INTELIGENCIA ARTIFICIAL PARA JUEGOS
CURSO 2019/2020

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

PRÁCTICA PUNTUABLE 2: Puzles

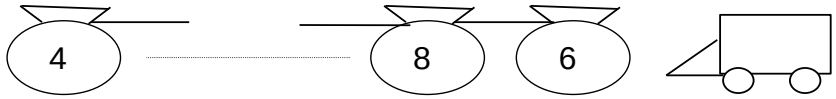
La práctica consta de tres partes independientes. Los resultados deben entregarse a través de la tarea habilitada en el campus virtual. La evaluación de la práctica debe hacerse en clase antes de la entrega y puede realizarse por partes.

A) PRIMERA PARTE

Programar las clases necesarias para representar y resolver problemas en el espacio de estados de **al menos uno** de los siguientes dominios:

El DILEMA DE LOS SACOS

Consideremos el siguiente concurso en el que un concursante se enfrenta a un robot. Se dispone de una fila de sacos de dinero conectados longitudinalmente por un cable, de forma que sólo pueden retirarse sacos de los extremos de la fila. Cada saco tiene impresa su cantidad de dinero (en millones de euros).



El concursante debe elegir entre los dos sacos de los extremos y retirar uno de ellos, pero cada vez que cogemos un saco el robot se apodera del que ha quedado más a la derecha y lo pone fuera de su alcance. El proceso se repite hasta que ya no quedan más sacos en la fila. El concursante desea apoderarse de la mayor cantidad posible de dinero.

Dada la definición anterior del problema anterior, se pide:

a) Definir las clases adecuadas para representar una configuración cualquiera del estado del problema, suponiendo que el número inicial de sacos puede ser cualquiera, pero siempre PAR.
b) Implementar los métodos necesarios para resolver el problema mediante la búsqueda con el algoritmo A*. Para la definición de la función de coste, nótese que maximizar el dinero que se lleva el concursante equivale a minimizar el dinero que sustrae el robot.

2. PROBLEMA DEL PUZLE XL

Considérese la siguiente definición del puzle XL: se dispone de un tablero de NxN, (siendo N impar y mayor o igual que 3) con fichas de tipo ‘X’ y ‘O’, de modo que inicialmente las fichas de tipo ‘X’ forman una X. En la figura se muestra un ejemplo para el caso N = 3. Se desea alcanzar una posición final, en la que las fichas tipo ‘X’ forman una L:

XOX	XOO
OXO	XOO
XOX	XXX
Posición inicial	Posición final

Los movimientos permitidos son intercambiar dos piezas adyacentes (en horizontal o vertical) siempre y cuando tengan fichas de distinto tipo. Por ejemplo, partiendo de la posición inicial mostrada en la figura anterior se pueden alcanzar 12 estados nuevos diferentes. Entre ellos se encuentran, por ejemplo, los siguientes, obtenidos de intercambiar la ficha superior izquierda con las que están a su derecha y debajo, respectivamente:

OXX	OOX
OXO	XXO
XOX	XOX

NOTA: En la generación de sucesores de un estado debe cuidarse de no generar sucesores repetidos.

B) SEGUNDA PARTE

Considérense las clases proporcionadas para la resolución de problemas en el dominio del N-Puzle. Programar las clases y métodos necesarios para resolver problemas en este dominio empleando los siguientes heurísticos:

1. Heurístico del número de piezas descolocadas.
2. Heurístico de la distancia Manhattan.

Programar un método que muestre por pantalla una comparativa del número de iteraciones necesarias para resolver los siguientes puzles con una búsqueda ciega, con cada uno de los heurísticos anteriores, y con heurísticos empleando bases de datos de patrones con 5 y 4 comodines.

```
3 2 5          8 1 7
6 0 4          4 5 6
7 8 1          2 0 3
```

C) TERCERA PARTE

Considérense los mapas del juego Baldur's Gate accesibles para la práctica en el campus virtual. Consideraremos que una posición en el mapa viene determinada por los valores de su fila y columna. Los movimientos permitidos consisten en desplazarse desde una posición a una de sus 8-vecinos, siempre y cuando no corresponda con un obstáculo o se salga del mapa.

Emplearemos como referencia el mapa `ar0011SR.map` Su formato es:

```
type octile\n
height 512\n
width 512\n
map\n
<matriz>
```

donde la `<matriz>` sería en este caso de 512 x 512 caracteres indicando:

@ - un obstáculo

. - una posición válida

El tipo 'octile' se refiere a una vecindad de 8-vecinos.

Programar las clases y métodos necesarios para representar el espacio de estados definido por un mapa del juego, leer un mapa dado, y resolver problemas de búsqueda en el mismo dados un estado inicial y otro final. Para la resolución de los problemas se utillizará el algoritmo A* y un heurístico adecuado.

Programar un método que muestre por pantalla únicamente el trozo del mapa que incluye el camino solución, mostrando el camino mediante 'o', y los obstáculos mediante 'X', tal como se muestra en el siguientes ejemplo:

- Origen: (370, 244), Destino: (394, 359)