



# Problemas de Inteligencia Artificial

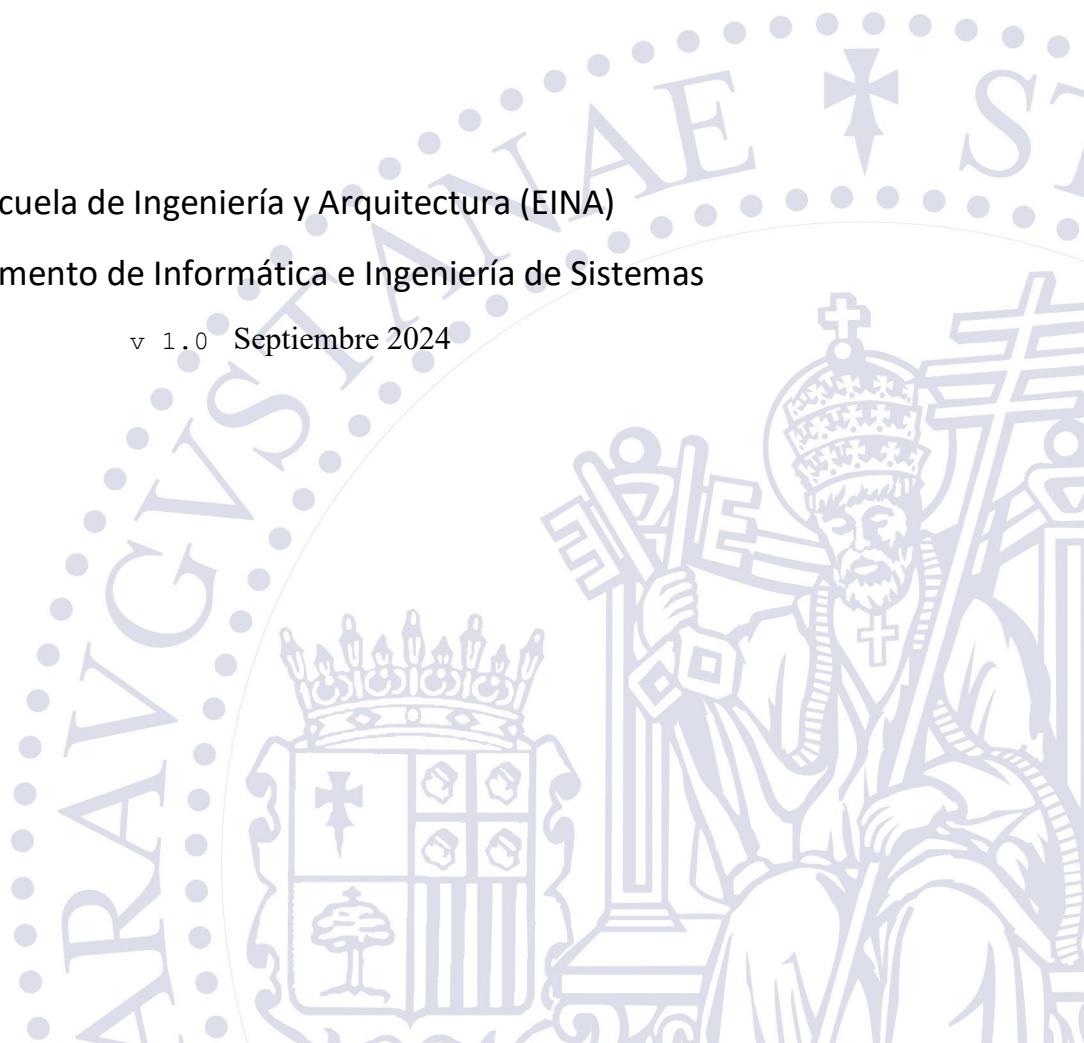
Tercer Curso Graduado en Ingeniería Informática

**Curso 2024-2025**

Escuela de Ingeniería y Arquitectura (EINA)

Departamento de Informática e Ingeniería de Sistemas

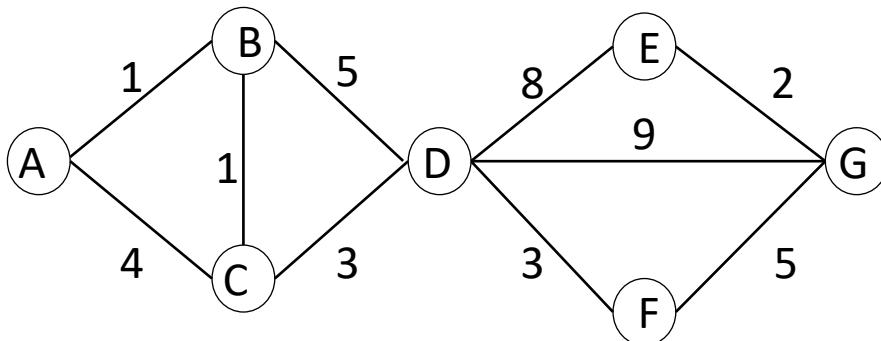
v 1.0 Septiembre 2024





### Problema1.

Considera el grafo que representa el espacio de estados donde A es el estado inicial, y G es el estado objetivo. El coste de cada arco es mostrado en el grafo. Cada arco puede ser atravesado en ambas direcciones.



**Posibles caminos encontrados.** Para cada uno de las siguientes estrategias de búsqueda, marca con una X que caminos de los listados podría devolver la estrategia indicada. Ten en cuenta que para algunas estrategias el camino devuelto dependerá de cómo resuelvas las situaciones en las que puedes escoger varias alternativas. En tales casos, asegúrate de que marcas todos los caminos que podrían devolverse resolviendo de distinta manera los empates.

Algoritmo de Búsqueda	A-B-D-G	A-C-D-G	A-B-C-D-F-G
Búsqueda en profundidad			
Búsqueda en anchura			
Búsqueda de coste uniforme			

### Problema2.

Un sistema puede encontrarse en un conjunto de estados  $\{S_0, \dots, S_7\}$ . Su estado inicial es  $S_0$  y su estado objetivo es  $S_7$ .

Para este problema se han encontrado los siguientes operadores y costes asociados a cada operador:

OP1: $S_0 \rightarrow S_1$ (coste 10)	OP2: $S_1 \rightarrow S_2$ (coste 10)	OP3: $S_0 \rightarrow S_3$ (coste 20)
OP4: $S_0 \rightarrow S_4$ (coste 20)	OP5: $S_1 \rightarrow S_5$ (coste 100)	OP6: $S_2 \rightarrow S_5$ (coste 80)
OP7: $S_5 \rightarrow S_6$ (coste 25)	OP8: $S_3 \rightarrow S_6$ (coste 20)	OP9: $S_6 \rightarrow S_7$ (coste 1)

Y se ha propuesto los siguientes valores de la función heurística  $h'$  que estima el menor coste desde cada estado al objetivo y que podemos asumir que es admisible:

$h'(S_0) = 40$	$h'(S_3) = 100$	$h'(S_6) = 10$
$h'(S_1) = 20$	$h'(S_4) = 110$	$h'(S_7) = 0$
$h'(S_2) = 40$	$h'(S_5) = 20$	

Se pide aplicar distintas estrategias de búsqueda y para cada una de ellas de pide indicar la secuencia de nodos (que se deben nombrar por sus respectivos estados) visitados y la secuencia de la solución.



**Representa los árboles de exploración en grafo y en árbol.** En caso de empate resuelve expandiendo el nodo que tengas dibujado más a la izquierda (Dibuja los nodos al expandir en orden alfabético de izquierda a derecha).

**a) búsqueda en anchura (grafo, árbol)**

- Secuencia de nodos visitados: .....
- Secuencia de operadores de la solución: .....

**b) búsqueda en profundidad (grafo, árbol)**

- Secuencia de nodos visitados: .....
- Secuencia de operadores de la solución: .....

**c) búsqueda en profundidad con profundización iterativa (con un paso de 1)**

- Secuencia de nodos visitados: .....
- Secuencia de operadores de la solución: .....

**d) búsqueda bidireccional**

- Secuencia de nodos visitados: .....
- Secuencia de operadores de la solución: .....

**e) búsqueda en escalada por la máxima pendiente**

- Secuencia de nodos visitados: .....
- Secuencia de operadores de la solución: .....

**f) búsqueda primero el mejor**

- Secuencia de nodos visitados: .....
- Secuencia de operadores de la solución: .....

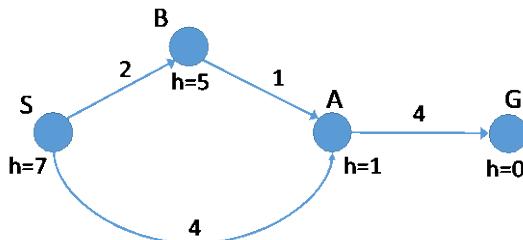
**g) algoritmo A\* (grafo, árbol)**

- Secuencia de nodos visitados: .....
- Secuencia de operadores de la solución: .....



### Problema 3

Considera el siguiente grafo dibujado que representa el espacio de estados donde S es el estado inicial, y G es el estado objetivo.



Para cada una de las siguientes estrategias de búsqueda (1-3), muestra el árbol de búsqueda, representando el orden de expansión de nodos en el árbol, y el estado de la FRONTERA y EXPLORADOS, si procede, en cada paso de la búsqueda.

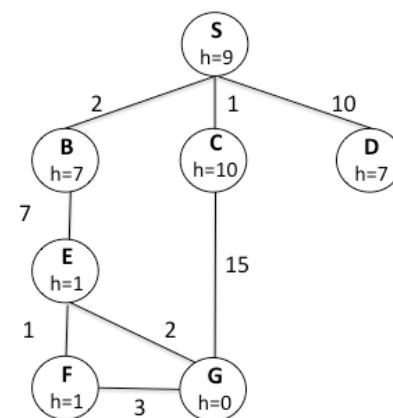
1. Búsqueda Coste Uniforme en Grafo.
2. Búsqueda A\* en árbol
3. Búsqueda A\* en grafo
4. ¿Se obtiene la solución óptima en todos los casos anteriores? Explica tu respuesta para cada caso.
5. Manteniendo los valores de la función heurística para  $h(S)=7$ ,  $h(B)=5$  y  $h(G)=0$ , ¿Cuál es el **menor** valor de  $h(A)$  para que la heurística sea consistente? ¿Cuál es el **mayor** valor de  $h(A)$  para que la heurística sea admisible?
6. En el grafo anterior, ¿La búsqueda en profundidad es completa incluso si el coste del arco S-A es cero?

### Problema 4.

Considera el grafo dibujado abajo que representa el espacio de estados donde S es el estado inicial, y G es el estado objetivo. Todos los arcos son bidireccionales.

Para cada una de las siguientes estrategias, muestra el camino devuelto, o escribe *ninguno* si no devuelve ningún camino solución. En caso de empates, asume que se expanden por orden alfabético. Representa los **árboles de búsqueda**, mostrando el **orden de expansión** de nodos y escribe el estado de la **FRONTERA** y **EXPLORADOS**, si procede, en cada ciclo de la búsqueda. Debes utilizar el **test de objetivo** al *generar* el nodo en la búsqueda en profundidad, y cuando proceda en el resto debes utilizar el **test-objetivo** al *expandir* el nodo.

- (a) Búsqueda en grafo en profundidad
- (b) Hill Climbing. Búscando el mínimo.
- (c) Búsqueda en grafo A\*.

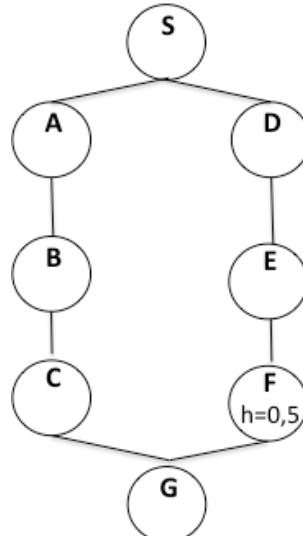




### Problema 5.

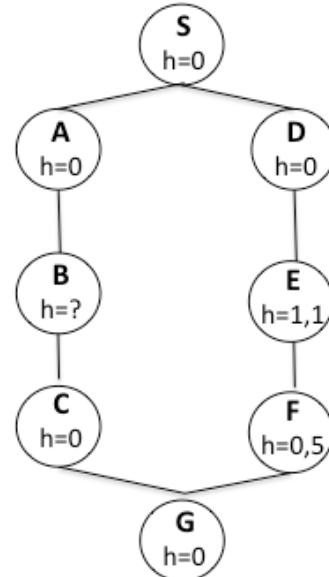
Para las siguientes cuestiones, todos los arcos en grafos tienen coste 1.

- (a) Supongamos que estás diseñando una heurística para el grafo de la derecha. Te dicen que  $h(F) = 0,5$ , pero no te dan más información. ¿Qué rango de valores son posibles para  $h(D)$  si se deben cumplir las siguientes condiciones?
- $h$  debe ser admisible
  - $h$  debe ser admisible y consistente



- (b) Ahora supongamos que  $h(F) = 0,5$  y  $h(E) = 1,1$ , y que todos los demás valores de heurísticas excepto  $h(B)$  son cero (como se muestra en la figura de la derecha). Para cada una de las siguientes partes, indica el rango de valores de  $h(B)$  que da lugar a una heurística **admisible** y da lugar al siguiente orden de expansión utilizando una **búsqueda en árbol A\***. Si el orden considerado es imposible con una heurística admisible tu respuesta debe ser **ninguna**. Los empates se resuelven alfabéticamente. Por supuesto, asumimos que  $h$  es no negativa.

- B se expande antes que E que se expande antes que F
  - E se expande antes que B que se expande antes que F
- (c) Responde al apartado al apartado (b) para una heurística **consistente** utilizando una **búsqueda en grafo A\***.





### Problema 6.

La reciente crecida del río Ebro ha provocado daños en las infraestructuras de los municipios ribereños que deben ser reparadas con urgencia. Concretamente, hay 4 obras por realizar y se ha pedido presupuesto a 4 empresas constructoras para cada una de las obras. El coste de encargar cada obra a cada empresa viene dado por la siguiente tabla, donde se indica el coste de encargar a la empresa  $E_i$  la obra  $O_j$ .

	Obra $O_1$	Obra $O_2$	Obra $O_3$	Obra $O_4$
Empresa $E_1$	2	3	2	4
Empresa $E_2$	5	5	4	5
Empresa $E_3$	6	5	4	3
Empresa $E_4$	10	8	6	6

La Confederación Hidrográfica del Ebro ha decidido asignar **una sola obra por empresa**. El problema consiste en decidir qué obra se asignará a cada empresa, de modo que se **minimice el coste total**. Los técnicos deciden utilizar el algoritmo A\* para resolver el problema.

- (a) Define una **representación “eficiente” del problema**, especificando el conjunto de posibles estados, estado inicial, estados finales, así como operador(es) y su coste.
- (b) Define una buena **función heurística  $h^*$**  optimista para el problema general. **Pista:** Una restricción que relaja el problema es permitir que a las empresas que no tienen tarea asignada se les permita realizar hacer más de un trabajo de los pendientes de asignar.
- (c) **Desarrolla el árbol de búsqueda que genera el algoritmo A\***. Indica el orden en el que se expanden los nodos y los valores de  $g$ ,  $h^*$  y  $f^*$  para cada nodo del árbol de búsqueda. En caso de empate en la función heurística, considera que se expanden primero los nodos que has dibujado más a la izquierda en tu árbol de búsqueda.
- (d) ¿Se puede resolver el problema con un algoritmo de búsqueda local, por ejemplo, Hill-Climbing?



### Problema 7.

Dada la siguiente figura, encontrar un camino de cuadros blancos y negros. Se comienza situado en el cuadrado más a la izquierda del camino, y el objetivo es salir del camino por la derecha en el menor número de movimientos. Se comienza en el primer cuadrado, pero el objetivo es estar fuera del camino. Si estás en un cuadrado blanco puedes avanzar 1 o 2 cuadradados a la derecha. Si estás en un cuadrado negro puedes avanzar 1 o 4 cuadradados a la derecha.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----

1. **Describe el problema:** Representación del estado, Estado inicial, Estado final, objetivo, acciones y función de coste)
2. ¿Cuál es el factor de ramificación?
3. **Desarrolla el árbol de búsqueda** que genera el algoritmo **búsqueda en anchura sobre un grafo**. ¿Cuál es la solución óptima?
4. **Plantea una heurística admisible** no trivial ( $h=0$  no vale) para el problema. Reflexiona sobre la admisibilidad de la heurística propuesta.
5. Desarrolla **el árbol de búsqueda que genera el algoritmo de búsqueda primero el mejor** en grafo utilizando la heurística propuesta.

### Problema 8.

Considera una búsqueda en árbol (es decir, no hay lista de expandidos) sobre un problema de búsqueda con un factor de ramificación  $b$ . Cada nodo del árbol de exploración tiene un coste acumulado desde el estado inicial al estado del nodo  $g(n)$ , una heurística admisible  $h(n)$  y una profundidad  $d(n)$ . Sea  $c$  un nodo objetivo con el mínimo coste y  $s$  el nodo objetivo menos profundo.

Para cada una de los siguientes puntos, da na expresión que caracterice el conjunto de nodos que son expandido antes de que termine la búsqueda. Por ejemplo, si pedimos el conjunto de nodos con valor heurístico positivo, puedes utilizar la expresión  $h(n) \geq 0$ . No te preocupes de los empates (así que no debes preocuparte de diferencias  $>$  de  $\geq$  en caso de empates). Si no hay nodos que cumplan la expresión debes escribir “ninguno”.

1. Da una expresión (es decir, una desigualdad en términos de las anteriores cantidades) para los nodos que serán expandidos en una búsqueda en anchura en árbol.
2. Da una expresión para los nodos expandidos en una búsqueda de coste uniforme.
3. Da una expresión para los nodos  $n$  expandidos en una búsqueda  $A^*$  en árbol con heurística  $h(n)$ .
4. Sean  $h_1$  y  $h_2$  dos heurísticas admisibles tal que  $\forall n, h_1(n) \geq h_2$ . Da una expresión para los nodos que se expandirán en la búsqueda  $A^*$  en árbol usando  $h_1$  pero no se expandirán cuando se use  $h_2$ .
5. Da una expresión para los nodos que serán expandidos por una búsqueda  $A^*$  en árbol usando  $h_2$  pero no cuando se use  $h_1$ .



### Problema 9.

El NIM es un juego con dos jugadores. Inicialmente hay  $n$  piedras en  $r$  cestos separados. El estado inicial puede expresarse como  $(n_1, n_2, \dots, n_r)$  donde  $n_1 + n_2 + \dots + n_r = n$ . En cada movimiento de juego, un jugador puede coger cualquier número de piedras ( $> 0$ ) de cualquiera de los  $r$  cestos (sólo de un cesto cada turno). En la versión de juego que se propone, **la última persona que hace movimiento gana**. Supondremos que el estado inicial del juego es  $(3,1)$ : dos cestos de piedras, con tres piedras en el primer cesto, y una piedra en el segundo.

- Representa el árbol de juego completo.
- Representa una evaluación **MINIMAX** del juego. A un nodo hoja se le da valor 1 si corresponde a un nodo Max ganador, y cero si representa un nodo Min ganador. No necesitas una función de estimación puesto que tienes un árbol de juego completo. Asegúrate de dar el valor correcto de 0, +1 o -1 a los nodos. Por ejemplo, si en un nodo MIN el estado es  $(0,1)$ , el jugador MIN retirará una piedra para dejar el estado terminal  $(0,0)$ , con lo que el nodo terminal resultante tendrá el valor -1 (MIN gana por haber realizado el último movimiento). Si, por ejemplo, en un estado MAX el estado es  $(0,1)$ , MAX retirará la piedra dejando un nodo hoja resultante  $(0,0)$ , con lo que el nodo hoja resultante tendrá el valor +1 (MAX gana por haber realizado el último movimiento).

### Problema 10.

El NIM-5 es un juego con dos jugadores con las siguientes reglas: Hay apilados 5 euros. Los jugadores se alternan coger 1, 2 o 3 euros de la pila. **El jugador que coge el último euro pierde**. Muestra el árbol de juego e indica que movimiento debe escoger el primer jugador para ganar.

### Problema 11.

La posición inicial del juego que se describe a continuación es la mostrada en la figura.

