EJERCICIO RESUELTO

EL ROBOT Y LAS SALAS

Un robot ha de moverse por un edificio compuesto por 12 salas dispuestas como muestra el dibujo. Todo par de salas contiguas está comunicado por una puerta con su portero. Este agente inteligente puede realizar movimientos a salas adyacentes por la derecha o hacia abajo, pero nunca vuelve atrás. El portero exige una pequeña propina para permitir el paso: 100 euros en las puertas con trazo simple, 200 euros en las puertas con trazo doble y 300 euros en las puertas con trazo triple según la figura.

1. **Realiza la formalización del problema** (en pseudocódigo).
   Describe el estado inicial, el estado final, la función testObjetivo y el conjunto de operadores para este problema (funciones esValido y aplicaOperador).

2. **Halla una función heurística admisible del coste empleado en ir desde cualquier sala a la sala de los Videojuegos. Justifica por qué es admisible, y pon un ejemplo de Consistencia (o de No Consistencia).**
3. **Aplica el algoritmo A* para hallar el camino menos costoso que lleva del vestíbulo a la sala de los Videojuegos. (Indica en cada paso la lista de Abiertos, Cerrados y las funciones de evaluación de los nodos).**
4. **Indica el recorrido resultante de acuerdo a la heurística propuesta y el Coste Real que le supone al Robot llegar a la sala de Videojuegos.**

**En caso de empate en la función de evaluación elegir como nodo-actual el primero por la izquierda y por abajo. Por ejemplo entre D y B se elige D primero.*

EJERCICIO RESUELTO

1. FORMALIZACIÓN

Espacio de estados: La descripción de un estado identifica la ubicación del robot en una sala. Los estados se pueden representar por un par de números (*i*,*j*) que representan las coordenadas de la celda en la que se encuentre el robot en cada momento, considerando la disposición de las salas como si de una matriz de MAX_FIL x MAX_COL (4x3 para este enunciado) se tratara. Dado que no se necesita almacenar ningún dato sobre las salas, no es necesario disponer de la matriz en la representación, sino únicamente de su tamaño.

Estado Inicial: Posición de partida para el robot. Para este enunciado sería el par (1,1), correspondiente al vestíbulo.

Estado Final: Posición final del robot, que es la correspondiente a la sala de videojuegos. Para este enunciado es (4,3).

Test_Objetivo: comprobar que (*i*,*j*)= Estado Final

Operadores: Movimientos de avance a salas adyacentes: abajo y derecha.

Coste: El coste de alcanzar una nueva posición puede ser 100, 200 ó 300 euros, dependiendo de la celda en que se encuentre.

Descripción de los operadores y reglas:

Mover abajo: el robot se mueve a la sala de abajo. Es aplicable a estados donde el robot no se encuentre en la última fila.

Mover derecha: el robot se mueve a la sala contigua por la derecha. Es aplicable a estados donde el robot no se encuentre en la última columna.

EJERCICIO RESUELTO

Diseño de las funciones en pseudocódigo:

```
const
        MAX_FIL = 4
        MAX_COL = 3

        ABAJO = 1
        DERECHA = 2

tipo
        registro: tEstado
                entero: fil, col
        fin_registro

// Funciones para crear el estado Inicial y el estado Final

tEstado: función EstadoInicial()
var
   tEstado: inicial
inicio
        inicial.fil←1,
        inicial.col←1

devolver estado
 fin_función

tEstado: función EstadoFinal()
var
   tEstado: final
inicio
        final.fil←MAX_FIL
        final.col←MAX_COL

devolver estado
fin_función


lógico: función testObjetivo(E tEstado: actual)
var
        lógico: objetivo
        tEstado: final
inicio
        final=EstadoFinal()
        devolver  objetivo← (actual.fil=final.fil)  ∧ ( actual.col=final.col)
fin_función
```

