Inteligencia Artificial
Hoja 1: Búsqueda y juegos

Publicación: 2018/02/13
Límite realización: 2018/02/20
Resolución en clase: 2018/02/20

---

1. Una empresa de distribución cuya central está en Madrid necesita mandar un camión una vez al día para repartir mercancías perecederas a Burgos, Soria y Teruel. La empresa nos pide que realicemos un software que determine la ruta óptima, de tal forma que el camión, partiendo de Madrid, visite las tres ciudades para repartir la mercancía en ellas, y tras ello regrese de vuelta a Madrid, realizando una suma total de horas de conducción en que sea **la menor posible**.

   La siguiente tabla refleja las horas de carretera que hay entre cada par de ciudades:

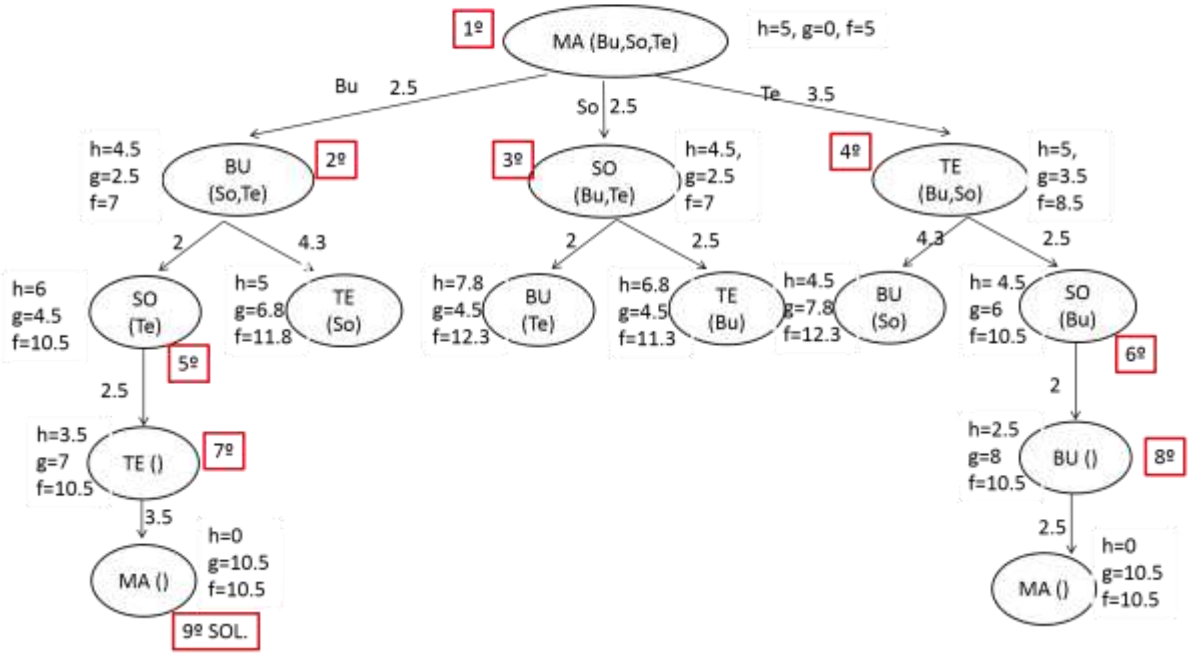
	Madrid	Burgos	Soria	Teruel
Madrid	0	2.5	2.5	3.5
Burgos	2.5	0	2.0	4.3
Soria	2.5	2.0	0	2.5
Teruel	3.5	4.3	2.5	0

   a) Formaliza los estados y acciones en este problema. ¿Cuál es S0? ¿y Sf?
   Ayuda: considera en cada estado la lista de ciudades en las que queda repartir la mercancía.
   b) Define una heurística admisible y no trivial para este problema.
   c) ¿Qué tipo de búsqueda usarías con A* para garantizar que con la heurística que has elegido la ruta obtenida sea la óptima? ¿A* con búsqueda en árbol o A* con búsqueda en grafo? ¿por qué?
   d) Encuentra la ruta óptima usando la búsqueda que has elegido en c).

Solución:

a) Cada estado lo formalizamos con el par
[ciudad en la que está el camión, lista de ciudades que quedan por visitar]
Así,
S0 = [Ma, (Ba, Gu, Te, To, Za)]
Sf = [Ma, ( )]
Las acciones posibles en cada estado son ir a una ciudad X y repartir la mercancía allí, donde X es una de las ciudades que quedan en la lista. Si la lista está vacía, la única acción posible es regresar a Madrid.

b) Una heurística admisible y sencilla es:
h(X, ( )) = horas de X a Madrid.
h(X, (A1, A2, …, An)) = menor tiempo de X a Ai + menor tiempo de Aj a Madrid
(se puede refinar mucho más).
c) No queda claro que h sea monótona, así que usamos A* con búsqueda en árbol.
d) (por detrás)


1º
MA (Bu,So,Te)
h=5, g=0, f=5
Bu 2.5
So 2.5
Te 3.5
h=4.5
g=2.5
f=7
BU
(So,Te)
2º
3º
SO
(Bu,Te)
h=4.5,
g=2.5
f=7
4º
TE
(Bu,So)
h=5,
g=3.5
f=8.5
2
4.3
2
2.5
4.3
2.5
h=6
g=4.5
f=10.5
SO
(Te)
h=5
g=6.8
f=11.8
TE
(So)
h=7.8
g=4.5
f=12.3
BU
(Te)
h=6.8
g=4.5
f=11.3
TE
(Bu)
h=4.5
g=7.8
f=12.3
BU
(So)
h= 4.5
g=6
f=10.5
SO
(Bu)
5º
6º
2.5
2
h=3.5
g=7
f=10.5
TE ()
7º
h=2.5
g=8
f=10.5
BU ()
8º
3.5
2.5
MA ()
h=0
g=10.5
f=10.5
9º SOL.
MA ()
h=0
g=10.5
f=10.5