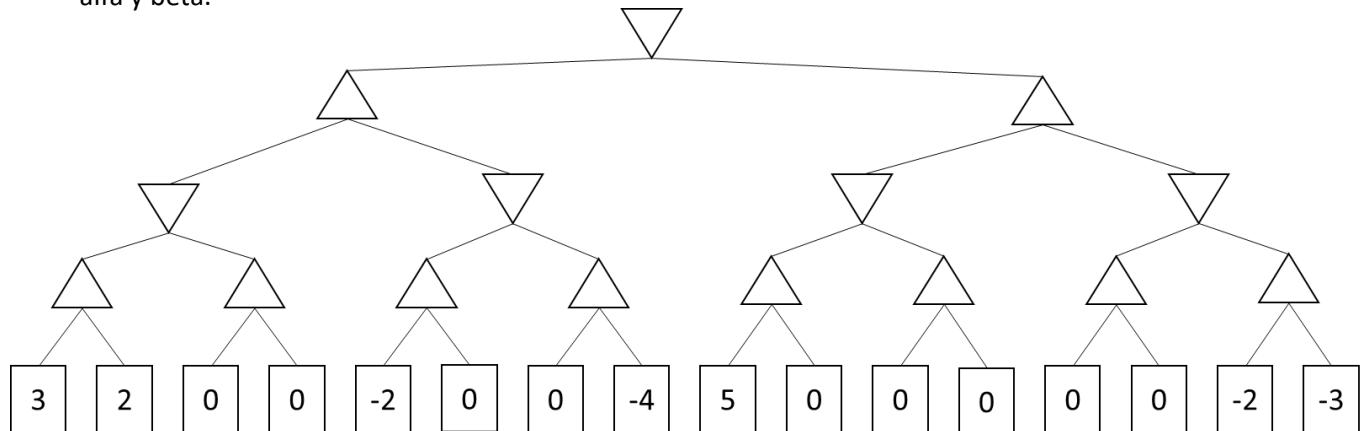
X	O	X	O	+	X	O	X	O
---	---	---	---	---	---	---	---	---

El jugador MAX juega con las fichas marcadas como X y el MIN con las fichas marcadas como O. MAX realiza el primer movimiento. La única ficha que pueden mover los jugadores es la marcada como +. Esta ficha se puede mover hacia la derecha y hacia la izquierda. Al intercambiarse con la ficha que ocupa la posición a la que se desplaza tiene el efecto de cambiar el signo de esa ficha y de la que pasa a ser su contigua. Por ejemplo, en la configuración XOXO+XOXO, si desplazamos la ficha + hacia la derecha obtenemos la configuración XOXOO+XXO. El objetivo de cada jugador es tener 5 fichas del mismo tipo seguidas. Como el juego es demasiado largo para desarrollar el árbol de juego completo en el examen, utilizaremos la siguiente función para la evaluación de las configuraciones:  $f'(n) = \text{Suma de los tamaños de los grupos de X mayores que } 1 - \text{Suma de los tamaños de los grupos de O mayores que } 1$ . Si la ficha + está entre dos fichas iguales, éstas no forman un grupo. Por ejemplo, la evaluación de la configuración XOXOO+XXO sería  $(2 - 2) = 0$ , la evaluación de la configuración XOOOXO+OO sería  $(0 - 3 - 2) = -5$ .



a) Representa el árbol de juego y sobre éste utiliza el **algoritmo MINIMAX** para evaluar el primer movimiento que debería hacer el jugador MAX. Haz la exploración hasta **el nivel 3** (dos jugadas de MAX y una de MIN). Aplica siempre el mismo orden: primero el movimiento hacia la izquierda y después el movimiento hacia la derecha.

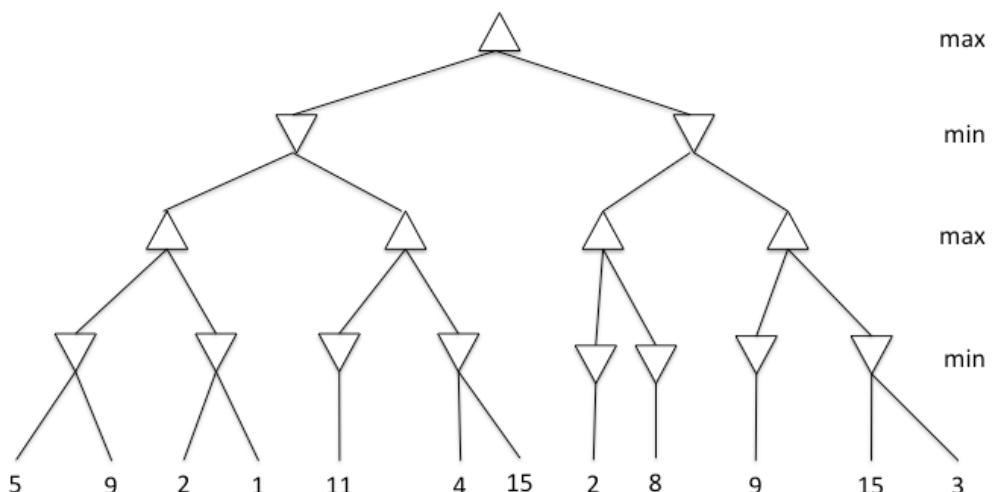
b) En la figura siguiente se muestra el árbol de juego hasta **el nivel 4** sin mostrar los estados, pero con la evaluación de los nodos terminales. Suponiendo que la raíz del árbol es el jugador MAX, muestra los nodos examinados usando la estrategia de **poda  $\alpha$ - $\beta$**  y las ramas podadas asumiendo recorrido de izquierda a derecha. Para cada nodo muestra claramente la evolución de los valores alfa y beta.



### Problema 12.

Suponiendo que la raíz del árbol es el jugador MAX:

1. Muestra los valores obtenidos para todos los nodos en el siguiente árbol de juego
2. Muestra los nodos tendrían que ser examinados usando la estrategia de poda  $\alpha$ - $\beta$  y las ramas que son podadas. Para cada rama podada, explica brevemente porque la poda  $\alpha$ - $\beta$  poda la rama.





### Problema 13.

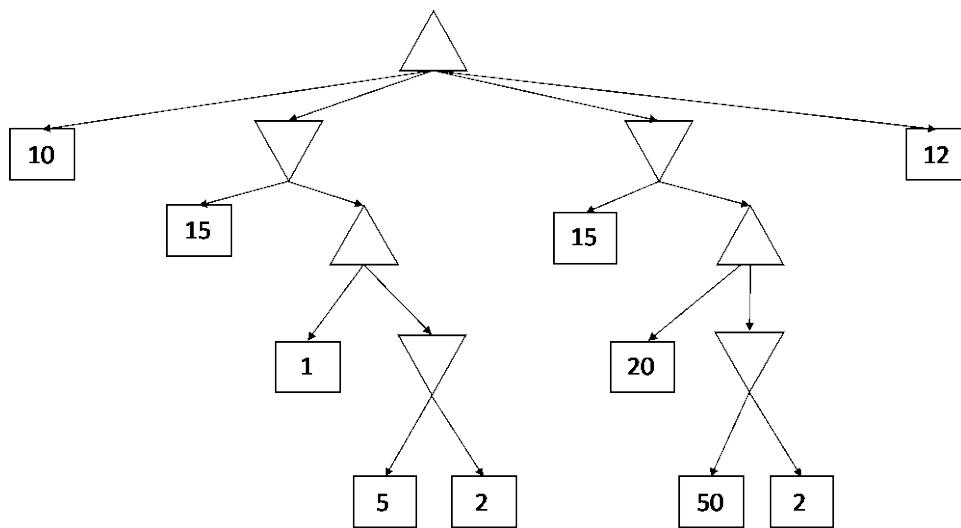
Dada la matriz 3x3 con los valores mostrados, dos jugadores se inventan un absurdo juego consistente en que el primero elige la fila, y el segundo después una columna. El juego finaliza tras los dos movimientos, y el segundo jugador dará la cantidad de euros que corresponda a la fila y columna elegida al segundo. Por ejemplo, el primero elige fila 1, y el segundo la columna 2. El juego finaliza y el segundo jugador le da 1 euro al primero.

8	1	6
3	5	7
4	9	2

1. Dibuja el árbol de juego, y sobre el árbol realiza la evaluación MINIMAX, indicando cual es la rama que elegirá el primer jugador para maximizar la ganancia.
2. Supongamos ahora que el jugador 2 no tiene opción de elegir la columna, y debe tirar un dado en el que, si sale 1 o 4 elige la primera columna, si sale 2 o 5 la segunda y si sale 3 o 6 elige la tercera columna. Dibuja el nuevo árbol de juego, la evaluación EXPECTIMAX, e indica que rama debe elegir el primer jugador para maximizar las ganancias.

### Problema 14.

Suponiendo que la raíz del árbol es el jugador MAX, muestra nodos examinados usando la estrategia de poda  $\alpha\text{-}\beta$  y ramas podadas asumiendo recorrido de izquierda a derecha.

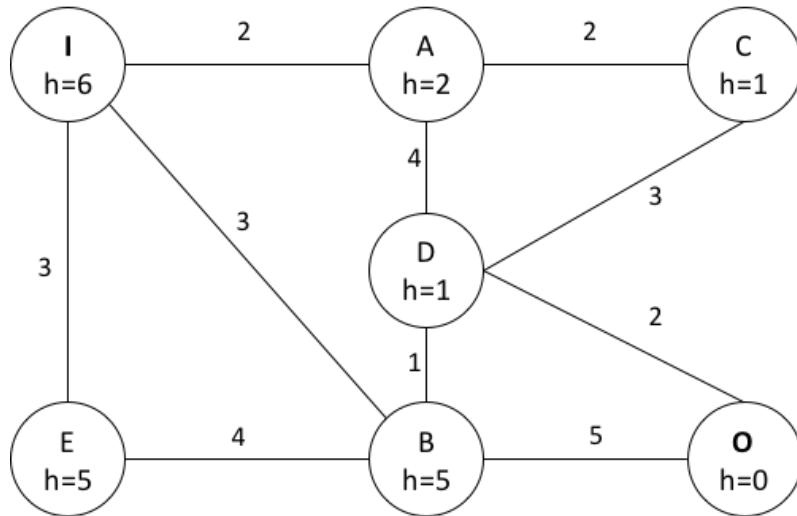




### Problema 15.

Considera el siguiente grafo dibujado que representa el espacio de estados donde **I** es el estado inicial, y **O** es el estado objetivo. Los arcos están etiquetados con el coste de ir de un nodo a otro, y los nodos con  $h$ , la estimación de cuánto cuesta ir del nodo al objetivo.

El agente de búsqueda utiliza la búsqueda A\* en grafo para ir del nodo inicial al objetivo.



1. Debes representar el árbol de exploración, etiquetando cada nodo con el valor de la función de evaluación, y marcando el orden de expansión de los nodos. Además, debes mostrar el estado de la FRONTERA y EXPLORADOS, si procede, en cada paso de la búsqueda. En caso de empate expande en orden alfabético.
2. ¿Es admisible la heurística? Si no lo es indica en qué nodos no lo es y propón un valor para que lo sea. ¿Una heurística admisible garantizaría el camino óptimo (menor coste) con la búsqueda A\* en grafo?
3. ¿Es consistente la heurística reflejada en el grafo? Si no lo es demuestra en qué nodos no se cumple. En el ejemplo propuesto, con la búsqueda A\* en grafo ¿Es posible cambiar la heurística en un solo nodo y garantizar que encontrará el camino óptimo?
4. Nuestro agente de búsqueda utiliza ahora la búsqueda Hill-climbing para ir del estado inicial al objetivo. Utilizando el grafo inicial, en el que todos los valores de la heurística los hemos incrementado en una unidad, debes representar el árbol de exploración, etiquetando cada nodo con el valor de la función de evaluación, y marcando el orden de expansión de los nodos. Además, debes mostrar el estado de la FRONTERA y EXPLORADOS, si procede, en cada paso de la búsqueda. En caso de empate expande en orden alfabético.



### Problema 16.

Considera el siguiente grafo dibujado que representa el espacio de estados donde **I** es el estado inicial, y **O** es el estado objetivo. Los arcos están etiquetados con el coste de ir de un nodo a otro, y los nodos con **h**, la estimación de cuánto cuesta ir del nodo al objetivo.

1. El agente de búsqueda utiliza la búsqueda A\* en grafo para ir del nodo inicial al objetivo. Debes representar el árbol de exploración, etiquetando cada nodo con el valor de la función de evaluación, y marcando el orden de expansión de los nodos. Además, debes mostrar el estado de la FRONTERA y EXPLORADOS, si procede, en cada paso de la búsqueda. En caso de empate expande en orden alfabético. Debes indicar el coste de cada nodo cuando es expandido.
2. ¿Es consistente la heurística reflejada en el grafo? Si no lo es indica algún ejemplo de nodo en el que la heurística no es consistente.
3. Define el valor de la función heurística en cada nodo para que sea consistente.
4. El agente de búsqueda utiliza la búsqueda A\* en árbol para ir del nodo inicial al objetivo con la heurística que has definido en el punto 3. Debes representar el árbol de exploración, etiquetando cada nodo con el valor de la función de evaluación, y marcando el orden de expansión de los nodos. Además, debes mostrar el estado de la FRONTERA y EXPLORADOS, si procede, en cada paso de la búsqueda. En caso de empate expande en orden alfabético.

### Problema 17.

Dada la siguiente figura, encontrar un camino de desde la casilla inicial a la casilla final. Se comienza situado en el cuadrado más a la izquierda (casilla 1), y el objetivo es llegar exactamente a la última casilla (casilla 8).

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

En cada casilla se pueden realizar dos acciones diferentes:

- **Andar** desde la casilla S a la casilla S+1 con **coste 1**.
- **Saltar** de la casilla S a las 2\*S con **coste 2**.

con la **restricción** de que no puede haber más acciones de saltar que de andar para llegar a un estado, y que no se permiten las acciones que mueven más allá de la casilla objetivo.

1. **Describe el problema:** Representación del estado, Estado inicial, Estado final, objetivo, acciones y función de coste. **(0,25 puntos)**
2. **Desarrolla el árbol de búsqueda** que genera el algoritmo de **búsqueda CU** (coste uniforme en grafo). **Debes representar** el árbol de exploración, etiquetando cada nodo con el estado, el valor de la función de evaluación, y marcando el orden de expansión de los nodos. Además, debes mostrar el estado de la FRONTERA y EXPLORADOS, si procede, en cada paso de la búsqueda. En caso de empate expande primero el nodo que estaba antes en la cola. **(1,25 puntos)**



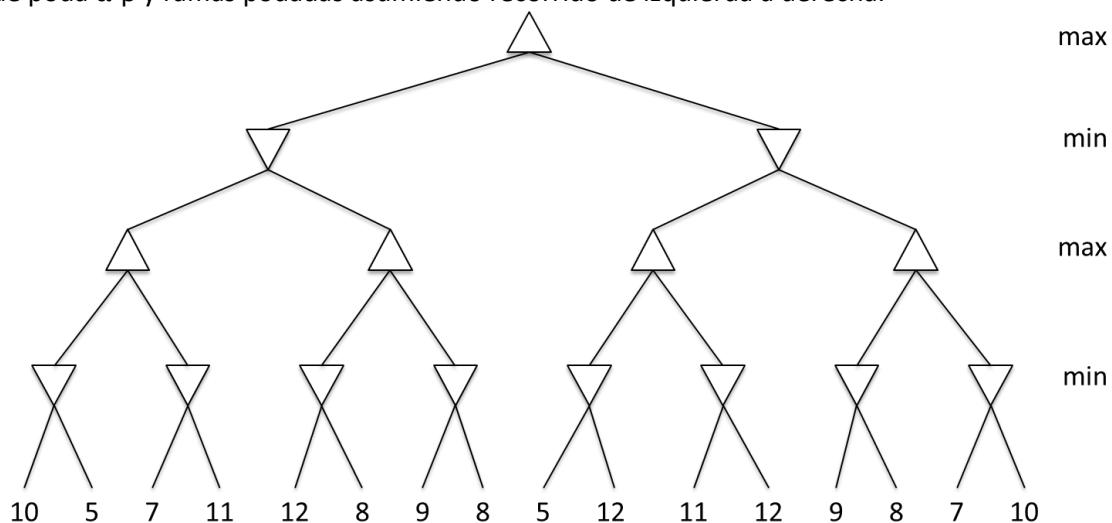
3. Explica razonadamente si la siguiente heurística es admisible:

$$h(S) = 2 * \log_2 \left( \frac{\text{Número Casilla Estado Objetivo}}{\text{Número Casilla Estado } S} \right)$$

**Plantea una heurística admisible no trivial ( $h=0$  no vale) para el problema. Explica razonadamente la admisibilidad de la heurística propuesta. (0,5 puntos)**

### Problema 18.

Suponiendo que la raíz del árbol es el jugador MAX, muestra nodos examinados usando la estrategia de poda  $\alpha$ - $\beta$  y ramas podadas asumiendo recorrido de izquierda a derecha.



### Problema 19.

El juego de “divide o reduce” consiste en empezar con un número entero  $N$ . Por turnos, cada jugador decide dividir  $N$  por dos (división entera), o decrementar  $N$  en una unidad. Comienza jugando MAX, y el jugador que obtiene el valor 0 al operar gana ( pierde el que en su turno de juego se encuentra con un valor 0). Suponiendo que partimos del valor  $N=6$ , Dibuja el árbol de juego, y sobre el árbol realiza la evaluación MINIMAX, indicando cual es la rama que elegirá el primer jugador para ganar.

Si fueras MAX y pudieras elegir otro número alternativo para empezar a jugar entre 1 y 6 ¿Cuál elegirías?



### Problema 20.

Dado el problema planteado en el ejercicio 17 se pide completar el siguiente programa CLIPS. Completa el código necesario deftemplate, reglas, etc. para implementar una búsqueda de CU.

**Funciones útiles:** explode\$, create\$, para guardar el camino. create\$ crea una lista de elementos, y explode\$ genera una cadena con los elementos de una lista. Ejemplo de uso:

```
CLIPS> (implode$ (create$ 1 0 0))
"1 0 0"
```

### Programa CLIPS a completar los recuadros:

```
; -----
; MODULO MAIN (COMPLETAR)
;-----
(defmodule MAIN
  (export deftemplate nodo))
```

```
(deftemplate MAIN::nodo
  ;;; Definición del estado.
  (multislot camino)
  (slot coste (default 0))
  (slot clase (default abierto)))
```

```
(deffacts MAIN::estado-inicial
  ;;; DEFINE Nodo inicial
)
```

```
(defrule MAIN::pasa-el-mejor-a-cerrado-CU
  ;;; IMPLEMENTA CU
)
```

```
;-----
; MODULO OPERADORES (COMPLETAR)
;-----
; Acciones andar y saltar con sus restricciones
(defmodule OPERADORES
  (import MAIN deftemplate nodo))
```

```
(defrule OPERADORES::Andar
  ;;; IMPLEMENTA

)
(defrule OPERADORES::Saltar
  ;;; IMPLEMENTA
)
```



```
; -----
; MODULO RESTRICCIONES (COMPLETAR)
; -----
; Nos quedamos con el nodo de menor coste
; La longitud del camino no es el coste

(defmodule RESTRICCIONES
  (import MAIN deftemplate nodo))

; eliminamos nodos repetidos
(defrule RESTRICCIONES::repeticiones-de-nodo
  ;;; IMPLEMENTA
)

; -----
; MODULO SOLUCION
; -----
;Definimos el modulo solución
(defmodule SOLUCION
  (import MAIN deftemplate nodo))

;Miramos si hemos encontrado la solucion
(defrule SOLUCION::encuentra-solucion
  (declare (auto-focus TRUE))
  ?nodo1 <- (nodo (casilla 8) (camino $?camino)
                    (clase cerrado))
=>
  (retract ?nodo1)
  (assert (solucion $?camino)))

;Escribimos la solución por pantalla
(defrule SOLUCION::escribe-solucion
  (solucion $?movimientos)
=>
  (printout t "La solucion tiene " (- (length $?movimientos) 1)
            " pasos" crlf)
  (loop-for-count (?i 1 (length $?movimientos))
    (printout t "(" (nth ?i $?movimientos) ")" " "))
  (printout t crlf)
  (halt))
```

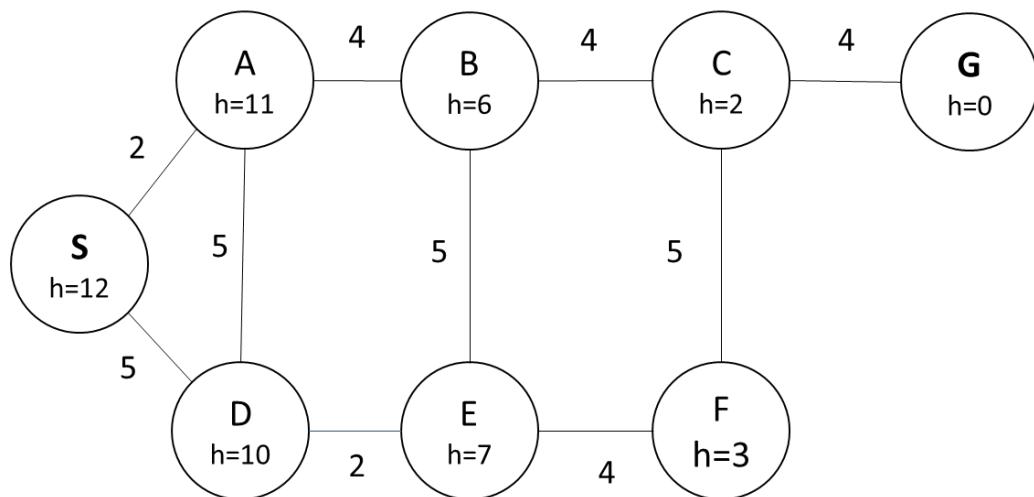


Problema 21.

