BLG435E, Artificial Intelligence, Fall 2015-2016
Assignment #1

Due: Oct 20, 2015

Submission Type: An archive file including the softcopy report and the source codes for Q3 to be submitted using Ninova. Note that each student must work individually for this assignment. Team work is not accepted!

Q1. For each of the following agents, develop a PEAS description of the task environment:

a. RoboCup 3D simulation league robot
b. Search and rescue robot
c. E-mail sorting application based on preferences
d. Activity recognition and anomaly detection software agent in an airport

For each of these agent types characterize the environment according to the properties of the environment (observability, dynamism, etc.) and determine the appropriate type of the agent architecture with reasonable arguments.

Q2. When do we need a consistent heuristic? Why?

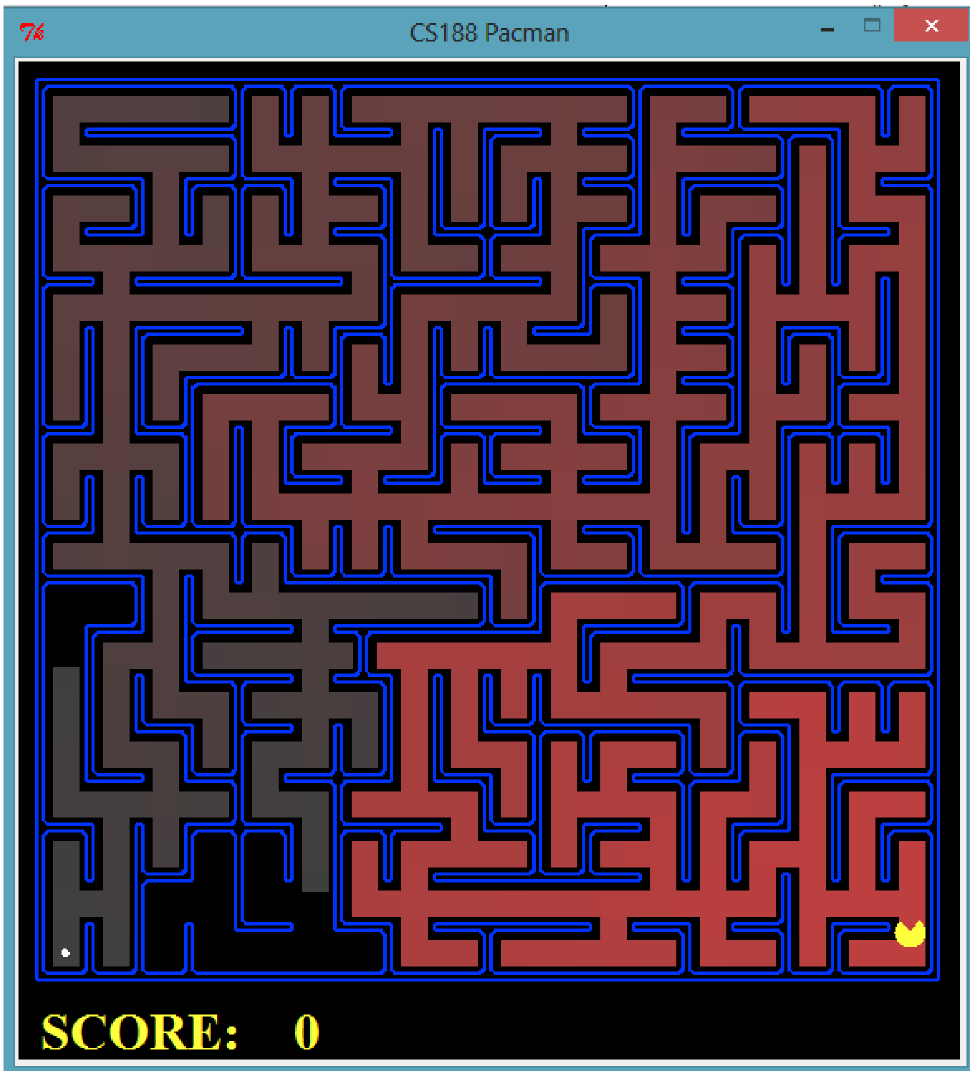
Q3. For this question, you are required to implement BFS, DFS and A* algorithms for the domain provided in the Pacman AI projects developed at UC Berkeley. You can obtain the base codes to be used from here.

`searchAgents.py` contains a fully implemented search agent (i.e., `SearchAgent`) which determines a path to a fixed position in the Pacman domain and executes that path step-by-step. For determining the path, `searchAgents.py` relies on `search.py` which includes declarations of the search algorithms. You will need to implement these functions. The implemented functions need to return a list of actions (valid directions, no moving through walls) that will lead the agent from the start to the goal. `pacman.py`, `game.py` and `util.py` are the files that contain necessary data types, structures and functions to be used (e.g., `GameState`, `Queue`, `Stack` etc.) for implementing the search algorithms.

The following command runs the already implemented dummy `tinyMazeSearch` function that only returns the predefined actions for `tinyMaze`:

```
python pacman.py -l tinyMaze -p SearchAgent -a fn=tinyMazeSearch
```

In this assignment, we are only interested in finding a path to a fixed location in the Pacman's maze. We are not interested in the other aspects of the domain such as the ghosts etc. The picture below shows the `BigMaze` layout where the Pacman starts at the lower right corner and the food (i.e., the goal location) is at the lower left corner.



Depth-First Search (DFS) and Breadth-First Search (BFS)

a. Implement both tree and graph search versions of BFS and DFS in `search.py` in a compatible format to be used in `searchAgents.py`. Test your functions in the `bigMaze` layout (given in the picture above) and analyze both tree and graph search versions of BFS and DFS in terms of:

- the number of nodes generated
- the number of nodes expanded
- the maximum number of nodes kept in the memory
- the running time.

If the corresponding algorithm does not last, please specify the reason.

A* Search

b. Is h(n)=1 an admissible heuristic for this problem? Why?

c. Implement the A* algorithm in `search.py` in a compatible format to be used in `searchAgents.py`. You will need to implement two different heuristics to be used in A*. One of the heuristics will be h(n)=1. You are also asked to implement an admissible heuristic as your second function. First, you need to show that this function is admissible for this problem. Then, you are asked to test these two functions in the `bigMaze` layout and compare them in terms of:

- the number of nodes generated
- the number of nodes expanded
- the maximum number of nodes kept in the memory
- the running time.

If the corresponding algorithm does not last, please specify the reason.

Important: In this question, your solution can rely on existing BFS, DFS or A* algorithm implementations such as the ones provided for "Artificial Intelligence: A Modern Approach" book. However, you need to explain how the algorithms work in this problem and perform the requested analyses above with sufficient explanations in your report. **Code usage without relevant references will be considered as Plagiarism.**