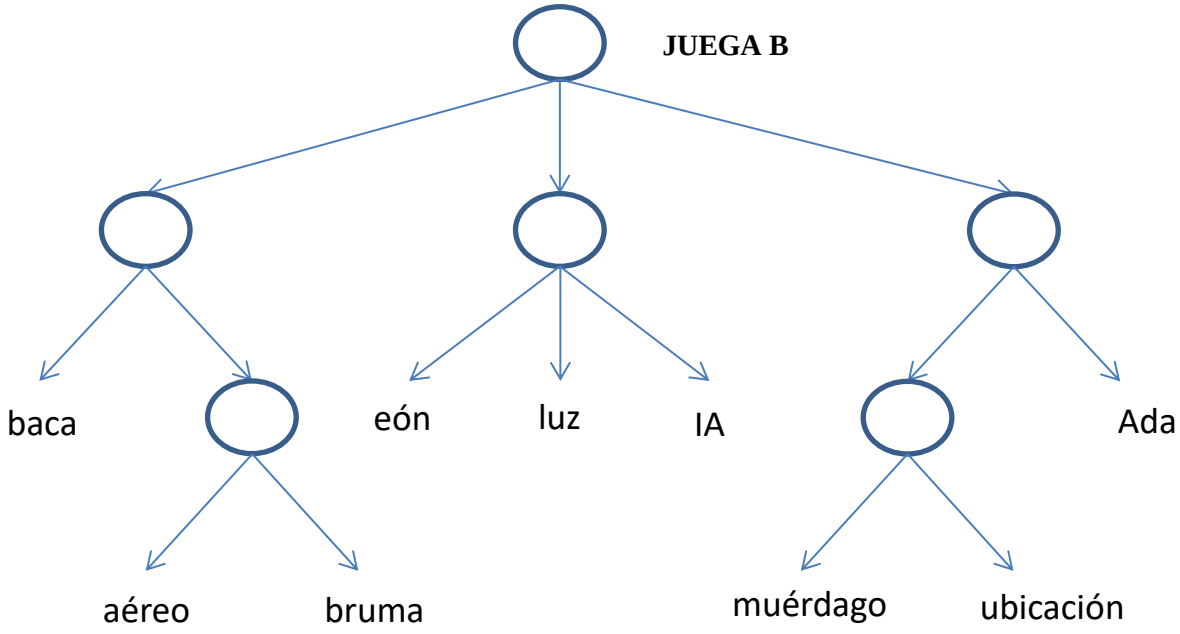
Inteligencia Artificial
Hoja 1: Búsqueda y juegos

Publicación:	2017/02/14
Límite realización:	2017/02/21
Resolución en clase:	2017/02/21

1. Consideremos un juego en el cual los jugadores forman palabras. El jugador A consigue puntos por las vocales y el B por las consonantes de la palabra formada en el nodo terminal. Consideremos el árbol de juego para un estado del juego en el que tiene el turno el jugador B:



Diseña una función de evaluación para completar la definición el juego y explica su significado.
Utiliza el algoritmo minimax con poda alfa-beta para determinar cuál es el valor minimax en el nodo raíz del árbol de juego y cuál es la jugada óptima en este estado del juego para B.

Indica con una etiqueta numérica el orden de recorrido de los nodos en el árbol. En cada nodo interno visitado, indica los valores [α, β]. En los nodos finales visitados (y **únicamente en ellos**) indica el valor de la función de evaluación. Indica asimismo los nodos en los que hay poda.

SOLUCIÓN:

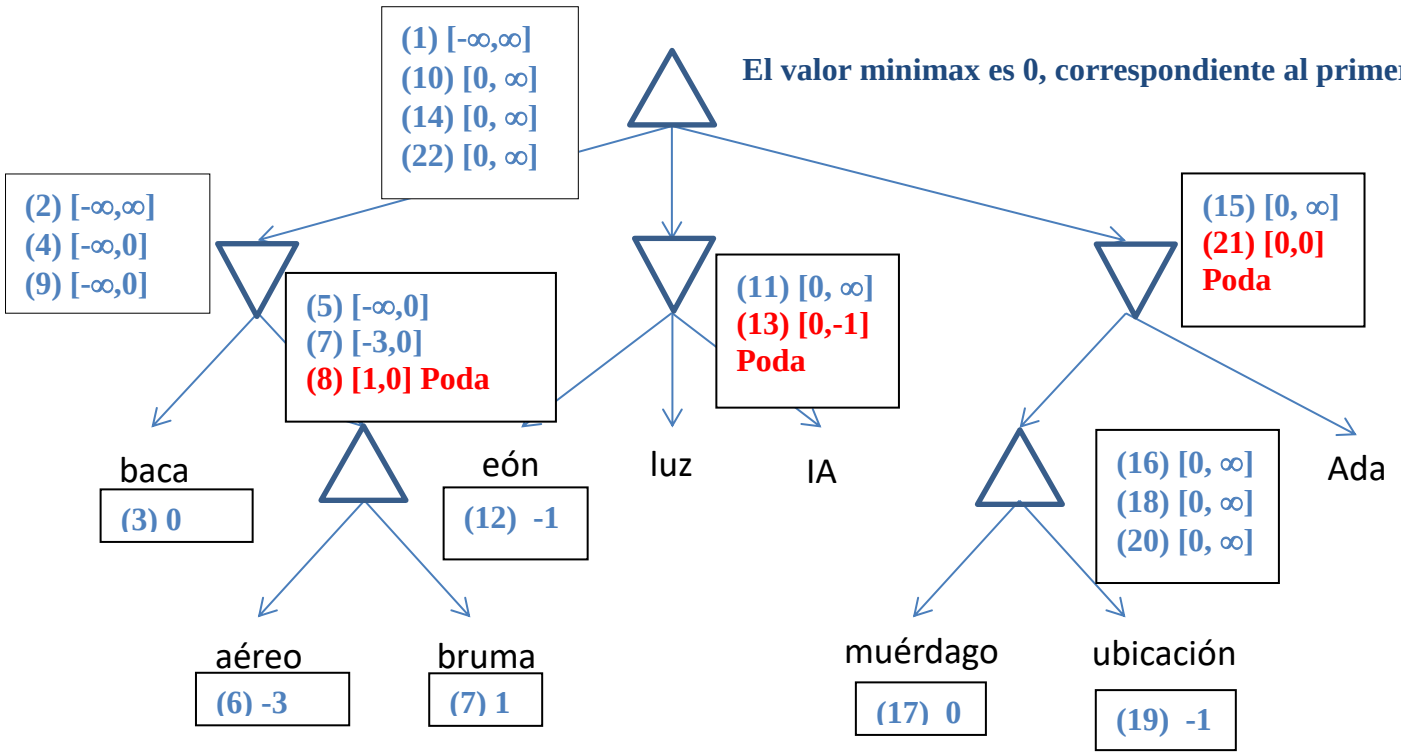
- El jugador B (tiene el turno) es MAX
- El jugador A es MIN.

