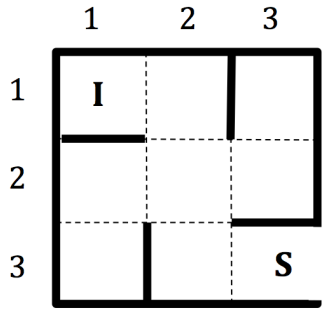
Inteligencia Artificial
Hoja 1: Búsqueda y juegos

Publicación:	2016/02/02
Límite realización:	2016/02/15
Resolución en clase:	2016/02/16

1. Considera el problema de buscar la salida en un laberinto:



La casilla de comienzo, marcada como **I**, es la (1,1). El objetivo es llegar a la casilla de salida **S**, situada en la posición (3,3), en el menor número de movimientos. Desde cada casilla sólo se podrá acceder a las casillas adyacentes que no tengan un muro entre medias (línea gruesa). Por ejemplo, desde la casilla (2,1) sólo se puede acceder a la (2,2), y a la (1,1). El orden de exploración de las acciones será derecha, abajo, izquierda, arriba. **En caso de empate se explora primero el nodo que ha sido generado antes.**

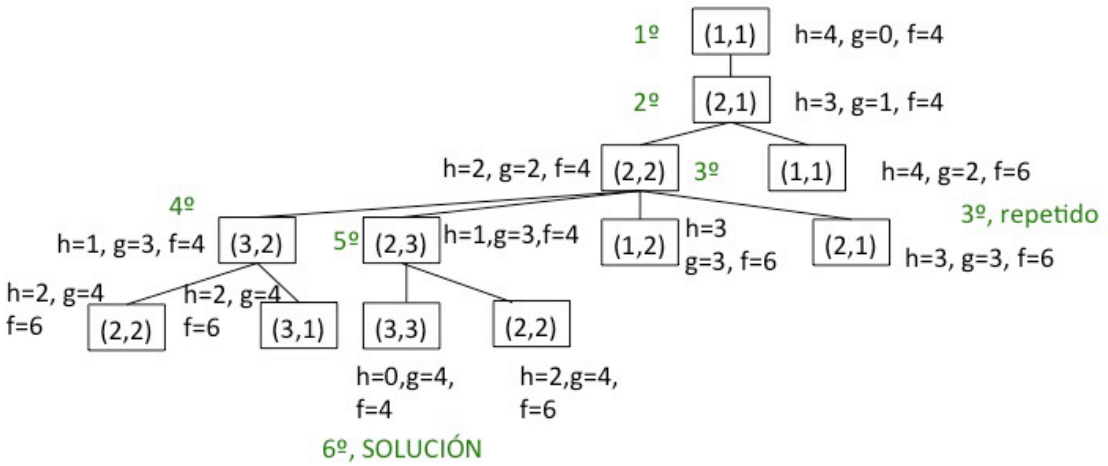
a) Define una heurística admisible para este problema
b) ¿Es monótona?
c) Usa dicha heurística en la búsqueda de una solución a este laberinto usando A* y búsqueda-en-grafo.
d) ¿Garantiza dicha estrategia encontrar la solución óptima para cualquier laberinto de este tipo? ¿Y para este laberinto en particular?

Solución:

a) |x-Sx|+|y-Sy|
donde Sx, Sy son las coordenadas de la casilla de salida **S**.

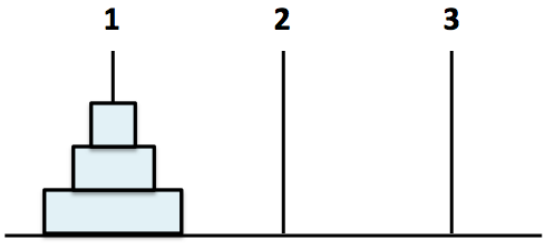
b) Monótona: el coste de cualquier movimiento es 1. Por otra parte el h como mucho baja 1 al moverse a cualquier vecino.
Por tanto $h(n) \leq c(n \rightarrow n') + h(n')$ y la heurística es monótona.

c)

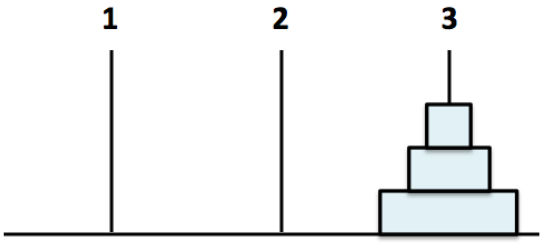


d) Sí, ya que la heurística es monótona y se usa A* con búsqueda-en-grafo.

2. Considera el problema de las Torres de Hanoi. En él se parte del estado en el que N discos están apilados en orden de tamaño en la varilla 1. Por ejemplo, para N=3:



El objetivo es, alcanzar el estado final en el que los N discos están apilados en orden en la varilla 3. Por ejemplo, para N=3:



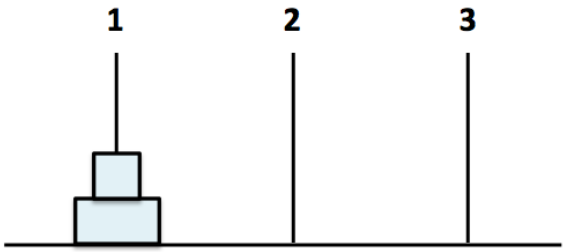
Las acciones permitidas son:

