

WUOLAH



guille31794

www.wuolah.com/student/guille31794



5307

Examen1.pdf

Examen enero 2015



2º Inteligencia Artificial



Grado en Ingeniería Informática



**Escuela Superior de Ingeniería
Universidad de Cádiz**



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.



Exámenes, preguntas, apuntes.



TIEMPO DE REALIZACIÓN: 3 HORAS

Se ha de entregar **cada ejercicio en folios distintos**, incluidos los ejercicios en blanco. Cada folio debe incluir el **nombre del alumno y el número** del ejercicio.

1. Completa el siguiente fragmento de código correspondiente a la función Expandir, dadas las especificaciones de las funciones para la formulación del problema del Puzle. (C)

```
Lista expandir(tElemento *nodo){
    unsigned op;
    Lista sucesores=CrearLista(MAXI);
    for (op=1;op<=NUM_OPERADORES;op++) {
```

```
        InsertarUltimo(nuevo,sucesores);
    } //bloque 1
} //for
return sucesores;
} //expandir
```

<p>puzle.h</p> <p>/* Comprueba si es posible aplicar un operador a una configuración determinada para el puzle. Devuelve 1 si el movimiento es válido y 0 en otro caso. */</p> <p>int esValido(unsigned op, tEstado* estado);</p> <p>/* Aplica un operador a una configuración concreta del puzle. Devuelve la nueva configuración del tablero tras el movimiento. */</p> <p>tEstado *aplicaOperador(unsigned op, tEstado *estado);</p>	<p>busqueda.h</p> <p>#ifndef _tElemento_</p> <p>#define _tElemento_</p> <p>typedef struct NodoBusqueda{</p> <p> tEstado *estado;</p> <p> struct NodoBusqueda *padre;</p> <p> unsigned operador;</p> <p> int costeCamino;</p> <p> int profundidad;</p> <p> } tNodoBusqueda;</p> <p>typedef tNodoBusqueda tElemento;</p> <p>#endif</p>
---	--

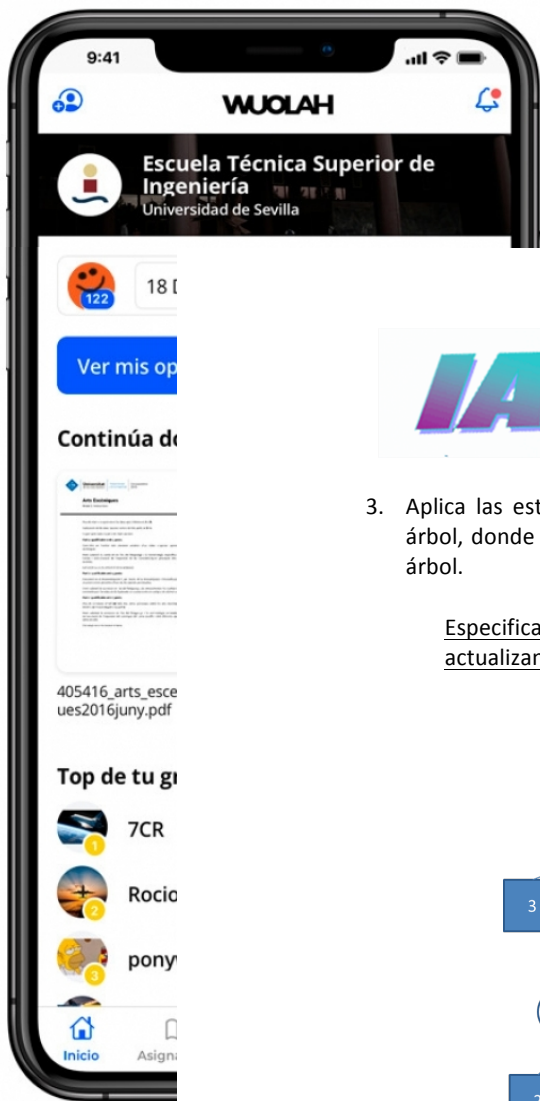
2. Ante la necesidad de comprar un coche, hemos visitado un concesionario, donde nos han indicado las características de un monovolumen, el cuál cumple con nuestras expectativas. El vendedor nos ha indicado que tiene capacidad para 7 personas: 2 en asientos delanteros (piloto y copiloto), 3 en una fila central (dos en una sola pieza que ocupan las posiciones central e izquierda y uno individual en el lado derecho) y otros 2 asientos en una tercera fila.
- Todos los asientos pueden abatirse, excepto el del piloto.
 - El asiento del copiloto podrá abatirse si y sólo si el asiento justo de detrás también se encuentra abatido (es decir, este asiento de la segunda fila no es posible que esté en posición normal si el del copiloto está abatido).
 - Los asientos de la segunda y tercera fila se pueden abatir cuando se desee.
 - Los asientos delanteros además, se pueden desplazar. Para que el asiento del piloto se pueda desplazar es necesario que los dos que se encuentran detrás (posiciones izquierda y central de la segunda fila) estén en posición normal.
 - Para que el asiento del copiloto puede desplazarse el asiento de detrás (derecho de la segunda fila) debe estar en posición normal.
 - Un asiento abatido no se puede desplazar.
 - Un asiento desplazado no se puede abatir.