EJERCICIO RESUELTO

```
lógico: función esValido(E entero: op, E tEstado: estado)
var
  lógico: valido
inicio
        según_sea (op) hacer
                        ABAJO:        valido ← estado.fil<MAX_FIL
                        DERECHA:      valido ← estado.col<MAX_COL
                        en_otro_caso: valido ← falso
        fin_según
        devolver valido
fin_función

tEstado: función aplicaOperador(E entero: op, E tEstado: actual)
var
   tEstado: nuevo
inicio
           nuevo←actual
           según_sea (op) hacer
                        ABAJO:        nuevo.fil ← nuevo.fil+1
                        DERECHA:      nuevo.col ← nuevo.col+1
           fin_según
           devolver nuevo
fin_función
```

//Calcula el coste de aplicar el operador desde la posición actual, estos cálculos son totalmente dependientes de este enunciado.

```
entero: función coste(E entero: op, E tEstado: actual)
var
  entero: coste
inicio
  según_sea (op) hacer
        ABAJO:          si actual.fil=3 entonces
                            coste← 300
                         si_no si actual.fil=2 ∧ (actual.col=1 ∨ actual.col=3) entonces
                                    coste← 200
                                si_no coste← 100
                                fin_si
                        fin_si
        DERECHA:        si (actual.fil=4 ∧ actual.col=3) entonces
                            coste← 300
                        si_no
                            si (actual.fil=2 ∧ actual.col=1) entonces
                                coste← 200
                            si_no coste← 100
                            fin_si
                        fin_si
        en_otro_caso: coste ← 0
  fin_según
  devolver coste
fin_función
```

EJERCICIO RESUELTO

2. HEURÍSTICA

Teniendo en cuenta que la disposición de las salas se considera como una matriz bidimensional y que el robot se mueve por las celdas, una heurística admisible podría estar relacionada con el número de celdas que restan hasta la solución, expresada mediante la distancia de Manhattan: Nº de Filas y Nº de Columnas desde la celda actual hasta la celda Objetivo.

Por otra parte, el enunciado indica que el coste mínimo para atravesar dos salas es de 100, por lo que, realizando una relajación de las condiciones del problema, se puede estimar de forma optimista, que se va a emplear el mínimo coste en cada movimiento que queda por realizar.

De esta manera una posible función heurística podría ser definida como:

h(n)= 100*distancia de Manhattan de *n* al estado Objetivo

siendo *n* cualquier estado del espacio del problema.

```
entero: función heuristica(E tEstado: actual)
var
        entero: h
inicio
         //distancia de Manhattan
        h ← 100* abs(final.col – actual.col) + abs(final.fil - actual.fil)
        devolver  h
fin_funcion
```

Admisibilidad de la Heurística: La función heurística es admisible si nunca sobreestima el coste real de llegar a la solución. Esta heurística establece el mínimo número de movimientos necesarios para llegar a la solución, ya que el robot se mueve por la cuadrícula a las salas adyacentes, de una en una y además suponiendo el mínimo coste que cada movimiento tiene, por tanto nunca sobreestimará el coste real de llegar a la solución.

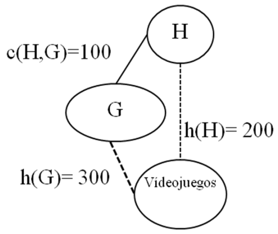
La condición de admisibilidad es suficiente cuando tratamos con árboles de búsqueda.
De forma general para resolver grafos de búsqueda (donde se ha de tener en cuenta la posibilidad de estados repetidos), existen dos soluciones para garantizar la optimalidad de A*:

- la primera opción es siempre desechar el camino más caro de entre dos posibles en un nodo repetido, dentro del algoritmo de búsqueda hay que implementar este proceso.
- la segunda opción consiste en diseñar funciones heurísticas que sean monótonas o consistentes, con lo cual se garantizará que la estrategia siempre optará por el camino de menor coste.

EJERCICIO RESUELTO

Ejemplo de Consistencia: La función es consistente si, en un estado *n* la estimación realizada para llegar al objetivo no es mayor que el coste de generar un sucesor de *n* más la estimación de este sucesor en llegar al objetivo. Así una función heurística consistente es también decreciente.