Considera el grafo dibujado abajo que representa el espacio de estados donde S es el estado inicial, y G es el estado objetivo. Todos los arcos son bidireccionales.

Para cada una de las siguientes estrategias, muestra el camino devuelto, o escribe ninguno si no devuelve ningún camino solución. En caso de empates, asume que se expanden por orden alfabético. Pinta los árboles de búsqueda, representando el orden de expansión de nodos y muestra el estado de la FRONTERA y EXPLORADOS, si procede, en cada ciclo de la búsqueda.

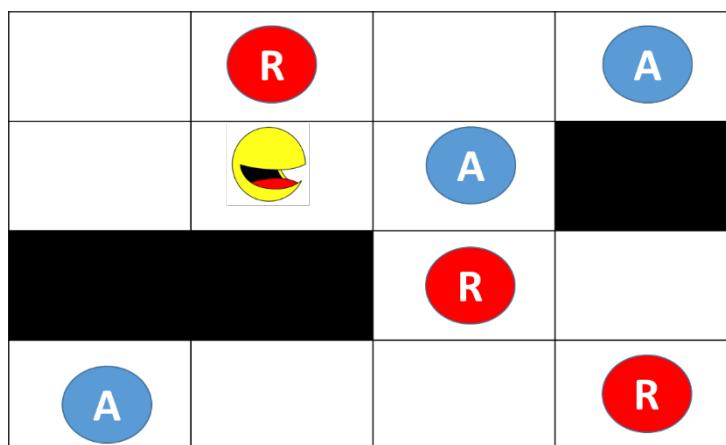
- (a) [0,5 ptos] Búsqueda local (Escalada, buscando el mínimo).
- (b) [0,5 ptos] Búsqueda en grafo A\*.
- (c) [0,5 ptos] ¿Es la heurística representada admisible?  
¿Es la heurística representada consistente?  
Explica tus respuestas.





### Problema 22.

El comecocos representado en el tablero puede comer dos clases de piezas de fruta representadas por los colores rojo (R) y azul (A). El objetivo del comecocos es saborear los dos tipos de frutas: Una vez que ha comido una fruta de cada tipo el juego finaliza (aunque el comecocos podría comer más de una pieza de cada fruta antes de que el juego acabe). Los cuatro movimientos posibles son arriba, abajo, izquierda y derecha. Debe moverse en todos los turnos. No es posible cruzar obstáculos (casillas negras). Se considera que la fruta que ocupa una casilla es comida una vez que el comecocos alcanza la casilla.



Se pide:

1. Define una **representación del problema**, especificando una representación del estado, estado inicial, estados finales, así como operador(es) y su coste. [0,5 ptos]
2. Dibuja el **árbol de búsqueda**, representando el orden de expansión de nodos y muestra el estado de la FRONTERA y EXPLORADOS, si procede, en cada ciclo de la **búsqueda en profundidad en grafo**. [0,25 ptos]
3. Para cada una de las siguientes heurísticas indica si es admisible o no [0,25 ptos]
  - a) El número de frutas restantes
  - b) La menor distancia de Manhattan a una de las frutas restantes.
  - c) La máxima distancia de Manhattan entre cualquiera de las dos frutas restantes.
  - d) La mínima distancia de Manhattan entre cualesquiera dos frutas de color opuesto.
4. Define una **función heurística diferente de las anteriores** que sea admisible. [0,5 ptos]

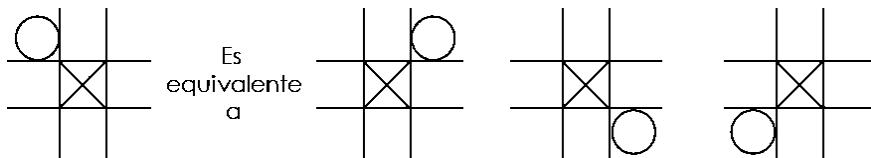


### Problema 23.

El juego del TRES EN LÍNEA, es un juego de dos jugadores (X e Y) que marcan alternadamente los espacios de un tablero de 3x3, empezando primero el jugador X. Gana el que logra colocar sus tres fichas en línea primera, y se empata si se rellenan todas las casillas sin que haya tres en línea. Se define  $X_n$  como el número de filas, columnas o diagonales con exactamente  $n$  Xs, y sin ninguna O. De la misma forma se define  $O_n$ , como el número de filas, columnas o diagonales con exactamente  $n$  Os, y ninguna X. La función de utilidad asigna +1 a cualquier posición con  $X_3 = 1$  y -1 a cualquier posición con  $O_3 = -1$ . Todas las demás posiciones terminales tienen el valor 0. *Para las posiciones no terminales*, utilizaremos la siguiente función de evaluación:  $\text{Eval}(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$ .

1. Muestra el árbol de juego desde la posición inicial con el tablero vacío hasta la profundidad 2 (Es decir, una X y una O en el tablero), teniendo en cuenta la simetría. [0,5 ptos].

Por ejemplo, las siguientes posiciones son equivalentes y sólo aparecerá la primera representándolas en el árbol de juego.



2. Marca en el árbol la evaluación de todas las posiciones en profundidad 2 con la función de evaluación dada. [0,5 ptos]
3. Realiza sobre el árbol la evaluación MINIMAX, indicando que rama elegirá el primer jugador X (MAX), para ganar. [0,5 ptos]
4. Realiza la poda alfa-beta sobre el árbol de juego, y marca rodeando con un círculo los nodos que no serán evaluados en tu árbol de juego con la poda. [0,5 ptos]



#### Problema 24.

La compañía “*Magacén*” utiliza empresas locales para transportar mercancías a su destino. Para una mercancía concreta cuenta con tres tramos que pueden realizarse por 3 distintas empresas. El reparto debe realizarse antes de 48 horas, y como política de la compañía nunca se contrata a la misma empresa de transporte para realizar más de un tramo. En el caso de la mercancía a transportar hay 3 empresas que pueden realizar el transporte en cada uno de los tres tramos necesarios para cubrir el transporte. La tabla siguiente representa los tramos a cubrir por las empresas candidatas, mostrando el coste en euros, y el tiempo empleado en horas.

	Tramo Tr <sub>1</sub>	Tramo Tr <sub>2</sub>	Tramo Tr <sub>3</sub>
Empresa E <sub>1</sub>	2 €, 35 h	3 €, 16 h	2 €, 5 h
Empresa E <sub>2</sub>	5 €, 10 h	5 €, 35 h	4 €, 25 h
Empresa E <sub>3</sub>	10 €, 6 h	8 €, 8 h	6 €, 4 h

El problema consiste en decidir qué tramo se asignará a cada empresa, de modo que se **minimice el coste total respetando las restricciones de tiempo y la política de empresa**. Los técnicos deciden utilizar el algoritmo A\* para resolver el problema.

- (a) Define una **representación “eficiente” del problema**, especificando el conjunto de posibles estados, estado inicial, estados finales, así como operador(es) y su coste (0,5 puntos)
- (b) Define dos buenas **funciones heurísticas h** optimistas (admisibles) para el problema. Comenta como has llegado a definir cada una de las heurísticas. (0,5 puntos)
- (c) **Desarrolla el árbol de búsqueda que genera el algoritmo A\* para las dos heurísticas propuestas.** Indica el orden en el que se expanden los nodos y los valores de f, g y h para cada nodo del árbol de búsqueda. Muestra el estado de la FRONTERA y EXPLORADOS, si procede, en cada ciclo de la búsqueda. En caso de empate en la función heurística, considera que se expanden primero los nodos que has dibujado más a la izquierda en tu árbol de búsqueda. (1,5 punto)



### Problema 25.

El juego Otero4 se juega en un tablero de 4x4 como el mostrado en la figura y consiste en cubrir todo el tablero con fichas blancas (B) y negras (N), colocándolas alternativamente, de manera que gane el que más fichas de su color consiga. Las reglas para jugar al Otero4 son las siguientes: **Para colocar una ficha en el tablero hay que tener en esa fila, columna o diagonal otra ficha propia que encierre fichas contrarias con la que colocamos.** Si no se encierra ficha contraria no es posible colocar ficha. Al colocar la ficha, todas las fichas de color contrario que queden atrapadas entre dos fichas propias cambian su color al nuestro. El tablero comienza con las fichas que aparecen en la **figura 1**.

	N	B	
	B	N	
		B	N

Fig. 1 Estado inicial del juego

4	2	2	4
2	1	1	2
2	1	1	2
4	2	2	4

Fig.2 Valoración de las piezas según posición

- 1) Representa el árbol de juego explorando hasta profundidad 2 (un movimiento de las blancas y otro de las negras).
- 2) Dibuja el árbol de juego, y sobre el árbol realiza la evaluación **MINIMAX**, indicando cual es la rama que elegirá el primer jugador para ganar.
- 3) Utiliza el algoritmo de **poda alfa-beta sobre el árbol de juego** para averiguar qué movimiento deberían hacer las blancas desde la posición inicial (marca en el árbol de juego los valores alfa y beta, dónde se produce poda, cruza una raya cortando las ramas no exploradas y tacha los nodos terminales no visitados).

Para la evaluación de los estados utiliza la tabla de la **figura 2** para asignar valor a cada ficha en función de la posición que ocupa en el tablero. El valor de un estado se obtiene sumando los valores de las fichas propias y restando las del contrario. **Orden de generación de sucesores:** Para expandir los nodos debes intentar los movimientos en el tablero desde la esquina superior izquierda hacia la derecha y siguiendo el orden por filas de arriba a abajo.



Problema 26.

Dada una lista de números, escribe un programa CLIPS que ordene dicha lista. Se pide completar las reglas necesarias para que funcione la siguiente función CLIPS:

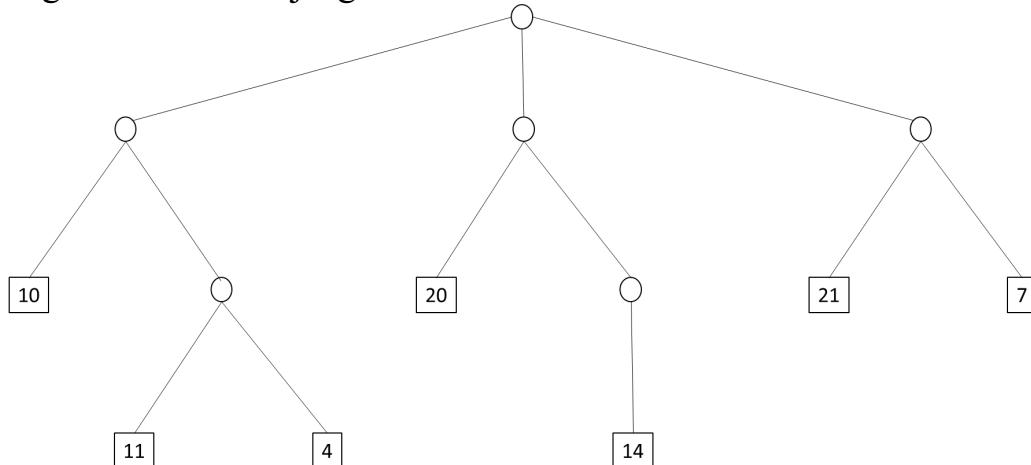
```
(deffunction ordena($?lista)
  (reset)
  (assert (lista ?lista))
  (run))
```

Ejemplo de uso:

```
CLIPS>(ordena 6 4 9 3 23 44 5 3)
VALORES ORDENADOS:(3 3 4 5 6 9 23 44)
CLIPS>
```

Problema 27.

Dado el siguiente árbol de juego:



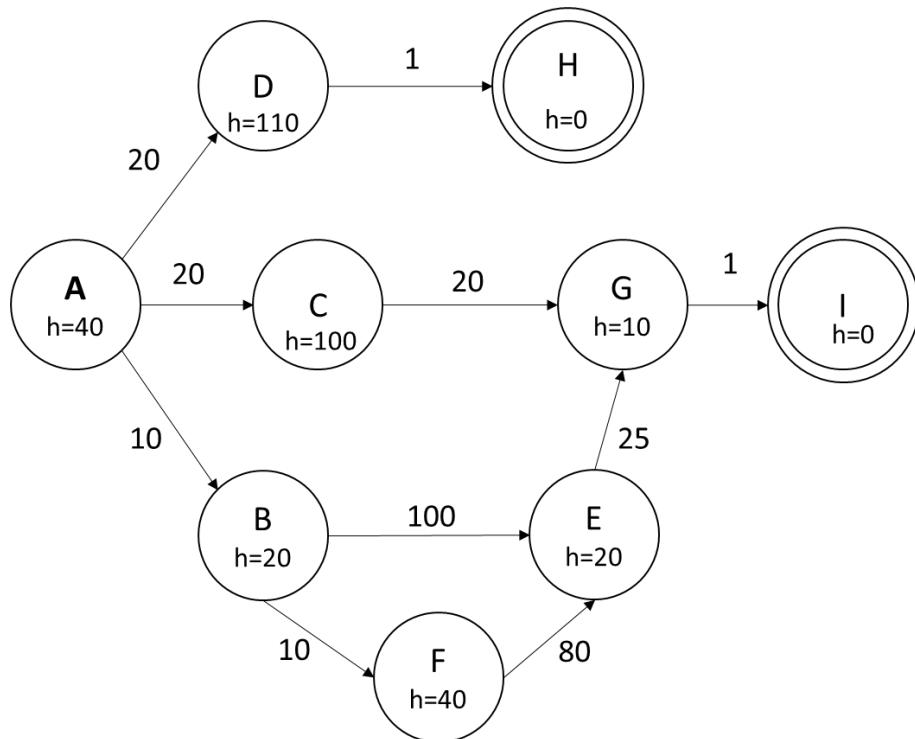
donde los valores numéricos que aparecen en los nodos hoja corresponden a estimaciones de lo prometedoras que son para el jugador MAX las situaciones de la partida representadas por dichos nodos. Aplicar el método de poda alfa-beta al árbol anterior para los siguientes casos:

- Suponiendo que el nodo raíz es un nodo MAX y el recorrido se realiza de izquierda a derecha. ¿Cuál es la decisión o jugada más acertada para MAX en éste caso? ¿Cuál es el valor obtenido por MAX?
- Suponiendo que el nodo raíz es un nodo MIN y el recorrido se realiza de derecha a izquierda. ¿Cuál es la decisión o jugada más acertada para MIN en éste caso? ¿Cuál es el valor obtenido por MIN?



### Problema 28

Considera el grafo dibujado que representa el espacio de estados donde A es el estado inicial, y los nodos H e I son los nodos objetivo. Cada arco está etiquetado con su coste y en cada nodo aparece la estimación de la menor distancia desde ese nodo al objetivo.



Para cada una de las siguientes estrategias de búsqueda, muestra el camino devuelto y su coste, o escribe ninguno si no devuelve ningún camino solución. Pinta los árboles de búsqueda, representando el orden de expansión de nodos y muestra el estado de la FRONTERA y EXPLORADOS, si procede, en cada ciclo de la búsqueda.

- [0,5 ptos] Búsqueda en **anchura en grafo**. En caso de empate expandir en orden alfabético. Test objetivo al crear nodo.
- [0,5 ptos] **Búsqueda local (Escalada**, buscando el mínimo).
- [1 ptos] Búsqueda en **grafo A\***. **En caso de empates**, asume que se expande **primero el último en entrar en la cola, o primero el último que actualiza el coste, para resolver el empate**.
- [0,5 ptos] ¿Es la heurística representada admisible?  
¿Es la heurística representada consistente?  
Explica tu respuesta.



### Problema 29.

El NIM-PARTE-PILAS es un juego con dos jugadores con las siguientes reglas: El juego comienza con una simple pila de monedas iguales. El movimiento de un jugador consiste en dividir una de las pilas **en dos pilas que contengan diferentes números de monedas**. Por ejemplo, si una pila contiene seis monedas, esta se puede dividir en pilas de 5 y 1, o 4 y 2, pero no en pilas de 3 y 3. En el siguiente movimiento de este ejemplo, partiendo de las pilas de 5 y 1 puedes dividir la pila de 5 en pilas de 4 y 1 obteniendo tres pilas (4,1,1), o dividir la pila de 5 en pilas de 3 y 2 obteniendo tres pilas (3,2,1). Partiendo de las pilas de 4 y 2, puede partir la pila de 4 en dos pilas de 3 y 1 obteniendo tres pilas (3,1,2). (No está permitido partir la pila de 4 en dos pilas de 2 monedas).

**El primer jugador que no puede realizar movimiento (porque no puede partir ninguna de las pilas en pilas de valor diferente) pierde.**

Dibujar el árbol de juego completo para esta versión del NIM si el juego **comienza por una pila de 7 monedas y comienza a jugar MIN**. Realiza la evaluación MINIMAX sobre este árbol de juego evaluando 1 un nodo terminal si MAX gana, y -1 si MIN gana. ¿Quién gana y qué movimiento debe realizar para ganar?

### Problema 30.

La siguiente figura representa el estado inicial con cinco monedas colocadas en línea donde la O indica CARA y la C CRUZ:

O	C	O	C	O
---	---	---	---	---

Los **movimientos posibles de coste 1** son dar la vuelta a dos monedas contiguas. El **objetivo** del problema es llegar al estado siguiente:

C	C	C	O	C
---	---	---	---	---

Dada la **función heurística**  $h(n) = \text{número de monedas mal colocadas}$

**Desarrolla el árbol de búsqueda que genera el algoritmo A\* en grafo para la heurística propuestas.** Indica claramente sobre el árbol de búsqueda el orden en el que se expanden los nodos, los valores de f, g y h para cada nodo, el tratamiento de nodos duplicados, el camino obtenido, y su coste. En caso de empate en la función heurística, considera que se expanden primero los nodos que has dibujado más a la izquierda en tu árbol de búsqueda.



### Problema 31.

La siguiente figura representa un tablero de juego con siete casillas que contienen seis monedas colocadas inicialmente como se muestra, donde O indica CARA y la C CRUZ. El jugador **MAX** tiene las fichas marcadas como O y el **MIN** las fichas marcadas como C. Los jugadores pueden mover sus fichas y las del contrario en cada turno

O	C	O		C	O	C
---	---	---	--	---	---	---

Los movimientos posibles son dos:

- **Contigua:** Una moneda se puede desplazar a la casilla libre contigua
- **Salto:** Una moneda puede saltar sobre otra moneda contraria (sólo una) cambiando de signo la ficha contraria, si a continuación hay una casilla vacía. Ejemplo:  
 $OCO\_COC \Rightarrow O\_CCCOC$

**Restricción** sobre las acciones: una jugada de desplazamiento no es válida si deshace la jugada inmediatamente anterior. Por ejemplo, si un jugador pasa de OC\_COC al estado OCC\_COC, el otro jugador no puede volver a pasar a OC\_COC.

El **objetivo de cada jugador** es tener 4 fichas del mismo tipo estrictamente consecutivas

Para la evaluación de los estados se utilizará la siguiente función:

$f'(n)$  = tamaño del mayor grupo de Os consecutivas - tamaño del mayor grupo de Cs consecutivas.

Ejemplos:

$$f'(OCO\_COC) = 1 - 1 = 0$$

$$f'(OCC\_COC) = 1 - 2 = -1$$

Utiliza el algoritmo de **poda alfa-beta** para averiguar cual debería ser el primer movimiento del jugador MAX desde la posición inicial. Haz la exploración hasta el nivel 2 (una jugada de MAX y una de MIN). Aplica siempre el mismo orden: movimientos posibles recorriendo el tablero de izquierda a derecha. Para cada nodo muestra claramente la evolución de los valores alfa y beta. ¿Cuál es el movimiento que debería escoger MAX?



