COSE361(03) - Final project 1/2

Due Date: 2023/5/22(Monday) 23:59

Python Version: 3.6.

All of the source code is from CS188 Berkeley.
(https://inst.eecs.berkeley.edu/~cs188/sp20/minicontest1/)

Submission policy is at the end of this PDF.

Overview

In this project, you will apply the search algorithms and problems implemented in Assignment 1 to handle more difficult scenarios that include controlling multiple ***Pacman*** agents and planning under time constraints. There is room to bring your own unique ideas, and there is no single set solution. We are very much looking forward to seeing what you come up with!

Extra Credit

Students that perform well in the **final** submission will receive the following extra credit:

- 1st to 5th place: 3 points
- 6th to 20th place: 2 points
- 21st to 50th place: 1 point

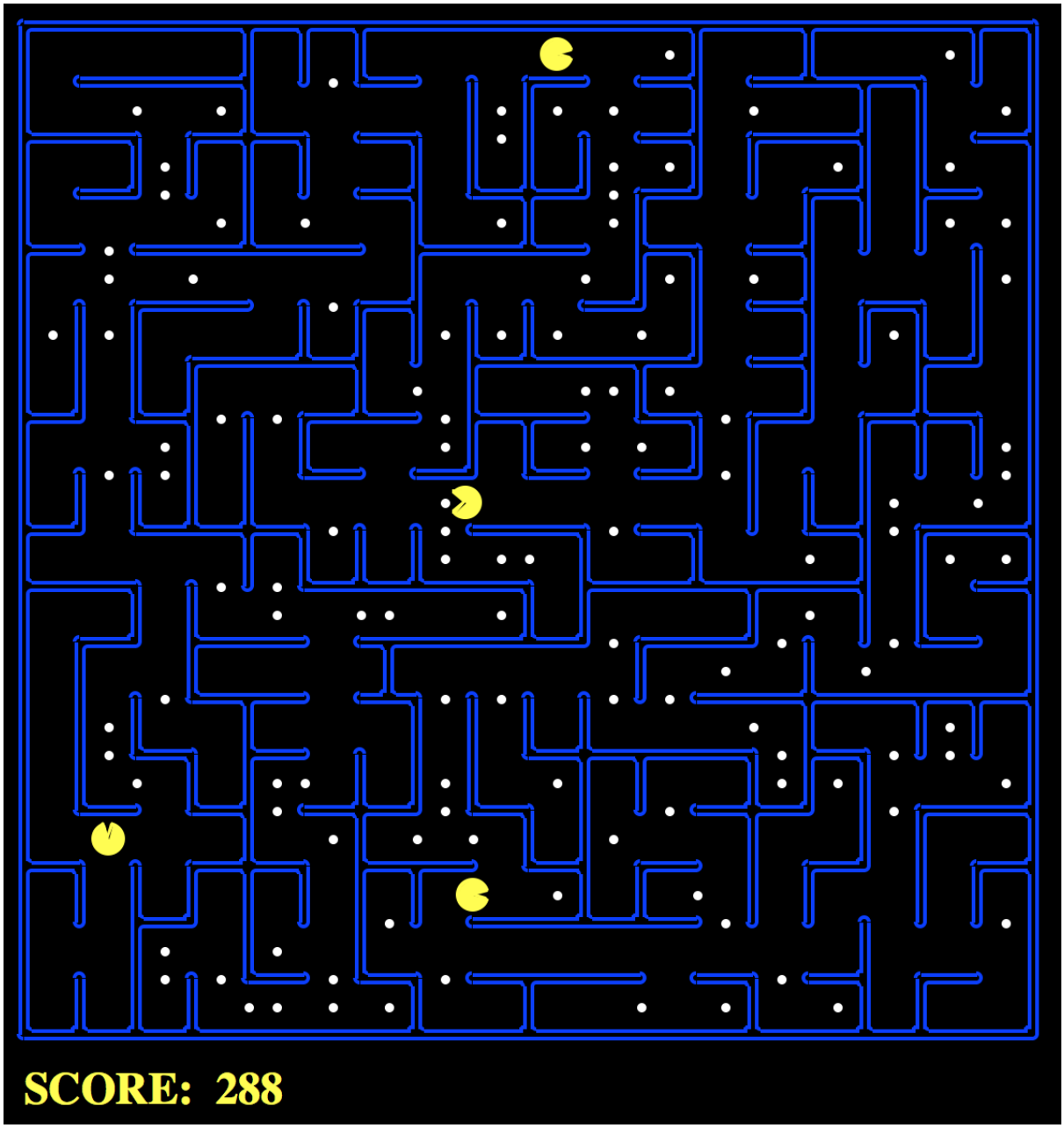
Note: A good implementation of the bot in the “Quick Start Guide” below should easily score more than 500 points on the leaderboard.

Quick Start Guide

Follow these 3 easy steps to quickly get involved in the contest!

1. Download the code (minicontest1.zip) from Blackboard, unzip it, and change to the directory. ***Note!*** There are few modifications from the original code, so download it from Blackboard.
2. Run `python pacman.py`. You should be able to see 4 Pacman agents traveling around the map collecting dots with the `ClosestDotAgent` in `myAgents.py`.
3. Submit the `myAgents.py` file and pdf file on Blackboard.

Important! You only need to submit `myAgents.py`. If you import from `search.py` or `searchProblems.py`, we will use the staff version of those files.