2. La distancia de Levenshtein es una medida de lo diferentes que son dos palabras, y se define como el mínimo número de operaciones que hay que realizar en una palabra para transformarla en la otra. Las operaciones permitidas son:

- Inserción de una letra: ola -> olla
- Eliminación de una letra: pasar -> pasa
- Sustitución de una letra: pasar -> posar

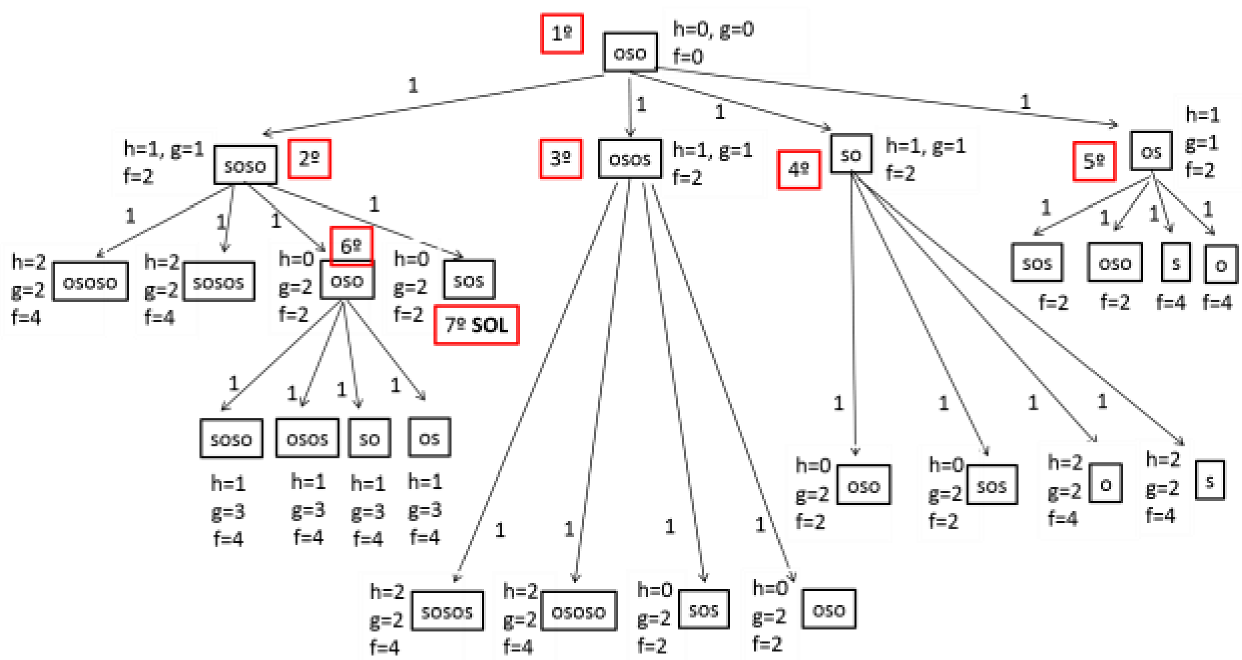
Cada una de estas operaciones tiene coste uno. Se puede demostrar que la distancia entre la palabra 1 y la palabra 2 es exactamente la misma que entre la palabra 2 y la 1.
Ejemplos:
- La distancia entre “ola” y “olla” es 1
- La distancia entre “as” y “sol” es 3

A continuación se implementará el cálculo de la distancia de Levenhstein a través de una búsqueda informada:
- ¿Cómo formalizas los estados? ¿cuál es S0? ¿y Sf?
- Propón una heurística que sea admisible para este problema y justifica su admisibilidad
- Calcula, a través de una búsqueda A* sin eliminación de estados repetidos y usando la heurística que has propuesto, la distancia de Levenshtein entre las palabras “oso” y “sos”

Para simplificar el desarrollo del árbol de búsqueda no permitiremos las inserciones que hagan que una letra aparezca seguida de ella misma (por ejemplo oso -> ooso no está permitida), y en las inserciones/sustituciones sólo se considerarán letras que pertenezcan a la palabra objetivo (por ejemplo oso -> poso no está permitida).
En caso de que haya empates se elegirá primero el nodo que haya sido generado antes. Si hay empates entre hermanos se considerará el siguiente orden de prioridad (de más a menos): insertar (de izquierda a derecha), eliminar (de izquierda a derecha), y sustituir (de izquierda a derecha).

SOLUCIÓN:

a) Cada estado corresponde a una palabra. S0: “oso”. Sf: “sos”.
b) h(n) = | longitud(palabra correspondiente a estado n) – longitud(palabra correspondiente a Sf) |
Como el coste de cada acción es 1 y cada acción cambia en 1 la longitud de la palabra, es imposible alcanzar Sf partiendo desde Sn con un número menor de acciones que h(n). Por tanto $h(n) \leq h^{*}(n)$. Hay otras posibles heurísticas más finas tales como la cardinalidad del conjunto diferencia entre el conjunto de caracteres de una y otra palabra (con o sin repeticiones).
c)



La distancia de Levenshtein entre las palabras “oso” y “sos” es el coste del camino óptimo del estado “oso” al final “sos”. Como hemos utilizado A* sin eliminación de estados repetidos usando una heurística admisible, el coste del camino encontrado por A* (2) es óptimo, y por tanto la distancia pedida.