### Problema 32.

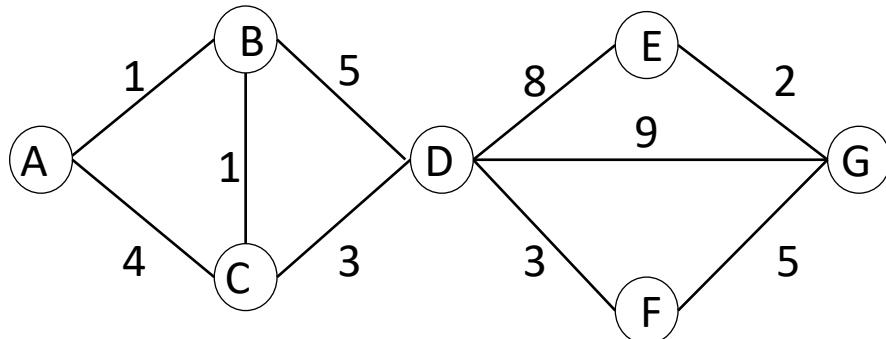
Dos algoritmos de búsqueda son equivalentes si y sólo si expanden los mismos nodos en el mismo orden y devuelven el mismo camino como solución. En este ejercicio se propone el algoritmo de **coste uniforme (CU)** utilizando como coste de las acciones para ir del nodo i al nodo j un valor, al que denominamos  $d_{ij}$ , que puede ser diferente del coste real de las acciones  $c_{ij}$ . Tienes que analizar si el algoritmo de coste uniforme CU utilizando las opciones propuestas de función de coste  $d_{ij}$  en las distintas cuestiones es equivalente a otro algoritmo de búsqueda. Para todas las cuestiones que siguen se supone: 1) los algoritmos de **búsqueda son en grafo**, 2) el valor del coste real  $c_{ij} > 0$  para ir del nodo i al j, 3) sólo hay un nodo objetivo en el problema, 4) en caso de empate se resolverán alfabéticamente asumiendo que los nodos tienen como nombre una letra o cadena de caracteres, y 5) las heurísticas son consistentes.

1. Marca todas las opciones de coste  $d_{ij}$  que hacen que ejecutar el algoritmo de **CU** con los costes  $d_{ij}$  propuestos lo hacen equivalente a ejecutar el algoritmo **búsqueda primero en anchura** [0,5 pts]  
  $d_{ij} = 0$         $d_{ij} = a, a > 0$   
  $d_{ij} = 1$         $d_{ij} = a, a < 0$   
  $d_{ij} = -1$        Ninguna de las anteriores
  
2. Marca todas las opciones de coste  $d_{ij}$  que hacen que ejecutar el algoritmo de **CU** con los costes  $d_{ij}$  propuestos lo hacen equivalente a ejecutar el algoritmo **búsqueda primero en profundidad** [0,5 pts]  
  $d_{ij} = 0$         $d_{ij} = a, a > 0$   
  $d_{ij} = 1$         $d_{ij} = a, a < 0$   
  $d_{ij} = -1$        Ninguna de las anteriores
  
3. Marca todas las opciones de coste  $d_{ij}$  que hacen que ejecutar el algoritmo de **CU** con los costes  $d_{ij}$  propuestos lo hacen equivalente a ejecutar el algoritmo **búsqueda Coste Uniforme** con el coste real  $c_{ij} > 0$  [0,5 pts]  
  $d_{ij} = c_{ij}^2$         $d_{ij} = c_{ij} + a, a > 0$   
  $d_{ij} = 1/c_{ij}$         $d_{ij} = a.c_{ij} + b, a > 0, b > 0$   
  $d_{ij} = a.c_{ij}, a > 0$        Ninguna de las anteriores
  
4. Marca todas las opciones de coste  $d_{ij}$  que hacen que ejecutar el algoritmo de **CU** con los costes  $d_{ij}$  propuestos lo hacen equivalente a ejecutar el algoritmo **búsqueda A\*** con el coste real  $c_{ij} > 0$  y la función heurística  $h$  [0,5 pts]  
  $d_{ij} = a.h(i), a > 0$         $d_{ij} = c_{ij} + h(i) - h(j)$   
  $d_{ij} = a.h(j), a > 0$         $d_{ij} = c_{ij} + h(j) - h(i)$   
  $d_{ij} = c_{ij} + h(i)$        Ninguna de las anteriores  
  $d_{ij} = c_{ij} + h(j)$



## SOLUCIONES

Problema 1.



Posibles soluciones:

Algoritmo de Búsqueda	A-B-D-G	A-C-D-G	A-B-C-D-F-G
Búsqueda en profundidad	X	X	X
Búsqueda en anchura	X	X	
Búsqueda de coste uniforme			X

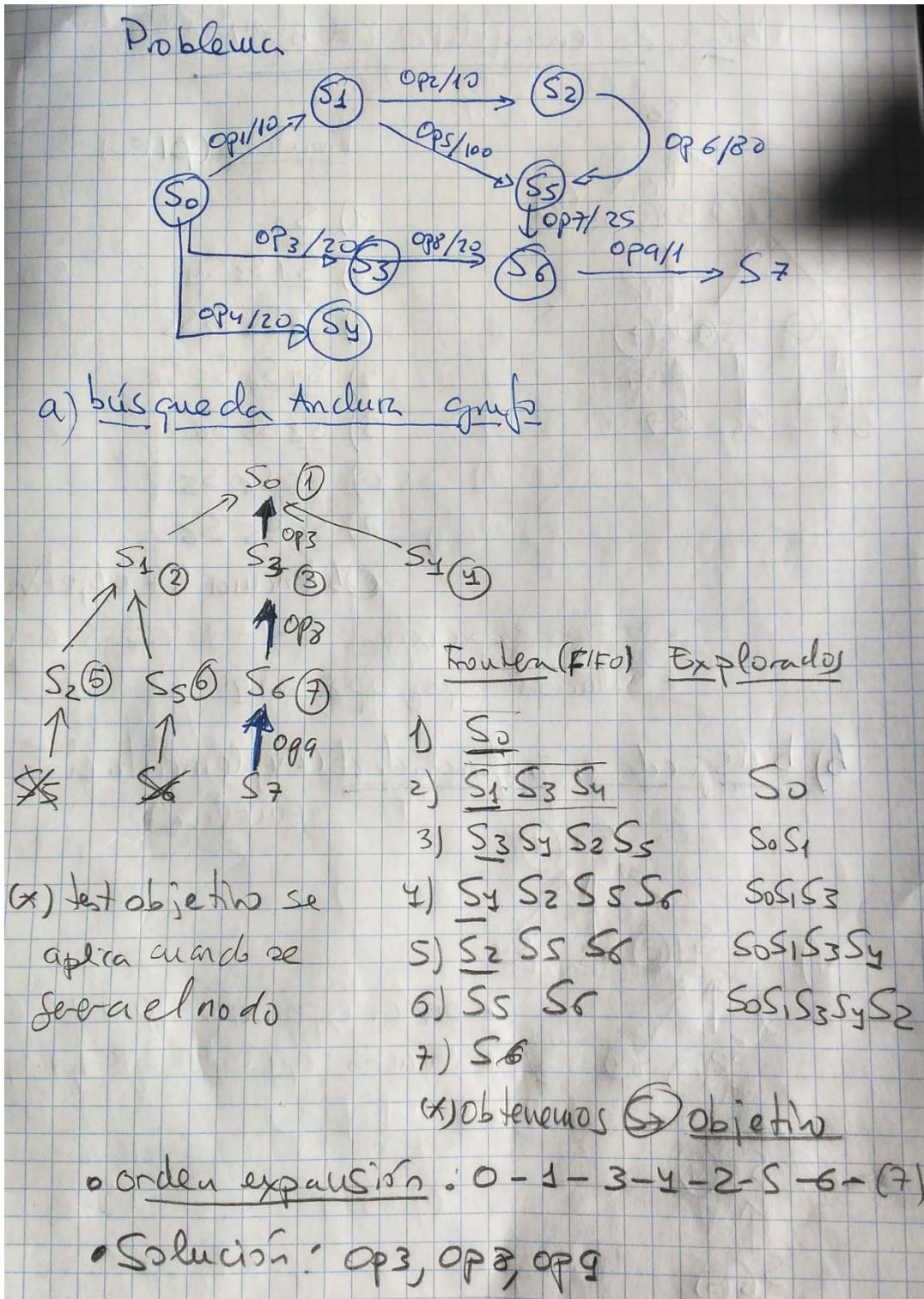
Los distintos caminos dependen de cómo se resuelven los empates:

- DFS cualquier camino
- BFS: los más cortos
- CU: el óptimo



Duración total del examen: 2 horas y media

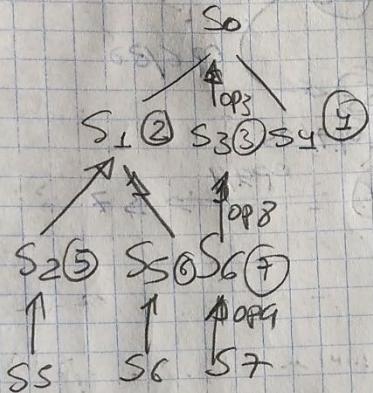
Problema 2.





Duración total del examen: 2 horas y media

### búsqueda en anchura (carbal)



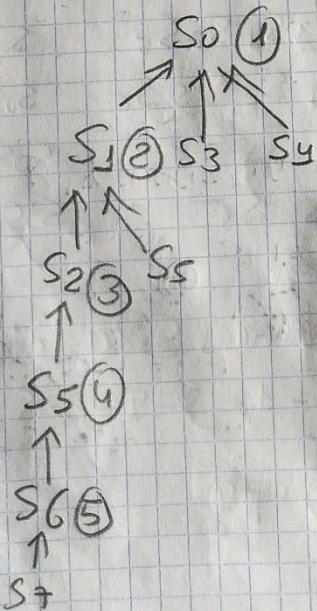
### Frontier (FIFO)

- 1) S<sub>0</sub>
- 2) S<sub>1</sub> S<sub>3</sub> S<sub>4</sub>
- 3) S<sub>3</sub> S<sub>4</sub> S<sub>2</sub> S<sub>5</sub>
- 4) S<sub>4</sub> S<sub>2</sub> S<sub>5</sub> S<sub>6</sub>
- 5) S<sub>2</sub> S<sub>5</sub> S<sub>6</sub>
- 6) S<sub>5</sub> S<sub>6</sub> S<sub>7</sub>
- 7) S<sub>6</sub> S<sub>7</sub> S<sub>6</sub>

Objetivo S<sub>7</sub> objeto

- orden expansión: 0-1-3-4-2-5-6-(7)
- Solución OP<sub>3</sub>, OP<sub>8</sub>, OP<sub>9</sub>

### búsqueda en profundidad (implementación recursiva)

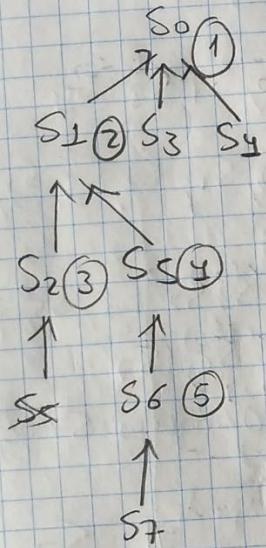


- orden expansión: 0-1-2-5-(7)
- Solución: OP<sub>1</sub>-OP<sub>2</sub>-OP<sub>6</sub>-OP<sub>7</sub>-OP<sub>9</sub>



Duración total del examen: 2 horas y media

- búsqueda en profundidad (grafo)



Frontera (LIFO) Explorados

- 1) S<sub>0</sub>
- 2) S<sub>1</sub> S<sub>3</sub> S<sub>4</sub>      S<sub>0</sub>
- 3) S<sub>2</sub> S<sub>5</sub> S<sub>3</sub> S<sub>4</sub>      S<sub>0</sub> S<sub>2</sub>
- 4) S<sub>5</sub> S<sub>3</sub> S<sub>4</sub>      S<sub>0</sub> S<sub>2</sub>
- 5) S<sub>6</sub> S<sub>3</sub> S<sub>4</sub>      S<sub>0</sub> S<sub>2</sub> S<sub>5</sub>

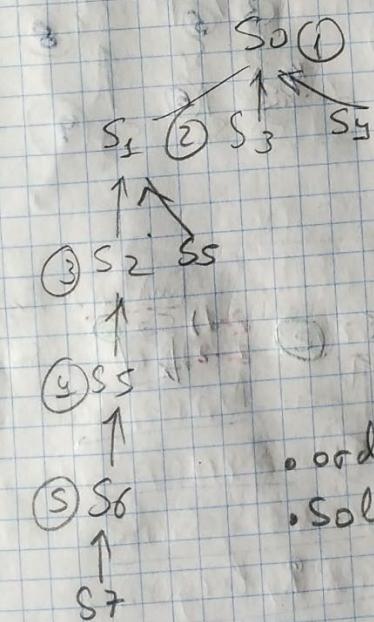
Obtenemos S<sub>7</sub> Objetivo

• orden expansión: 0 - 1 - 2 - 5 - 6 - (7)

• Solución: op<sub>1</sub> - op<sub>3</sub> - op<sub>7</sub> - op<sub>9</sub>

- búsqueda en profundidad (árbol)

Frontera (LIFO)



- 1) S<sub>0</sub>
- 2) S<sub>1</sub> S<sub>3</sub> S<sub>4</sub>
- 3) S<sub>2</sub> S<sub>5</sub> S<sub>3</sub> S<sub>4</sub>
- 4) S<sub>5</sub> S<sub>3</sub> S<sub>4</sub>
- 5) S<sub>6</sub> S<sub>3</sub> S<sub>4</sub>

Obtenemos S<sub>7</sub> Objetivo

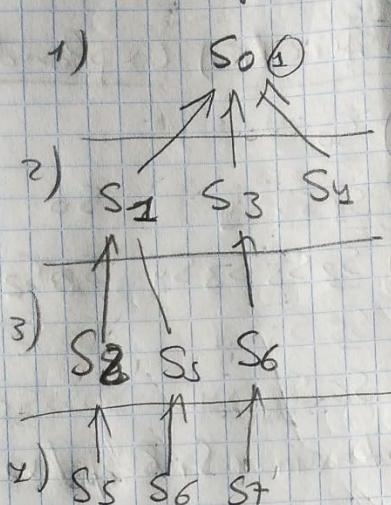
• orden expansión: 0 - 1 - 2 - 5 - 6 - (7)

• Solución: op<sub>1</sub> - op<sub>2</sub> - op<sub>5</sub> - op<sub>7</sub> - op<sub>9</sub>



**Duración total del examen: 2 horas y media**

c) búsqueda e profundización literatura

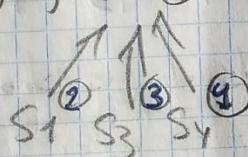


- Nodos visitados por iteración

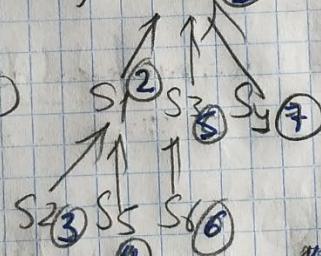
  - 1) S<sub>0</sub>
  - 2) S<sub>0</sub>, S<sub>1</sub>, S<sub>3</sub>, S<sub>4</sub>
  - 3) S<sub>0</sub>, S<sub>1</sub>, S<sub>2</sub>, S<sub>5</sub>, S<sub>3</sub>, S<sub>6</sub>, S<sub>4</sub>
  - 4) S<sub>0</sub>, S<sub>1</sub>, S<sub>2</sub>, S<sub>5</sub>, S<sub>5</sub>, S<sub>6</sub>, S<sub>3</sub>, S<sub>6</sub>, S<sub>7</sub>

Aholes de exploración profunda en Techu

- 1)  $S_9$  2)  $S_0$  3)  $S_0$



- 3) 50 ①



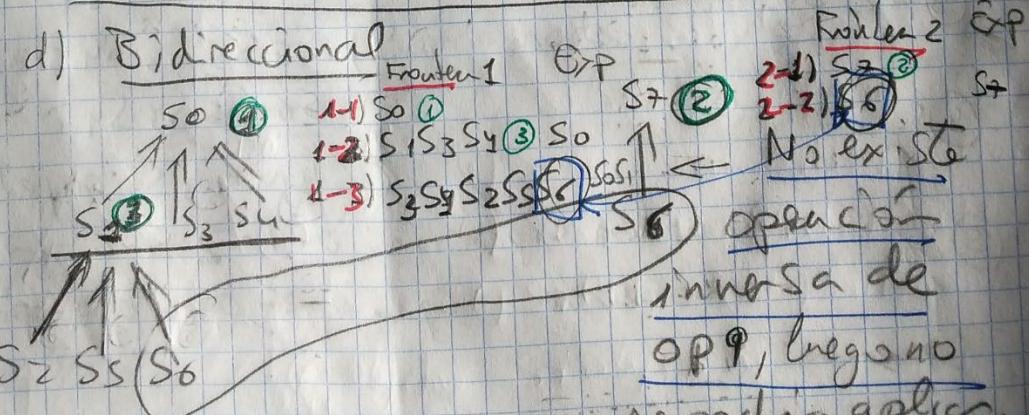
- $$1) \quad \text{So } 3n \text{ (1)}$$



Cada iteración hace una búsqueda

en profundidad (abolo recursiva)

d) Bidireccional



Si existen inversores, el orden de expansión será



Duración total del examen: 2 horas y media

$S_0, S_7, S_1$  y encontráramos en las fronteras el nodo  $S_6$  común a ambas.

c) búsqueda escalada por la máxima pendiente (algoritmo de búsqueda local)

$$h(S_0) = 40$$

$$h(S_1) = ?$$

$$h(S_2) = 40$$

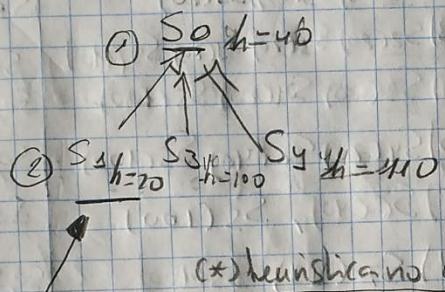
$$h(S_3) = 100$$

$$h(S_4) = 110$$

$$h(S_5) = 20$$

$$h(S_6) = 10$$

$$h(S_7) = 0$$



(\*) heurística no admisible

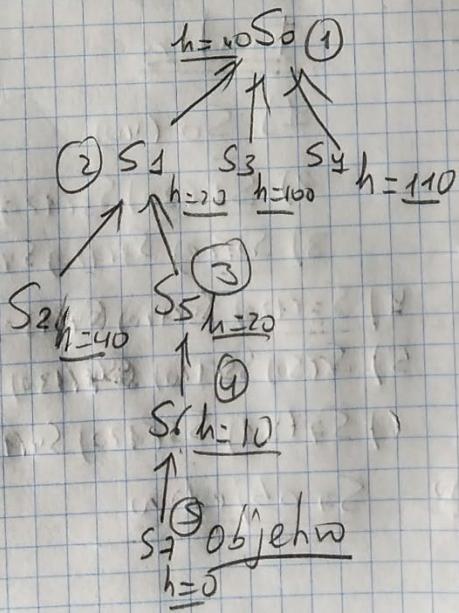
$S_5 \ h=20$  No mejora, así que para y detiene

$S_1$

- Secuencia nodo visitado  $S_0, S_1$

- Solución: —

f) Primero el mejor / razon (árbol)



Traer

$$1) \underline{S_0(40)}$$

$$2) \underline{S_1(20)} S_3(100) S_4(110)$$

$$3) \underline{S_5(20)} S_2(40) S_3(100) S_4(110)$$

$$4) S_6(10) S_2(40) S_3(110) S_4(110)$$

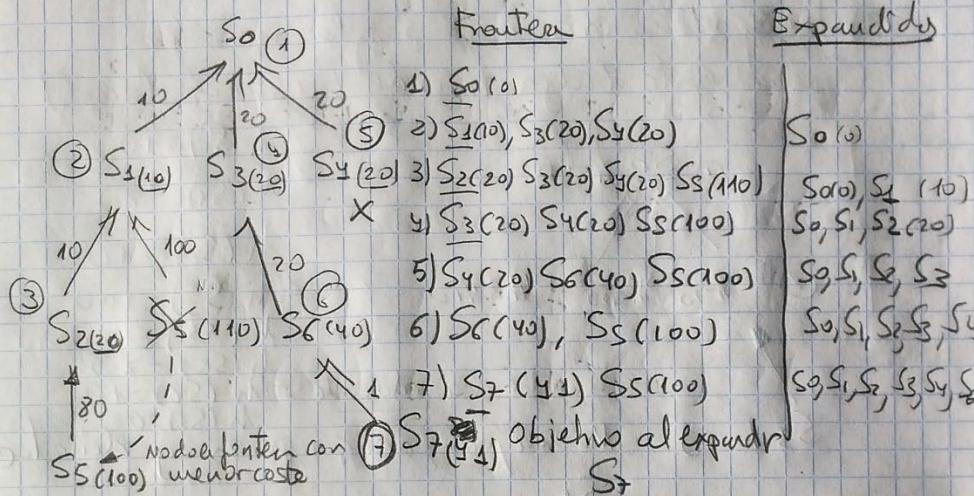
$$5) \underline{S_7(0)} S_2(40) S_3(100) S_4(110)$$

Objetivo



Duración total del examen: 2 horas y media

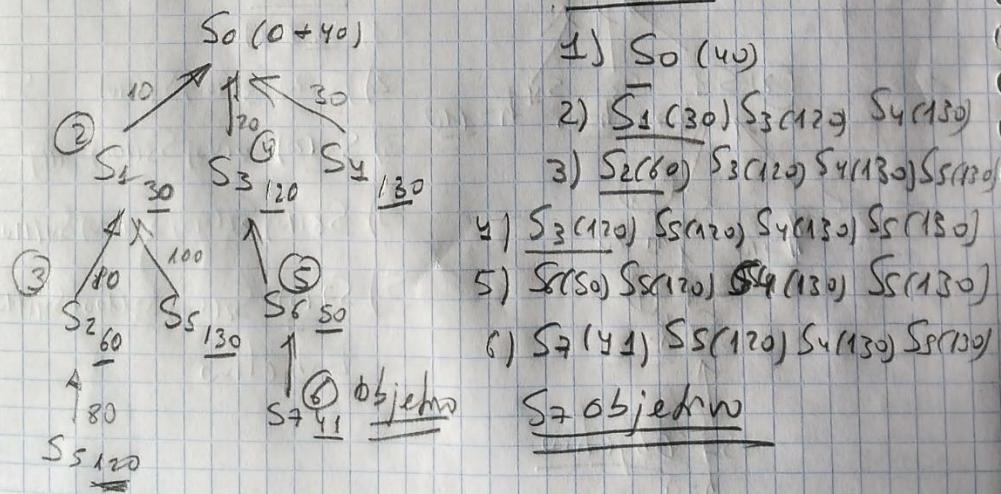
### Coste Uniforme (Grafos)



\* Fijarse que el algoritmo de CU mira nodos con menor coste para actualizar sólo en la frontera, no en expansión.