- Mover disco superior (si lo hubiera) de varilla 1 a varilla 2
- Mover disco superior (si lo hubiera) de varilla 1 a varilla 3
- Mover disco superior (si lo hubiera) de varilla 2 a varilla 1
- Mover disco superior (si lo hubiera) de varilla 2 a varilla 3
- Mover disco superior (si lo hubiera) de varilla 3 a varilla 1
- Mover disco superior (si lo hubiera) de varilla 3 a varilla 2

Se permite la mover un disco de varilla sólo si la varilla destino se encuentra vacía o el disco superior de dicha varilla es mayor que el que queremos mover. Por tanto el número de acciones posibles desde cada estado puede variar. El orden de exploración de las acciones será el orden en el que están enumeradas en la lista de arriba. **En caso de empate se explora primero el nodo que ha sido generado antes.**

a) Define una heurística admisible para el problema general de N discos y tres varillas, en la que el objetivo es llevar todos los discos a la varilla 3.

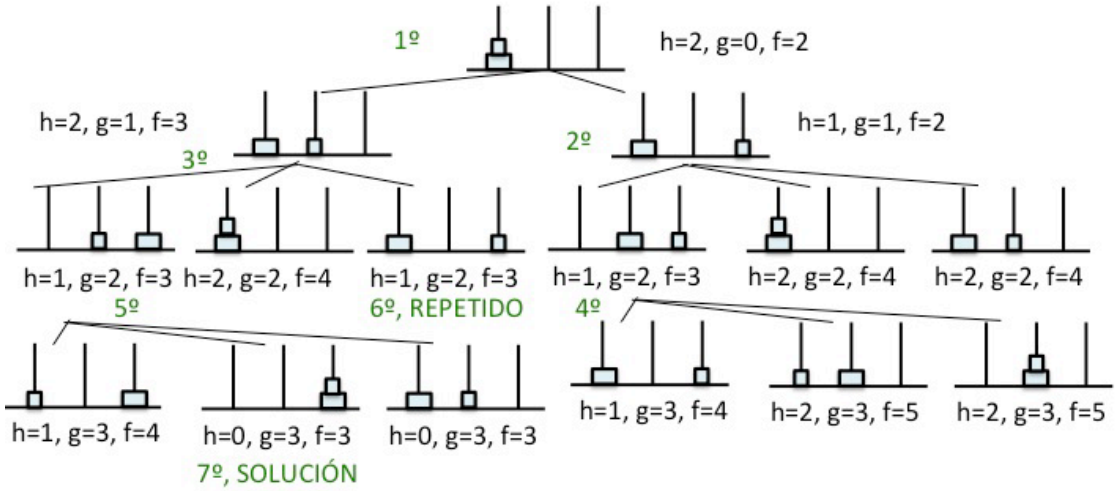
b) Encuentra la solución óptima (cada movimiento cuesta 1) con A*, búsqueda-en-grafo y dicha heurística para N=2 y el siguiente estado inicial:



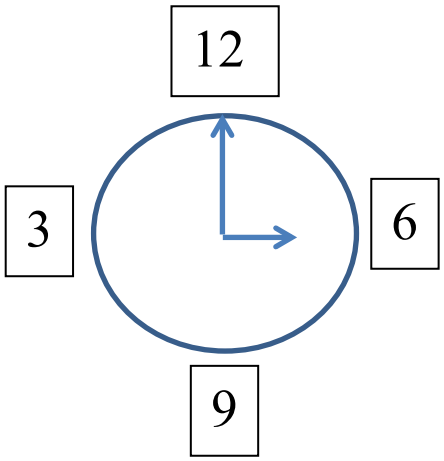
Solución:

a) Heurística: número de discos en 1 + número de discos en 2

b)


1º
h=2, g=0, f=2
h=2, g=1, f=3
h=1, g=1, f=2
2º
3º
h=1, g=2, f=3
h=2, g=2, f=4
h=1, g=2, f=3
6º, REPETIDO
h=1, g=2, f=3
h=2, g=2, f=4
h=2, g=2, f=4
4º
5º
h=1, g=3, f=4
h=2, g=3, f=5
h=2, g=3, f=5
h=1, g=3, f=4
h=0, g=3, f=3
h=0, g=3, f=3
7º, SOLUCIÓN

3. Hemos heredado un reloj de nuestro antepasado, que había trabajado en Londres con Charles Babbage y Ada Lovelace en el problema de cómo programar la máquina analítica diseñada por el primero.



Las etiquetas en esta esfera de reloj están descolocadas. Para colocarlas de nuevo, es necesario intercambiarlas dos a dos hasta formar el orden correcto. El coste de intercambiar dos etiquetas es la suma de los valores numéricos de dichas etiquetas.

a. Formaliza los estados de búsqueda.

x_0
x_3 | x_1
x_2

con $x_i \in \{3,6,9,12\}$ y $x_i \neq x_j;\ i,j = 1,2,3,4$

Utilizando la formalización propuesta:

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

Dados $i,j \in \{1,2,3,4\}$ con $i \neq j$, actualizar los valores de las variables únicamente si x_i, x_j tienen ambas valores incorrectos

$(x_i, x_j) \leftarrow (x_j, x_i)$ coste: $x_i + x_j$

c. Especifica el estado inicial.

12
3 | 6
9

d. Especifica el test objetivo.