3. En un bosque de pinos que estamos gestionando se ha detectado una plaga de orugas. Es necesario enviar un equipo para fumigar y limpiar el bosque en el mínimo tiempo posible.

El estado del bosque es el siguiente:

0	1
2	X

X: Zona rocosa (inaccesible por tierra y libre de la plaga)
0: Zona sin plaga
1: Zona con población de oruga baja
2: Zona con población de oruga alta

N
O E
S

El equipo de fumigación posee un mapa en el que se indican únicamente los sectores del bosque. El grado de infección solo se puede conocer sobre el terreno.

- Para eliminar la plaga en una zona con población de oruga baja, necesitamos realizar una única operación de limpieza. Para eliminar la plaga en una zona con población de oruga alta, necesitamos realizar dos operaciones de limpieza con una separación de, al menos, 4 horas.
- Para comenzar su tarea, el equipo puede acceder al bosque en cualquier sector, ya que será transportado en helicóptero. Para generar el árbol de búsqueda el orden de exploración para el acceso es el de las agujas del reloj, comenzando con el sector al noroeste. Es decir: NO, NE, SE, SO.
- Para realizar la limpieza, tienen que hacerlo a pie. El tiempo para realizar una operación de limpieza de un sector es 1 hora.
- El tiempo para desplazarse entre sectores adyacentes es de 2 horas en horizontal (oeste ↔ este) o vertical (norte ↔ sur) y 3 horas en diagonal (sudoeste ↔ nordeste, sudeste ↔ nororeste). Para generar el árbol de búsqueda el orden de exploración de desplazamientos es el de las agujas del reloj, empezando por el movimiento que produce un desplazamiento hacia el norte. Es decir, el orden de exploración es N, NE, E, SE, S, SO, O, NO.
- Una vez que el equipo ha eliminado la plaga en el bosque el equipo será transportado de nuevo por helicóptero desde el sector en el que se encuentre.

a. Formaliza los estados de búsqueda.

Utilizando la formalización propuesta:

b. Especifica las acciones para generar sucesores, las precondiciones necesarias para aplicarlas 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. 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. De acuerdo con el resultado obtenido, ¿cuál es la estrategia óptima? ¿cuál es el coste óptimo?
h. ¿Garantiza A* con heurística admisible y eliminación de estados repetidos encontrar la solución óptima?
i. ¿Encuentra A* la solución óptima en este ejemplo? Justifica tu respuesta.
j. Repite el ejercicio desde el estado inicial

1	1
2	X

SOLUCIÓN:

a. Formaliza los estados de búsqueda.
Cada estado se especifica indicando la posición del equipo de limpieza y el grado de infección en cada uno los sectores

A_{11}	A_{12}
A_{21}	X

$0 \leq A_{11} \leq 1;\ 0 \leq A_{12} \leq 1;\ 0 \leq A_{21} \leq 2;$

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

Se pueden realizar cinco tipos de acciones:

- Comenzar la exploración [coste 0]: El
    - Precondiciones: Solo se puede aplicar en el estado inicial.
- Realizar un desplazamiento a alguna de las celdas adyacentes que sean accesibles.
    - Precondiciones: Sector en el que se encuentra el equipo de limpieza libre de plagas (nivel 0) + algún de los otros sector por limpiar
    - Movimientos N, S, E, O [Coste 2]
    - Movimientos NE,NO SE, SO [Coste 3]
- Limpiar el sector en la que se encuentra el equipo de limpieza sin desplazar [Coste 1].
    - Precondiciones: Sector en el que se encuentra el equipo de limpieza con una población baja de orugas (nivel 1) + restos de sectores libres de plaga (nivel 0)
- Realizar un desplazamiento a alguna de las celdas adyacentes que sean accesibles.
    - Precondiciones: Sector en el que se encuentra el equipo de limpieza libre de plagas (nivel 0) + algún de los otros sectores por limpiar
    - Movimientos N, S, E, O [Coste 2]
    - Movimientos NE,NO SE, SO [Coste 3]
- Limpiara la celda en la que se encuentra el equipo y realizar un desplazamiento a alguna de las celdas adyacentes que sean accesibles.
    - Precondiciones: Sector en el que se encuentra el equipo de limpieza con población de orugas (nivel > 0) + algún de los otros sectores por limpiar
    - Limpieza + movimientos N, S, E, O [Coste 1 + 2 = 3]
    - Limpieza + movimientos NE,NO SE, SO [Coste 1 + 3= 4]

c. Especifica el estado inicial.

0	1
2	X

El equipo de limpieza aún no ha accedido al bosque.

d. Especifica el test objetivo.

0	0
0	X

El equipo de limpieza puede estar en cualquier sector.