Puedes ver el razonamiento

### $A^*$ en árbol





Duración total del examen: 2 horas y media

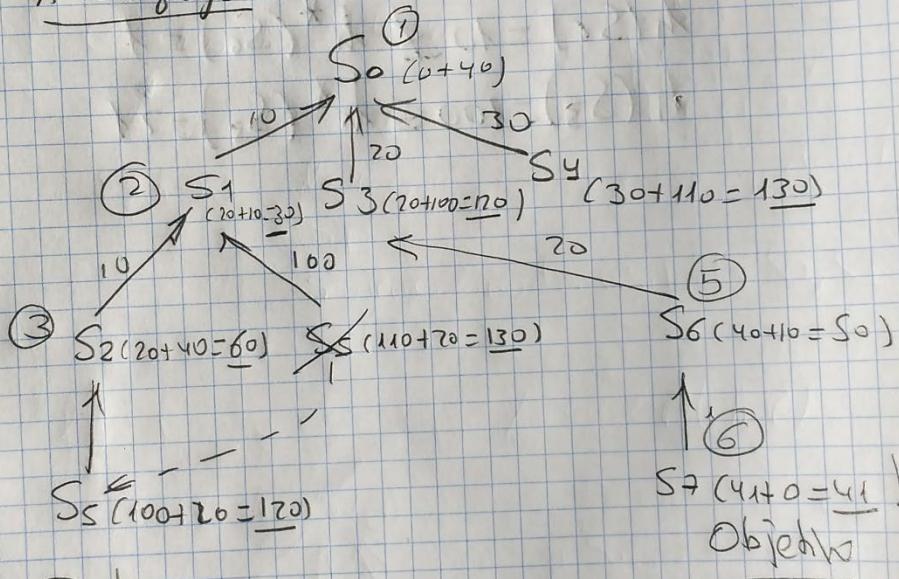
\* La heurística no es admisible.

$$h(S_3) = 100 > h^*(S_3) = 21$$

$$h(S_6) = 10 > h^*(S_6) = 1$$

No hay garantías de encontrar óptimo.  
Pero en este caso se encuentra solución óptima

A\* en grafo



Tareas

1)  $S_0(40)$

2)  $S_1(30), S_3(170), S_4(130)$

(\*) 3)  $S_2(60), S_3(120), S_4(130), S_5(130)$

4)  $S_3(120), S_5(120), S_4(130)$

5)  $S_6(50), S_5(120), S_4(130)$

6)  $S_7(41), S_5(170), S_4(130)$

7) Expansión  $S_7$  objetivo

En A\* en grafo, si la heurística es consistente se expande en orden creciente de f. (No ocurre así porque la heurística no es consistente.)

Ejemplos de que no es consistente:

$$h(S_6) = 10 \not\leq h(S_7) + c(S_6, S_7) = 0 + 1 = 1$$

$$h(S_3) = 100 \not\leq h(S_6) + c(S_3, S_6) = 10 + 20 = 30$$

Ejemplos de que no es admisible:

$$h(S_6) = 10 \not\leq h^*(S_6) = 1$$

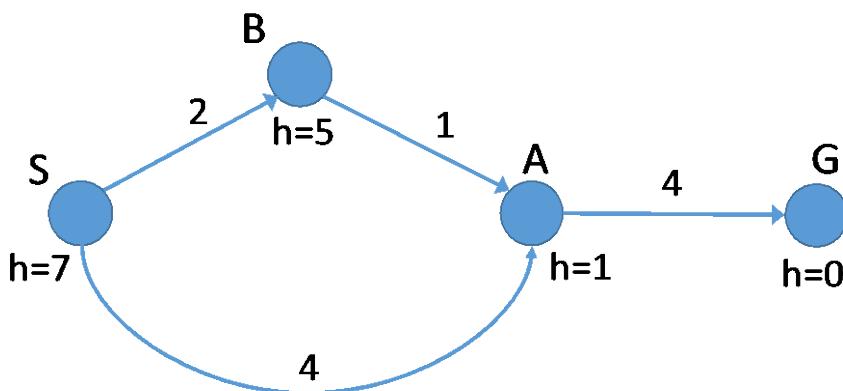
$$h(S_3) = 100 \not\leq h^*(S_3) = 21$$



Duración total del examen: 2 horas y media

### Problema 3

Considera el siguiente grafo dibujado que representa el espacio de estados donde S es el estado inicial, y G es el estado objetivo.

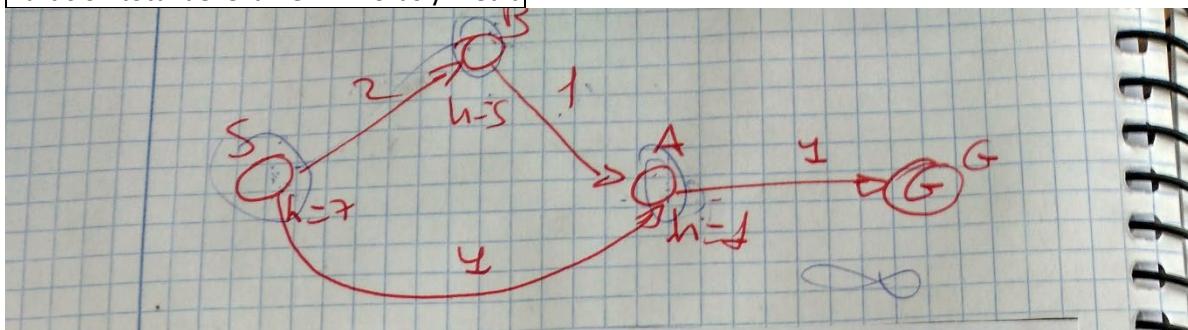


Para cada una de las siguientes estrategias de búsqueda (1-3), muestra el árbol de búsqueda, representando el orden de expansión de nodos en el árbol, y el estado de la FRONTERA y EXPLORADOS, si procede, en cada paso de la búsqueda.

1. Búsqueda Coste Uniforme en Grafo.
2. Búsqueda A\* en árbol
3. Búsqueda A\* en grafo
4. ¿Se obtiene la solución óptima en todos los casos anteriores? Explica tu respuesta para cada caso.
5. Manteniendo los valores de la función heurística para  $h(S)=7$ ,  $h(B)=5$  y  $h(G)=0$ , ¿Cuál es el **menor** valor de  $h(A)$  para que la heurística sea consistente? ¿Cuál es el **mayor** valor de  $h(A)$  para que la heurística sea admisible?
6. En el grafo anterior, ¿La búsqueda en profundidad es completa incluso si el coste del arco S-A es cero?



Duración total del examen: 2 horas y media



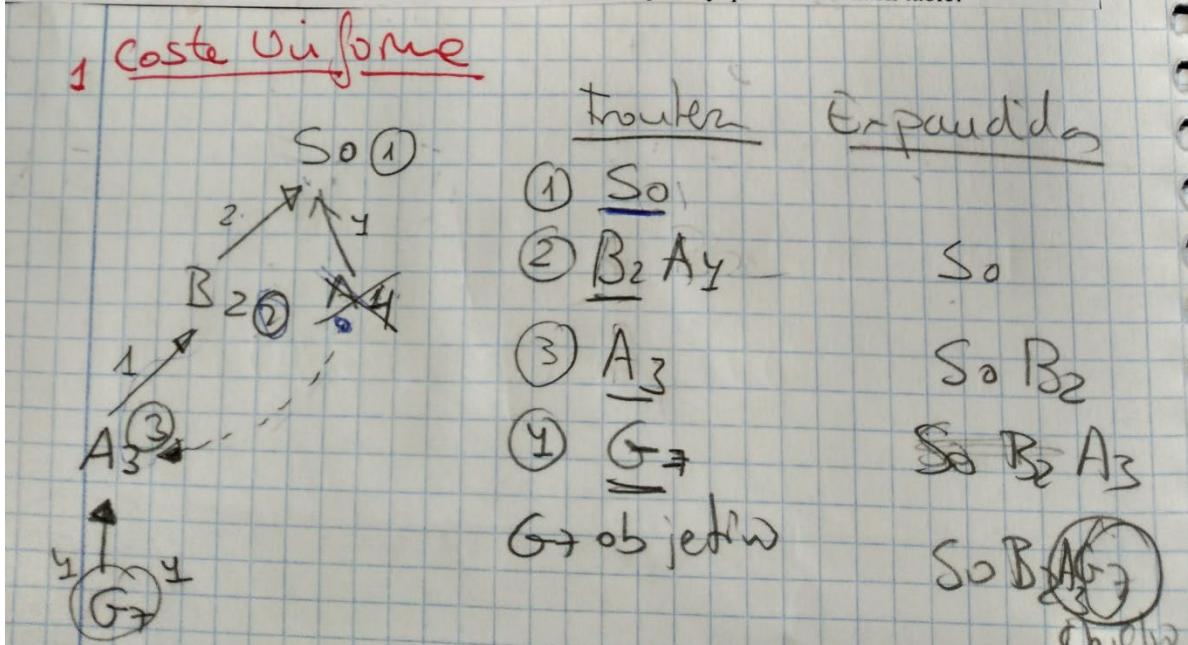
```

function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node  $\leftarrow$  a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier  $\leftarrow$  a priority queue ordered by PATH-COST, with node as the only element
  explored  $\leftarrow$  an empty set
  loop do
    if EMPTY?(frontier) then return failure
    node  $\leftarrow$  POP(frontier) /* chooses the lowest-cost node in frontier */
    if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
    add node.STATE to explored
    for each action in problem.ACTIONS(node.STATE) do
      child  $\leftarrow$  CHILD-NODE(problem, node, action)
      if child.STATE is not in explored or frontier then
        frontier  $\leftarrow$  INSERT(child, frontier)
      else if child.STATE is in frontier with higher PATH-COST then
        replace that frontier node with child

```

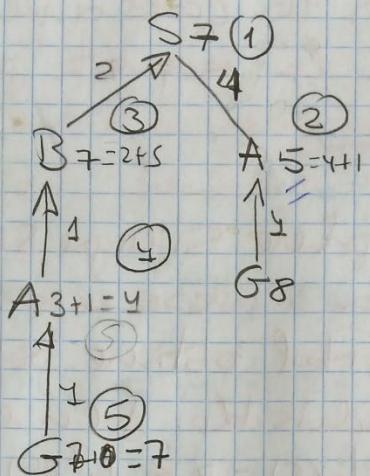
$\rightarrow$  A\*

Figure 3.13 Uniform-cost search on a graph. The algorithm is identical to the general graph search algorithm in Figure ??, except for the use of a priority queue and the addition of an extra check in case a shorter path to a frontier state is discovered. The data structure for *frontier* needs to support efficient membership testing, so it should combine the capabilities of a priority queue and a hash table.



*Duración total del examen: 2 horas y media*

## 2) Buisgredes en át'ool A&K

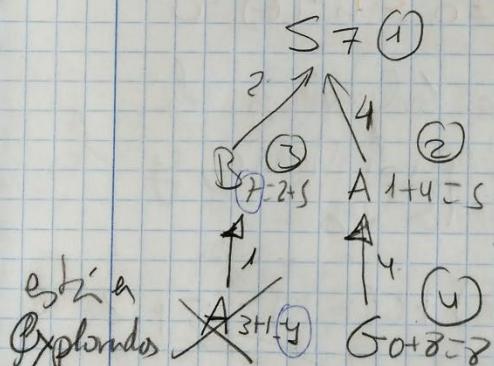


Boulder

- 1) S<sub>7</sub>
  - 2) As B<sub>7</sub>
  - 3) B<sub>7</sub> G<sub>8</sub>
  - 4) A<sub>4</sub> G<sub>8</sub>
  - 5)  $\overbrace{F_7}^{\downarrow}$  G<sub>8</sub>

Objects

### 3) Büssgede en Grub A\*



Troubles

## Experiments

- 1) S<sub>7</sub>  
 2) As B<sub>7</sub> S<sub>7</sub>  
 3) B<sub>7</sub> G<sub>8</sub> S<sub>7</sub>As  
 4) G<sub>8</sub> S<sub>7</sub>AsB<sub>7</sub>  
 ↓  
 Objektiv



Duración total del examen: 2 horas y media

**1.4 Coste Uniforme.** Obtiene siempre la solución óptima en espacio de estados finitos.

**A\* en árbol.** Obtiene la solución óptima si la heurística es admisible, y en este caso lo es  
 $h(S) = 7 \leq h^*(S) = 7$   
 $h(B) = 5 \leq h^*(B) = 5$   
 $h(A) = 1 \leq h^*(A) = 4$   
 $h(G) = 0 \leq h^*(G) = 0$

**A\* en grafo.** Es óptima si la heurística es consistente, y en este caso no lo es. Si fuera consistente, en la búsqueda en grafo del punto 1.3 la función  $f=g+h$  sería siempre creciente en un camino. En la expansión vemos el camino S(7), B(7) y A(4) como f no es creciente.

Para que sea consistente se tendría que cumplir que  $h(n) \leq h(n') + c(n,n')$  para todo sucesor  $n'$  de  $n$ . En los siguientes casos no se cumple:

$$h(S)=7 \not\leq h(A) + c(S,A) = 4+1 = 5$$

$$h(B)=5 \not\leq h(A) + c(B,A) = 1+1 = 2$$

**1.5 Menor valor de  $h(A)$  para que la heurística sea consistente**

$$h(B) = 5 \leq 1 + h(A) \Rightarrow h(A) \geq 4$$

$$h(S) = 7 \leq 4 + h(A) \Rightarrow h(A) \geq 3$$

$$h(A) \leq h(G)+4 = 4$$

por lo tanto  $4 \geq h(A) \geq 4 \Rightarrow h(A) = 4$

**Mayor valor de  $h(A)$  para que la heurística sea admisible**

$0 \leq h(A) \leq h^*(A) = 4 \Rightarrow$  el mayor valor admisible es  $h(A)=4$

**1.5 En el grafo anterior, ¿La búsqueda en profundidad es completa incluso si el coste del arco S-A es cero?**

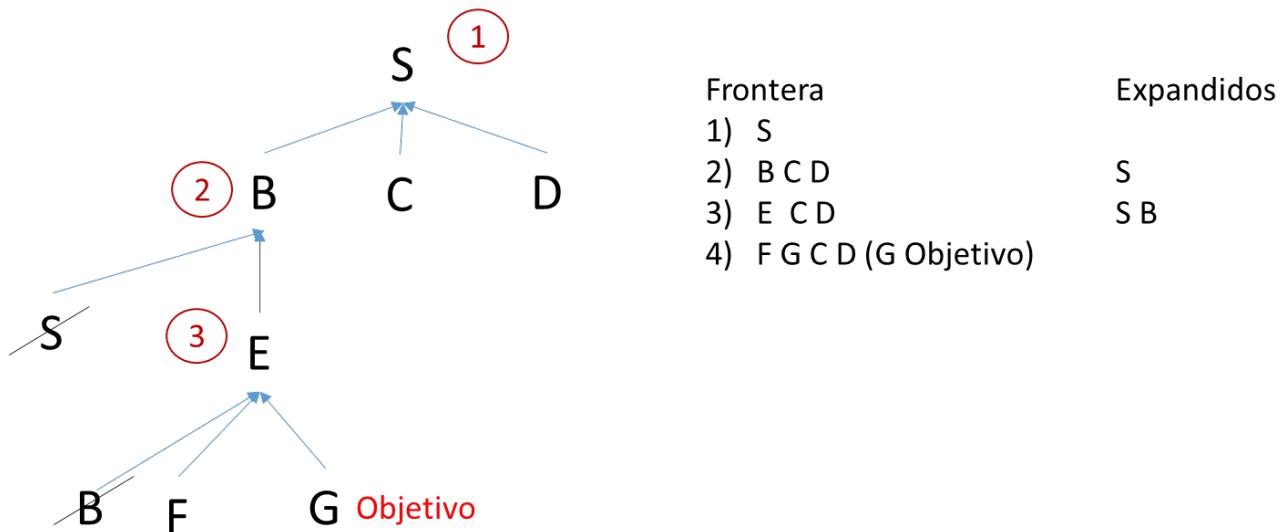
La búsqueda en profundidad es no informada y no considera el coste. Completa quiere decir que encuentra solución, y en el grafo anterior siempre la encontrará. En general la búsqueda en profundidad NO es completa. Pero en espacios de estados finitos (Conexos) y sin ciclos, la búsqueda en profundidad acabará recorriendo todo el espacio de estados, y por lo tanto si es completa como en el ejemplo propuesto.



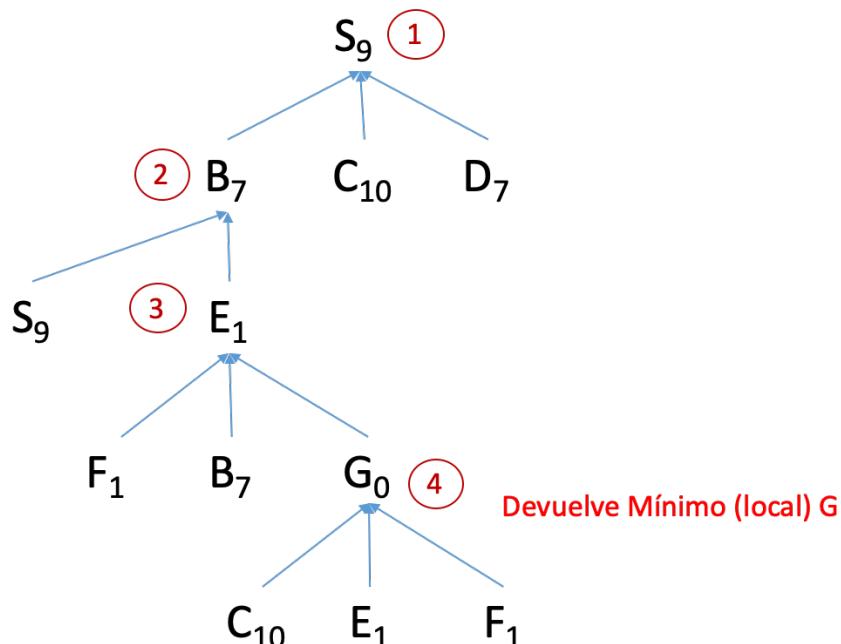
Duración total del examen: 2 horas y media

Problema 4.

(a) Búsqueda en grafo. Test al generar nodo.



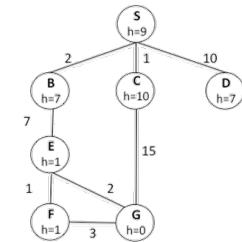
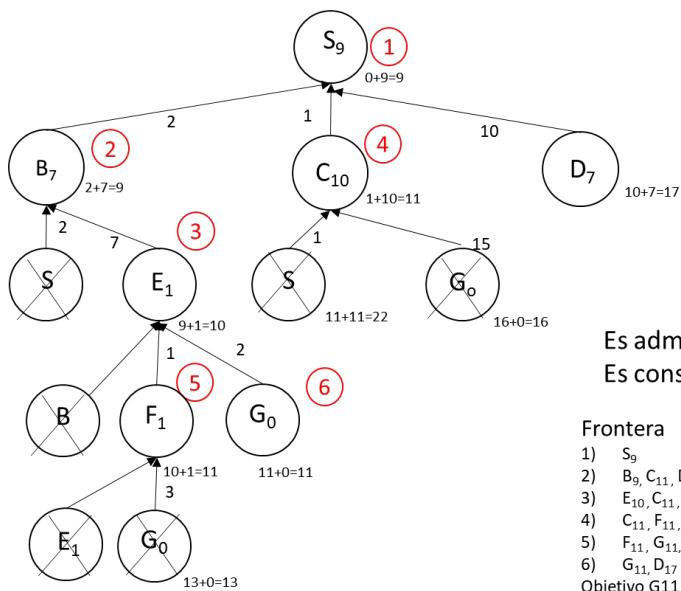
(b) Hill-Climbing. Escalada, buscando el mínimo..





