Primera práctica de Inteligencia Artificial
Curso 2020-2021

Departament d'Informàtica i Enginyeria Industrial
Universitat de Lleida
carlos@diei.udl.cat
eduard.torres@udl.cat
josep.alos@udl.cat

1. Enunciado

El objetivo de esta práctica es evaluar el conocimiento del alumno sobre los algoritmos de búsqueda informada y no informada descritos hasta el momento en la asignatura. Dichos algoritmos se aplicarán en el contexto de los proyectos docentes de inteligencia artificial sobre el popular juego, Pac-Man [1].

1.1. Descripción de los proyectos Pac-Man

En la página central de los proyectos Pac-Man podemos leer: "Los proyectos Pac-Man se desarrollaron para un curso introductorio de inteligencia artificial (CS 188) de UC Berkeley. Se aplican una serie de técnicas de IA para jugar al Pac-Man. Sin embargo, estos proyectos no se centran en la construcción de la IA de los videojuegos. En su lugar, se enseñan conceptos fundamentales, tales como búsqueda informada en el espacio de estados de un problema, inferencia probabilística y aprendizaje por refuerzo. Estos conceptos subyacen en áreas de aplicación del mundo real, tales como el procesamiento de lenguaje natural, la visión artificial y robótica."

Nosotros nos centraremos en esta práctica en el proyecto de búsqueda [2].

1.2. Tareas a realizar: 10 puntos

1.2.1. Algoritmo A* (búsqueda en grafo): 6 puntos

Implementad el algoritmo A* (búsqueda en grafo). Utilizad la función `aStarSearch` que se encuentra dentro del fichero `search.py` .

Observaciones:

- Considerar que el coste de transición de un estado a otro es 1.
- Utilizar como heurísticas: distancia de Manhattan y distancia Euclídea.
- Siempre que sea posible debes dar la versión optimal del algoritmo.

En este apartado se valora la corrección del algoritmo implementado.

[1] http://ai.berkeley.edu/project_overview.html
[2] http://ai.berkeley.edu/search.html

1.2.2. Implementación Heurística para *Corners Problem*: 1.5 puntos

Proponed e implementad una heurística no trivial y consistente para el *Corners Problem*. Utilizad la función `cornersHeuristic` que se encuentra dentro del fichero `searchAgents.py` .

Para ejecutar este problema podéis utilizar el siguiente comando:

```
python2 pacman.py -l <layout> -p AStarCornersAgent -z 0.5
```

Podéis utilizar como *layout* los mapas *tinyCorners*, *mediumCorners* y *bigCorners*, o crear otros mapas que tengan un punto accesible en cada esquina.

1.2.3. Implementación Heurística para *Food Search Problem*: 2.5 puntos

Proponed e implementad una heurística no trivial y consistente para el problema *FoodSearchProblem*, el cual se encuentra ya implementado dentro del fichero `searchAgents.py` .

En este problema Pacman deberá intentar comer toda la comida que haya repartida por el mapa utilizando el mínimo número de pasos posible.

Utilizad la función `foodHeuristic`, que también se encuentra dentro del fichero `searchAgents.py` .

Para ejecutar este problema podéis utilizar el siguiente comando:

```
python2 pacman.py -l <layout> -p AStarFoodSearchAgent
```

Como *layout* podéis utilizar los mapas *testSearch* y *trickySearch*, y en general cualquier mapa que tenga más de un ítem de comida y que no tenga enemigos.

1.2.4. OPCIONAL: Búsqueda Subóptima

Como tarea opcional se os pide la implementación de un agente que siempre vaya a comerse el ítem de comida que tenga más cerca.

Para ello debéis completar el método `findPathToClosestDot` de la clase `ClosestDotSearchAgent`, dentro del fichero `searchAgents.py` .

Para ejecutar este problema podéis utilizar el siguiente comando:

```
python2 pacman.py -l <layout> -p ClosestDotSearchAgent -z 0.5
```

Como *layout* podéis utilizar el mapas *bigSearch* y en general cualquier mapa que tenga más de un ítem de comida y que no tenga enemigos.

1.3. Implementación

Se valora la calidad y eficiencia de los algoritmos implementados. Es necesario tener en cuenta los siguientes aspectos de la implementación:

- El lenguaje de programación es Python.
- La utilización de un diseño orientado a objetos.
- La simplicidad y legibilidad del código.
- Se recomienda seguir la guía de estilo [3].

[3] `https://www.python.org/dev/peps/pep-0008/`

Nota 1: Tal como se ha comentado en clase, la corrección inicial será realizada con una herramienta automática, por lo que es importante que mantengáis los nombres de las funciones que se han especificado en el enunciado del problema (los cuales corresponden a los que se encuentran en el fichero Python original).

Nota 2: La salida por pantalla del programa que espera la herramienta es la que tiene el proyecto Pac-Man por defecto, si para hacer *debug* necesitáis imprimir algo por pantalla, tomad precauciones para evitar confundir a la herramienta. Dos posibles soluciones:

- Quitad las lineas que imprimen por pantalla antes de entregar la práctica.
- Utilizad la salida estándar de errores '`print >> sys.stderr, "texto"`'.

1.4. Documentación

Documento en pdf (máximo $\approx$ 4 páginas) que incluya:

- una descripción de las decisiones que se han tomado en la implementación del algoritmo A*
- una descripción de las heurísticas propuestas y de las decisiones que se han tomado en su implementación

La página inicial ha de contener el nombre de los integrantes del grupo. La práctica puede realizarse en grupos de dos personas o individualmente.

Se valorará la redacción y presentación del documento.

1.5. Material a entregar

El material evaluable de esta práctica es:

- Todos los archivos de código fuente, modificados o añadidos. Estos ficheros se descargarán sobre el contenido del directorio *search*.
- Documento de la práctica.

Todo el material requerido se entregará en el paquete de nombre `ia-prac1.[tgz|tar.gz|zip]`

```
ia-prac1.[tgz|tar.gz|zip]
├── pacman
│   └── *.py
└── informe.pdf
```