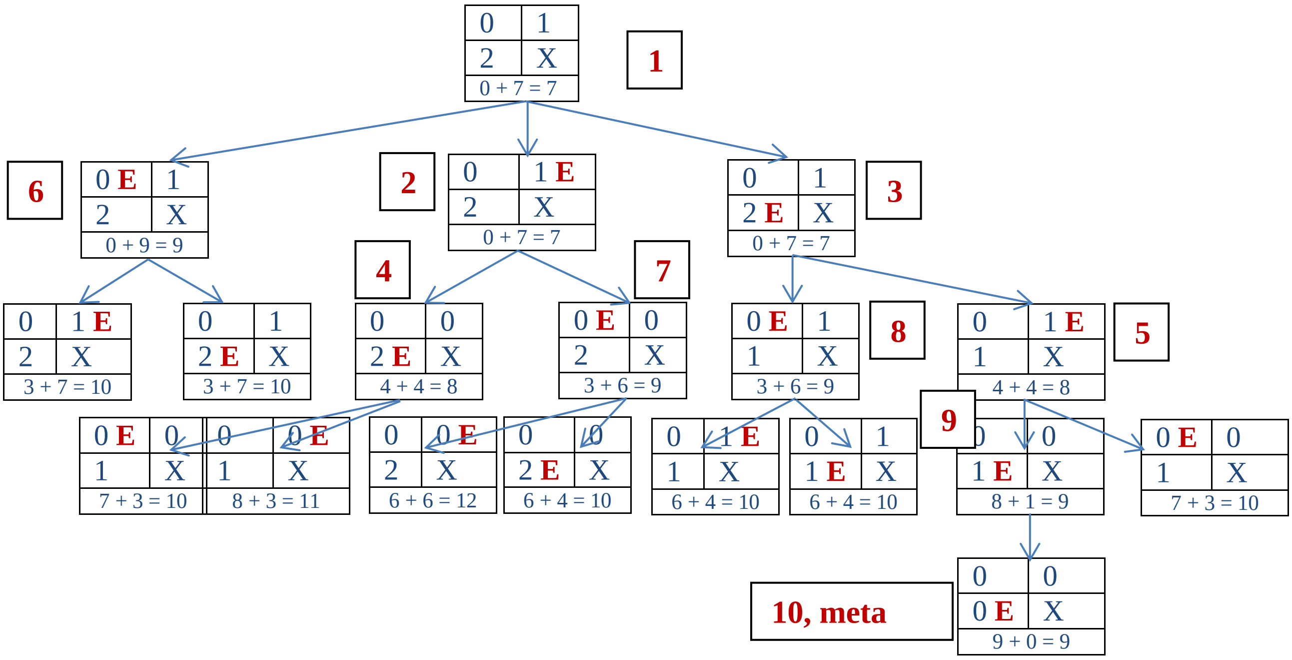
e. Define una heurística admisible no trivial para resolver el problema y demuestra su admisibilidad.

Por relajación, suponer que:

- cada sector (i,j) requiere A_{ij} limpiezas [coste de operación de limpieza 1],
- el número de desplazamientos es igual a $(A_{11}+A_{12}+A_{21})$ en caso de que el sector en el que se encuentra el equipo está libre de orugas, $(A_{11}+A_{12}+A_{21}) - 1$, si es el estado inicial o el sector en el que se encuentra el equipo de limpieza está infestado.
- en el problema relajado, el coste de un desplazamiento es igual al coste mínimo de los desplazamiento posibles en el problema original [coste de desplazamiento 2]

(1+2)*(A_{11}+A_{12}+A_{21}) [– 2, si es el estado inicial o si el sector en el que se encuentra el equipo de limpieza está infestado, ya que no es necesario desplazarse a ese sector para realizar la operación de limpieza]

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. De acuerdo con el resultado obtenido, ¿cuál es la estrategia óptima? ¿cuál es el coste óptimo?

SO [limpia, 1] →, 3 NE [limpia, 1] →, 3 SO [limpia, 1]. Coste total mínimo: 9

h. ¿Garantiza A* con heurística admisible y eliminación de estados repetidos encontrar la solución óptima?

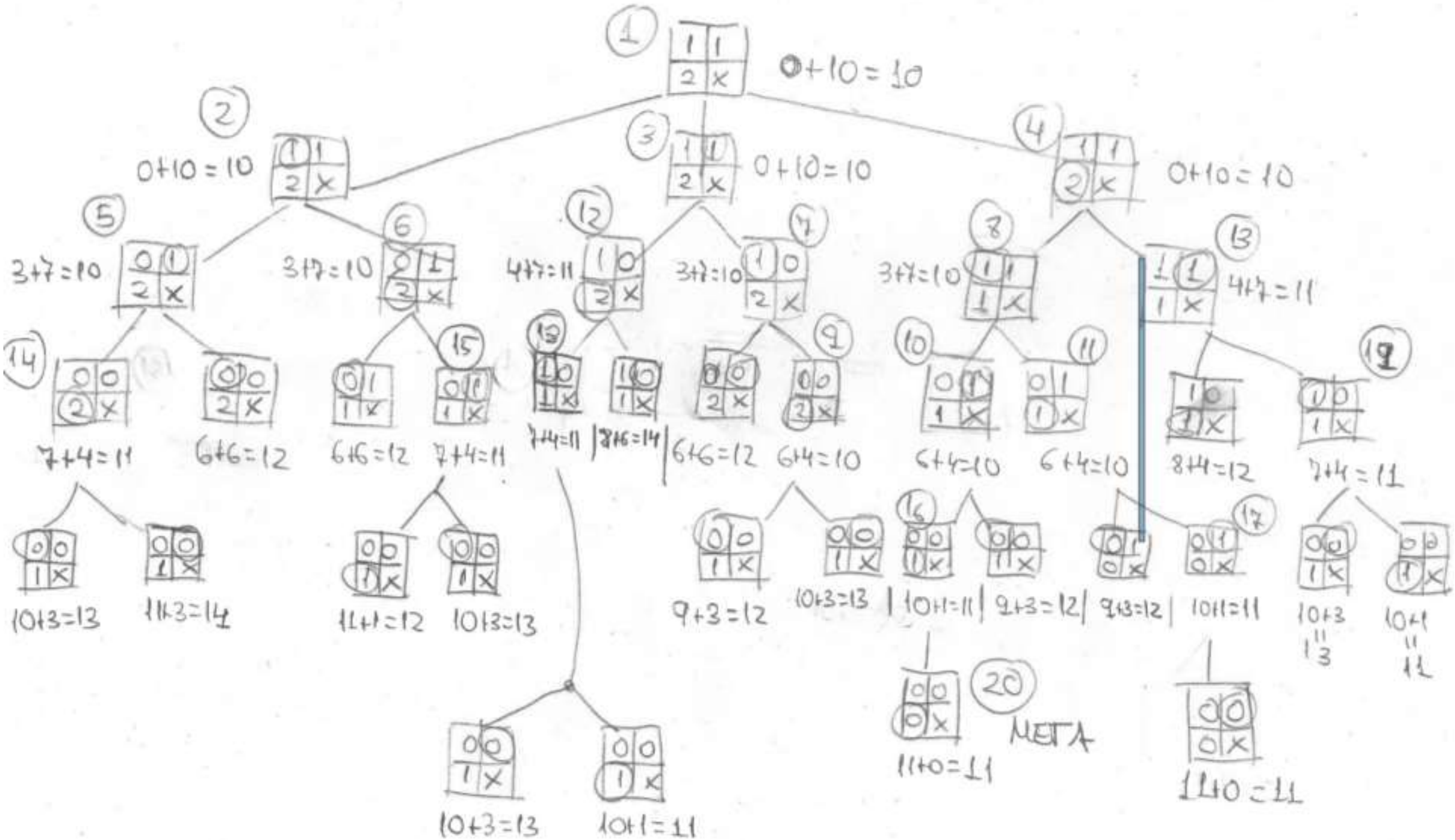
No, A* + eliminación de estados repetidos + h admisible no es necesariamente óptima.