El juego está parcialmente definido por el enunciado.

Se podrían definir las siguientes funciones de evaluación en términos de los números de vocales ($n_{vocales}$) y de consonantes ($n_{consonantes}$) de la palabra formada en el nodo final

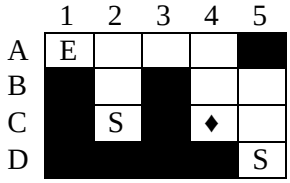
	$n_{consonantes} > n_{vocales}$ [gana B (MAX)]	$n_{consonantes} = n_{vocales}$	$n_{consonantes} < n_{vocales}$ [gana A (MIN)]
JUEGO 1	1	0	-1
JUEGO 2	$n_{consonantes}$	0	$-n_{vocalees}$
JUEGO 3	$n_{consonantes}-n_{vocalees}$		

El árbol para el juego 3 es



El valor minimax es 0, correspondiente al primer movimiento explorado.

2. Un jugador IA necesita salir del laberinto partiendo de la entrada (casilla A1, marcada E). Las casillas en negro no son accesibles. Para poder salir del laberinto, el jugador necesita haber pasado antes por la casilla C4, para recoger la llave (♦) que le permitirá abrir la cerradura en cualesquiera de las salidas del laberinto (casillas C2 y D5, marcadas con S).



El jugador solo se puede desplazar a las casillas adyacentes horizontal y verticalmente dentro del laberinto. Para coger la llave no es necesario que el jugador realice ninguna acción especial: basta con posicionarse en la casilla C4. Una vez que el jugador ha pasado por dicha casilla lleva consigo la llave.
En caso de que existan diversas opciones, el orden de aplicación de las acciones será **abajo, izquierda, arriba, derecha**. **En caso de empate se explora primero el nodo que ha sido generado antes. Todas las acciones tienen el mismo coste.**

a) Formaliza los estados de búsqueda.
b) Formaliza las acciones.
c) En términos de la formalización realizada ¿Cuál es el test objetivo?
d) Define una buena heurística para este problema basada en la distancia de Manhattan y que sea monótona.
e) Utiliza dicha heurística en la búsqueda de una solución al problema usando A* sin eliminación de estados repetidos. Indica en el árbol de búsqueda el valor de los valores g, h y f de cada nodo y el orden en el que se exploran los nodos expandidos.
f) ¿Se ha encontrado la solución óptima? ¿Garantiza esta estrategia con la heurística propuesta encontrar la solución óptima para cualquier laberinto de este tipo? ¿Por qué?

SOLUCIÓN:

a) Formaliza los estados de búsqueda.

[<fila> <columna> <llave>];
<fila> ∈ {A,B,C,D}
<columna> ∈ {1,2,3,4,5}
<llave> ∈ {-,♦}

Excepto los estados correspondientes a casillas en negro:
[A5 <llave>],
[B1 <llave>], [B3 <llave>],
[C1 <llave>], [C3 <llave>],
[D1 <llave>], [D2 <llave>], [D3 <llave>], [D4 <llave>]

b) Formaliza las acciones.
Abajo: incrementa fila A → B → C → D
Izquierda: decrementa columna 5 → 4 → 3 → 2 → 1
Arriba: decrementa fila D → C → B → A
Izquierda: incrementa columna 1 → 2 → 3 → 4 → 5

Estas acciones solo se pueden aplicar en caso de que la casilla de destino no esté bloqueada (color negro). No afectan al valor <llave> del estado.

c) En términos de la formalización realizada ¿Cuál es el test objetivo?

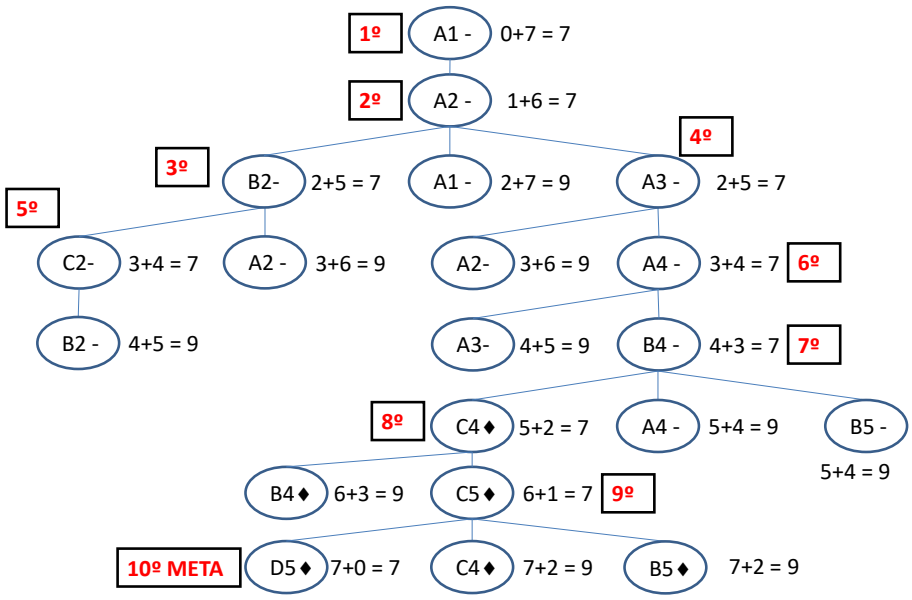
¿Se encuentra el jugador en un celda de salida (C2 o D5) y
lleva la llave (<llave> == ♦)?

d) Define una heurística monótona para este problema.
Nota: Es necesario elegir una buena heurística. En caso contrario, los siguientes apartados serán complejos de resolver.

h([<fila> <columna> <llave>]) =
Si llave == -
[distancia de Manhattan desde la casilla del jugador a ♦ (casilla C4)] +
[menor distancia de Manhattan de ♦ (casilla C4) a S]
Si llave == ♦
[menor distancia de Manhattan desde la casilla del jugador a S]

Es monótona, ya que es el coste de la solución óptima de un problema relajado, en el que se han eliminado las paredes dentro del laberinto.

e) Utiliza dicha heurística en la búsqueda de una solución al problema usando A* sin eliminación de estados repetidos.



f) ¿Se ha encontrado la solución óptima? ¿Garantiza esta estrategia con la heurística propuesta encontrar la solución óptima para cualquier laberinto de este tipo? ¿Por qué?