Duración total del examen: 2 horas y media

(c) Búsqueda A\* en grafo. En caso de empate orden alfabético.



Es admisible porque ningun  $h(n)$  sobreestima

Es consistente porque para todo nodo  $h(n)=h(n') + c(n,n')$

#### Frontera

- 1)  $S_9$
- 2)  $B_9, C_{11}, D_{17}$
- 3)  $E_{10}, C_{11}, D_{17}$
- 4)  $C_{11}, F_{11}, G_{11}, D_{17}$
- 5)  $F_{11}, G_{11}, D_{17}$
- 6)  $G_{11}, D_{17}$
- Objetivo  $G_{11}$

#### Explorados

- 1)  $S_9$
- 2)  $S_9, B_9$
- 3)  $S_9, B_9, E_{10}$
- 4)  $S_9, B_9, E_{10}, C_{11}$
- 5)  $S_9, B_9, E_{10}, C_{11}, F_{11}$
- 6)  $S_9, B_9, E_{10}, C_{11}, F_{11}, G_{11}$
- 7)  $S_9, B_9, E_{10}, C_{11}, F_{11}, G_{11}$

Expandido Objetivo  $G_{24}$



Duración total del examen: 2 horas y media

Problema 5.

- (a) i)  $h$  debe ser admisible

$0 \leq h(D) \leq 3$  (si voy desde  $D$  a  $G$  pasando por  $E$ )

$0 \leq h(D) \leq 5$  (si voy desde  $D$  a  $G$  pasando por  $A$ )

Por lo tanto, para que  $h$  sea admisible  $0 \leq h(d) \leq 3$

- ii)  $h$  debe ser admisible y consistente:

Admisible  $0 \leq h(d) \leq 3$

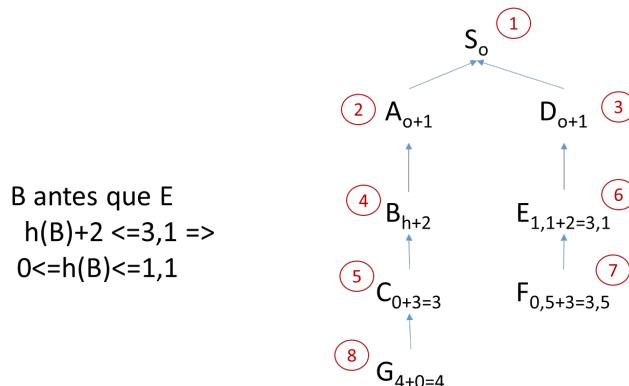
Consistente:

$h(E) \leq h(F) + 1 \Rightarrow 0 \leq h(E) \leq 0,5 + 1 = 1,5$

$0 \leq h(D) \leq h(E) + 1 \leq 1,5 + 1 = 2,5$  Suponiendo que cogemos el máximo valor de  $h(E)$  para que sea consistente.

- (b) i)  $B$  se expande antes que  $E$  que se expande antes que  $F$ . ( $A^*$  árbol,  $h(B)$  admisible)

$$0 \leq h(B) \leq 1,1$$



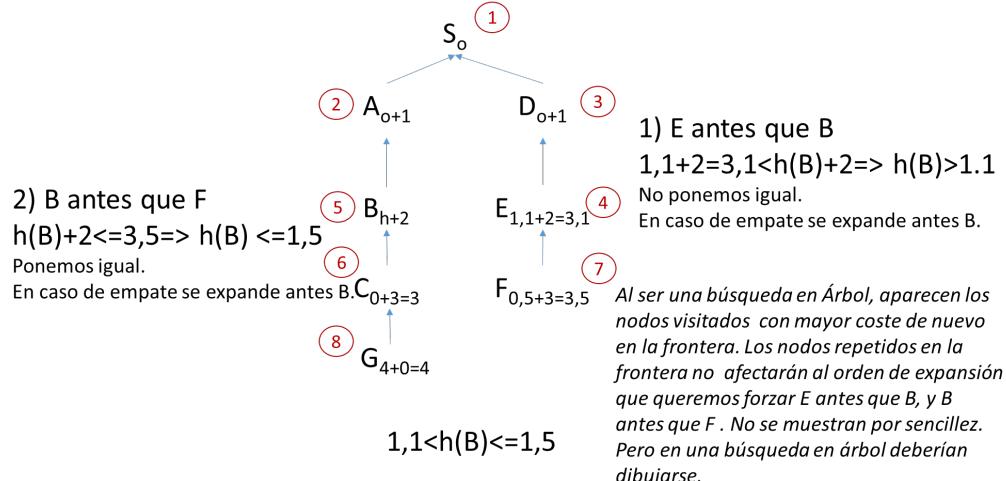
Al ser una búsqueda en Árbol, aparecen los nodos visitados con mayor coste de nuevo en la frontera. Los nodos repetidos en la frontera no afectarán al orden de expansión que queremos forzar  $B$  antes que  $E$ . No se muestran por sencillez. Pero en una búsqueda en árbol deberían dibujarse.

- (c) ii)  $E$  se expande antes que  $B$ , que se expande antes que  $F$ . ( $A^*$  árbol,  $h(B)$  admisible)



Duración total del examen: 2 horas y media

$$1,1 \leq h(B) \leq 1,1$$

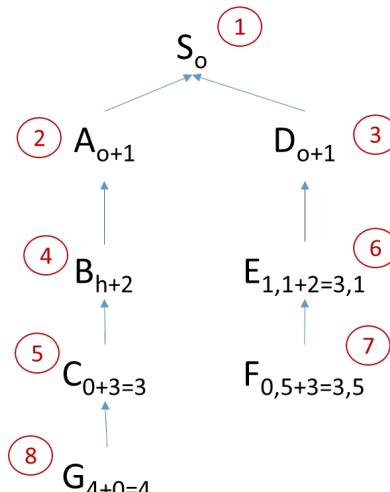


- (d) i) B se expande antes que E que se expande antes que F. ( $A^*$  grafo,  $h(B)$  consistente)  
 $0 \leq h(B) \leq 1$

Se pide  $h(B)$  consistente  
 $h(B) \leq h(C) + 1 = 0 + 1$   
 $h(B) \leq h(A) + 1 = 0 + 1$   
 $\Rightarrow h(B) \leq 1$

B antes que E  
 $2 + h(B) \leq 3,1$   
 $h(B) \leq 1,1$

Por lo tanto  $0 \leq h(B) \leq 1$

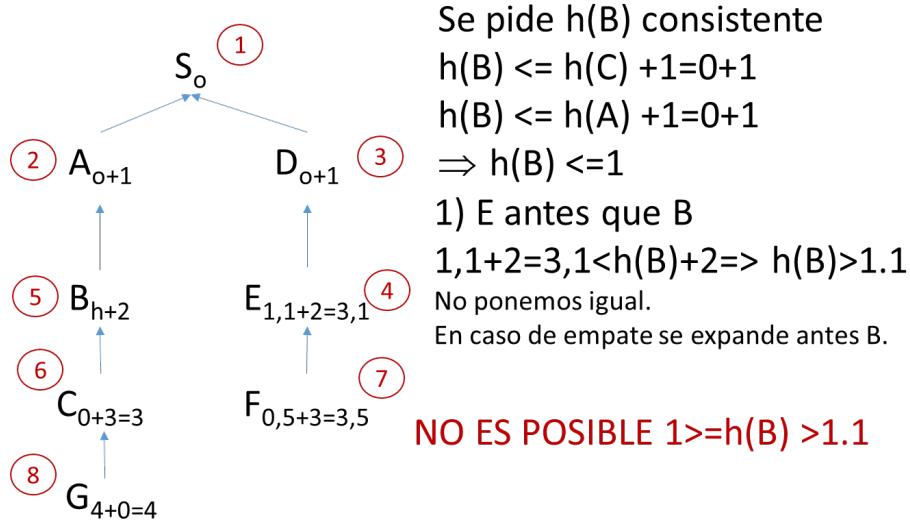


- ii) E se expande antes que B, que se expande antes que F. ( $A^*$  grafo,  $h(B)$  consistente)



Duración total del examen: 2 horas y media

NO ES POSIBLE





Duración total del examen: 2 horas y media

### Problema 6

(a) Representación del Problema:

- **Estado:** Lista de empresas a las que se adjudican por orden las obras.

Por ejemplo {E3, E1} representa E3 se le adjudica O1, y E1 se le adjudica obra O2.

**Estado inicial {}**

**Estado Final:** Cualquier lista de longitud 4 sin repetición de empresas.

**Acciones:** Añadir empresa a lista **Añadir (Ei, lista)**

Precondición:  $(Ei \notin \text{lista}) \wedge (\text{longitud}(\text{lista}) < 4)$

Postcondición:  $\text{lista} = \text{lista} \circ \{Ei\}$  ( $\circ$  concatenación de listas)

**Coste:** Suma de costes de las obras realizadas por las empresas en la lista

$$\text{coste}(\text{lista}) = \sum_{i=1}^{\text{longitud}(\text{lista})} \text{Costes}[\text{lista}(i)][i]$$

Siendo lista(i) el elemento i-ésimo de la lista y Costes la matriz

Con primer índice indicando la fila de la empresa y el segundo índice la columna de la obra.

	Obra O <sub>1</sub>	Obra O <sub>2</sub>	Obra O <sub>3</sub>	Obra O <sub>4</sub>
Empresa E <sub>1</sub>	2	3	2	4
Empresa E <sub>2</sub>	5	5	4	5
Empresa E <sub>3</sub>	6	5	4	3
Empresa E <sub>4</sub>	10	8	6	6

(b) Heurística: Suma de costes mínimos de las obras no asignadas. Relajamos el problema permitiendo repetir empresas entre las pendientes de adjudicar obras.

$$h(\text{lista}) = \sum_{i=\text{longitud}(\text{lista})+1}^4 \text{mínimo\_columna}(i, \text{Costes}), \text{ donde}$$

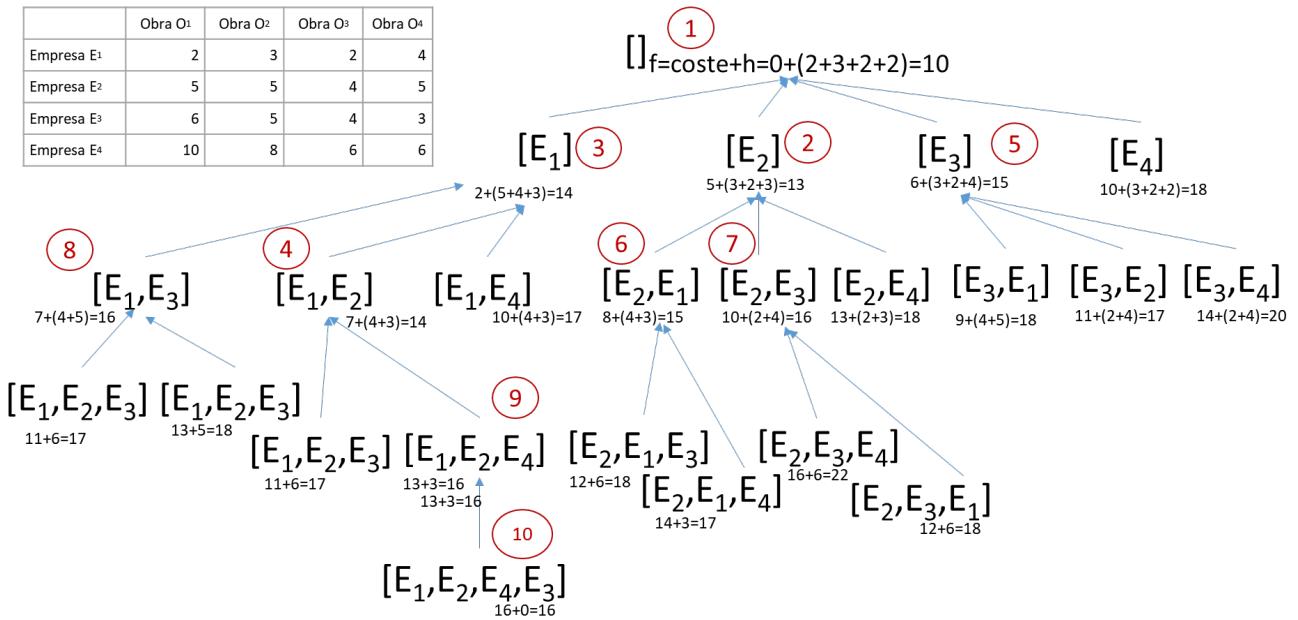
**mínimo\_columna(i,Costes)** es el valor mínimo de la obra i-ésima (sin asignar) en la tabla de costes **exceptuando las filas de empresas en lista**.



Duración total del examen: 2 horas y media

(c) Búsqueda A\* en grafo

	Obra O <sub>1</sub>	Obra O <sub>2</sub>	Obra O <sub>3</sub>	Obra O <sub>4</sub>
Empresa E <sub>1</sub>	2	3	2	4
Empresa E <sub>2</sub>	5	5	4	5
Empresa E <sub>3</sub>	6	5	4	3
Empresa E <sub>4</sub>	10	8	6	6



#### Frontera

- 1)  $([], 10)$
  - 2)  $([E_2], 13), ([E_1], 14), ([E_3], 15), ([E_4], 18)$
  - 3)  $([E_1], 14), ([E_3], 15), ([E_2, E_1], 15), ([E_4], 18), ([E_2, E_4], 18)$
  - 4)  $([E_1, E_2], 14), ([E_3], 15), ([E_2, E_1], 15), ([E_2, E_3], 16), ([E_1, E_3], 16), ([E_1, E_4], 17), ([E_4], 18), ([E_2, E_4], 18)$
  - 5)  $([E_3], 15), ([E_2, E_1], 15), ([E_2, E_3], 16), ([E_1, E_3], 16), ([E_1, E_2, E_4], 16), ([E_1, E_4], 17), ([E_1, E_2, E_3], 17), ([E_4], 18), ([E_2, E_4], 18)$
  - 6)  $([E_2, E_1], 15), ([E_2, E_3], 16), ([E_1, E_3], 16), ([E_1, E_2, E_4], 16), ([E_1, E_4], 17), ([E_1, E_2, E_3], 17), ([E_2, E_1, E_3], 17), ([E_3, E_2], 17), ([E_3, E_2], 17), ([E_2, E_1, E_4], 17), ([E_4], 18), ([E_2, E_4], 18), ([E_3, E_1], 18), ([E_3, E_4], 20)$
  - 7)  $([E_2, E_3], 16), ([E_1, E_3], 16), ([E_1, E_2, E_4], 16), ([E_1, E_4], 17), ([E_1, E_2, E_3], 17), ([E_3, E_2], 17), ([E_3, E_2], 17), ([E_2, E_1, E_4], 17), ([E_4], 18), ([E_2, E_4], 18), ([E_3, E_1], 18), ([E_2, E_1, E_3], 18), ([E_3, E_4], 20)$
  - 8)  $([E_1, E_3], 16), ([E_1, E_2, E_4], 16), ([E_1, E_4], 17), ([E_1, E_2, E_3], 17), ([E_1, E_3, E_2], 17), ([E_3, E_2], 17), ([E_2, E_1, E_4], 17), ([E_4], 18), ([E_2, E_4], 18), ([E_3, E_1], 18), ([E_2, E_1, E_3], 18), ([E_1, E_3, E_4], 18), ([E_3, E_4], 20)$
  - 9)  $([E_1, E_2, E_4], 16), ([E_1, E_2, E_4, E_3], 16), ([E_1, E_4], 17), ([E_1, E_2, E_3], 17), ([E_1, E_3, E_2], 17), ([E_3, E_2], 17), ([E_2, E_1, E_4], 17), ([E_4], 18), ([E_2, E_4], 18), ([E_3, E_1], 18), ([E_2, E_1, E_3], 18), ([E_1, E_3, E_4], 18), ([E_3, E_4], 20)$
  - 10)  $([E_1, E_2, E_4, E_3], 16), ([E_1, E_4], 17), ([E_1, E_2, E_3], 17), ([E_1, E_3, E_2], 17), ([E_3, E_2], 17), ([E_2, E_1, E_4], 17), ([E_4], 18), ([E_2, E_4], 18), ([E_3, E_1], 18), ([E_2, E_1, E_3], 18), ([E_1, E_3, E_4], 18), ([E_3, E_4], 20)$
- Expandido Objetivo**

#### Expandidos

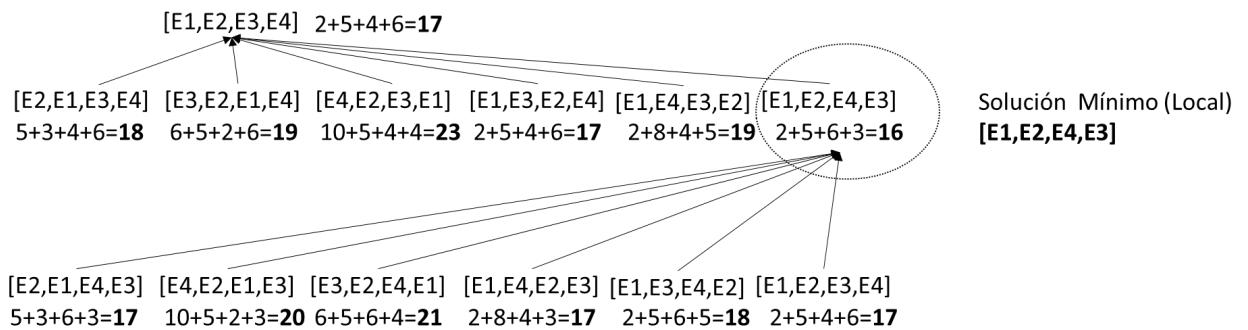
- 1)  $([], 10)$
- 2)  $([], 10), ([E_1], 14)$
- 4)  $([], 10), ([E_1], 14), ([E_1, E_2], 14)$
- 5)  $([], 10), ([E_1], 14), ([E_1, E_2], 14), ([E_3], 15)$
- 6)  $([], 10), ([E_1], 14), ([E_1, E_2], 14), ([E_3], 15), ([E_2, E_1], 15)$
- 7)  $([], 10), ([E_1], 14), ([E_1, E_2], 14), ([E_3], 15), ([E_2, E_1], 15), ([E_2, E_3], 16)$
- 8)  $([], 10), ([E_1], 14), ([E_1, E_2], 14), ([E_3], 15), ([E_2, E_1], 15), ([E_2, E_3], 16), ([E_1, E_3], 16)$
- 9)  $([], 10), ([E_1], 14), ([E_1, E_2], 14), ([E_3], 15), ([E_2, E_1], 15), ([E_2, E_3], 16), ([E_1, E_3], 16), ([E_1, E_2, E_4], 16)$
- 10)  $([], 10), ([E_1], 14), ([E_1, E_2], 14), ([E_3], 15), ([E_2, E_1], 15), ([E_2, E_3], 16), ([E_1, E_3], 16), ([E_1, E_2, E_4], 16), ([E_1, E_2, E_4, E_3], 16)$



Duración total del examen: 2 horas y media

Problema 6 (d)  
Hill Climbing/Escalada

- Se parte de representación de estado completa
- Función objetivo  $h$ = suma costes asignación obras a empresas
- Operador: Genera sucesores cambiando pares en la tupla





Duración total del examen: 2 horas y media

Problema 7

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----

1. **Describe el problema:** Representación del estado, Estado inicial, Estado final, objetivo, acciones y función de coste)

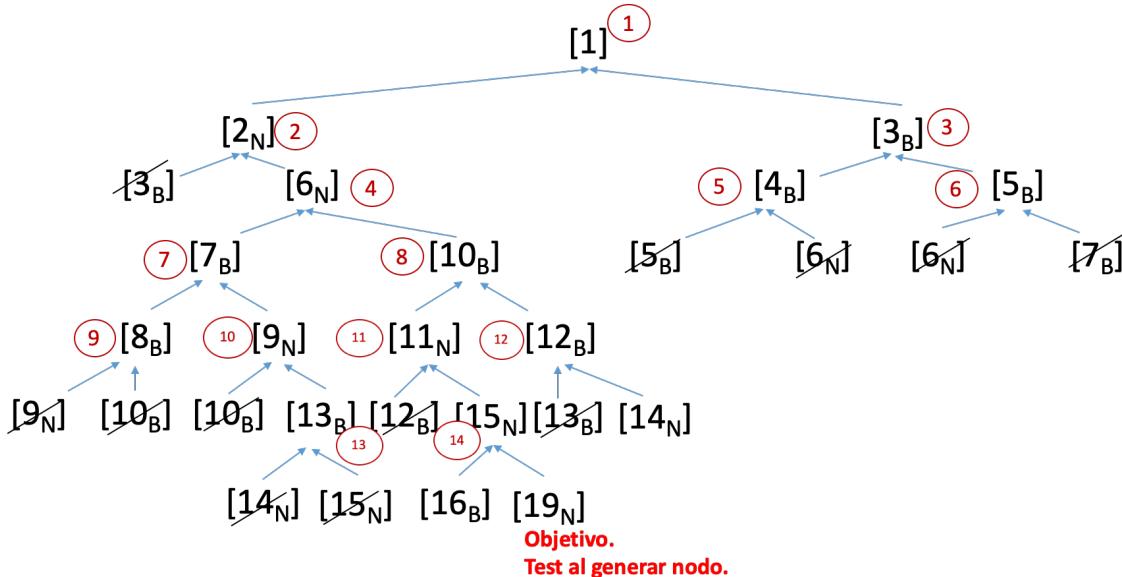
- El **estado** viene representado por la posición, dado que la figura con cuadrados blancos y negros no cambia.  
Estado inicial: 1  
Estado final: n con  $n > 18$
- Acciones:

	Precondición	Resultado
Mueve1(n)	$n \geq 1 \& n \leq 18$	$n = n + 1$
Mueve2(n)	$n \geq 1 \& n \leq 18$ & color(n) = blanco	$n = n + 2$
Mueve4(n)	$n \geq 1 \& n \leq 18$ & color(n) = negro	$n = n + 4$
- Función coste: 1 por movimiento. Coste = longitud camino.