Obviamente, todas las operaciones de abatir y desplazar sólo se pueden llevar a cabo si los asientos se encuentran en posición normal y, además, son operaciones individuales, es decir, no se pueden abatir ni desplazar dos asientos simultáneamente. La operación de abatir tiene un coste 2, al igual que la operación inversa (volver a posición normal si está abatido) y las operaciones de desplazamiento tienen un coste 1.

Nos indica el vendedor que el coche nos lo entregan con los asientos de la primera desplazados, los de la segunda fila en posición normal y los dos de la tercera fila abatidos. Sin embargo, nuestro coche dispone de un sistema inteligente que es capaz de configurar automáticamente las posiciones del vehículo para que se adapte a nuestras necesidades. Concretamente, para poder llevarnos el coche tenemos que tener en cuenta que seremos dos adultos (acomodados en los asientos de piloto y copiloto) y dos niños, los cuales serán acomodados en la tercera fila y abatiremos los asientos intermedios para colocar las maletas. El sistema inteligente necesita conocer la secuencia de operadores que ha de aplicar para poder acomodarnos todos en el coche, y esto lo realizará con alguna de las estrategias de búsqueda estudiadas en esta asignatura.

Realiza la formalización como un problema cuya resolución puede realizarse mediante búsqueda en espacios de estados tal y como se ha estudiado en la asignatura (en C o en pseudocódigo):

- A. Describe el tipo de datos tEstado para este problema, indicando cuál serían los valores para el estado inicial y final.
- B. Describe el conjunto de operadores.
- C. Diseña las funciones testObjetivo para comprobar si un estado es estado final, la función esVálido, para determinar si es o no posible la aplicación de un operador a partir de un estado concreto, y la función aplicaOperador que lleva a cabo la aplicación de cualquiera de los posibles operadores.
- D. Diseña la función Coste, que calcula el coste de aplicar cualquier operador.
- E. Define una función heurística para este problema, indicar si es o no admisible.
- F. Realiza la expansión del árbol de búsqueda en Anchura hasta profundidad 2



Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.

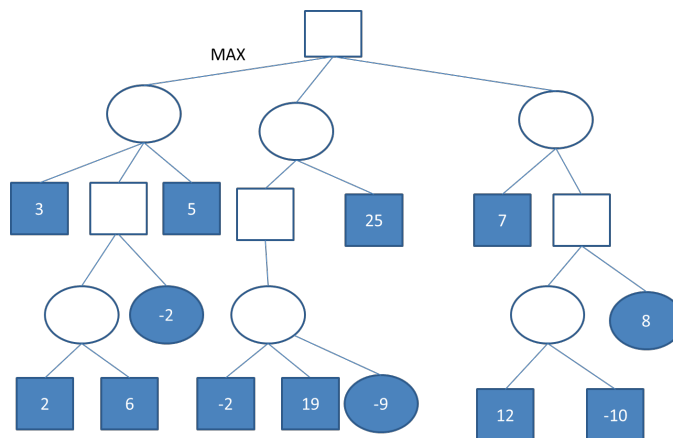


Grado en Ingeniería Informática
Departamento de Ingeniería Informática



3. Aplica las estrategias Minimax y de Poda alfa-beta de Izquierda a Derecha al siguiente árbol, donde los valores de la función de evaluación aparecen en cada nodo terminal del árbol.