Sí, porque una heurística monótona es admisible.

A* sin eliminación de estados repetidos + heurística admisible = óptima

3. Consideremos el juego de las **tres en raya con gravedad**. Se juega en un tablero vertical de dimensiones 3 x 3 celdas. En él dos jugadores, a los que denominaremos X y O, por los símbolos que aparecen en las fichas que colocan en el tablero, alternan sus movimientos. Inicialmente el tablero está vacío. El primer movimiento en una partida corresponde a X. En cada turno, el jugador que realiza el movimiento deja caer una de sus fichas en una de las columnas libres del tablero. Por efecto de la gravedad las fichas caen hasta la celda desocupada más baja. Los movimientos se exploran comenzando por la columna libre que se encuentre más a la izquierda. El objetivo es colocar tres fichas alineadas (horizontal, verticalmente o en diagonal) antes de que lo haga el oponente.

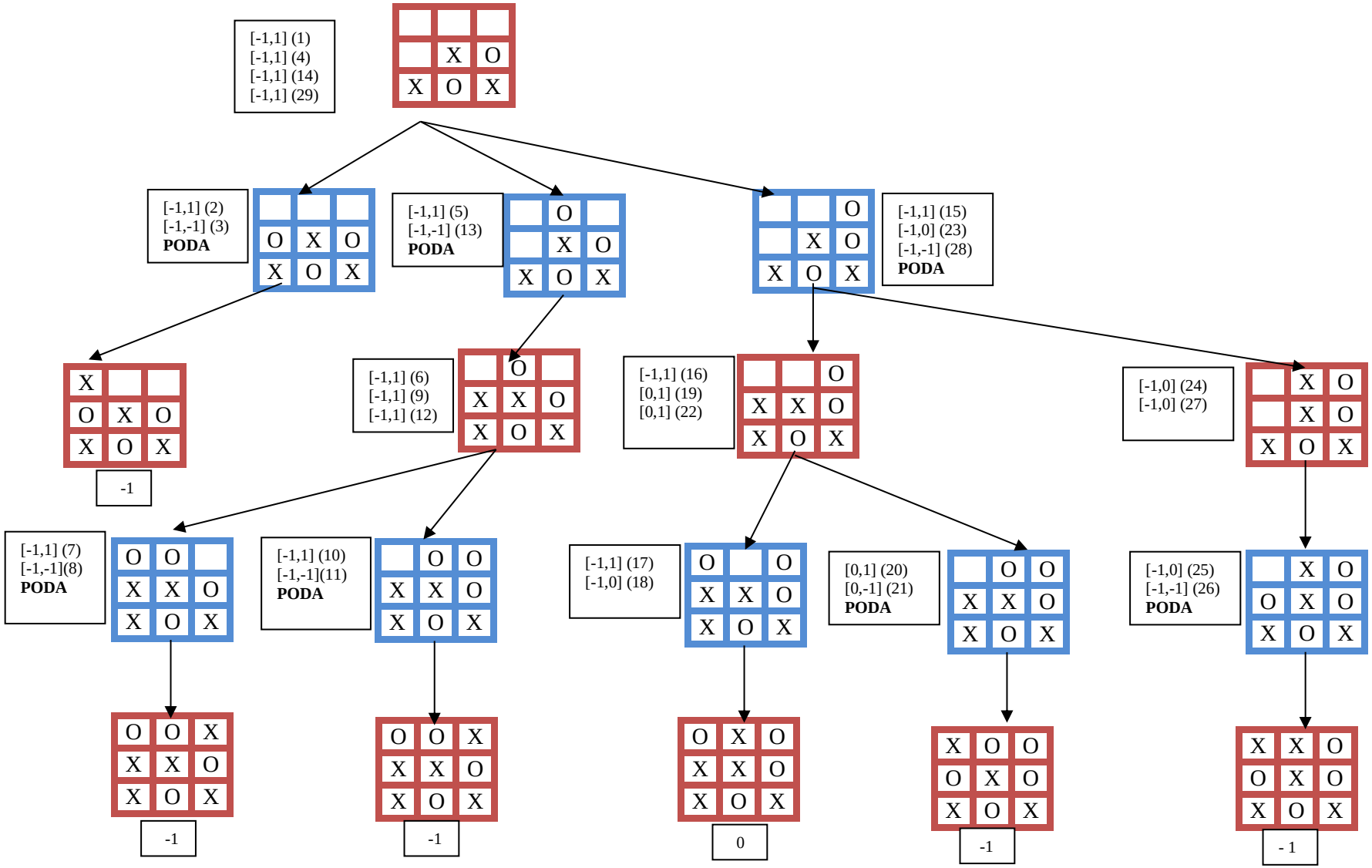
   Partimos de la posición intermedia del juego

	X	O
X	O	X

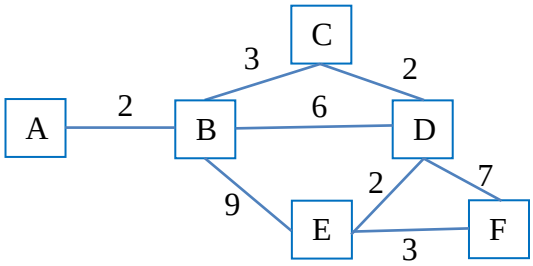
   (i) ¿Quién es MAX y quién es MIN en este estado de la partida?
   (ii) Realiza paso a paso el despliegue del árbol de juego para encontrar la estrategia óptima de MAX mediante el algoritmo minimax con poda alfa-beta.
   Para ello:
      a. En el árbol de juego, **dibuja únicamente los nodos explorados, indicando dónde se realiza poda**.
      b. Indica, para cada nodo, el **estado del tablero** y si el nodo es **MAX o MIN**.
      c. Indica en los **nodos terminales** el valor de la función de **utilidad**.
      d. Para los **nodos internos**, indica en cada paso **los valores del intervalo [α,β],** etiquetándolo con un entero que se incremente cada vez que este intervalo se propague hacia algún nodo hijo o se actualice con la información que proviene de algún nodo hijo.
   (ii) ¿Cuál es la estrategia minimax para el jugador que tiene el turno? ¿Cuál es el resultado del juego, de acuerdo con esta estrategia?
   (iii) ¿Se puede formalizar este juego como un juego de suma 0? ¿Por qué?
   (iv) ¿Garantiza minimax encontrar la estrategia óptima en este juego? ¿Por qué?

