Mini-Concurso 1: Multi-Agent Pacman

(Adaptado de CS188 Berkeley)

Tabla de Contenido

Idea general

En este mini concurso, aplicarás los algoritmos de búsqueda y los problemas implementados en el Proyecto 1 para manejar escenarios más difíciles que incluyen el control de múltiples agentes pacman y la planificación con limitaciones de tiempo. Hay espacio para aportar vuestras propias ideas únicas, y no existe una solución única. ¡Esperamos ver lo que se os ocurre!

Puntuación

Los puntos adicionales se obtienen además de los 25 puntos disponibles en P1. P.ej. si ganas 1 punto de EC a través del mini concurso y obtuviste un 25/25 en P1, entonces tendrás 26/25 en P1

Nota: Una buena implementación del bot en la "Guía de inicio rápido" a continuación debería sumar fácilmente más de 500 puntos.

Guía rápida

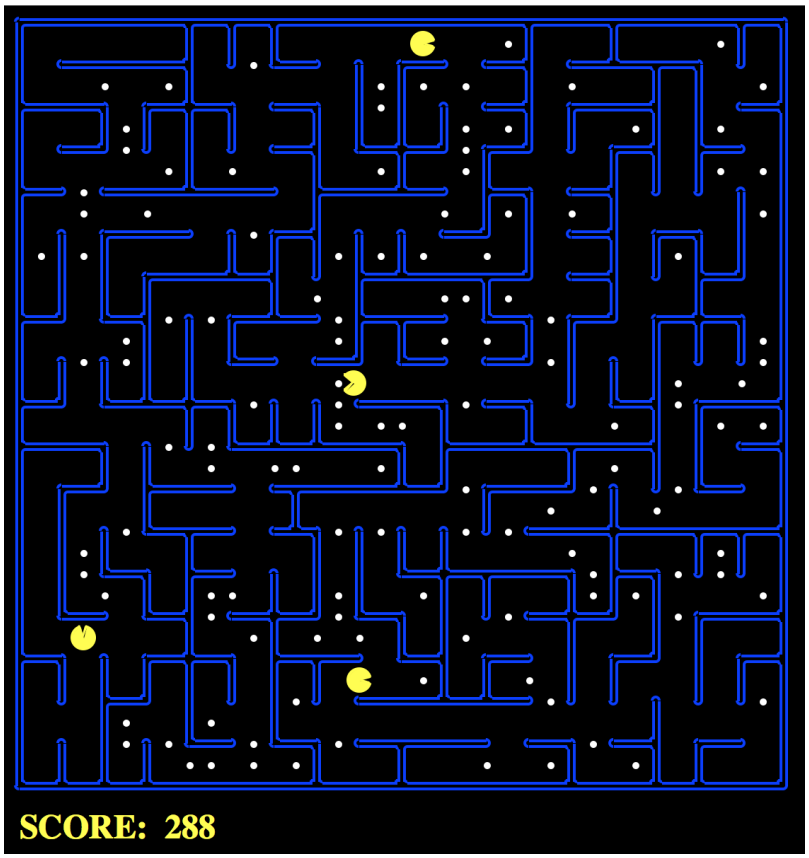
Estos son los pasos a seguir:

1. Descarga el código (`minicontest1.zip`), descomprímelo, y muévete a ese directorio.

2. Copia `search.py` desde el Proyecto 1 al directorio del mini concurso (reemplazando `search.py`).

3. Ve a `myAgents.py` y completa `findPathToClosestDot` en `ClosestDotAgent`, y `isGoalState` en `AnyFoodSearchProblem`. Basta con copiar vuestra solución para el proyecto 1.

4. Ejecuta `python pacman.py`, y deberías ver 4 agentes pacman recorriendo el tablero y comiendo puntos.



Introducción

El código base es casi idéntico al proyecto 1, pero con ligeras modificaciones para incluir más de un agente pacman. Algunas diferencias:

- Vas a controlar N agentes pacman al mismo tiempo. Tu código soportará múltiples pacmans.
- Hay un coste asociado al tiempo que pacman "piensa" (coste de computación). Mira el apartado de puntuación para más detalles.

Ficheros que debes editar:

`myAgents.py`	Contiene todo el código necesario para el agente.

Ficheros que puede interesar mirar:

`searchProblems.py`	El mismo código de P1, con alguna ligera modificación
`search.py`	El mismo código de P1
`pacman.py`	El fichero principal que ejecuta pacman. Este fichero describe un GameState de Pacman, que usaremos en el proyecto.

