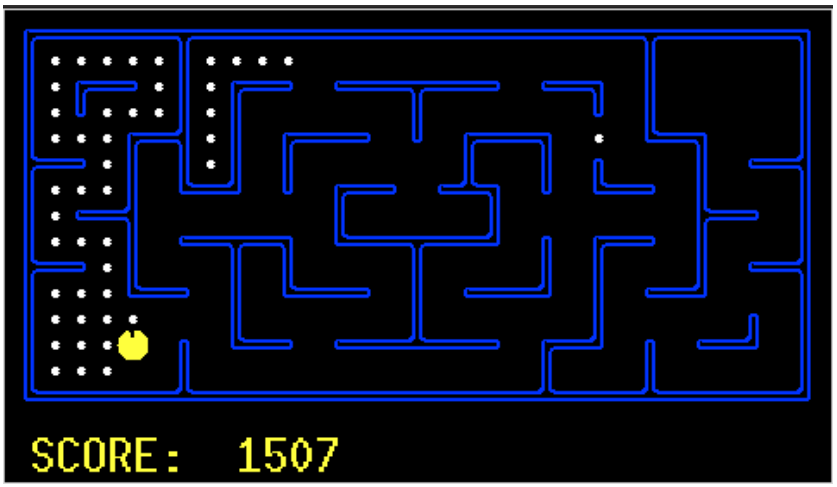
Question 4:

The logic of the implemented heuristic function is similar to that of the 3rd query, except this time the dots are not only at the corners of the track. It again returns the maximum manhattan distance, maintaining the consistency and acceptability of the solution (as shown before). The nodes expanded in this way are 9551.

Question 5:

ClosestDotSearchAgent will not always find the optimal solution, as it only looks at the closest dots, thus avoiding those that are in a more advantageous position at that moment. An example is the behavior of the agent when we run: python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5.

We observe the following:



pacman has left the dot on the right, as it only saw those closest to it when it was there (ClosestDotSearchAgent). In the end it is forced to traverse the entire track from left to right, as it is the only "nearby" dot left, thus making the solution suboptimal.