SOLUCIÓN:

(i) MAX es ‘O’ (tiene el turno) y MIN es ‘X’.
(ii) El primer movimiento explorado. El jugador ‘O’ pierde.
(iii) Se puede formalizar como un juego de suma cero ya que los valores de la función de utilidad de una posición terminal desde el punto de vista de cada uno de los dos jugadores es igual en magnitud, pero de signo opuesto.
(iv) Minimax garantiza encontrar la estrategia óptima en juegos de suma 0.
En este caso no existe una estrategia ganadora. El jugador O pierde con cualquiera de las jugadas que realice.
Es decir, la estrategia óptima (cualquiera de los tres movimientos) es una estrategia perdedora.


[-1,1] (1)
[-1,1] (4)
[-1,1] (14)
[-1,1] (29)
[-1,1] (2)
[-1,-1] (3)
PODA
[-1,1] (5)
[-1,-1] (13)
PODA
[-1,1] (15)
[-1,0] (23)
[-1,-1] (28)
PODA
-1
[-1,1] (6)
[-1,1] (9)
[-1,1] (12)
[-1,1] (16)
[0,1] (19)
[0,1] (22)
[-1,0] (24)
[-1,0] (27)
[-1,1] (7)
[-1,-1](8)
PODA
[-1,1] (10)
[-1,-1](11)
PODA
[-1,1] (17)
[-1,0] (18)
[0,1] (20)
[0,-1] (21)
PODA
[-1,0] (25)
[-1,-1] (26)
PODA
-1
-1
0
-1
- 1

4. Consideremos el siguiente grafo no dirigido, en el que deseamos encontrar el camino de menor coste entre A y F utilizando búsqueda informada



(i) Define una heurística que coincida en todos los nodos con la óptima, excepto en D, nodo en el que vale 0.
(ii) ¿Es esta heurística admisible? Demuéstralo.
(iii) ¿Es esta heurística monótona? Demuéstralo.
(iv) Utilizando dicha heurística, realiza búsqueda A* en este grafo para encontrar un camino entre A y F utilizando búsqueda en grafo (es decir, eliminando estados repetidos). Los sucesores de un nodo se generan en orden alfabético. En igualdad de condiciones, se exploran primero los nodos que han sido generados primero.
Indica en el árbol de búsqueda el valor de los valores g, h y f de cada nodo y el orden en el que se exploran los nodos expandidos, incluyendo los estados repetidos.
(v) ¿Se ha encontrado el camino óptimo (de menor coste)? ¿Garantiza esta estrategia con la heurística propuesta encontrar caminos óptimos para cualquier grafo de este tipo? ¿Por qué?

SOLUCIÓN:

(i) Define una heurística que coincida en todos los nodos con la óptima, excepto en D, nodo en el que vale 0.

Elegimos una heurística que coincide con la óptima en todos los nodos excepto en D, nodo en el que vale 0.

nodo	h(n)	h*(n)
A	12	12
B	10	10
C	7	7
D	0	5
E	3	3
F	0	0

(ii) ¿Es esta heurística admisible?
Dado que subestima la el coste óptimo, la heurística es admisible.

(iii) ¿Es esta heurística monótona?

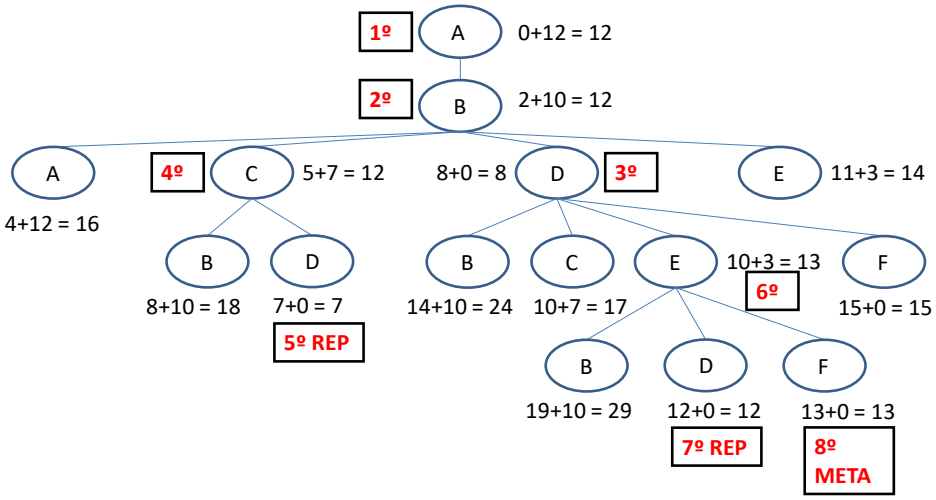
La condición de monotonicidad es que para todas las aristas (n→n'), en la que n' es sucesor de n, se cumple $h(n) \leq coste(n \rightarrow n') + h(n')$

La heurística propuesta no es monótona. Consideremos por ejemplo la arista C→D: h(C) = 7; coste(C→D) = 2; h(D) = 0

$h(C) > coste(C \rightarrow D) + h(D)$

Se vulnera la condición de monotonicidad.

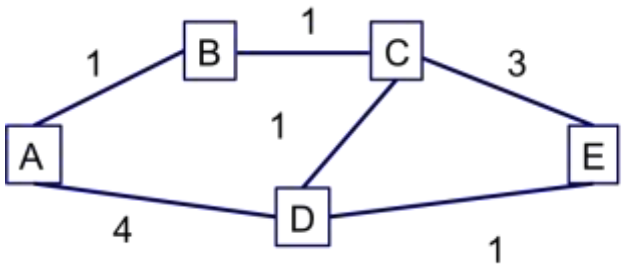
(iv) Utilizando dicha heurística, realiza búsqueda A* en este grafo para encontrar un camino entre A y F utilizando búsqueda en grafo (es decir, eliminando estados repetidos). Los sucesores de un nodo se generan en orden alfabético. En igualdad de condiciones, se exploran primero los nodos que han sido generados primero.
Indica en el árbol de búsqueda el valor de los valores g, h y f de cada nodo y el orden en el que se exploran los nodos expandidos, incluyendo los estados repetidos.



(v) ¿Se ha encontrado el camino óptimo (de menor coste)? ¿Garantiza esta estrategia con la heurística propuesta encontrar caminos óptimos para cualquier grafo de este tipo? ¿Por qué?

No se encuentra la solución óptima.
A* + eliminación de estados repetidos + heurística no monótona no garantiza encontrar la solución óptima.