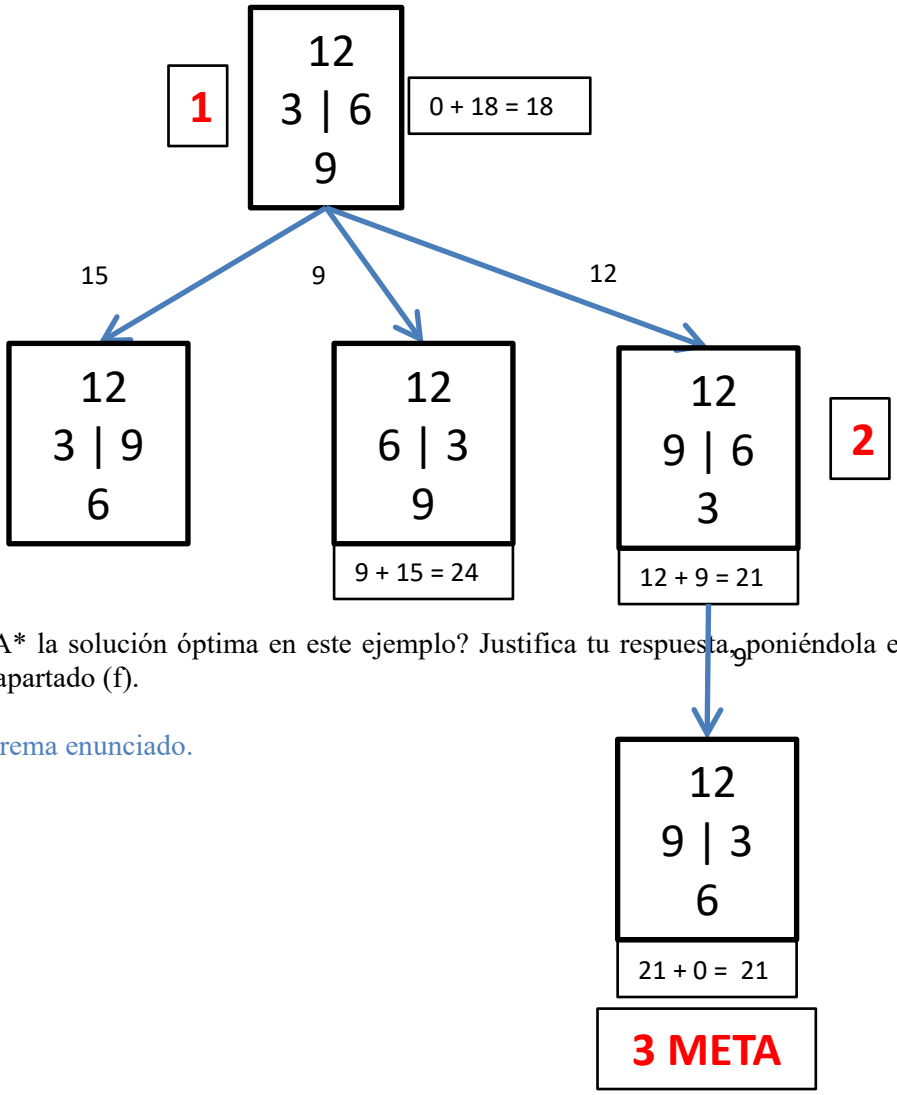
x_0
x_3 | x_1
x_2
$==$
12
9 | 3
6

e. Define una heurística admisible no trivial para resolver el problema y demuestra su admisibilidad.
Suma de los valores de las etiquetas mal colocadas.
Es admisible porque es el coste de un problema que se obtiene eliminando la restricción de que sólo se pueden intercambiar dos etiquetas en cada acción.

f. ¿Garantiza A* con esta heurística y sin eliminación de estados repetidos encontrar la solución óptima? Explica la razón.

Sí, A* sin eliminación de estados repetidos + heurística admisible es óptima.

g. Detalla el árbol generado por A* sin 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 poniéndola en relación con tu respuesta al apartado (f).

Sí, por el teorema enunciado.

4. Tenemos dos cántaros vacíos. El primero tiene una capacidad de 4 litros. El segundo, de 3 litros. También disponemos de un grifo para llenar los cántaros de agua. Los cántaros se pueden llenar completamente, o vaciar completamente. También se puede verter el contenido de un cántaro en el otro hasta que o bien el primero se vacíe o que el segundo se llene (lo que ocurra antes). El coste de cada operación es el número de litros que se transfieren. Inicialmente ambos cántaros están vacíos. El objetivo es llegar a la siguiente situación: que el cántaro de 4 litros de capacidad contenga 1 litro y que el cántaro de 3 litros quede vacío.
   La heurística a utilizar es la suma de los valores absolutos de las diferencias entre los volúmenes de agua en cada uno de los cántaros y los correspondientes objetivos.

   Responde las siguientes cuestiones:
   i. Formaliza los estados de búsqueda.

   Utilizando la formalización propuesta:
   j. Especifica las acciones para generar sucesores: Condiciones que debe cumplir el estado inicial, estado que resulta de aplicar la acción y coste.
   k. Especifica estado inicial.
   l. Especifica el test objetivo.
   m. Define la función que permite calcular el valor de la heurística para un estado dado ¿Es la heurística propuesta admisible? ¿y monótona?
   n. ¿Garantiza A* con esta heurística y eliminación de estados repetidos encontrar la solución óptima? Explica la razón.
   o. 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.
   p. ¿Encuentra A* la solución óptima en este ejemplo? Justifica tu respuesta, poniéndola en relación con tu respuesta al apartado (g).