i. ¿Encuentra A* la solución óptima en este ejemplo? Justifica tu respuesta.

Sí, porque la heurística es monótona (es la solución óptima de un problema relajado):
A* + eliminación de estados repetidos + h monótona = óptima.

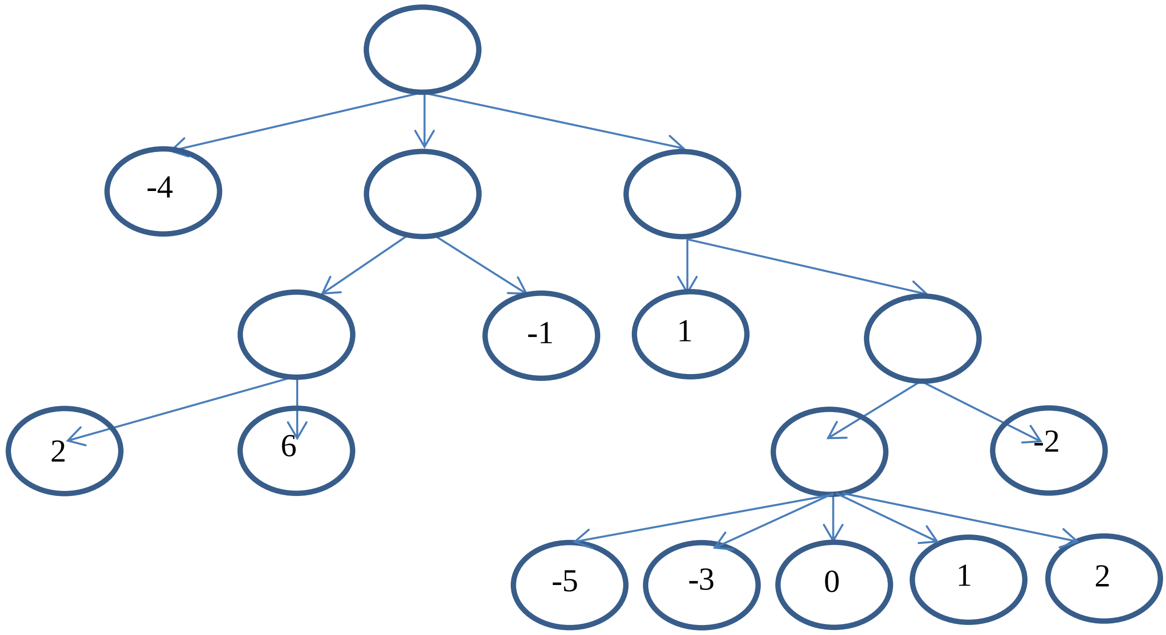
j. Repite el ejercicio desde el estado inicial

1	1
2	X



SO [limpia,1] →,2 NO [limpia,1] →,2 NE [limpia,1] →,3 SO [limpia,1]. Coste total mínimo: 11

4. Consideremos el siguiente árbol para un juego de suma nula entre dos jugadores que realizan movimientos en turnos alternos. Supondremos que ambos jugadores son óptimos. Los nodos terminales del árbol de juego están etiquetados con las puntuaciones de las posiciones terminales del juego, desde el punto de vista del jugador X, quien es el que comienza la partida. En el recorrido del árbol las jugadas se explorarán de izquierda a derecha.