5. El siguiente grafo no dirigido representa las transiciones entre los diferentes estados en un problema y los costes asociados a dichas transiciones. A través de búsqueda informada, se desea encontrar el camino óptimo al nodo final (E). Para ello se consideran dos posibles heurísticas, h1 y h2.



nodo	h1
A	3
B	2
C	3
D	0
E	0

nodo	h2
A	2
B	2
C	2
D	1
E	0

(i) ¿Es h1 admisible? ¿y h2? Demuéstralo.

(ii) ¿Es h1 monótona? ¿y h2? Demuéstralo.

(iii) ¿Qué heurística deberías escoger si realizas búsqueda A* **en grafo** (es decir, eliminando estados repetidos) para encontrar el camino óptimo entre A y E? ¿por qué?

(iv) Desarrolla el árbol de dicha búsqueda. Los sucesores de un nodo se generan en orden alfabético. En igualdad de condiciones, se exploran primero los nodos que han sido generados primero.
Indica en el árbol de búsqueda el valor de los valores g, h y f de cada nodo y el orden en el que se exploran los nodos expandidos, incluyendo los estados repetidos.

SOLUCIÓN:

(vi) ¿Es h1 admisible? ¿y h2? Demuéstralo.

nodo	h1
A	3
B	2
C	3
D	0
E	0

nodo	h2
A	2
B	2
C	2
D	1
E	0

nodo	h*
A	4
B	3
C	2
D	1
E	0

h1(C) > h*(C) luego h1 no admisible

h2(n) ≤ h*(n) luego h2 admisible

(vii) ¿Es h1 monótona? ¿y h2? Demuéstralo.

h1 no es admisible (pregunta anterior), luego no es monótona.

h2 es admisible (pregunta anterior), veamos si es monótona:

n	m	h2(n)	h2(m)	c(n->m)	h2(m)+c	OK?
A	B	2	2	1	3	Sí
A	D	2	1	4	5	Sí
B	A	2	2	1	3	Sí
B	C	2	2	1	3	Sí
C	B	2	2	1	3	Sí
C	D	2	1	1	2	Sí
C	E	2	0	3	3	Sí
D	A	1	2	4	6	Sí
D	C	1	2	1	3	Sí
D	E	1	0	1	1	Sí
E	C	0	2	3	5	Sí
E	D	0	1	1	2	Sí

Luego h2 sí es monótona.

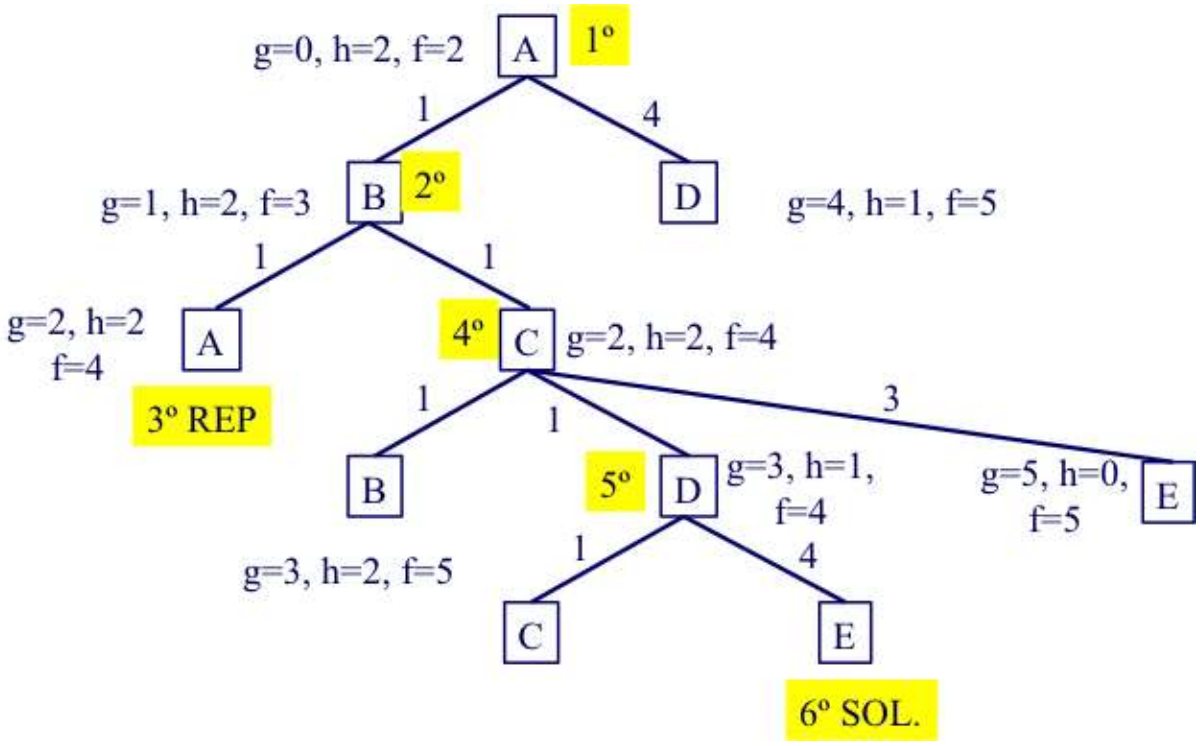
(viii) ¿Qué heurística deberías escoger si realizas búsqueda A* **en grafo** (es decir, eliminando estados repetidos) para encontrar el camino óptimo entre A y E? ¿por qué?

h1 no es admisible ni monótona luego no garantiza encontrar el camino óptimo con A* + búsqueda en grafo.

h2 sí es monótona, luego garantiza encontrar el camino óptimo con A* + búsqueda en grafo.

(ix) Desarrolla el árbol de dicha búsqueda. Los sucesores de un nodo se generan en orden alfabético. En igualdad de condiciones, se exploran primero los nodos que han sido generados primero.

Indica en el árbol de búsqueda el valor de los valores g, h y f de cada nodo y el orden en el que se exploran los nodos expandidos, incluyendo los estados repetidos.


g=0, h=2, f=2
A
1º
1
4
g=1, h=2, f=3
B
2º
D
g=4, h=1, f=5
1
1
g=2, h=2
f=4
A
3º REP
4º
C
g=2, h=2, f=4
1
1
3
B
5º
D
g=3, h=1,
f=4
g=5, h=0,
f=5
E
g=3, h=2, f=5
1
4
C
E
6º SOL.