SOLUCIÓN:

Responde las siguientes cuestiones:

a. Formaliza los estados de búsqueda.

$[x_1, x_2] \qquad 0 \le x_1 \le 4; \qquad 0 \le x_2 \le 3;$

Utilizando la formalización propuesta:

b. Especifica las acciones para generan sucesores: Condiciones que debe cumplir el estado inicial, estado que resulta de aplicar la acción y coste.

$[x_1, x_2]$	$[x_1 < 4]$	$llenar_1$	→ $[4, x_2]$		coste: $4- x_1$
$[x_1, x_2]$	$[x_2 < 3]$	$llenar_2$	→ $[x_1, 3]$		coste: $3- x_2$
$[x_1, x_2]$	$[x_1 > 0]$	$vaciar_1$	→ $[0, x_2]$		coste: x_1
$[x_1, x_2]$	$[x_2 > 0]$	$vaciar_2$	→ $[x_1, 0]$		coste: x_2
$[x_1, x_2]$	$[x_1 > 0]$	$verter_{1\to2}$	→ $[0, x_1+x_2]$	$x_1 + x_2 \le 3$	coste: x_1
			→ $[x_1+ x_2 -3,3]$	$x_1 + x_2 \ge 3$	coste: $3-x_2$
$[x_1, x_2]$	$[x_2 > 0]$	$verter_{2\to1}$	→ $[x_1+x_2, 0]$	$x_1 + x_2 \le 4$	coste: x_2
			→ $[4, x_1+ x_2 -4]$	$x_1 + x_2 \ge 4$	coste: $4-x_1$

c. Especifica el estado inicial.

$[0, 0]$

d. Especifica el test objetivo.

$[x_1, x_2] == [1,0]$

e. Define la función que permite calcular el valor de la heurística para un estado dado ¿Es la heurística propuesta admisible? ¿y monótona?

$h([x_1 , x_2]) = | x_1 - 1 | + x_2$

No es admisible.
Contraejemplo: El coste de ir del estado [0,1] (heurística = 2) al [1,0] (meta. heurística =0) es 1. Por lo tanto la heurística sobreestima el coste real de la solución óptima y no es admisible.
Dado que no es admisible tampoco es monótona.

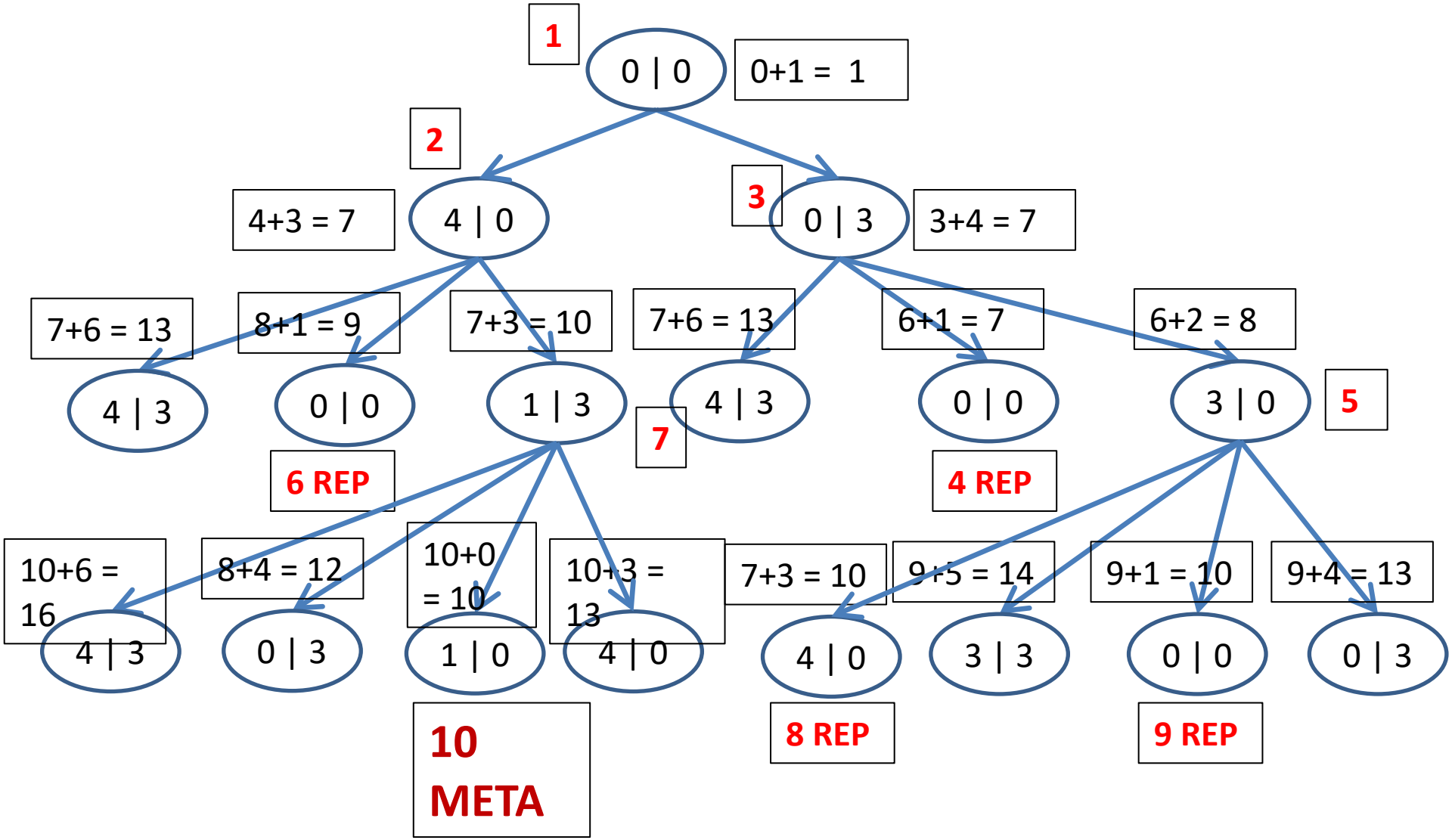
f. ¿Garantiza A* con esta heurística y eliminación de estados repetidos encontrar la solución óptima? Explica la razón.