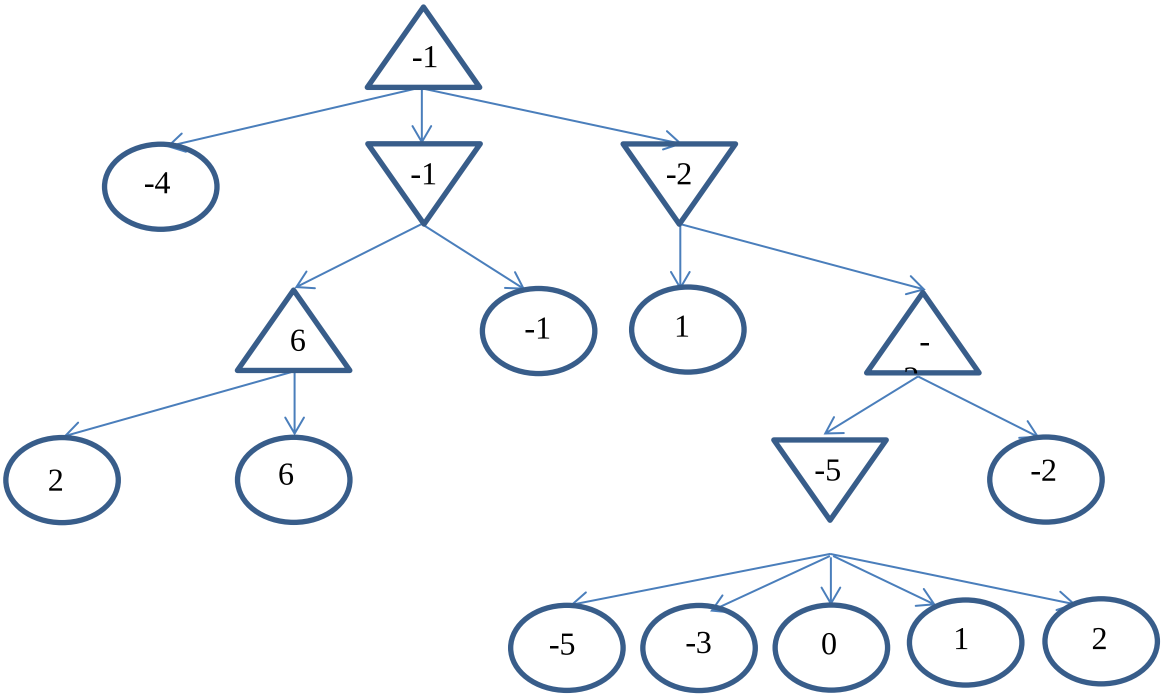
a. Aplica el algoritmo MINIMAX.
   De acuerdo con este algoritmo, ¿cuál sería la puntuación final para X? ¿Cuál es la jugada óptima para el jugador X? ¿Cómo continuaría la partida hasta el final?

b. Aplica el algoritmo MINIMAX con poda alfa-beta.
   De acuerdo con este algoritmo, ¿cuál es el valor minimax para X? ¿Cuál es la jugada óptima para el jugador X? ¿Cómo continuaría la partida hasta el final?

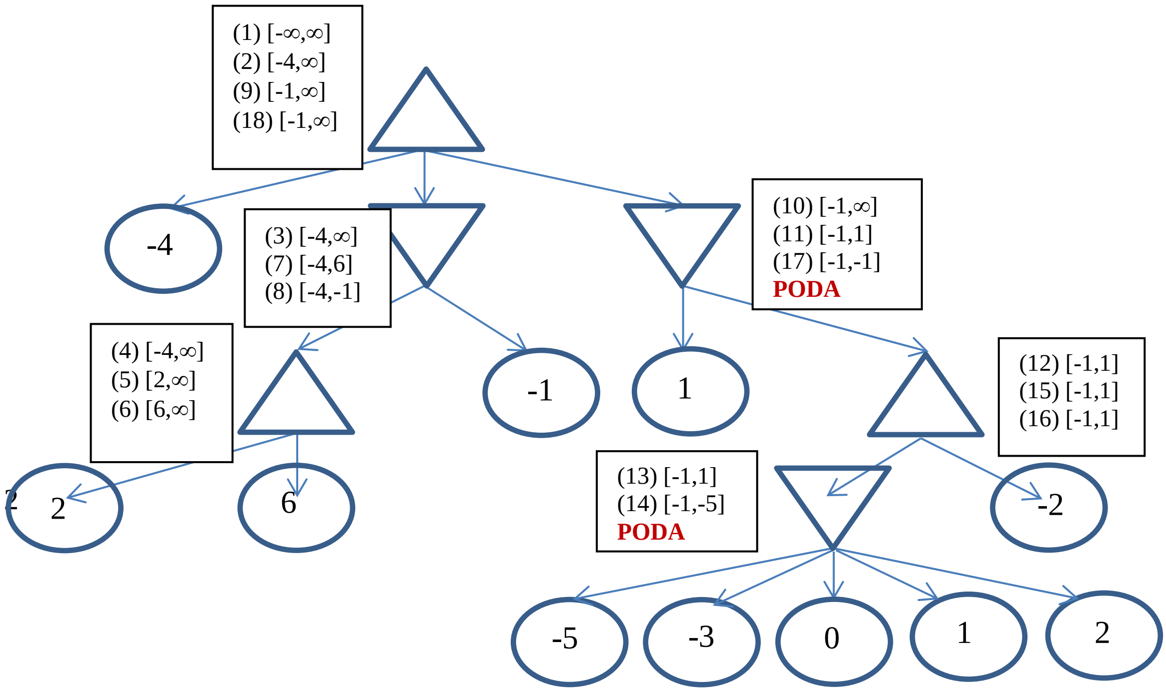
c. En un caso general, ¿podrían variar las respuestas a las preguntas realizadas entre MINIMAX sin poda y MINIMAX con poda alfa-beta? En caso de que la respuesta sea negativa, indicad las razones para ello. En caso de que sea positiva, indicad los cambios posibles y las razones para ello.

d. En un caso general, ¿podrían variar las respuestas a las preguntas realizadas si alteramos el orden de exploración de las jugadas (por ejemplo de derecha a izquierda, en lugar de de izquierda a derecha? En caso de que la respuesta sea negativa, indicad las razones para ello. En caso de que sea positiva, indicad los cambios posibles y las razones para ello.