6. Queremos patentar un móvil prácticamente irrompible desarrollado en nuestra empresa. Para ello necesitamos averiguar desde qué altura puede caerse el móvil sin que se rompa. Para averiguar esta altura utilizaremos dos móviles idénticos, que podemos dejar caer desde alguna planta de un edificio de 3 plantas. Cuando un móvil se rompe al dejarlo caer desde el piso n, también se rompería al dejarlo caer de un piso k > n. Si dejamos caer el móvil desde una altura dada y no se rompe, este móvil puede ser reutilizado en el experimento. Si se rompe, no podríamos volver a utilizarlo.

   Vamos a modelizar el problema como un juego entre dos adversarios que realizan acciones por turnos. Suponemos que el móvil 1 es el que se deja caer primero. El móvil 2 se utiliza únicamente si el móvil 1 se rompe en un ensayo. Las jugadas posibles para cada jugador son:

   Jugador A: Elige el piso desde el que dejar caer el móvil 1, con la restricción de que, en cada turno, el piso elegido tiene que ser más elevado que los elegidos en jugadas anteriores de la partida.
   Jugador B: Decide si se rompe o no el móvil en el turno correspondiente

   Hay dos tipos de posiciones terminales:
   - El móvil 1 se rompe en una caída desde el piso n: En este caso, adicionalmente a los ensayos con el móvil 1, hay que considerar el número de ensayos con el móvil 2 necesarios, en el peor de los casos, para determinar la altura desde la que se rompe un móvil.
   - El móvil 1 no se rompe en una caída desde el piso 3. En ese caso, la altura desde la que se rompe el móvil es superior a la del edificio. No es necesario realizar ensayos adicionales con el móvil 2.

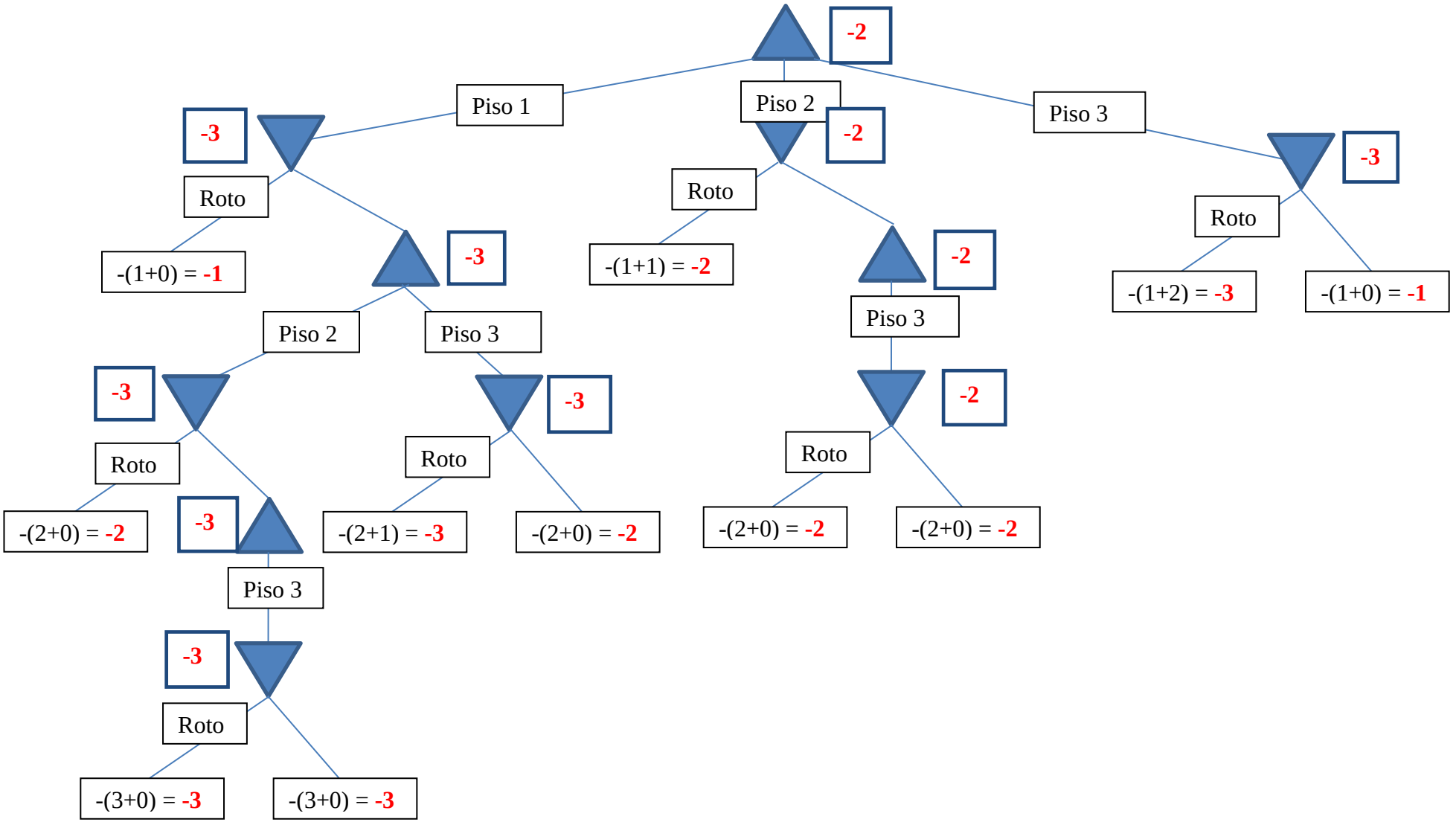
   ¿Cuál es la estrategia óptima para minimizar el número de veces que hay que dejar caer los móviles?
   En el peor caso, ¿cuál es el número de veces que hay que dejar caer los móviles?

   Utiliza minimax para decidir cuál es la estrategia de pruebas óptima y responder a las preguntas planteadas.

   SOLUCIÓN:

   Los nodos terminales se etiquetan con la suma de los ensayos realizados con el móvil 1 y ensayos pendientes con el móvil, cambiada de signo, ya que el objetivo del juego es minimizar el número total de ensayos.

   La estrategia óptima es dejar caer el móvil 1 desde el segundo piso. En caso de que se rompiera bastaría con dejar caer el móvil 2 desde el piso 1. En caso de que no se rompiera, habría que intentarlo desde el piso 3.


-2
Piso 1
Piso 2
Piso 3
-3
Roto
-(1+0) = -1
-3
Piso 2
Piso 3
-3
-3
Roto
-(2+0) = -2
-3
-(2+1) = -3
-(2+0) = -2
Piso 3
-3
Roto
-(3+0) = -3
-(3+0) = -3
-2
Roto
-(1+1) = -2
-2
Piso 3
-2
Roto
-(2+0) = -2
-(2+0) = -2
-3
Roto
-(1+2) = -3
-(1+0) = -1

7. Tenemos a nuestra disposición reglas de longitudes 1 m y 0.3 m. El precio de las reglas es proporcional a su longitud. Las reglas no tienen marcadas divisiones inferiores a su longitud. Sin embargo, necesitamos medir con ellas una longitud de 0.4 m con un coste mínimo.