h*(n) no es monótona (ni admisible), por lo tanto
A* + eliminación de estados repetidos (búsqueda en grafo) + heurística no monótona
= algoritmo de búsqueda no garantiza encontrar la solución óptima

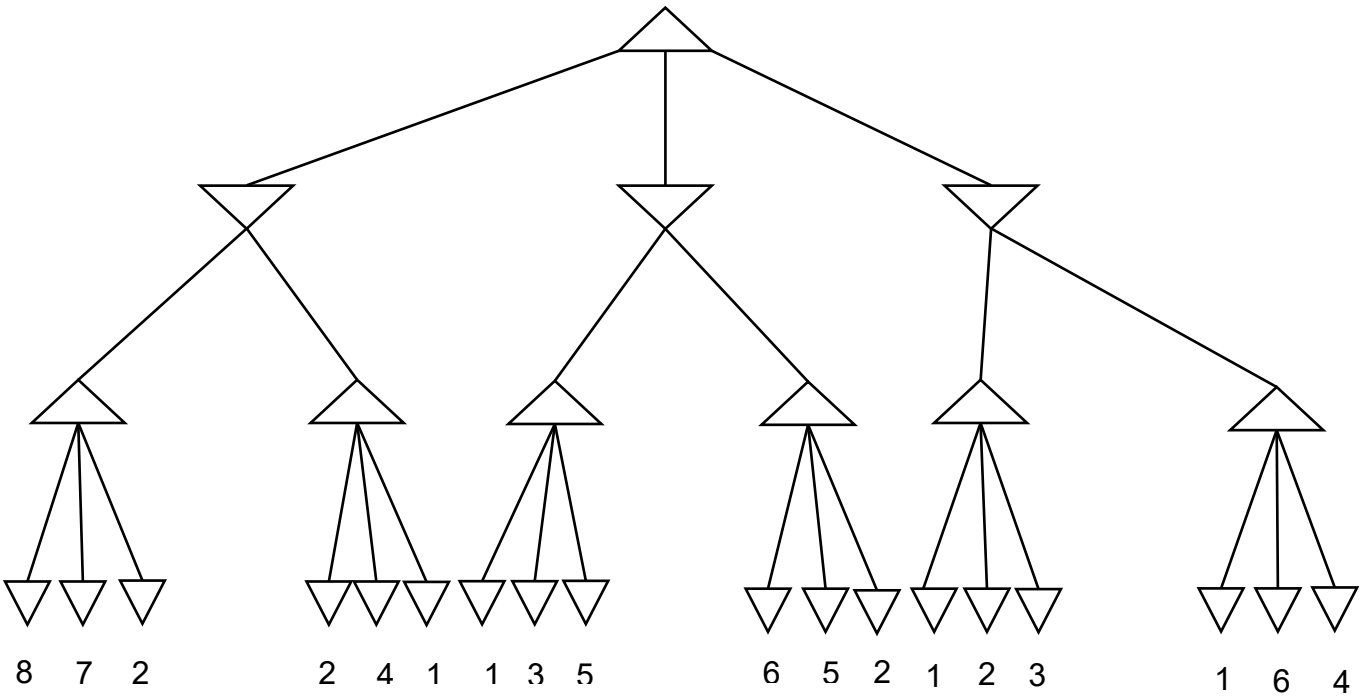
h. ¿Encuentra A* la solución óptima en este ejemplo? Justifica tu respuesta, poniéndola en relación con tu respuesta al apartado (g).