SOLUCIONES:
a. Algoritmo MINIMAX



Algoritmo MINIMAX con poda alfa-beta:



En ambos casos el valor minimax es -1.
La jugada óptima para el jugador X es la central. El jugador Y respondería optando por la jugada de la derecha. La puntuación de X en su jugada óptima es -1.

El valor minimax y la secuencia óptima de jugadas obtenidas con o sin poda debe ser los mismos.
Con la el orden de exploración se podrían seleccionar secuencias de movimientos distintas, en caso de que hubiera más de una secuencia de jugadas óptima. En todo caso, el valor minimax no cambiaría.

5. Consideremos una variante del nim, en el que, partiendo de una pila de 7 monedas, dos jugadores toman turnos para realizar movimiento. En cada turno el jugador elige una pila de entre las existentes y la divide en **dos** subpilas **no vacías** y **de tamaños diferentes**. Pierde el jugador que se queda sin movimientos.

   Dibuja el árbol de juego completo

   - Las jugadas que generan subpilas de tamaños más dispares se generan primero.
   - En el despliegue del árbol es necesario que tengas en cuenta las simetrías para evitar que este sea excesivamente grande.
   - Etiqueta los terminales con el valor de la función de utilidad respecto a MAX.

   De acuerdo con la formalización realizada

   a. ¿Es un juego de suma cero, de suma constante, o de suma no nula y no constante?
   b. En caso de que el oponente sea óptimo ¿Cuál sería la estrategia óptima para el jugador que inicia el juego?
   c. Sobre el árbol de juego desplegado aplica el algoritmo minimax.
      De acuerdo con este algoritmo, ¿cuál sería el valor minimax en la raíz del árbol? ¿cuál sería la jugada óptima?
   d. Sobre una copia del árbol de juego, aplica algoritmo minimax con poda alpha-beta. En esta copia incluye únicamente los nodos visitados.
      De acuerdo con este algoritmo, ¿cuál sería el valor minimax en la raíz del árbol? ¿cuál sería la jugada óptima?

SOLUCIÓN:

a. ¿En un juego de suma cero, de suma constante, o de suma no nula y no constante?

Con la formalización realizada, es un juego de suma constante.

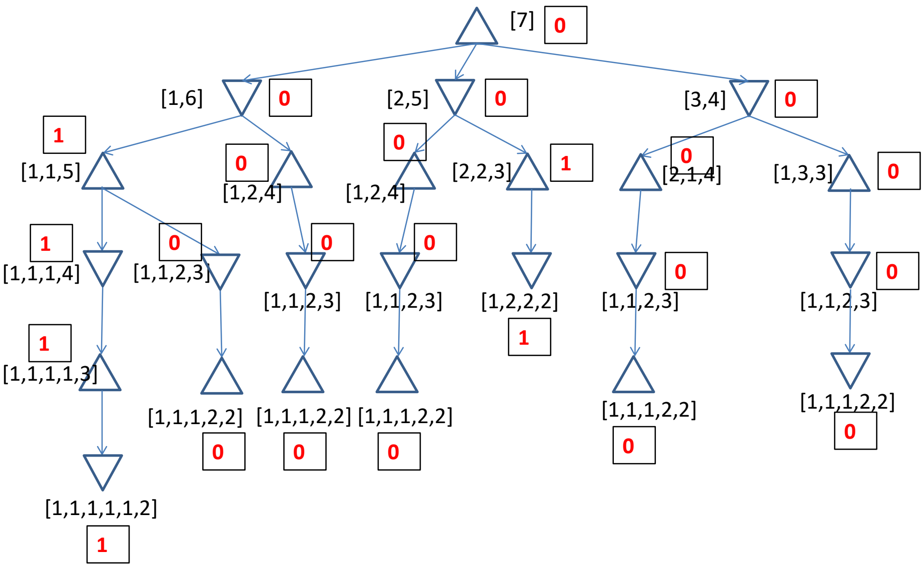
b. En caso de que el oponente sea óptimo ¿Cuál sería la estrategia óptima para el jugador que inicia el juego?

La estrategia que se derive de aplicar el algoritmo minimax.

c. Sobre el árbol de juego desplegado aplica el algoritmo minimax.

De acuerdo con este algoritmo, ¿cuál sería el valor minimax en la raíz del árbol? ¿cuál sería la jugada óptima?

