Project 1: Search in Pacman

CS 4300: Artificial Intelligence
University of Utah

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 p01.zip which will contain all the code and supporting files.

1 Files to edit

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

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

2 Supporting Files

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

- **pacman.py:** The main file that runs Pacman games. This file describes a Pacman GameState type, which you use 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.

3 Search in Pacman (42 pts)

For all the problem titles described below, please refer to this search project page 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 Programming Implementation (25 pts)

1. Finding a Fixed Food Dot using Depth First Search (3 pts)
2. Breadth First Search (3 pts)
3. Varying the Cost Function (Uniform Cost Search) (3 pts)
4. A* search (3 pts)
5. Finding All the Corners (3 pts)
6. Corners Problem: Heuristic (3 pts)
7. Eating All The Dots (4 pts)
8. Suboptimal Search (3 pts)

3.2 Written Questions (12 pts)

In a sentence or two, answer the following questions about your implementation and/or observations of the programming portions above.

3.2.1 Depth First Search

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

3.2.2 Breadth First Search

1. (1 pt) Does BFS find a least cost solution? If so, explain why.

3.3 Varying the Cost Function (Uniform Cost Search)

1. (1 pt) Specify the data structure used from util.py for uniform cost search

3.4 A* search

1. (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

1. (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

1. (1 pt) Describe the heuristic you used for the implementation.

3.7 Eating All Dots

1. (2 pts) Describe the heuristic you used for the FoodSearchProblem.

3.8 Suboptimal Search

1. (1 pt) Explain why the ClosestDotSearchAgent won't always find the shortest possible path through the maze.

4 Self Analysis (5 pts)

Each group member must answer these questions individually.

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 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. If your code passes all the test cases in the autograder you will receive full points for the implementation.

However even if your code does not necessarily pass all the test cases, we will 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 will be turning in the following items (please do not zip them together but instead submit them separately):

  1) **search.py**

2) **searchAgents.py**

4) one PDF document *per group* containing your responses to questions. Please ensure that the pdf has two separate answers for the Self-evaluation section, but one answer for the rest.

- Please ensure all the submissions are done through Gradescope. Please do not email the instructor or the TA's with your submission. Submissions made via email will not be considered for grading.
- **Written Answers:** Place all your written answers in a single PDF document. This should be clearly named in the format ⟨uid1⟩-⟨uid2⟩-Proj⟨number⟩-answers.pdf, where ⟨uid1 and 2⟩ are the Utah uids of the two group members (if applicable) and ⟨number⟩ is the Project number. Ex: u0004300-u0001337-Proj1-answers.pdf. Please make sure to write your and your partner's name at the top of the document!
- **Naming:** Your python file(s) upload should be kept under the original name(s), i.e. **search.py, searchAgents.py**
- For this project fill in portions of the files to edit. Once you have completed the code, name them as per the conventions stated above and submit the requested files via Gradescope.
- In the pdf, each group member is expected to provide their own responses to the self analysis, but can work together on the code and other written responses.
- Do **NOT** install/use additional libraries or packages other than what is provided and the python standard library.