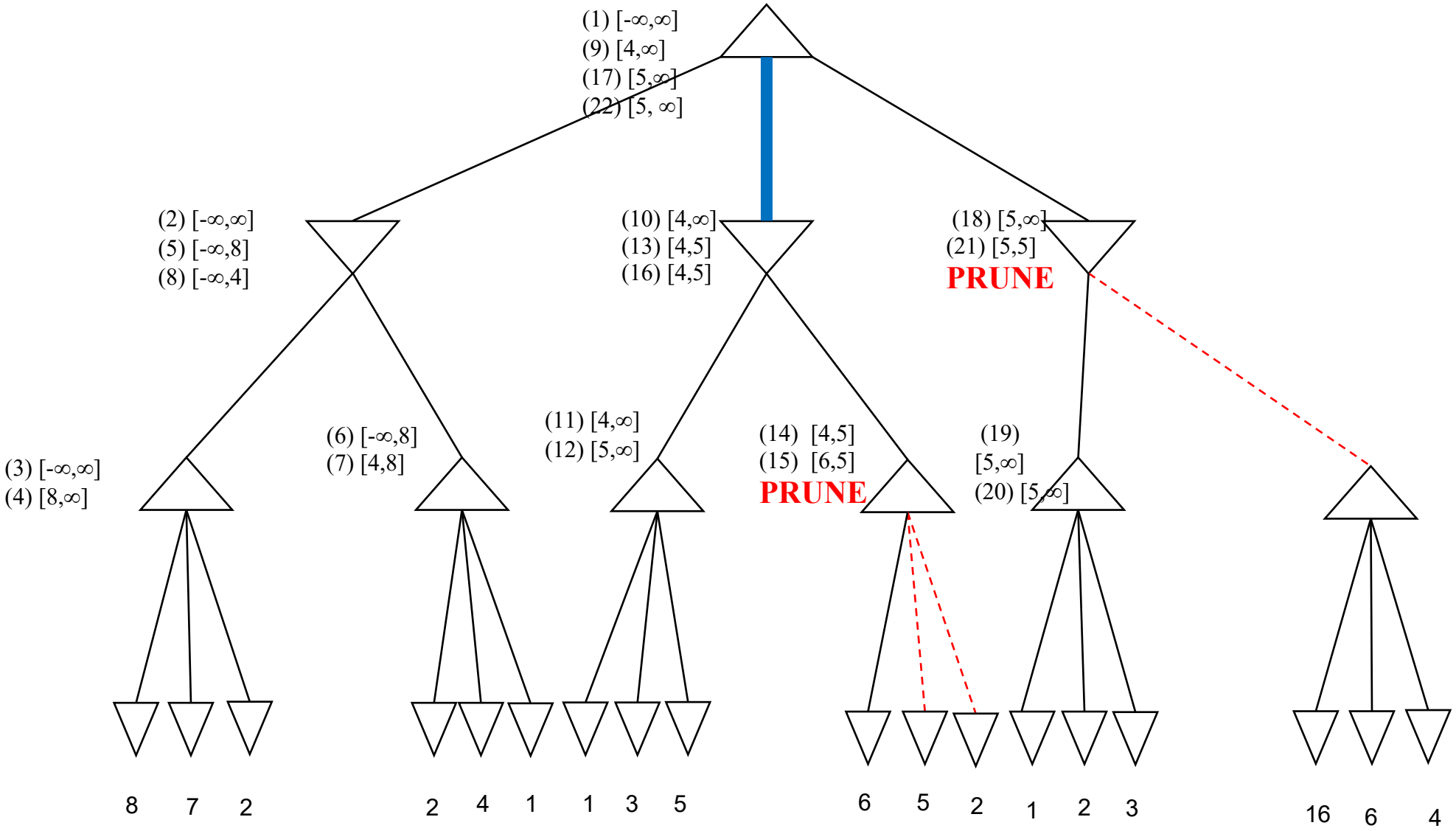
Sí, a pesar de que la heurística no sea admisible

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.



5. **[1 punto]** Utiliza el algoritmo minimax con poda alfa-beta para determinar el valor minimax en la raíz del árbol y la jugada óptima suponiendo que el oponente es óptimo. Para ello, indica el orden de recorrido en el árbol, numerando los nodos visitados junto con las actualizaciones de los valores [α,β] en cada paso




(1) [-∞,∞]
(9) [4,∞]
(17) [5,∞]
(22) [5, ∞]
(2) [-∞,∞]
(5) [-∞,8]
(8) [-∞,4]
(10) [4,∞]
(13) [4,5]
(16) [4,5]
(18) [5,∞]
(21) [5,5]
PRUNE
(3) [-∞,∞]
(4) [8,∞]
(6) [-∞,8]
(7) [4,8]
(11) [4,∞]
(12) [5,∞]
(14) [4,5]
(15) [6,5]
PRUNE
(19)
[5,∞]
(20) [5,∞]
8 7 2
2 4 1
1 3 5
6 5 2
1 2 3
16 6 4

6. 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 mimimax encontrar la estrategia óptima en este juego? ¿Por qué?

