EXAMEN PRÁCTICAS INTELIGENCIA ARTIFICIAL

CONVOCATORIA DE JUNIO

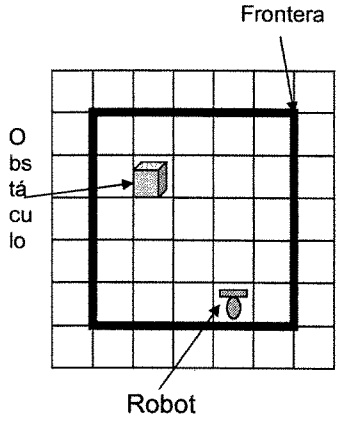
Segundo curso del Grado en Ingeniería Informática, Curso 2012-2013

Nombre:

DNI:

Grupo:

1. **(3 puntos)** Supongamos que un agente trabaja sobre un tablero formado por *nxn* casillas. Sobre este tablero se definen dos zonas: una "zona interior" formada por un tablero de (n-2)x(n-2) casillas inscrito en el tablero general, y una "zona exterior" formada por el resto de las casillas. Separando ambas zona aparece una línea gruesa negra denominada "Frontera". En la figura se muestra un ejemplo de la configuración de un tablero 7x7.



El cometido del robot consiste en llevar todos los obstáculos que se encuentren en la zona interior a la zona exterior. El robot siempre se debe encontrar en la zona interior, y no debe nunca traspasar la frontera.

Para realizar esta tarea, el robot dispone de 3 sensores, un sensor de choque "BUMPER" que le permite detectar el obstáculo, un sensor de infrarrojos "CNY70" que permite ver donde está la línea de la Frontera, y una brújula digital "Brujula" que le indica su orientación en el avance. Los dos primeros sensores se encuentran situados en la parte frontal del robot. La brújula sólo devuelve 4 valores: 0, 1, 2 y 3, representando respectivamente Norte, Este, Sur y Oeste.

Las acciones que puede realizar el robot son las siguientes:

- ***Avanzar***: Avanza una casilla en la dirección que marca su brújula siempre que no tenga un obstáculo delante.
- ***Retroceder***: Retrocede una casilla en la dirección contraria a la que indica su brújula, siempre que no tenga un obstáculo detrás.
- ***Girar***: Gira sin moverse de la casilla en el sentido de la agujas del reloj.
- ***Empujar***: Avanza una casilla en la dirección que marca su brújula. Para que está acción tenga efecto, debe estar activado el sensor de choque.

Se pide:

a) Definir las variables de estado y las reglas de producción necesarias para diseñar un agente reactivo con memoria que, partiendo de una casilla desconocida dentro de la zona interior de un tablero de dimensiones también desconocidas (nunca superiores a 100x100), sea capaz de encontrar el obstáculo y expulsarlo hacía la zona exterior.

b) Añadir sobre el problema del apartado anterior, las variables de estado y las reglas de producción necesarias que permitan al robot expulsar el obstáculo hacía la zona exterior, arrastrándolo ***por el camino más corto de casillas***.

2. **(2 puntos)** Considere el 8-puzzle cuyo estado inicial y estado meta se muestran en la siguiente figura:

2	4	6
1	3	5
8	7	

1	2	3
8		4
7	6	5

Desarrolle el árbol de búsqueda que expande el algoritmo de escalada por máxima pendiente, utilizando las siguientes heurísticas. El orden de aplicación de los movimientos es el siguiente: Arriba, Abajo, Izquierda, Derecha.

a) Heurística 1: **h1(n) = número de piezas mal colocadas.**

b) Heurística 2: **h2(n) = suma de distancia Manhattan de cada pieza sobre su posición correcta.** (Nota: *La distancia Manhattan de una pieza es la suma de las distancias vertical y horizontal a su posición final.*)

Indique en el desarrollo del árbol junto al tablero el valor de la heurística. En caso, de dos tableros con la misma heurística se expande el que primero se generó.

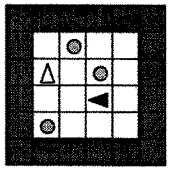
3. **(2 puntos)** Implementar el algoritmo MINIMAX en C++. Para su implementación, suponer que existen dos funciones:

a) **double Valoration (const state & tablero, int jug)** que devuelve la valoración de un tablero desde el punto de vista de *jug* y
b) **state Next_Child(const state &tablero)** que genera el siguiente hijo, aún no generado, del tablero que se pasa como argumento.

Se puede suponer además la existencia de una constante **const state NULO** que es el valor que devuelve la función *Next_Child* cuando ya no tiene más descendientes no generados del tablero que se pasa como argumento.

Si con la información anterior no fuese posible hacer la implementación, añadir tantas suposiciones como se crean necesarias.

4. **(3 puntos)** En la siguiente figura se muestra una situación concreta en el juego de las aspiradoras. El juego consiste en que dos aspiradoras (una blanca y otra negra) que compiten por llevarse la mayor cantidad de suciedad disponible en el tablero.



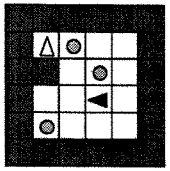
Los elementos que aparecen en el tablero son tres:

a) Las aspiradoras, una blanca y otra negra.
b) La suciedad, representada por casillas con un círculo inscrito.
c) Obstáculos, marcados como casillas oscuras y representan casillas que no puede ocupar una aspiradora.

Las tres acciones que puede realizar cada aspiradora son las siguientes:

- ***Avanzar***: la aspiradora avanza a la casilla adyacente según indica su orientación. El avance sólo es posible si la casilla a la que va a acceder no está ocupada por un obstáculo.
- ***Girar_Derecha***: la aspiradora se mantiene en su posición y cambia su orientación de avance hacia la derecha.
- ***Girar_Izquierda***: la aspiradora se mantiene en su posición y cambia su orientación de avance hacia la derecha.

Como situación especial, cada vez que una aspiradora avanza, deja en la casilla origen un obstáculo, dejando dicha casilla como no transitable. Así, por ejemplo, el efecto de que la aspiradora de la figura anterior avance, el tablero resultante es el siguiente:



Se pide:

a) Construir el árbol del juego hasta profundidad 4 (es decir, desde el nivel 0 hasta nivel 3), tomando este último tablero como nodo raíz, y suponiendo que el primero en jugar es la aspiradora negra.

b) La función de valoración en los nodos hoja es la siguiente:

M(Aspiradora Negra) - M(Aspiradora Blanca)+Dsc(Aspiradora Blanca)

Siendo *M(x)* una función que mide la suciedad obtenida hasta el momento por la aspiradora x que se pasa como argumento y *Dsc(x)* es la distancia de Manhattan de la aspiradora pasada como argumento a la casilla más cercana con suciedad.

c) Calcular el valor MINIMAX de la raíz del árbol construido usando la poda ALFA-BETA. Se debe marcar con claridad que nodos son podados al realizar el proceso.