Introduction

The base code is nearly identical to Assignment 1, but with some minor modifications to include support for more than one Pacman agent. You can download all the code and supporting files in Blackboard. Some key differences:

- You will control `N` Pacman agents on the board at a time. Your code must be able to support a multiple number of agents
- There is a cost associate with how long Pacman "thinks" (compute cost). See `Scoring` for more details.

Files you'll edit:

`myAgents.py` : What will be submitted. Contains all of the code needed for your agent.

Files you might want to look at:

`searchProblems.py` : The same code as in Assignment 1, with some slight modifications.

`search.py` : The same code as in Assignment 1.

`pacman.py` : The main file that runs Pacman games. This file describes a Pacman GameState type, which you use in this project.

Files to Edit and Submit: You will fill and submit `myAgents.py`.

***Evaluation*:** Your code will be graded to determine your final score. Please do not change the names of any provided functions or classes within the code, or you will wreak havoc on the grade rule. If necessary, we will review and grade assignments individually to ensure that you receive due credit for your work.

Rules

Layout

There are a variety of layouts in the `layouts` directory. Agents will be exposed to a variety of maps of different sizes and amounts of food.

Scoring

The scoring from Assignment 1 is maintained, with a few modifications.

Kept from Assignment 1:

- +10 for each food pellet eaten
- +500 for collecting all food pellets

Modifications:

- 0.4 for each action taken (Assignment 1 penalized -1)
- −1×total compute used to calculate next action (in seconds)×1000
- −1×total compute used to calculate next action (in seconds)×1000

Each agent also starts with 100 points.

Observations

Each agent can see the entire state of the game, such as food pellet locations, all pacman locations, etc. See the `GameState` section of the code for more details.

Winning and Losing

Win: You win if you collect all food pellets. Your score is the current amount of points.

Lose: You lose if your score reaches zero. This can be caused by not finding pellets quickly enough, or spending too much time on compute. Your score for this game is zero. If your agent crashes, it automatically receives a score of zero.

Designing Agents

File Format

You should include your agents in a file of the same format as `myAgents.py`. Your agents must be completely contained in this one file, **although you may use the functions in `search.py`**

Interface

The `GameState` in `pacman.py` should look familiar but contains some modifications to support multiple Pacman agents. The major change to note is that many `GameState` methods now have an extra argument, `agentIndex`, which is to identify which Pacman agent it needs. For example, `state.getPacmanPosition(0)` will

get the position of the first Pacman agent. For more information, see the `GameState` class in `pacman.py`.

Agent

To get started designing your own agent, we recommend subclassing the `Agent` class in `game.py` (this has already been done by default). This provides to one important variable, `self.index`, which is the `agentIndex` of the current agent. For example, if we have 2 agents, each agent will be created with a unique index, `[MyAgent(index=0), MyAgent(index=1)]`, that can be used when deciding on actions.

We will call the `createAgents` function to create your team of pacmen. By default, it is set to create `N` identical pacmen, but you may modify the code to return a diverse team of multiple types of agents.

Restrictions

Please respect the APIs and keep all of your implementation within `myAgents.py`.

Getting Started

By default, the game runs with the `ClosestDotAgent` implemented in the Quick Start Guide. To run your own agent, change `agent` for the `createAgents` method in `myAgents.py`

```
$ python pacman.py
```

A wealth of options are available to you:

```
$ python pacman.py --help
```

To run a game with your agent, do:

```
$ python pacman.py --agent MyAgent
```

Testing

The `layouts` folder contains all of the test cases that will be executed on the autograder. **There are no hidden tests.** To see how you perform on any single map, you can run:

```
$ python pacman.py --agent MyAgent --layout test71.lay
```

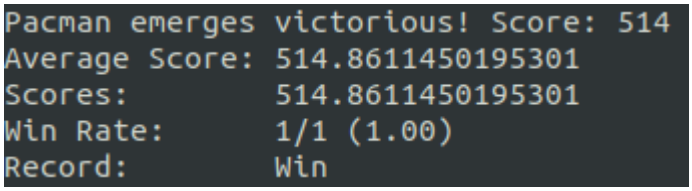
Submission

We will grades . To get full credit, run `pacman.py` and it should look like below (your score must exceed 500).

```
cose361@one:~/minicontest1$ python pacman.py --agent MyAgent --layout test71.lay
...
Pacman emerges victorious! Score: 515
Average Score: 515.3870964050282
Scores:        515.3870964050282
Win Rate:      1/1 (1.00)
Record:        Win
```

▼ There are two files you need to submit.

1. Submit `myAgents.py` on Blackboard.
2. Submit a 2 page report(pdf) file containing
   a. capture the result of `pacman.py` with layout `test71.lay` (4 points)



   b. Description of your agents. (2 points)
   c. Three discussions when playing Pacman (2 point for each discussion. Total 6 points.)

- Discuss cases where the agent implemented by yourself is ***better*** than the baseline.

- Discuss cases where the agent implemented by yourself is ***worse*** than the baseline.
- Ask & Answer your own question about the above discussion.

Note:

- Do not compress your result files. Upload them respectively
- It is okay to make out a report in Korean

About Plagiarism

We know that it is easy to find source code for this assignment. However, if we find out that you just copied from one of the source code, grade for the assignment will be zero. You can refer to those source codes, but do not just copy and paste them.

Q & A

If you have any questions, please upload your question on discussion board in Blackboard.