Project 1: Search in Pacman

CS 6300 Artificial Intelligence

University of Utah Due: Fri, Feb 5, 2015 by 11:59:59 P.M

In this project, your Pacman agent will find paths through his maze world, both to reach a particular location and to collect food efficiently. You will build general search algorithms and apply them to Pacman scenarios. As in Project 0, this project includes an autograder for you to grade your answers on your machine. This can be run with the command:

python autograder.py

The code for this project consists of several Python files, some of which you will need to read and understand in order to complete the assignment, and some of which you can ignore. Download search.zip from here `http://ai.berkeley.edu/search.html` which will contain all the code and supporting files.

1 Files to edit

For all the problems in this project, you would have to edit just two python files namely:

1. **search.py:** where all of your search algorithms will reside.
2. **searchAgents.py:** where all of your search-based agents will reside.

2 Supporting files

The following python files would help you in understanding the problem and the get you familiar with the different data structures and games states in Pacman.

1. pacman.py: The main file that runs Pacman games. This file describes a Pacman GameState type, which you use in this project.
2. game.py: The logic behind how the Pacman world works. This file describes several supporting types like AgentState, Agent, Direction, and Grid.
3. util.py: Useful data structures for implementing search algorithms.

3 Search in Pacman (120pts)

For all the problem titles described below, please refer to the link `http://ai.berkeley.edu/search.html` for the problem description and what is expected of each problem. As always autograder has different test cases against which you can run your program to check the correctness. For the questions asked below, please ensure your response is brief and to the point. Please don't write paragraphs of text as responses to these questions.

3.1 Depth First Search (12pts)

1. (10pts) Code Implementation
2. (1pt) Is the exploration order what you would have expected? Does Pacman actually go to all the explored squares on his way to the goal?
3. (1pt) Is this a least cost solution? If not, think about what depth-first search is doing wrong.

3.2 Breadth First Search (11pts)

1. (10 pts) Code Implementation
2. (1 pt) Does BFS find a least cost solution? If so explain why ?

3.3 Uniform Cost Search (11pts)

1. (10pts) Code Implementation
2. (1 pt) Specify the data structure used from the util.py for the uniform cost search

3.4 A* search (12pts)

1. (10pts) Code Implementation
2. (2 pts) What happens on openMaze for the various search strategies? Describe your answer in terms of the solution you get for A* and Uniform cost search.

3.5 Finding All the Corners (12pts)

1. (10pts) Code Implementation
2. (2 pts) Describe in few words/ lines the state representation you chose or how you solved the problem of finding all corners?

3.6 Corners Problem: Heuristic (11pts)

1. (10pts) Code Implementation
2. (1 pt) Describe the heuristic you had used for the implementation.

3.7 Eating All Dots (15pts)

1. (13pts) Code Implementation
2. (2 pt) Describe the heuristic you had used for the FoodSearchProblem.

3.8 Suboptimal Search (11pts)

1. (10pts) Code Implementation
2. (1 pt) Explain why the ClosestDotSearchAgent won't always find the shortest possible path through the maze.

4 Self Analysis (5pts)

1. What was the hardest part of the assignment for you?
2. What was the easiest part of the assignment for you?
3. What problem(s) helped further your understanding of the course material?
4. Did you feel any problems were tedious and not helpful to your understanding of the material?
5. What other feedback do you have about this homework?

5 Evaluation

Your code will be auto-graded 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. If your code passes all the test cases in the autograder you would receive full points for the implementation.

However even if your code does not necessarily pass all the test cases, we would evaluate your code and then award you partial points accordingly. In such cases it would be even more beneficial if you could give a short description of what you tried and where you had failed and that would help us in giving you better points.

6 Submission Instructions

- For the final submission you would be turning in a single zipped folder which should contain:
  - Folder having all the python files.(Search folder in this case)
  - PDF document containing your responses to the questions in Section 3 and 4.
- For those of you who are doing it in teams, it is enough that one of the team members makes a submission. We would soon have groups created in Canvas for this and you could use that to upload your submission.
- Please ensure all the submissions are done through canvas. Please do not email the instructor or the TA's with your submission. Submissions made via email would not be considered for grading.
- **Naming:** Your final upload should be named in the format $< uid >$-Proj1.zip where $< uid >$ is your Utah uid. **Ex:** u0006300-Proj1.zip

- For this project you should first unzip search.zip and then will fill in portions of search.py and searchAgents.py. Do not delete any of the files or change the names of any of those files in the project directory.
- **Written Answers:** Place all written answers to questions in Section 3 and 4 in a single PDF document. This should be clearly named in the format $< uid >$-Proj1-answers.pdf, where $< uid >$ is your Utah uid. **Ex:** u0006300-Proj1-answers.pdf. Please make sure to write your name at the top of the document!