Ejemplo: Utilizando una regla de 1 m y otra de 0.3 m, podemos medir una longitud de 0.7 m con un coste proporcional a 1.3.

0.3 m: XXX← 0.7 m →
1.0 m: XXXXXXXXXX

a. Formaliza los estados de búsqueda.

Utilizando la formalización propuesta:

b. Especifica las acciones para generar sucesores y su coste.
c. Especifica el estado inicial.
d. Especifica el test objetivo.
e. Define una heurística admisible no trivial para resolver el problema y demuestra su admisibilidad.
f. ¿Garantiza A* con esta heurística con eliminación de estados repetidos encontrar la solución óptima?
g. Detalla el árbol generado por A* con eliminación de estados repetidos. Indica para cada nodo los valores g + h = f y el orden en el que el **intento de exploración** se realiza (es decir, los nodos repetidos y la meta también reciben numeración). **En caso de que haya empates, se elegirá primero en la exploración el nodo que haya sido generado antes.**
h. ¿Encuentra A* la solución óptima en este ejemplo? Justifica tu respuesta.

SOLUCIÓN:

a. Formaliza los estados de búsqueda.

Cada estado correspondería a 2 carriles. En el primero de los carriles, solo se pueden colocar reglas de tamaño 0.3 y en el otro únicamente reglas de tamaño 1

2*1.0	$l_1 = 2.0$
1*0.3	$l_2 = 0.3$

Por lo tanto, los estados se pueden representar mediante pares de reales (l_1, l_2).

Con esta construcción se consigue medir la distancia $|l_1-l_2|$.

b. Especifica las acciones para generar sucesores y su coste.

A partir de un estado (l_1, l_2), se pueden realizar las siguientes acciones:

Añadir regla 0.3 a carril 1: $(l_1, l_2) \rightarrow (l_1+0.3, l_2), 0.3$
Añadir regla 1.0 a carril 2: $(l_1, l_2) \rightarrow (l_1, l_2+1.0), 1.0$

c. Especifica el estado inicial.

Estado: $(0.0, 0.0)$.

d. Especifica el test objetivo.

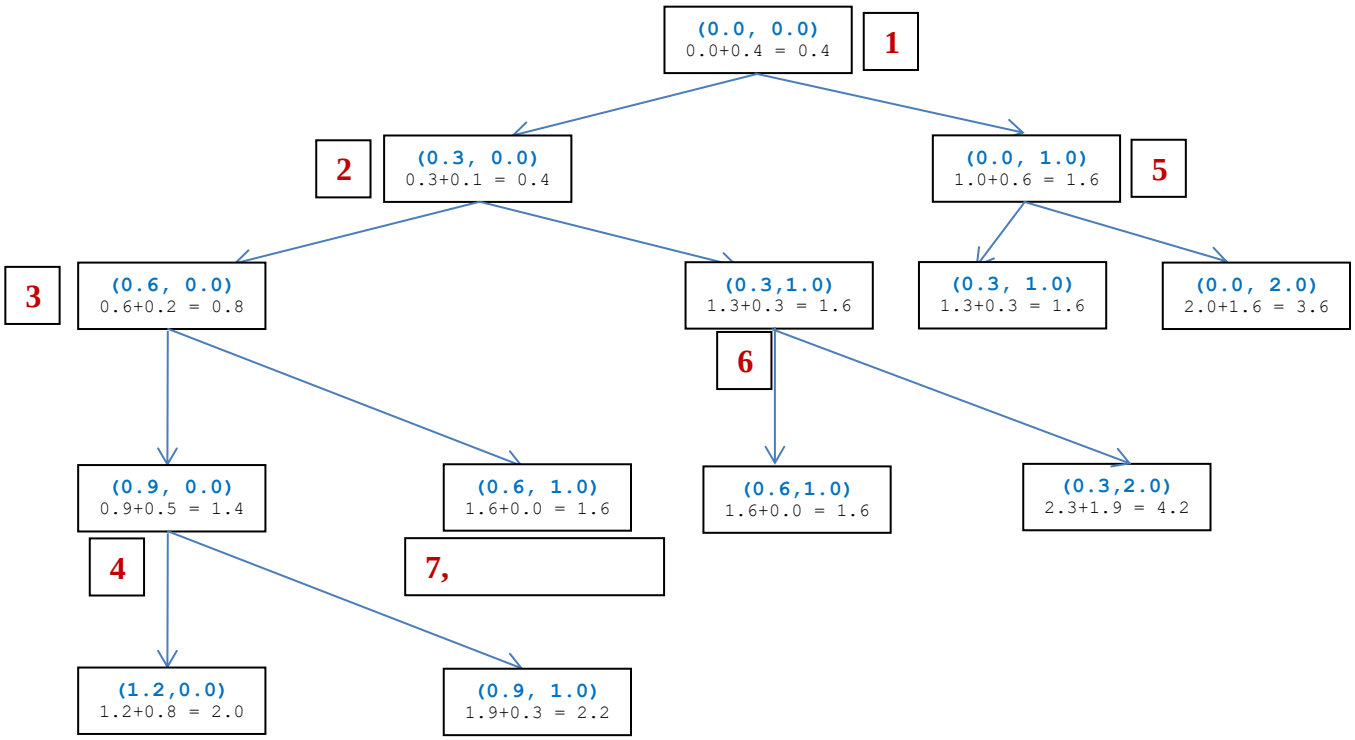
Estado: (l_1, l_2).
Test objetivo: $|l_1-l_2| == 0.4$

e. Define una heurística admisible no trivial para resolver el problema y demuestra su admisibilidad.

Estado: (l_1, l_2).
Heurística: $h(l_1, l_2) = abs(|l_1-l_2| - 0.4)$

La heurística es admisible y monótona, ya que es la solución de un problema relajado, en el que las reglas a colocar pueden tener longitud arbitraria.