Por ejemplo, el estado central H:



$$h(H) <= h(G) + c(H,G)$$

La estimación del coste del nodo H al objetivo (h(H)=200), es una estimación más optimista que si se eligiera un camino alternativo, por el nodo G cuyo coste es mayor. Por tanto, al ser la heurística consistente, A* tomará el camino más directo a la solución.

3. RESOLUCIÓN MEDIANTE A*

Comienza en el estado inicial, el vestíbulo, etiquetado con la letra A, y se correspondería con el primer Nodo Actual. Para cada nodo actual se ha de verificar, mediante la función testObjetivo, que no es el estado final que se busca, y se procedería a la expansión de sus sucesores calculando la función de evaluación para cada nodo e introduciendo estos nodos en la lista de Abiertos. (El movimiento que genera al padre del nodo actual no se considera)

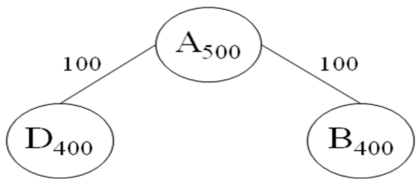
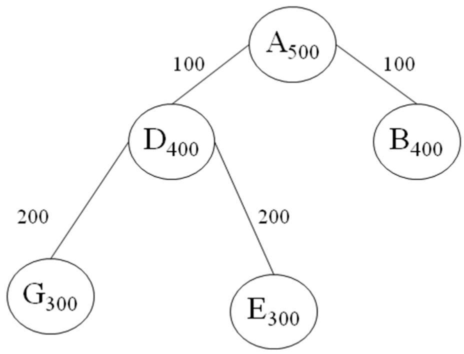
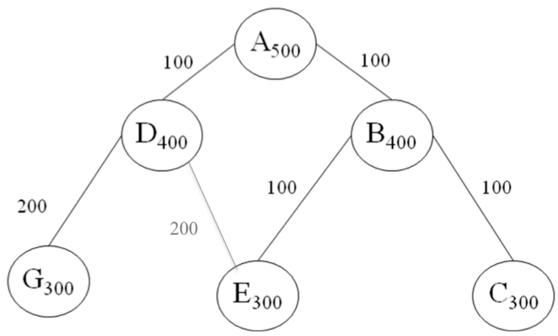
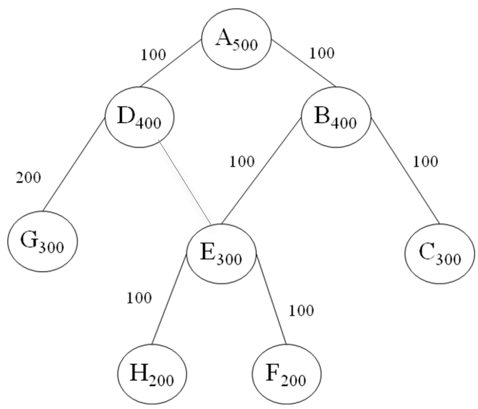
**En caso de empate en la función de evaluación elegir como nodo-actual el primero por la izquierda.*

CONTROL DE ESTADOS REPETIDOS:

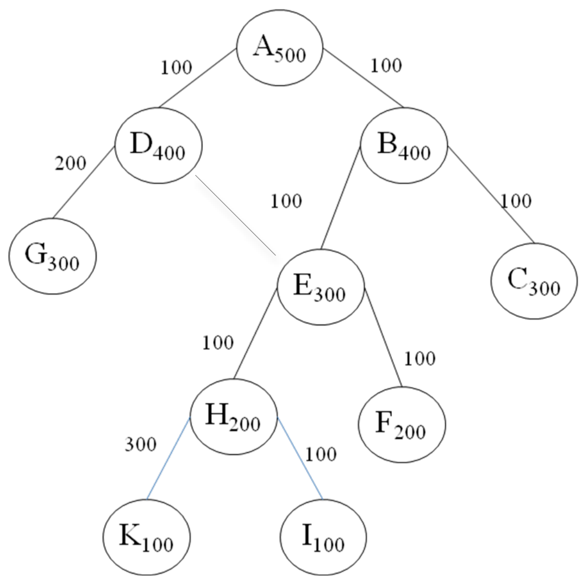
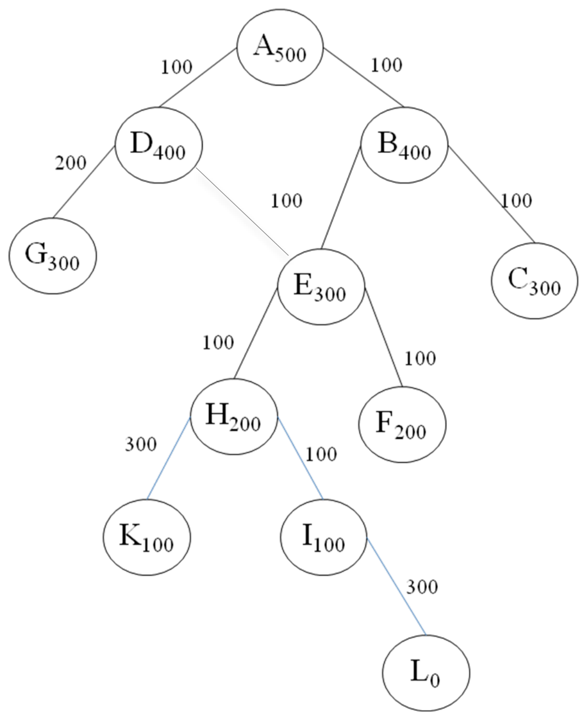
Todos los posibles sucesores se almacenan en la lista de Abiertos de forma ordenada. Para cada nodo Actual, se comprueba si se encuentra en la lista de Cerrados:

- Si *Actual* es un estado repetido que está en la lista de CERRADOS:
    - Si la nueva función de evaluación *f(Actual)* es menor que la del nodo en Cerrados, se considera para su expansión guardando los sucesores en Abiertos (es un mejor camino)
    - Si no (el valor de *f(Actual)* es mayor o igual) no se considera este nodo para su expansión ni se guarda en ninguna lista
    - **¡Atención! No se hace nada más con los sucesores.**

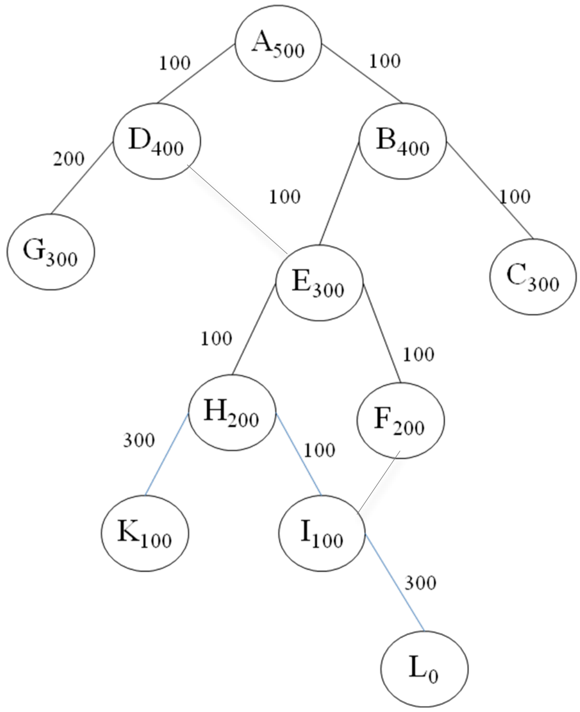
EJERCICIO RESUELTO

Nodo Actual: A(500)	
Abiertos: D(500), B(500) Cerrados: A(500)	
Nodo Actual: D(500) Abiertos: B(500), G(600) E(600) Cerrados: A(500), D(500)	
Nodo Actual: B(500) Abiertos: E(500), C(500), G(600), E(600), Cerrados: A(500), D(500), B(500) **E se guarda otra vez en Abiertos*	
Nodo Actual: E(500) Abiertos: H(500), F(500), C(500), G(600), E(600) Cerrados: A(500), D(500), B(500), E(500)	