6. ¿Cuál es el factor de ramificación?

En cualquier nodo se pueden ejecutar **2 acciones**.

7. Desarrolla el árbol de búsqueda que genera el algoritmo **búsqueda en anchura sobre un grafo**.  
¿Cuál es la solución óptima?



Solución: Acciones: 1 4 4 2 4 4 / Casillas: 1 2 6 10 11 15 19

8. Plantea una heurística admisible no trivial ( $h=0$  no vale) para el problema. Reflexiona sobre la admisibilidad de la heurística propuesta.

Relajamos problema suponiendo todas las casillas son negras. De esta manera, siempre se puede aplicar la acción Mueve4(n)



Duración total del examen: 2 horas y media

Heurística propuesta:  $h(n) = \frac{(N+1) - \text{posición}}{4}$  división entera. N=última casilla.

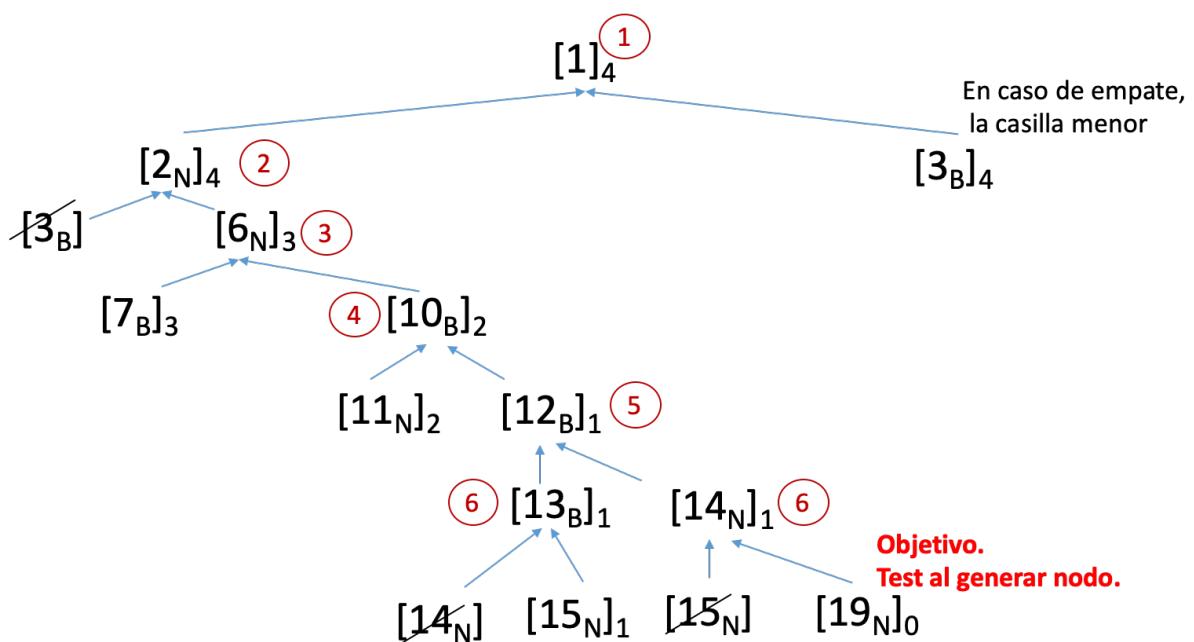
Por ejemplo  $h(1) = \frac{19-1}{4} = 4$

Por ejemplo  $h(14) = \frac{19-14}{4} = 1$

Por ejemplo  $h(15) = \frac{19-15}{4} = 1$

Es admisible porque subestima.

9. Desarrolla el árbol de búsqueda que genera el algoritmo de búsqueda primero el mejor en grafo utilizando la heurística propuesta.





Duración total del examen: 2 horas y media

### Problema 8

Considera una búsqueda en árbol (es decir, no hay lista de expandidos) sobre un problema de búsqueda con un factor de ramificación  $b$ . Cada nodo del árbol de exploración tiene un coste acumulado desde el estado inicial al estado del nodo  $g(n)$ , una heurística admisible  $h(n)$  y una profundidad  $d(n)$ . Sea  $c$  un nodo objetivo con el mínimo coste y  $s$  el nodo objetivo menos profundo.

Para cada una de los siguientes puntos, da una expresión que caracterice el conjunto de nodos que son expandidos antes de que termine la búsqueda. Por ejemplo, si pedimos el conjunto de nodos con valor heurístico positivo, puedes utilizar la expresión  $h(n) \geq 0$ . No te preocupes de los empates (así que no debes preocuparte de diferencias  $>$  de  $\geq$  en caso de empates). Si no hay nodos que cumplan la expresión debes escribir “ninguno”.

6. Da una expresión (es decir, una desigualdad en términos de las anteriores cantidades) para los nodos que serán expandidos en una búsqueda en anchura en árbol.  
 **$d(n) \leq d(s)$ :** BFS expande todos los nodos con profundidad menor que la del objetivo menos profundo. En una búsqueda en profundidad se puede realizar el test de objetivo al generarlos o al extraerlos de la frontera, por lo que puede que no todos los nodos de profundidad  $d(s)$  sean expandidos antes de encontrar el objetivo.
7. Da una expresión para los nodos expandidos en una búsqueda de coste uniforme.  
 **$g(n) \leq g(c)$ :** CU expande todos los nodos con coste menor que el objetivo. Recuerda que el test-objetivo se realiza cuando se saca el nodo de la frontera para garantizar optimidad. Esto quiere decir que podemos expandir algún nodo de coste  $g(c)$  antes de expandir el nodo objetivo.
8. Da una expresión para los nodos  $n$  expandidos en una búsqueda A\* en árbol con heurística  $h(n)$ .  
 **$g(n) + h(n) \leq g(c)$ :** Todos los nodos con suma de coste más función heurística menor o igual al coste del test objetivo con menor coste serán expandidos.
9. Sean  $h_1$  y  $h_2$  dos heurísticas admisibles tal que  $\forall n, h_1(n) \geq h_2$ . Da una expresión para los nodos que se expandirán en la búsqueda A\* en árbol usando  $h_1$  pero no se expandirán cuando se use  $h_2$ .  
**Ninguno.** Sea  $N_1 = \{n : g(n) + h_1(n) \leq g(c)\}$  el conjunto de nodos expandidos por  $h_1$ , y Sea  $N_2 = \{n : g(n) + h_2(n) \leq g(c)\}$  el conjunto de nodos expandidos por  $h_2$ . El conjunto de nodos expandidos por  $N_1$  pero no por  $N_2$  es igual a  $N_1 - N_2 = N_1 \cap \bar{N}_2$ . Puesto que  $h_1 \geq h_2$ ,  $h_1$  es una heurística más informada y expandirá menos nodos de forma que  $N_1 \subseteq N_2$ , y por lo tanto  $N_1 \cap \bar{N}_2 = \emptyset$ .
10. Da una expresión para los nodos que serán expandidos por una búsqueda A\* en árbol usando  $h_2$  pero no cuando se use  $h_1$ .  
**Como antes,** sea  $N_1 = \{n : g(n) + h_1(n) \leq g(c)\}$  el conjunto de nodos expandidos por  $h_1$ , y Sea  $N_2 = \{n : g(n) + h_2(n) \leq g(c)\}$  el conjunto de nodos expandidos por  $h_2$ . El conjunto de nodos expandidos por  $N_2$  pero no por  $N_1$  es igual a  $N_2 - N_1 = N_2 \cap \bar{N}_1 = \{n : g(n) + h_2(n) \leq g(c) \text{ y } g(n) + h_1(n) > g(c)\}$ .

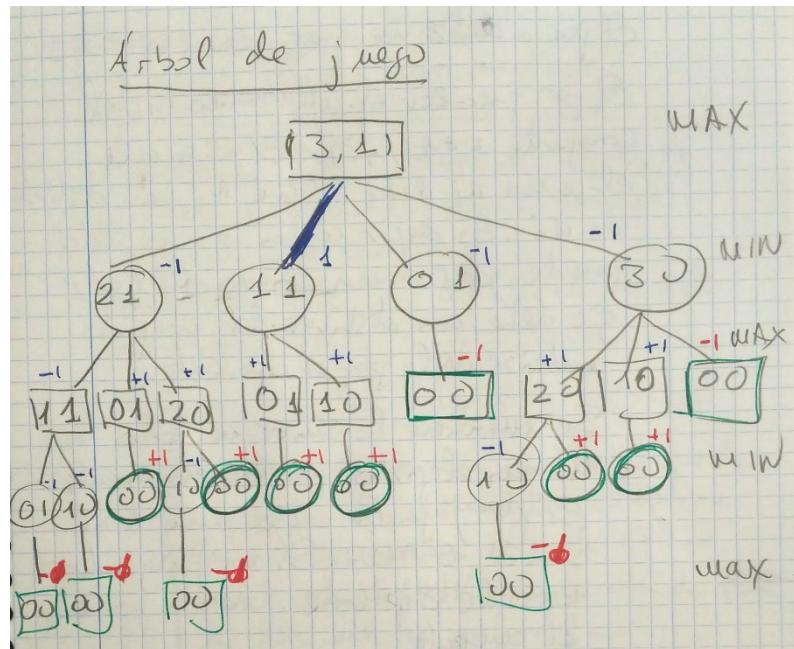


Duración total del examen: 2 horas y media

Problema. 9.

El NIM es un juego con dos jugadores. Inicialmente hay  $n$  piedras en  $r$  cestos separados. El estado inicial puede expresarse como  $(n_1, n_2, \dots, n_r)$  donde  $n_1 + n_2 + \dots + n_r = n$ . En cada movimiento de juego, un jugador puede coger cualquier número de piedras ( $> 0$ ) de cualquiera de los  $r$  cestos (sólo de un cesto cada turno). En la versión de juego que se propone, **la última persona que hace movimiento gana**. Supondremos que el estado inicial del juego es  $(3, 1)$ : dos cestos de piedras, con tres piedras en el primer cesto, y una piedra en el segundo.

- Representa el árbol de juego completo.
- Representa una evaluación **MINIMAX** del juego. A un nodo hoja se le da valor 1 si corresponde a un nodo Max ganador, y cero si representa un nodo Min ganador. No necesitas una función de estimación puesto que tienes un árbol de juego completo. Asegúrate de dar el valor correcto de 0, +1 o -1 a los nodos. Por ejemplo, si en un nodo MIN el estado es  $(0, 1)$ , el jugador MIN retirará una piedra para dejar el estado terminal  $(0, 0)$ , con lo que el nodo terminal resultante tendrá el valor -1 (MIN gana por haber realizado el último movimiento). Si, por ejemplo, en un estado MAX el estado es  $(0, 1)$ , MAX retirará la piedra dejando un nodo hoja resultante  $(0, 0)$ , con lo que el nodo hoja resultante tendrá el valor +1 (MAX gana por haber realizado el último movimiento).

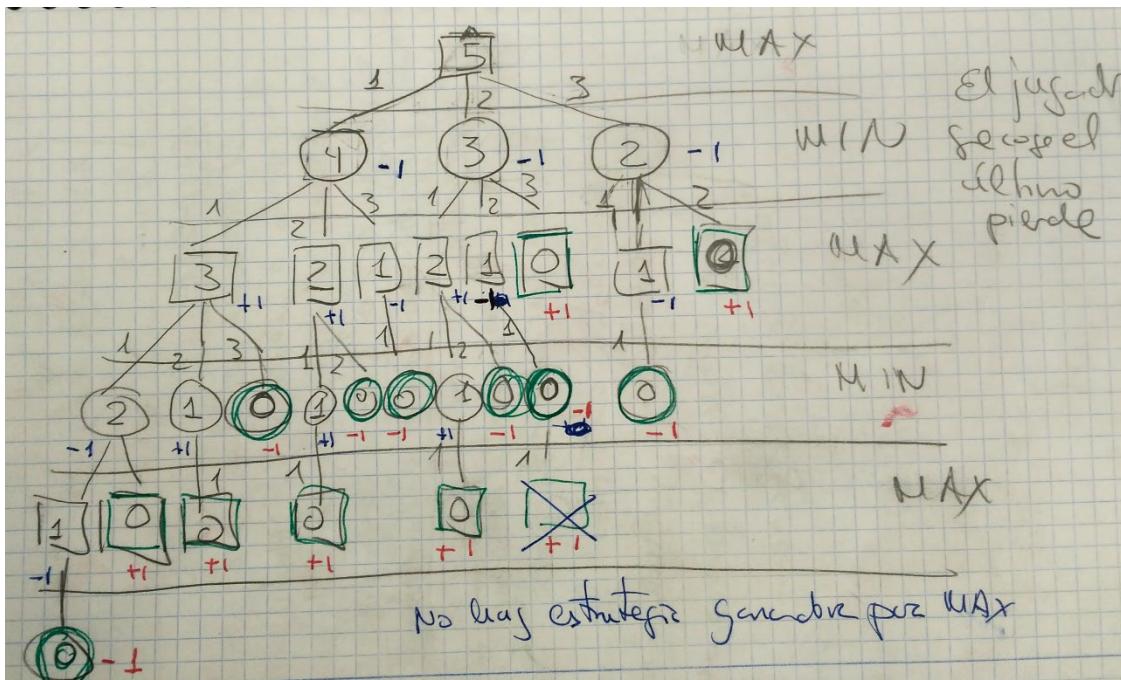




Duración total del examen: 2 horas y media

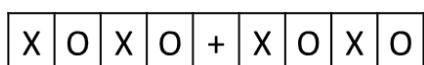
### Problema 10.

El NIM-5 es un juego con dos jugadores con las siguientes reglas: Hay apilados 5 euros. Los jugadores se alternan coger 1, 2 o 3 euros de la pila. **El jugador que coge el último euro pierde.** Muestra el árbol de juego e indica que movimiento debe escoger el primer jugador para ganar.



### Problema 11.

La posición inicial del juego que se describe a continuación es la mostrada en la figura.

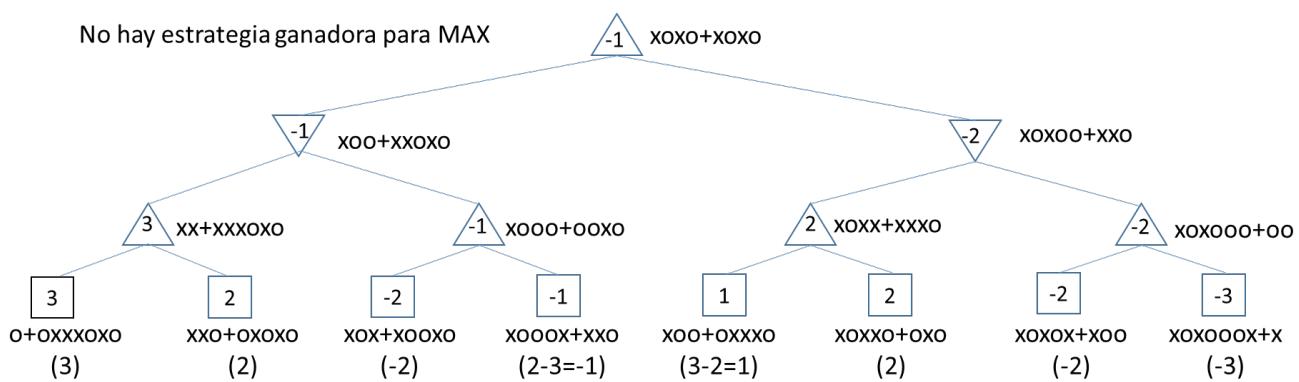


El jugador MAX juega con las fichas marcadas como X y el MIN con las fichas marcadas como O. MAX realiza el primer movimiento. La única ficha que pueden mover los jugadores es la marcada como +. Esta ficha se puede mover hacia la derecha y hacia la izquierda. Al intercambiarse con la ficha que ocupa la posición a la que se desplaza tiene el efecto de cambiar el signo de esa ficha y de la que pasa a ser su contigua. Por ejemplo, en la configuración XOXO+XOXO, si desplazamos la ficha + hacia la derecha obtenemos la configuración XOXOO+XXO. El objetivo de cada jugador es tener 5 fichas del mismo tipo seguidas. Como el juego es demasiado largo para desarrollar el árbol de juego completo en el examen, utilizaremos la siguiente función para la evaluación de las configuraciones:  $f'(n) = \text{Suma de los tamaños de los grupos de X mayores que } 1 - \text{Suma de los tamaños de los grupos de O mayores que } 1$ . Si la ficha + está entre dos fichas iguales, éstas no forman un grupo. Por ejemplo, la evaluación de la configuración XOXOO+XXO sería  $(2 - 2) = 0$ , la evaluación de la configuración XOOOXO+OO sería  $(0 - 3 - 2) = -5$ .

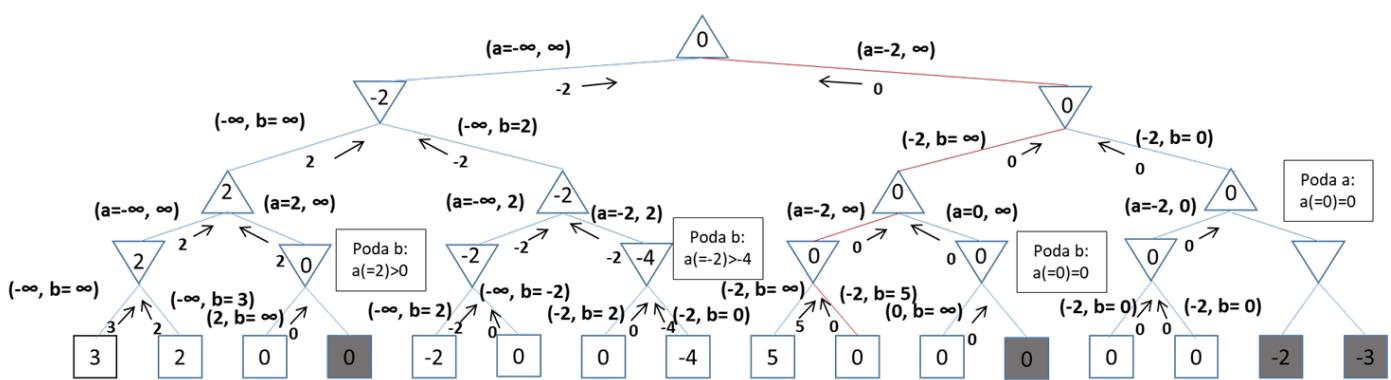


Duración total del examen: 2 horas y media

- a) Representa el árbol de juego y sobre éste utiliza el **algoritmo MINIMAX** para evaluar el primer movimiento que debería hacer el jugador MAX. Haz la exploración hasta el **nivel 3** (dos jugadas de MAX y una de MIN). Aplica siempre el mismo orden: primero el movimiento hacia la izquierda y después el movimiento hacia la derecha.



- b) En la figura siguiente se muestra el árbol de juego hasta el **nivel 4** sin mostrar los estados, pero con la evaluación de los nodos terminales. Suponiendo que la raíz del árbol es el jugador MAX, muestra los nodos examinados usando la estrategia de **poda  $\alpha$ - $\beta$**  y las ramas podadas asumiendo recorrido de izquierda a derecha. Para cada nodo muestra claramente la evolución de los valores alfa y beta.