Especifica los valores alfa y beta para cada nodo, cómo y en qué orden se van actualizando a medida que se va aplicando la estrategia de poda.



4. RETE (1,5 puntos)

Dado la siguiente Base de Reglas y la Base de Hechos:

- Construye la Red de Redundancia Temporal (RETE).
- Realiza una simulación de la ejecución correspondiente con la red construida y aplicando las siguientes estrategias de Resolución de Conflictos:
 - Especificidad
 - Refracción

Al igual que la acción Assert se representa con el símbolo (+), la acción Modify se representa con los símbolos (+-).

Base de Reglas

<pre>(defrule R1 ?f1<-(valvula (nombre ?v) (T1 ?t1)(T2 ?t2) (presion ?P1) (estado abierta)) ?f2<-(alarma (nombre ?v) (presion ?p2) (limite_tempe ?lt) (estado ON)) (test (>= ?P1 ?p2)) => (modify ?f1 (presion 0)))</pre>	<pre>(defrule R2 ?f1<-(valvula (nombre ?v) (T1 ?t1) (T2 ?t2) (presion ?p) (estado cerrada)) (test (<= ?p 5)) (test (> ?t2 ?t1)) => (modify ?f1 (estado abierta)) (assert (alarma (nombre ?v) (presion ?p) (limite_tempe ?t1) (estado ON)))))</pre>
<pre>(defrule R3 ?f1<-(valvula (nombre ?v1) (T1 ?t1) (T2 ?t12) (presion 0) (estado abierta)) ?f2<-(valvula (nombre ?v2) (T1 ?t1) (T2 ?t22) (presion 0) (estado abierta)) ?f3<-(alarma (nombre ?v1) (presion ?p3) (limite_tempe ?lt1) (estado ON)) ?f4<-(alarma (nombre ?v2) (presion ?p4) (limite_tempe ?lt2) (estado ON)) (test (neq ?v1 ?v2)) =>)</pre>	



```
(modify ?f1 (presion (+ ?p3 ?p4)) (estado cerrada))
(modify ?f2 (presion (+ ?p3 ?p4)) (estado cerrada))
(retract ?f3 ?f4)
```

Base de Hechos

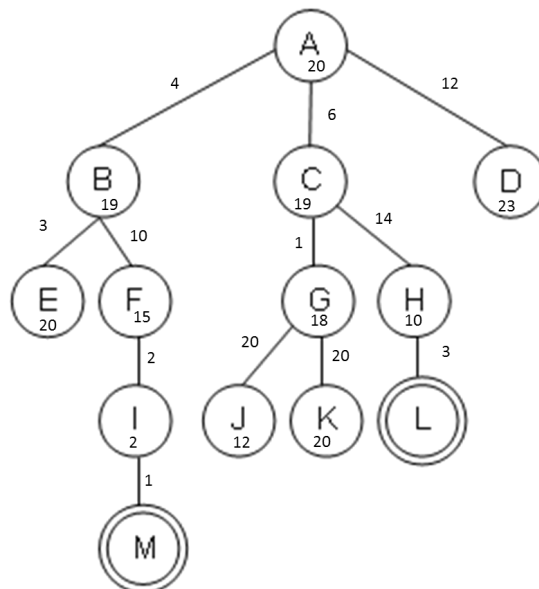
(valvula (nombre Salida) (T1 10) (T2 55) (presion 5) (estado cerrada)))
(valvula (nombre Pasillo) (T1 41) (T2 37) (estado abierta))
(valvula (nombre Entrada) (T1 10) (T2 35) (presion 2) (estado cerrada))

5. El siguiente grafo representa un problema de espacio de estados, donde los nodos representan los estados del problema, los arcos el coste real de ir de un nodo a otro, y el número junto al nombre de cada nodo representa el valor de la heurística en cada estado. El estado inicial es A y estados finales L y M.