Ficheros a editar y entregar: Deberás rellenar y entregar `myAgents.py`

Evaluación: No se pueden cambiar los nombres de las funciones o clases proporcionadas dentro del código, o causará conflictos en el autograder. Sin embargo, la corrección de nuestra implementación, no los juicios del calificador automático, será el juez final de la puntuación recibida. Si es necesario,

se revisarán y calificarán las tareas individualmente para asegurar que se reciba el debido crédito por su trabajo.

Deshonestidad: Se comprobará el código contra otras entregas en la clase. Si se copia el código de otra persona y se envía con cambios menores, lo sabremos. Estos detectores de trampas son bastante difíciles de engañar, así que no lo intentéis. Confiamos en que todos enviéis vuestro propio trabajo solamente; por favor no nos decepcionéis.

Ayuda: ¡No estas solo/a! Si os encontráis atascados en algo, debéis poneros en contacto con los profesores del curso para obtener ayuda. Las tutorías y el foro de discusión están a vuestra disposición; por favor usadlos. Si no podéis amoldaros a nuestro horario, informadnos y encontraremos un momento. Queremos que estos proyectos sean gratificantes e instructivos, no frustrantes y desmoralizadores. Pero no sabemos cuándo o cómo ayudar a menos que nos lo solicitéis.
Discusión: Por favor, no enviar *spoilers* al foro.

Reglas

Tableros

Hay un montón de posibilidades en el directorio `layouts`. Los agentes se verán expuestos a mapas de diferentes tamaños y disposiciones de comida.

Puntuación

Se mantendrá el esquema de puntuación del proyecto 1, con algunas modificaciones:

Se mantienen desde el proyecto 1

- +10 por cada punto comido
- +500 por comer todos los puntos

Modificaciones

- -0.4 por cada acción tomada (en el proyecto 1 se penalizaba -1)
- `-1 * total computación usada para calcular la siguiente acción (en segundos) * 1000`

Cada agente empieza con 100 puntos.

Observaciones

Cada agente puede ver el estado completo del juego, incluyendo las posiciones de comida, las posiciones de todos los pacman, etc. Ver la sección de GameState para más detalles.

Ganar y perder

Win: Ganas si recolectas todos los puntos. Tu puntuación es el valor de los puntos actuales.

Lose: Pierdes si tu puntuación llega a cero. Esto puede ser por no encontrar los puntos de manera suficientemente rápida, o por pasar demasiado tiempo calculando. La puntuación será cero. Si tu agente genera una excepción, recibirá una puntuación de cero.

Diseñando agentes

Formato del fichero

Deberías incluir tus agentes en el mismo formato de `myAgents.py`. Tus agentes deben estar completamente contenidos en este fichero. Aunque puedes usar funciones de **`search.py`**

Interface

El `GameState` de `pacman.py` debería ser familiar, pero contiene varias modificaciones para soportar múltiples agentes pacman. El mayor cambio es que muchos métodos de GameState tienen un argumento extra, agentIndex, que sirve para identificar el agente pacman necesario. Por ejemplo, `state.getPacmanPosition(0)` obtendrá la posición del primer agente pacman. Para más información, se debe mirar la clase `GameState` class en `pacman.py`.

Agente

Para empezar a diseñar tu propio agente, te recomendamos hacer subclases de la clase Agent en game.py (esto se hecho ya por defecto). Esto proporciona una variable importante, self.index, que es el agentIndex del agente actual. Por ejemplo, si tenemos 2 agentes, cada agente será creado con un índice único, `[MyAgent(index=0), MyAgent(index=1)]`, que podrá ser usado al decidir las acciones.

El autograder llamará a la función `createAgents` para crear tu grupo de pacmans. Por defecto, está pensado para crear N pacman idénticos, pero puedes modificar el código para devolver un equipo diverso formado por distintos tipos de agentes.

Restricciones

Se deben respetar las APIs y guardar toda la implementación en `myAgents.py`.

Cómo empezar

Por defecto, el juego se ejecuta con `ClosestDotAgent` implementado en la Guía rápida. Para ejecutar tu propio agente, cambia `agent` para el método `createAgents` en `myAgents.py`

```
python pacman.py
```

Hay muchas opciones disponibles:

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

Para ejecutar un juego con un agente:

```
python pacman.py --pacman myAgents.py
```

Tableros (Layouts)

El directorio `layouts` contiene todos los casos de prueba que serán ejecutados en el autograder. Para ver cómo se comporta en un tablero, puedes ejecutar:

```
python pacman.py --layout test1.lay
```

Prueba

Puedes ejecutar un agente en una sola prueba con el siguiente comando. Mira el directorio layouts para comprobar el conjunto completo de pruebas. No hay pruebas ocultas.

```
python pacman.py --l test1.lay
```

Puedes ejecutar el autograder con el siguiente comando.

```
python autograder.py --pacman myAgents.py
```