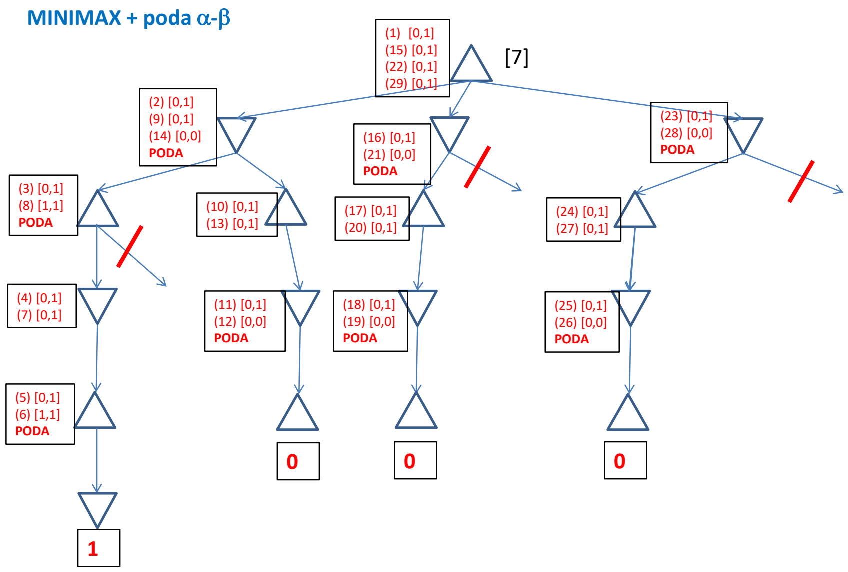
MINIMAX



Valor minimax en la raíz: 0

Jugada óptima: Cualquiera de las tres

d. Sobre una copia del árbol de juego, aplica algoritmo minimax con poda alpha-beta. En esta copia incluye únicamente los nodos visitados.
De acuerdo con este algorimo, ¿cuál sería el valor minimax en la raíz del árbol? ¿cuál sería la jugada óptima?



Valor minimax en la raíz: 0
Jugada óptima: Cualquiera de las tres

6. En la segunda mitad el siglo XX el compositor Iannis Xenakis utilizó el marco de teoría de juegos para realizar tres composiciones: *Duel* (1959), *Stratégie* (1962) y *Linaia Agon* (1972). Las primeras dos consisten en batallas musicales entre dos orquestas, en las que cada uno de los directores elige entre distintos materiales sonoros (por ejemplo silencio, *glissandi* de cuerdas, secuencias estocásticas de percusión, anillos, clústeres, etc.) con el fin de maximizar una función de utilidad. Con el fin de dotar de variabilidad a la pieza, las composiciones de Xenakis son juegos de suma no nula y no constante.
En este ejercicio, trataremos de realizar una composición musical para dos orquestas (A y B) basada en un juego en el que dos directores eligen, alternando sus turnos, el material sonoro a interpretar por cada orquesta. La obra tiene 3 movimientos. En el primer y tercer movimiento, elige el director A entre dos opciones (A1 o A2). En el segundo es el director B es que elige el material sonoro (B1 o B2). En la pieza puede haber repeticiones. Gana el duelo la orquesta que obtenga la mayor puntuación, siendo importante el margen de ganancia. Para calcular las puntuaciones obtenidas por cada orquesta se utilizan estas matrices:

Puntuaciones para A
Ai_Bj; i,j $\in\{1,2\}$

	B1	B2
A1	3	4
A2	2	1

Puntuaciones para B
Bj_Ak, j,k $\in\{1,2\}$

	A1	A2
B1	5	1
B2	3	3

y la regla: "La secuencia Ai_Bj_Ak conduce a las puntuaciones
Orquesta A: (Ai_Bj) + (Ak_Bj)
Orquesta B: (Bj_Ai) + (Bj_Ak)
i,j,k $\in\{1,2\}$

Por ejemplo, para la secuencia A1_B1_A2, las puntuaciones son: Orquesta A: (A1_B1) + (A2_B1) = 3+2 = 5
Orquesta B: (B1_A1) + (B1_A2) = 5+1 = 6
Gana la orquesta B.

(i) ¿Qué tipo de juego es?

Es un juego de suma cero, ya que lo que gana una orquesta lo pierde la otra.

(ii) Para este tipo de juego, ¿cuál es el algoritmo que determina cuál será la forma final de ejecución de la obra, suponiendo que tanto el director A como el B realizan elecciones óptimas?

Minimax.

(iii) En esas condiciones determina, desplegando el árbol de juego, la forma final de la obra.
Se debe utilizar poda alfa-beta.
El árbol de juego debe ser desplegado a medida que se va realizando la exploración, indicando claramente los pasos del algoritmo, numerando y etiquetando cada uno de ellos con la información relevante e indicando claramente los momentos de poda.
IMPORTANTE: Únicamente se deben incluir y etiquetar los nodos descubiertos en la exploración del algoritmo. Puede que sea más cómodo dibujar el árbol en una hoja con formato apaisado, con el fin de tener suficiente espacio para el etiquetado.

Ver el árbol de juego en la siguiente página.

(iv) ¿Cuál es el valor minimax en la raíz del árbol de juego?

-1

(v) ¿Cuál es la puntuación de cada orquesta?

Orquesta A: 5, Orquesta B: 6

(vi) ¿Qué orquesta ha ganado el duelo?

Orquesta B

(vii) ¿Cuál es la elección óptima para la orquesta A?

La orquesta A elegirá A1

(viii) De acuerdo con el algoritmo minimax ¿Cuál es la forma final de la obra?

(A1-B1-A2)

(ix) De acuerdo con el árbol generado ¿Cuál es la forma o las posibles formas finales de la obra?

(A1-B1-A2), (A2-B1-A1), (A2-B2-A1)

(x) ¿Garantiza minimax con poda alfa-beta encontrar todas las posibles formas finales de la obra?

No, ya que al realizar poda, puede que no se exploren algunas alternativas que son igualmente óptimas para la orquesta A.

	B1	B2
A1	3	4
A2	2	1

	A1	A2
B1	5	1
B2	3	3