Determina qué solución y qué coste real se obtienen aplicando el algoritmo de búsqueda de A*.

***** En caso de empate, los nodos se almacenan y extraen de cualquier lista en Orden Alfabético**

Especifica el proceso de búsqueda paso a paso, estableciendo en cada iteración cuál es el nodo Actual, y el contenido de las listas de nodos Abiertos y Cerrados junto con los valores de la función de evaluación.



6. Compra en un Supermercado

Se pretende implementar mediante un sistema basado en reglas un agente inteligente que realice de forma automática la compra en un supermercado. Para ello se dispone de un agente inteligente que se desplaza por los pasillos del supermercado con su propio carro buscando los productos de la lista de la compra. Supongamos los siguientes hechos que deberán definirse de forma estructurada mediante plantillas:

- **Productos:** contiene la información de los productos del supermercado: un identificador y un nombre, para guardar su ubicación se tendrá en cuenta el número del pasillo (numerados de 1 a 12), además se almacenará la cantidad de ese producto que hay en stock y el precio unitario.
- **Pedidos:** los clientes pueden realizar pedidos que en el sistema entrarán como hechos con la siguiente información: identificador del cliente que realiza el pedido, el producto y las unidades que desea comprar. Por ejemplo:
(pedido (id_cliente 33) (id_producto 1) (unidades 3))

- **Carro de la compra:** existe un carro de la compra por cada cliente, contendrá el identificador del cliente, el número de artículos que lleva comprados, el importe total en euros así como el pasillo actual donde se encuentra el agente.

Las acciones que se van a realizar son las siguientes:

- **Asignar un carro:** cuando un cliente nuevo se registra en el sistema introduce su identificador (se realizará con instrucciones del tipo: (assert (nuevo_cliente id_cliente)) y automáticamente se le asigna un carro que saldrá del pasillo 1, y la factura y número de productos comprados estará a 0.

A partir de este momento, los pedidos que este cliente realiza se representarán con instrucciones del tipo: (assert (pedido (id_cliente xx) (id_producto yy) ...etc)

Las acciones que se podrán realizar entonces serán:

- **Mover carro:** cuando se realiza el pedido de un producto y el carro no está en ese pasillo, se mueve el carro a un pasillo contiguo en orden ascendente, es decir, si el pasillo actual del carro es el 3, esta regla mueve el carro al pasillo 4. Cuando se encuentre en el pasillo 12, entonces pasará al pasillo 1.
- **Comprar:** se puede comprar un producto pedido cuando el carro se encuentre en el mismo pasillo que este producto y exista stock suficiente. El pedido realizado se borra, y se calcula el precio que se ha de añadir a la factura del carro. La cantidad en stock en el producto también debe actualizarse. La compra deberá tener prioridad sobre el desplazamiento del carro por el supermercado.
- **Existencias Insuficientes:** Cuando no hay stock suficiente para realizar la compra, se avisa al cliente con un mensaje en pantalla y se elimina este pedido.

- Define las reglas necesarias para que el agente inteligente pueda llevar a cabo su labor de forma correcta .
- Usa plantillas para definir las propiedades de las cajas y de los artículos.
- Define y usa al menos una función para la realización de los cálculos aritméticos.
- Realiza una traza de ejecución con los siguientes hechos de ejemplo:

```
(deffacts productos
  (producto (id_producto 1) (nombre leche)(pasillo 3) (stock 45) (precio 1))
  (producto (id_producto 2) (nombre galletas) (pasillo 4)(stock 10) (precio 2.20))
  (producto (id_producto 3) (nombre cafe) (pasillo 5) (stock 2) (precio 2.6))
  (producto (id_producto 4) (nombre arroz) (pasillo 12) (stock 30) (precio 1.98))
)
(assert (nuevo_cliente 33))
(assert (pedido (id_cliente 33) (id_producto 1) (unidades 3)))
(assert (pedido (id_cliente 33) (id_producto 2) (unidades 30)))
```