

# Homework 3 - Multi-Agent Search

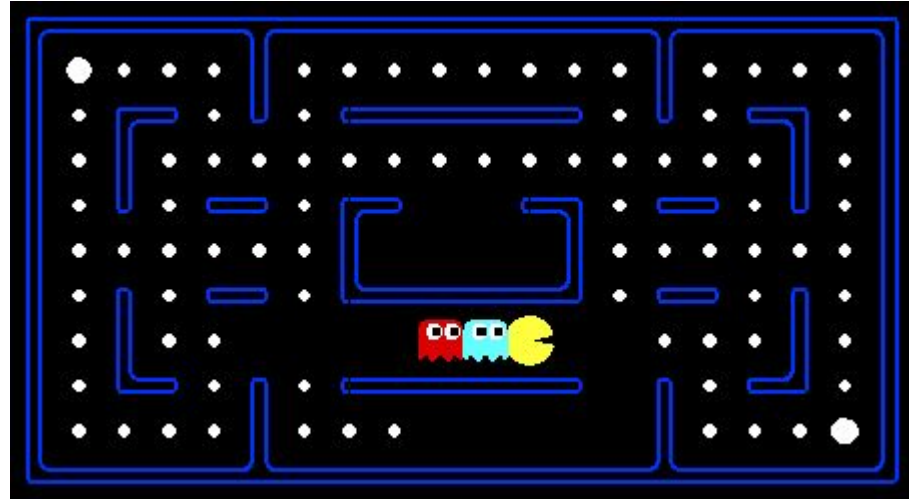
Due Date: 4/21 (Friday) 23:59



# Introduction

In this assignment, you will design agent for the classic version of Pac-Man by implementing three adversarial search algorithms, i.e.,

- Minimax search
- Alpha-beta pruning
- Expectimax search



# Introduction

- The code base of Pac-Man was developed at UC Berkeley.  
(<https://inst.eecs.berkeley.edu/~cs188/sp21/project2/>)
- Since the game uses GUI, Google Colab cannot execute it. You can only run the codes on a local machine. **Please install python 3 on your own machine** and be familiar with the code with CLI.

# Start the Game

First, play a game of classic Pac-Man by running the following command:

```
python pacman.py
```

and using the arrow keys to move.

Next, run the given [ReflexAgent](#) in [multiAgents.py](#):

```
python pacman.py -p ReflexAgent
```

Note that the ReflexAgent performs poorly even on simple layouts:

```
python pacman.py -p ReflexAgent -l testClassic
```

Other Options:

1. Default ghosts are random. You can also play for fun with slightly smarter directional ghosts using `-g DirectionalGhost`.
2. You can play multiple games in one command with `-n`.
3. You can turn off graphics with `-q` to run games quickly.
4. Use `-h` to know more options.

# File Structure

<b>Files you will edit:</b>	
<a href="#">multiAgents.py</a>	All of your multi-agent search agents will be resided.
<b>Files you might want to look at:</b>	
<a href="#">pacman.py</a>	The main file that runs Pac-Man games. This file also describes a pacman <a href="#">GameState</a> type, which you will use extensively in this assignment.
<a href="#">game.py</a>	The logic behind how the Pac-Man world works. This file describes several supporting types like <a href="#">AgentState</a> , <a href="#">Agent</a> , <a href="#">Direction</a> , and <a href="#">Grid</a> .
<a href="#">util.py</a>	Useful data structures for implementing search algorithms. You don't need to use these for this assignment, but may find other functions defined here to be useful.
<b>Other files you might want to look at, if you are interested in the details of this game.</b>	

# Autograding

In this assignment, TAs will use an autograder to grade your implementation. The autograder has been included in the code base. You can use the following command to test by yourself.

```
python autograder.py
```

The autograder will check your code to determine whether it explores the correct number of game states. This will show what your implementation does on some simulated trees and Pac-Man games. After that, it will show the score you get.

# Requirements

1. Please modify the codes in `multiAgents.py` between `# Begin your code` and `# End your code`. In addition, do not import other packages.
2. All agents you implement should work with `any number of ghosts`. In particular, your search tree will have multiple min/chance layers (one for each ghost) for every max layer.
3. Your code should also expand the game tree to `arbitrary depth` with the supplied `self.depth`. A single level of the search is considered to be one pacman move and all the ghosts' responses, so depth 2 search will involve pacman and each ghost moving twice.

**For more detail requirements, please check the spec!**

# Implementation (90%)

- Part 1: Minimax search(20%)
  - Implement the **MinimaxAgent** class stub in **multiAgents.py**
- Part 2: Alpha-Beta Pruning (25%)
  - Implement the **AlphaBetaAgent** class in **multiAgents.py**
- Part 3: Expectimax Search (25%)
  - Implement the **ExpectimaxAgent** class in **multiAgents.py**
- Part 4: Evaluation Function (20%)
  - Write a better evaluation function for pacman in **betterEvaluationFunction** in **multiAgents.py**
  - The evaluation function should evaluate only states not including actions



# Report (10%)

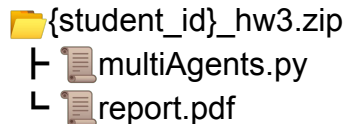
- You should write your report following the report template
- The report should be written in **English**.
- Please save the report as a **.pdf** file. (font size: 12)
- For part 1 ~ 4, please take some screenshots of your code and explain how you implement codes **in detail**.

# Submission

**Due Date: 2023/4/21 23:55**

Please compress your [multiAgents.py](#) and report (.pdf) into `STUDENTID_hw3.zip`.

The file structure should look like:



```
{student_id}_hw3.zip
├── multiAgents.py
└── report.pdf
```

The diagram shows a file structure within a rectangular box. At the top level is a folder icon followed by the text "{student\_id}\_hw3.zip". Below this folder, there are two file icons, each preceded by a small square icon representing a file. The first file is labeled "multiAgents.py" and the second is labeled "report.pdf".

**Wrong submission format leads to -10 point.**

## Late Submission Policy

**20% off per late day**

Please check out the spec  
for more details!