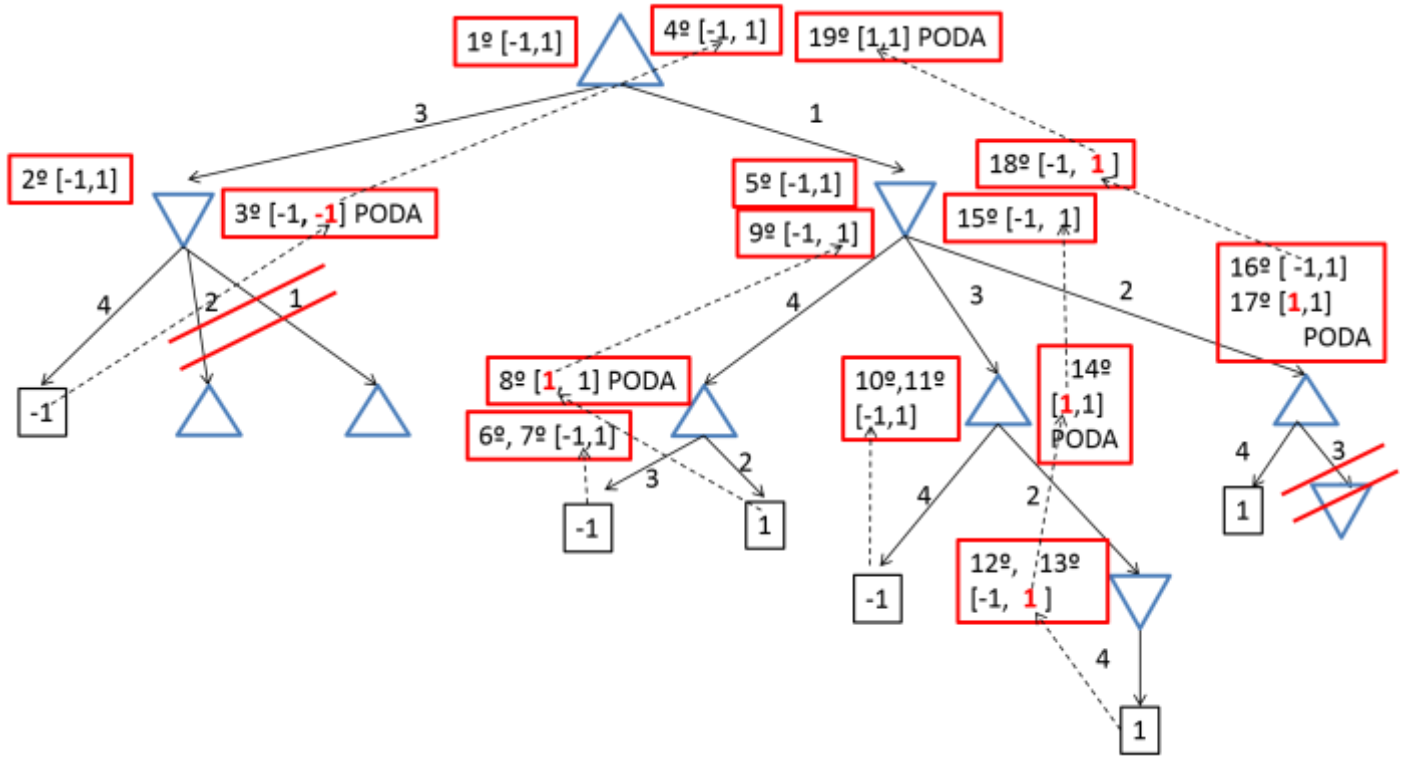
f. Detalla el árbol generado por A* con eliminación de estados repetidos. Indica para cada nodo los valores g + h = f y el orden en el que el **intento de exploración** se realiza (es decir, los nodos repetidos y la meta también reciben numeración). **En caso de que haya empates, se elegirá primero en la exploración el nodo que haya sido generado antes.**



g. ¿Garantiza A* con esta heurística con eliminación de estados repetidos encontrar la solución óptima, en caso de que exista?
Sí, porque la heurística es monótona:
A* + eliminación de estados repetidos + h monótona = óptima.

h. ¿Qué ocurre en caso de que la solución no existe?
En la formalización realizada, en caso de que haya alguna relación de congruencia entre las reglas y si no hay solución el algoritmo no terminaría. Sólo terminaría si se eliminan estados equivalentes. Dos estados (x_1, x_2), (y_1, y_2) son equivalentes si $|x_1-x_2| = |y_1-y_2|$.

8. Juanma y Sergio son dos compañeros de trabajo que conducen una furgoneta de reparto y van alternando su uso (el primer recado lo hace Juanma, el segundo lo hace Sergio, el siguiente Juanma de nuevo, etc.).
   Hoy deben realizar un total de cuatro recados (dos cada uno): un recado a Sol (coste en combustible de 1 litro), otro a Lavapiés (coste de 2 litros), otro a Ventas (3 litros) y otro a Moratalaz (4 litros). Cuando le toca el turno a un repartidor puede elegir el recado que quiera dentro de los que quedan pendientes y en los que esté el cliente disponible.
   A primera hora de la mañana, que es cuando va a realizar su primer recado del día Juanma, ni el cliente de Lavapiés ni el de Moratalaz están disponibles, por lo que en su primer recado Juanma sólo puede elegir ir a Sol o a Ventas.
   Si un repartidor va a coger la furgoneta y no tiene combustible, o se queda sin combustible en medio de un recado, le tocará llenar el depósito.
   Si los dos repartidores quieren evitar a toda costa tener que llenar el depósito, y este está cargado al comienzo del día con 7 litros, ¿cuál es la estrategia óptima para Juanma?
   Utiliza minimax con poda alfa-beta y un orden de exploración en el que **los recados con mayor coste se exploran primero**.
   En el árbol de juego, **dibuja únicamente los nodos explorados, indicando dónde se realiza poda**. Para cada nodo indica si es **MAX o MIN**. Indica en los **nodos terminales** el valor de la función de **utilidad**, y para los **nodos internos**, indica en cada paso **los valores del intervalo [α,β],** etiquetándolo con un entero que se incremente cada vez que este intervalo se propague hacia algún nodo hijo o se actualice con la información que proviene de algún nodo hijo.



La estrategia óptima para Juanma es elegir hacer el recado de Sol, ya que el hijo izquierdo del nodo raíz (al que se llegaría tras realizar la acción "hacer recado de Ventas") devuelve -1, y el hijo derecho de la raíz (al que se llegaría tras realizar la acción "hacer recado de Sol") devuelve 1.

Nota: en esta solución se ha codificado la utilidad desde el punto de vista de MAX (Juanma) como -1 (tiene que llenar el depósito) y 1 (no tiene que llenarlo). El resultado hubiera sido el mismo si se hubiese elegido una codificación $-\infty$, $+\infty$. Por otra parte, como se puede observar siempre se marcan las situaciones de poda (aunque no haya más acciones que explorar en ese nodo).