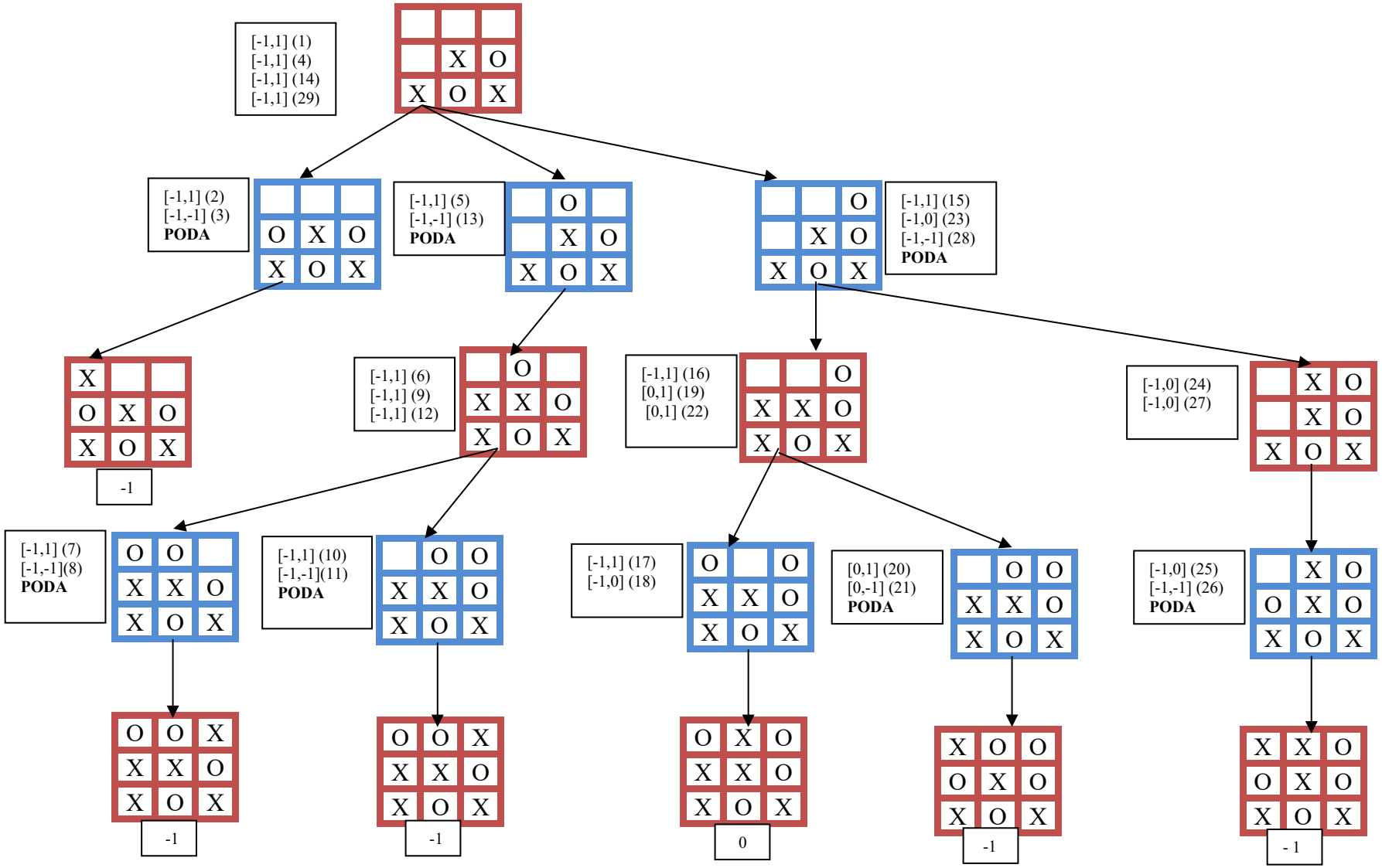
SOLUCIÓN:

(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

7. Tenemos dos jugadores que se alternan poniendo fichas en un tablero de dimensiones 3x3. Uno de los jugadores coloca fichas etiquetadas con ‘1’. El otro jugador coloca fichas etiquetadas con ‘0’. Inicialmente el tablero está vacío. El primer movimiento en una partida corresponde a ‘1’. Los jugadores alternan sus movimientos. En cada turno, el jugador que realiza el movimiento coloca una de sus fichas en una de las celdas libres del tablero. Los movimientos se exploran comenzando por la columna libre que se encuentre más a la izquierda y por la fila inferior.

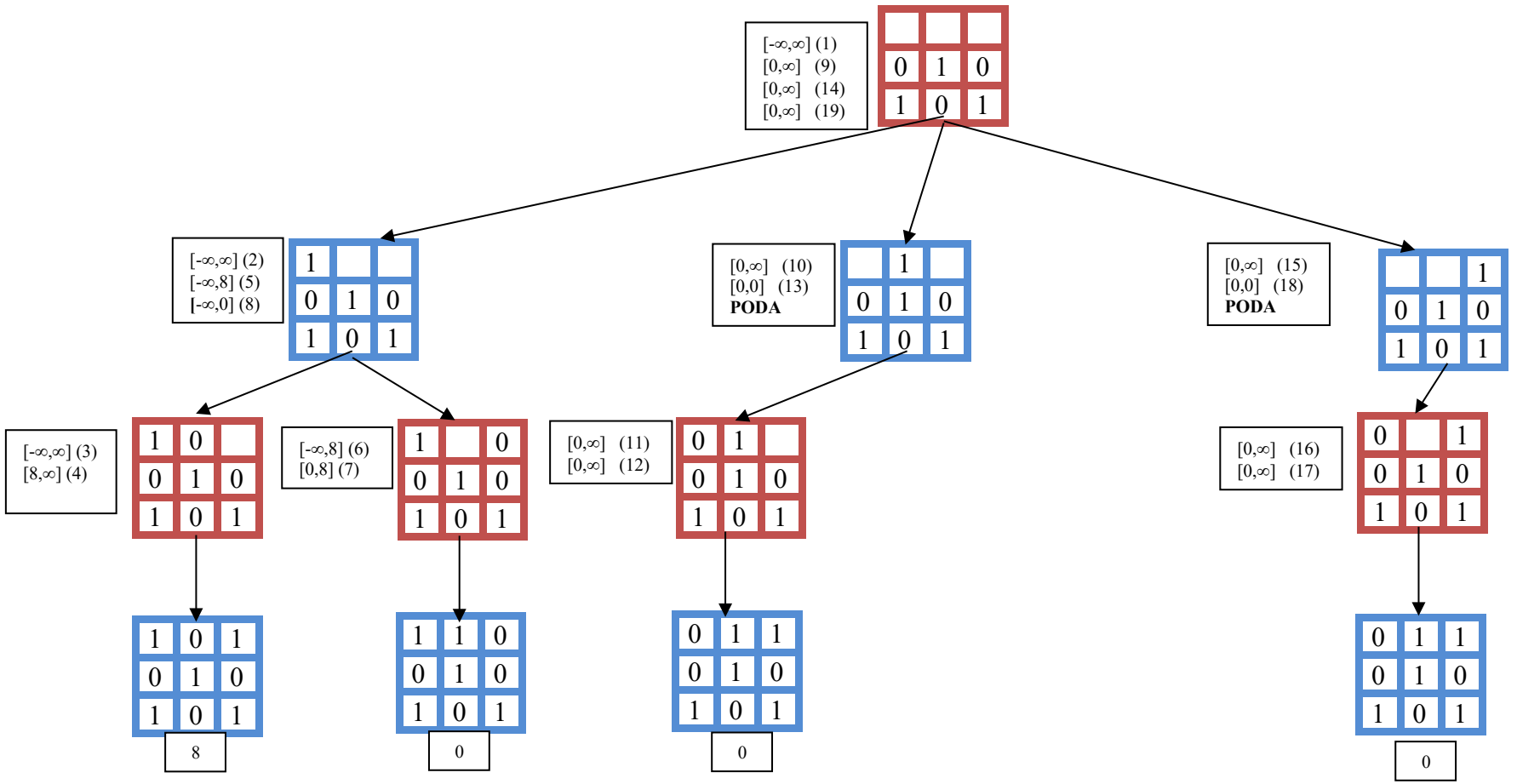
   La partida termina cuando el tablero está completo. Los puntos para el jugador ‘1’ son el número de ternas de bits capicúas (es decir: 000, 010, 101, 111) que se pueden leer en horizontal, en vertical o en diagonal. Los puntos para el jugador ‘0’ son el número de ternas de bits no capicúas (es decir: 001, 100, 110, 011).