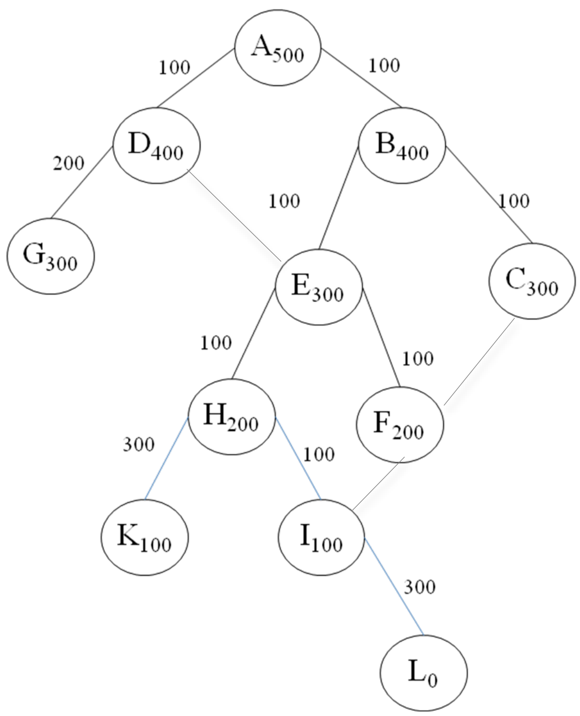
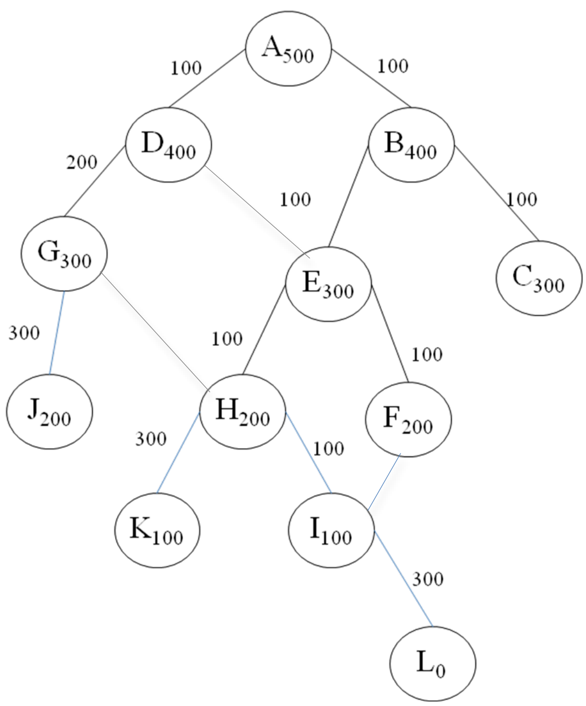
EJERCICIO RESUELTO

Nodo Actual: H(500) Abiertos: I(500), F(500), C(500), G(600), E(600) K(700) Cerrados: A(500), D(500), B(500), E(500), H(500),	
Nodo Actual: I(500) Abiertos: F(500), C(500), G(600), E(600), K(700), L(700) Cerrados: A(500), D(500), B(500), E(500), H(500), I(500)	

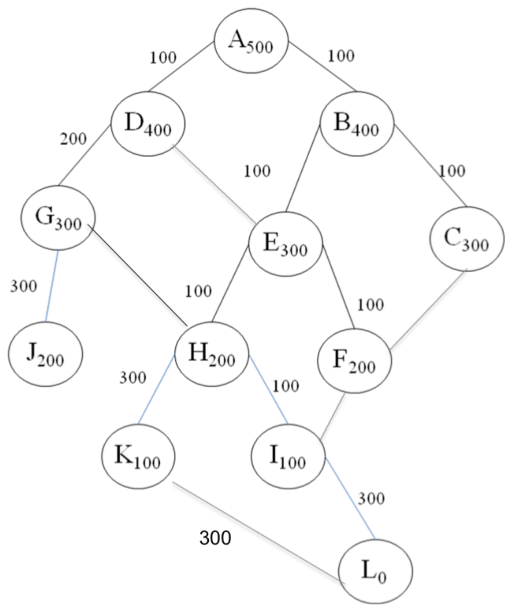
EJERCICIO RESUELTO

Nodo Actual: F(500) Abiertos: I(500) C(500), G(600), E(600), K(700), L(700) Cerrados: A(500), D(500), B(500), E(500), H(500), I(500), F(500)	
Nodo Actual: I(500) Abiertos: C(500), G(600), E(600), K(700), L(700) Cerrados: A(500), D(500), B(500), E(500), H(500), I(500), F(500)	Nodo repetido, se encuentra en Cerrados con idéntica función de evaluación, por tanto no se considera su re-expansión

