CSC460 Artificial Intelligence Spring 2023

Project – Phase 2

Description: This document contains details and requirement of Phase 2 of your project which is **due Friday, April 26th, at 11:59pm**.

Late submissions will be subject to grade penalty. You can work in a group of maximum two. Due to the fact that some students have dropped the course, you may change your group. Make sure to fill in your name or the names of your group members in the allocated sheet of the Excel file that was shared with you earlier, by Friday, April 7th, at 11:59pm.

The competition will take place the week after – date and time to be confirmed.

Background

In this phase of your project, your goal is to compete as Pacman against several Ghosts. The code basics are similar to the ones of the previous phase. Each group will play three consecutive rounds of Pacman on randomly selected mazes.

The game

In this is a multi-agent environment, Pacman seeks to eat as many dots as possible while avoiding being eaten by Ghosts.

The trick: Initially, several Ghosts appear on the maze, and only one of them starts moving. Then, every three seconds, an additional Ghost starts moving. The number of Ghosts is relative to the size of the maze which could be small, medium, and large. The code files include a sample of mazes of different sizes.

Every maze has its specific number of ghosts to consider. The following is the command line to specify the maze name (after -l) and the ghost number (-k).

The maze names and their respective number of ghosts are:

- smallClassic, 10 ghosts
- trappedClassic, 2 ghosts
- minimaxClassic, 2 ghosts
- mediumClassic, 12 ghosts
- capsuleClassic, 7 ghosts
- contestclassic, 12 ghosts

Assessment: will be based on the quality of your proposed solution as well as the outcomes of the games your agent plays in the competition. That will be in terms of the number of times that you win out of three games, the score of each game, and the amount of time through which you manage to keep Pacman alive in each game.

Grading: will be based on the resulting distribution of the above mentioned assessment metrics.

Keep in mind that the competition will run in the lab and not on your own computer. Therefore, it is very important to submit the required files, as detailed in the next sections.

```
python autograder.py
```

It can be run for one particular question, such as q2, by:

```
python autograder.py -q q2
```

It can be run for one particular test by commands of the form:

```
python autograder.py -t test_cases/q2/0-small-tree
```

You can use the autograder as a code proof for your project, but it will not be essential for grading as the game is evaluated differently now. The challenge is to have a better performance scores than your colleagues and not to achieve a better overall score in absolute.

The code for this project contains the following files which are available as an attached zipped folder on Blackboard. This list will direct you to what you need to work on, but you are welcome to check the remaining files.

Caution: Do not download any project provided by UC Berkeley as the codes are modified and tailored to your class competition. Although they share fair amount of similarity but they are programmed differently now.

File you will edit:

multiAgents.py	Where all of your multi-agent search agents will reside.

Files you might want to look at:

pacman.py	The main file that runs Pacman games. This file also describes a Pacman GameState type, which you will use extensively in this project.
game.py	The logic behind how the Pacman world works. This file describes several supporting types like AgentState, Agent, Direction, and Grid.
util.py	Useful data structures for implementing search algorithms. You don't need to use these for this project, but may find other functions defined here to be useful.

Supporting files you can ignore:

graphicsDisplay.py	Graphics for Pacman
graphicsUtils.py	Support for Pacman graphics
textDisplay.py	ASCII graphics for Pacman
ghostAgents.py	Agents to control ghosts
keyboardAgents.py	Keyboard interfaces to control Pacman
layout.py	Code for reading layout files and storing their contents
autograder.py	Project autograder
testParser.py	Parses autograder test and solution files
testClasses.py	General autograding test classes
test_cases/	Directory containing the test cases for each question
multiagentTestClasses.py	Project 3 specific autograding test classes

Files to Edit and Submit: Fill in missing portions of multiAgents.py according to your solution. Once you complete your project, submit **multiAgents.py file ONLY** on Blackboard (no need to zip anything). Do not change the other files in this distribution.

Also, **submit a report** explaining the components and steps of your solution.

Evaluation: Your code will be autograded for technical correctness. Please do not change the names of any provided functions or classes within the code, or you will wreak havoc on the autograder. However, the correctness of your implementation and the above listed metrics – not the autograder's judgements – will be the final judge of your score.

Academic Dishonesty: We will be checking your code against other submissions in the class for logical redundancy. If you copy someone else's code and submit it with minor changes, we will know. These cheat detectors are quite hard to fool, so please don't try. We trust you all to submit your own work only; please don't let us down. If you do, we will pursue the strongest consequences available to us.

Getting Help: You are not alone! If you find yourself stuck on something, you can contact the course graduate assistants, Simon Abi Younes and Elissa Lichaa El Khoury, for help. You can reach them at simon.abiyounes@lau.edu and elissa.lichaaelkhoury@lau.edu. However, note that they are here for technical support and not to solve the project on your behalf.

Discussion: You are encouraged to discuss with your colleagues but be careful not to give away your solution! Remember that this is a competition!