CIS*6020: Assignment 1

Total Marks: 15 – [X] indicates marks for each question. (X) indicates breakdown of marks.

Files to edit and submit:

searchAgents.py – Where your search agents will end up. (foodHeuristic is what you will fill in)

Important files to look at:

search.py – Search algorithms are located here.

util.py – Useful data structures and utility functions.

Might be useful to look at:

pacman.py – Main files that run the game.

game.py – Logic for the game, and supporting data structures.

Not so useful to look at:

Everything else

To submit:

i. **Your version of searchAgents.py**
searchAgents.py

ii. **A pdf format file with your explanation of why your heuristic is admissible (1 paragraph)**.

Name your files as follows:

searchAgents.py
yourlastname_yourfirstname_a1.pdf

Submit the above files to the CourseLink dropbox as a .zip file named A1.zip.

A1 Questions: (Written and Programming)

1. **[12]** Write an **admissible** (and non-trivial) heuristic within foodHeuristic that solves the problem of eating all of the dots in a maze.

   python pacman.py -l testSearch -p AStarFoodSearchAgent
   python pacman.py -l tinySearch -p AStarFoodSearchAgent
   python pacman.py -l trickySearch -p AStarFoodSearchAgent

   (4) Finds optimal path for tinySearch, trickySearch, any other search task
   (2) Nodes Expanded for trickySearch is less than 15000
   (1) Nodes Expanded for trickySearch is less than 12000
   (1) Nodes Expanded for trickySearch is less than 9000
   (1) Nodes Expanded for trickySearch is less than 7000
   (1) Nodes Expanded for trickySearch is less than 6000
   (1) Nodes Expanded for trickySearch is less than 5000
   (1) Nodes Expanded for trickySearch is less than 4000 (is this even possible?)
   Note that a non-admissible heuristic will be considered outside this scoring scheme and will receive a maximum of 5 marks for this section – so be careful!

   The code counts the number of nodes visited as a result of a search using the heuristic you choose. Be careful about using a search inside the search as part of your heuristic. For example, in the extreme, one could do a uniform cost search at each node to determine the heuristic value and visit a minimum number of nodes in the "outer" search. This isn't ideal. While a shallow search is allowed as part of the heuristic, ask the instructor if you are uncertain of whether your "inner" search is ok. (One way of answering this is asking yourself how much computation this is actually saving)

   Given the above – your code must not take longer than the starter code to complete the trickySearch layout.

2. **[3]** Explain/justify briefly why your chosen heuristic is consistent in a separate pdf file. Ideally this should be in the form of an informal (or formal) proof.