   Partimos de la posición intermedia del juego

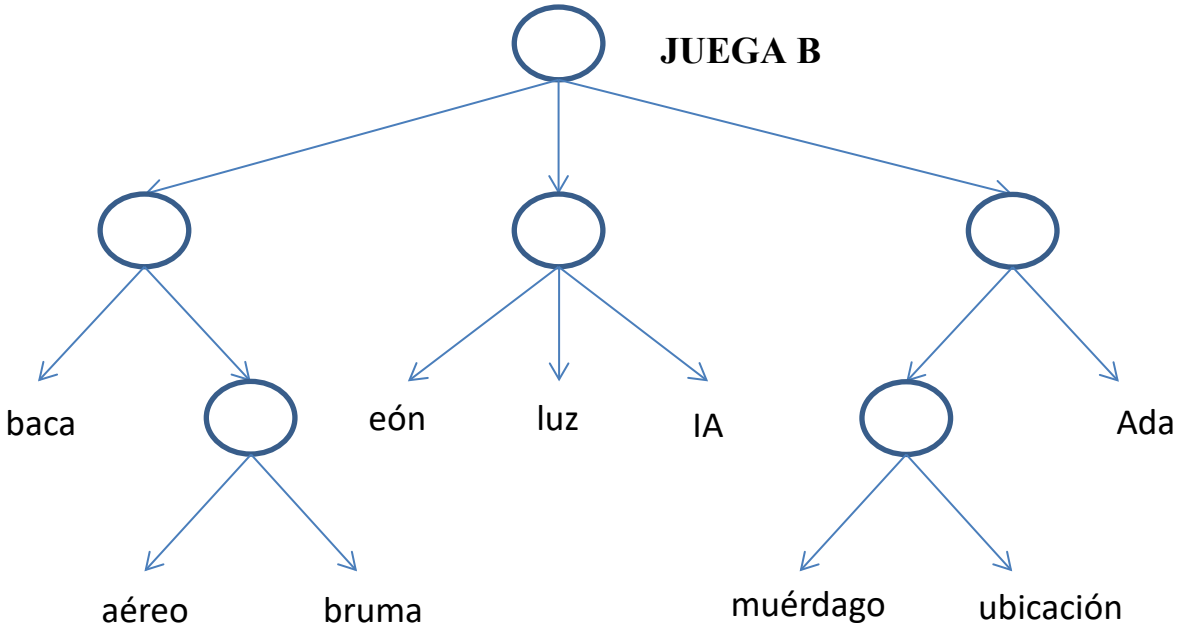
0	1	0
1	0	1

   (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.
   (iii) ¿Cuál es la estrategia minimax para el jugador que tiene el turno? ¿Cuál es el resultado del juego, de acuerdo con esta estrategia?
   (iv) ¿Se puede formalizar este juego como un juego de suma 0? ¿Por qué?
   (v) ¿Garantiza mimimax encontrar la estrategia óptima en este juego? ¿Por qué?

(v) MAX es ‘1’ (tiene el turno) y MIN es ‘0’.
(vi) El primer movimiento explorado. El resultado es empate.
(vii) 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.
(viii) Minimax garantiza encontrar la estrategia óptima en juegos de suma 0.
En este caso la estrategia óptima para el jugador ‘1’ es una estrategia en la que hay empate.


[-∞,∞] (1)
[0,∞] (9)
[0,∞] (14)
[0,∞] (19)
[-∞,∞] (2)
[-∞,8] (5)
[-∞,0] (8)
[0,∞] (10)
[0,0] (13)
PODA
[0,∞] (15)
[0,0] (18)
PODA
[-∞,∞] (3)
[8,∞] (4)
[-∞,8] (6)
[0,8] (7)
[0,∞] (11)
[0,∞] (12)
[0,∞] (16)
[0,∞] (17)
8
0
0
0

8. 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 ábol 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