EJERCICIO RESUELTO

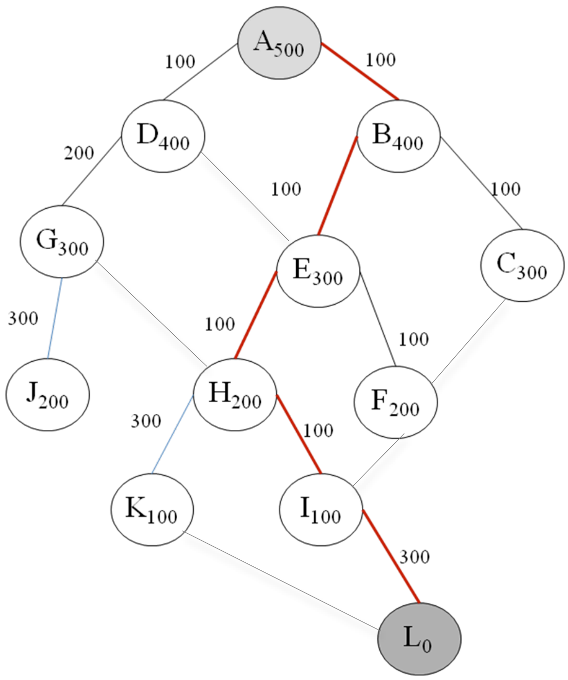
<table>
<tr>
<td>Nodo Actual: C(500)
Abiertos: F(500), G(600), E(600), K(700), L(700)
Cerrados:
A(500), D(500), B(500), E(500), H(500), I(500), F(500), C(500)</td>
<td>
</td>
</tr>
<tr>
<td>Nodo Actual: F(500)
Abiertos: G(600), E(600), K(700), L(700)
Cerrados:
A(500), D(500), B(500), E(500), H(500), I(500), F(500), C(500)</td>
<td>Nodo repetido, se encuentra en Cerrados con idéntica función de evaluación, por tanto no se considera su re-expansión</td>
</tr>
<tr>
<td>Nodo Actual: G(600)
Abiertos: H(600), E(600), K(700), L(700), J(800)
Cerrados:
A(500), D(500), B(500), E(500), H(500), I(500), F(500), C(500), G(600),</td>
<td>
</td>
</tr>
</table>

EJERCICIO RESUELTO

Nodo Actual: H(600) Abiertos: E(600), K(700), L(700), J(800) Cerrados: A(500), D(500), B(500), E(500), H(500), I(500), F(500), C(500), G(600),	Nodo repetido, se encuentra en Cerrados con función de evaluación menor a la actual, por tanto no se considera su re-expansión
Nodo Actual: E(600) Abiertos: K(700), L(700), J(800) Cerrados: A(500), D(500), B(500), E(500), H(500), I(500), F(500), C(500), G(600),	Nodo repetido, se encuentra en Cerrados con función de evaluación menor a la actual, por tanto no se considera su re-expansión
Nodo Actual: K(700) Abiertos: L(700) J(800) L(900) Cerrados: A(500), D(500), B(500), E(500), H(500), I(500), F(500), C(500), G(600), K(700)	

EJERCICIO RESUELTO

Nodo Actual: L(700) **OBJETIVO**



4. CAMINO SOLUCIÓN

A, B, E, H, I, L con un coste real de 700 euros

El orden de expansión en este problema puede ser diferente cuando las condiciones en caso de empate en la función de evaluación se establecen de acuerdo a otro criterio: orden alfabético, el nodo último se expande primero, etc. Pero el coste del camino a la solución aplicando A* con esta heurística admisible y consistente debe ser el mismo.