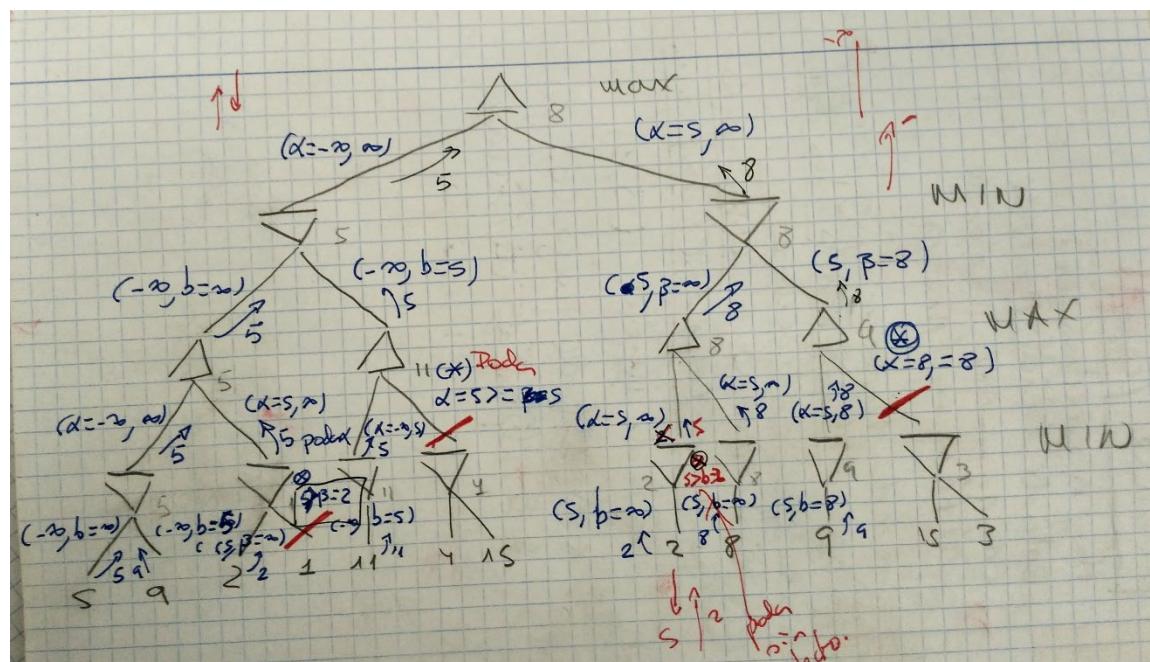
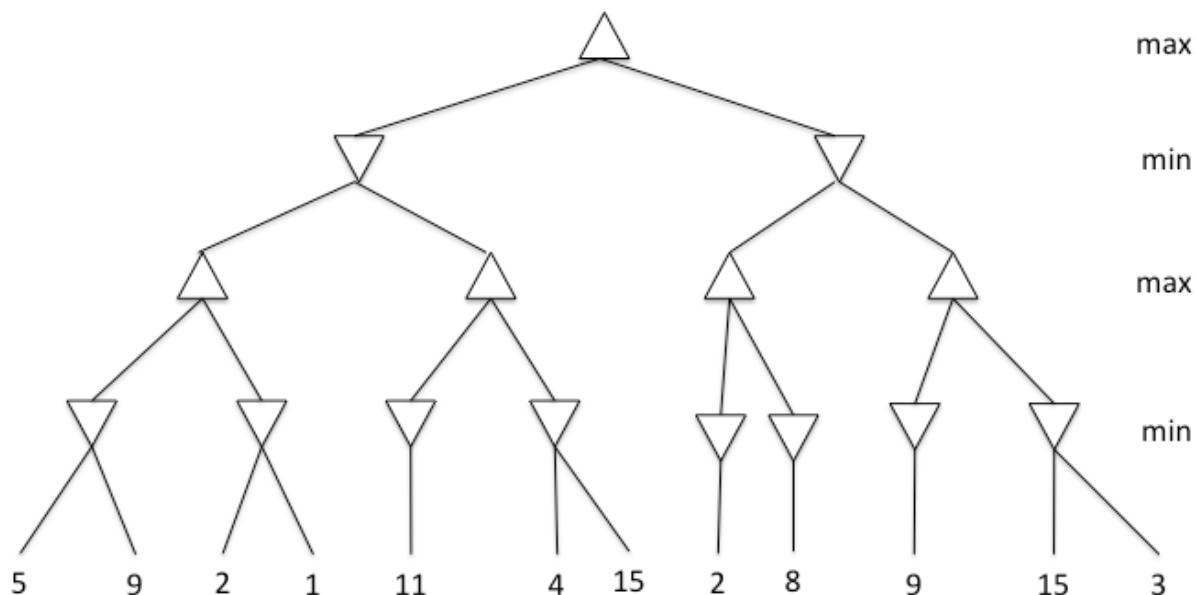


Duración total del examen: 2 horas y media

Problema 12.

Suponiendo que la raíz del árbol es el jugador MAX:

1. Muestra los valores obtenidos para todos los nodos en el siguiente árbol de juego
2. Muestra los nodos tendrían que ser examinados usando la estrategia de poda  $\alpha\beta$  y las ramas que son podadas. Para cada rama podada, explica brevemente porque la poda  $\alpha\beta$  poda la rama.





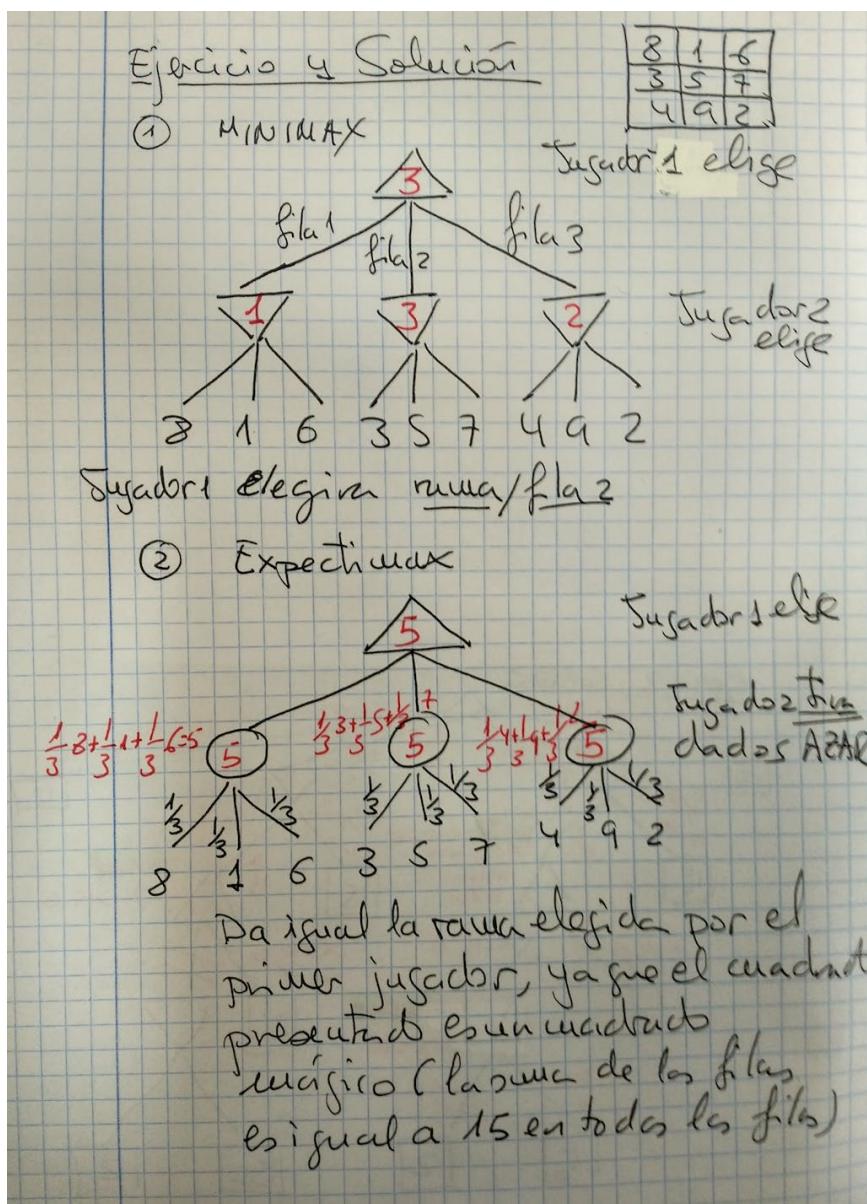
Duración total del examen: 2 horas y media

8	1	6
3	5	7
4	9	2

### Problema 13.

Dada la matriz 3x3 con los valores mostrados, dos jugadores se inventan un absurdo juego consistente en que el primero elige la fila, y el segundo después una columna. El juego finaliza tras los dos movimientos, y el segundo jugador dará la cantidad de euros que corresponda a la fila y columna elegida al segundo. Por ejemplo, el primero elige fila 1, y el segundo la columna 2. El juego finaliza y el segundo jugador le da 1 euro al primero.

1. Dibuja el árbol de juego, y sobre el árbol realiza la evaluación MINIMAX, indicando cual es la rama que elegirá el primer jugador para maximizar la ganancia.
2. Supongamos ahora que el jugador 2 no tiene opción de elegir la columna, y debe tirar un dado en el que, si sale 1 o 4 elige la primera columna, si sale 2 o 5 la segunda y si sale 3 o 6 elige la tercera columna. Dibuja el nuevo árbol de juego, la evaluación EXPECTIMAX, e indica que rama debe elegir el primer jugador para maximizar las ganancias.

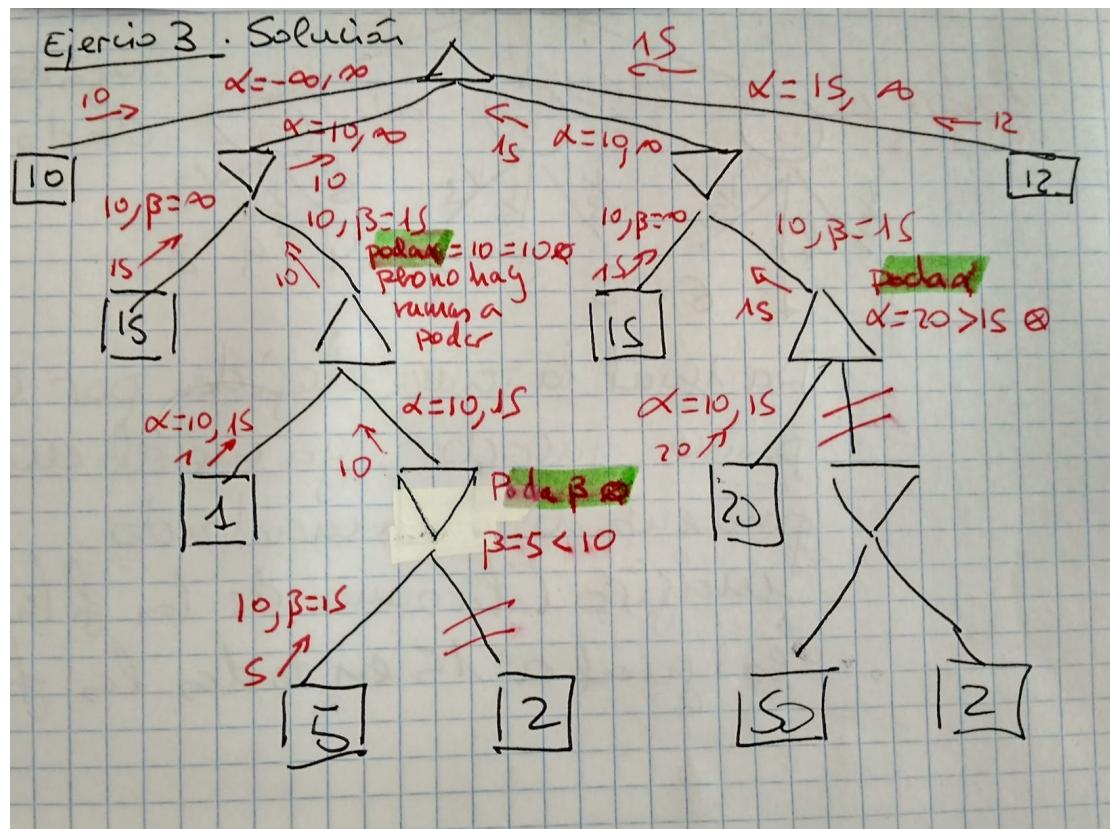
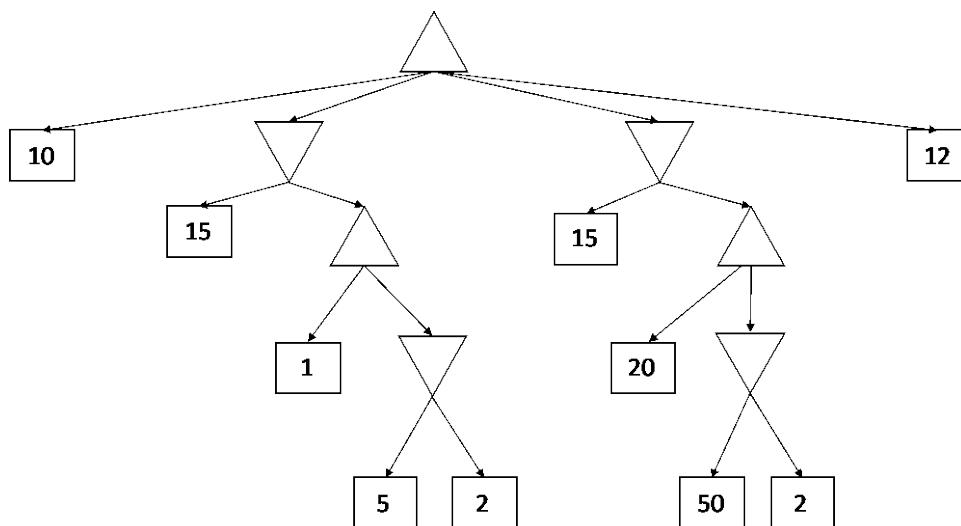




Duración total del examen: 2 horas y media

Problema 14.

Suponiendo que la raíz del árbol es el jugador MAX, muestra nodos examinados usando la estrategia de poda  $\alpha\text{-}\beta$  y ramas podadas asumiendo recorrido de izquierda a derecha.



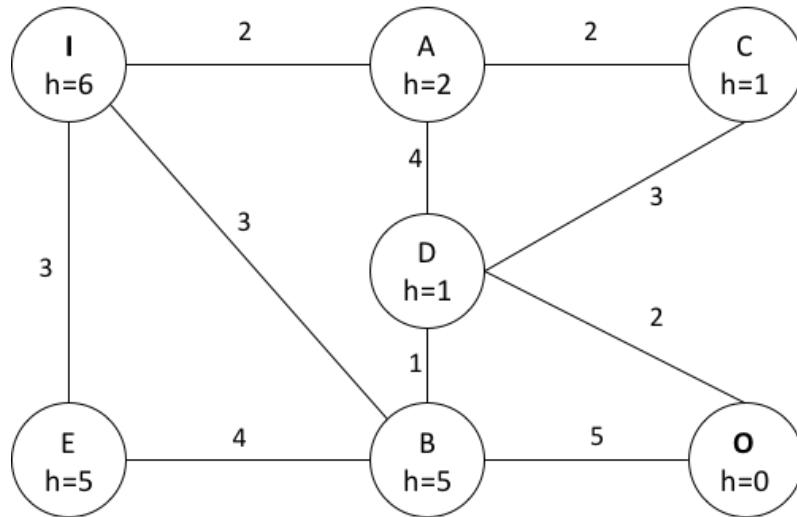


Duración total del examen: 2 horas y media

Problema 15.

Considera el siguiente grafo dibujado que representa el espacio de estados donde **I** es el estado inicial, y **O** es el estado objetivo. Los arcos están etiquetados con el coste de ir de un nodo a otro, y los nodos con  $h$ , la estimación de cuánto cuesta ir del nodo al objetivo.

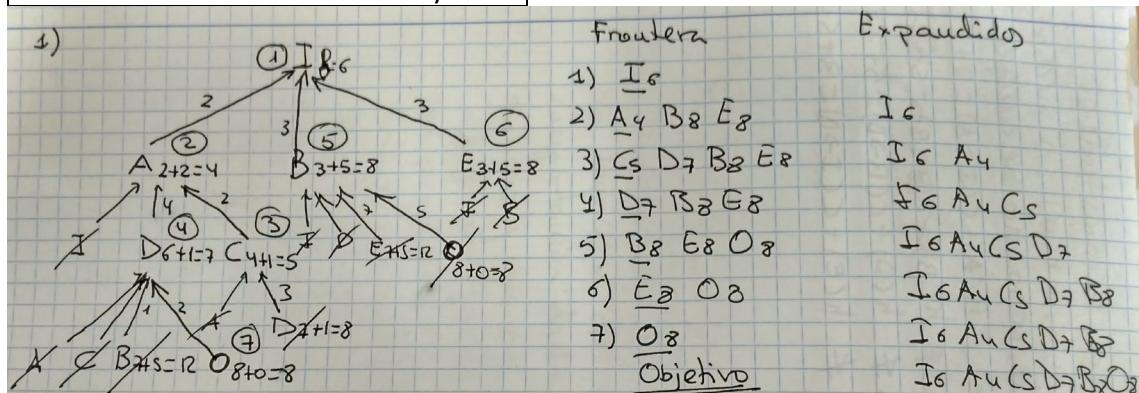
El agente de búsqueda utiliza la búsqueda A\* en grafo para ir del nodo inicial al objetivo.



1. Debes representar el árbol de exploración, etiquetando cada nodo con el valor de la función de evaluación, y marcando el orden de expansión de los nodos. Además, debes mostrar el estado de la FRONTERA y EXPLORADOS, si procede, en cada paso de la búsqueda. En caso de empate expande en orden alfabético.
2. ¿Es admisible la heurística? Si no lo es indica en qué nodos no lo es y propón un valor para que lo sea. ¿Una heurística admisible garantizaría el camino óptimo (menor coste) con la búsqueda A\* en grafo?
3. ¿Es consistente la heurística reflejada en el grafo? Si no lo es demuestra en qué nodos no se cumple. En el ejemplo propuesto, con la búsqueda A\* en grafo ¿Es posible cambiar la heurística en un solo nodo y garantizar que encontrará el camino óptimo?
4. Nuestro agente de búsqueda utiliza ahora la búsqueda Hill-climbing para ir del estado inicial al objetivo. Utilizando el grafo inicial, en el que todos los valores de la heurística los hemos incrementado en una unidad, debes representar el árbol de exploración, etiquetando cada nodo con el valor de la función de evaluación, y marcando el orden de expansión de los nodos. Además, debes mostrar el estado de la FRONTERA y EXPLORADOS, si procede, en cada paso de la búsqueda. En caso de empate expande en orden alfabético.



Duración total del examen: 2 horas y media



Camino: I-A-D-O, coste 8

2. No es admisible.  $h(B) = 5 \nleq h^*(B)=3$ . Elegimos el mayor valor (heurística más informada) de las posibles  $h(B)=3$ .

Una heurística admisible no garantiza el óptimo en una búsqueda A\* en grafo.

3. La heurística es consistente si para todos los nodos  $h(n) \leq c(n,n') + h(n')$ . En los siguientes casos no se cumple:

$$h(I) = 6 \nleq h(A) + c(I,A) = 2 + 2 = 4$$

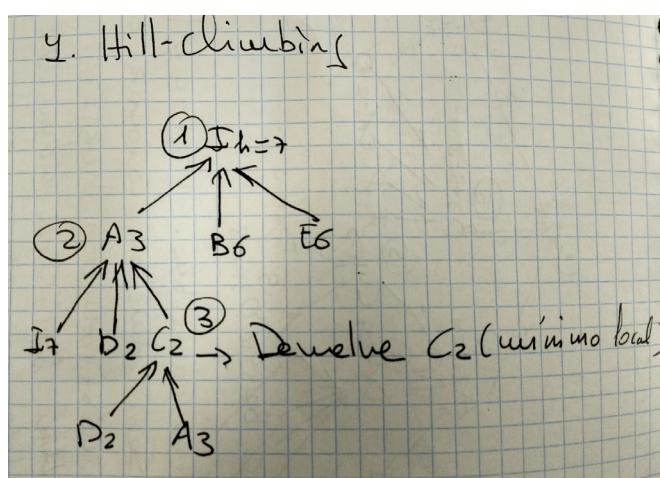
$$h(B) = 5 \nleq h(D) + c(B,D) = 1 + 1 = 2$$

Para que sea consistente  $h(B) \leq 2$ . Cogemos  $h(B)=2$ .

$$h(I) \leq h(A) + C(I,A) = 4$$

$$h(I) \leq h(B) + C(I,B) = h(B)+3=5. \text{ Cogemos } h(I) = 4.$$

La heurística del nodo I no tiene importancia dado que es el primer nodo expandido y ya no se tiene en cuenta al formar parte de la lista de expandidos. Si cambiamos la heurística de B por un valor  $0 \leq h(B) \leq 2$  podemos garantizar que la búsqueda A\* en grafo encontrará el camino óptimo de I a O.



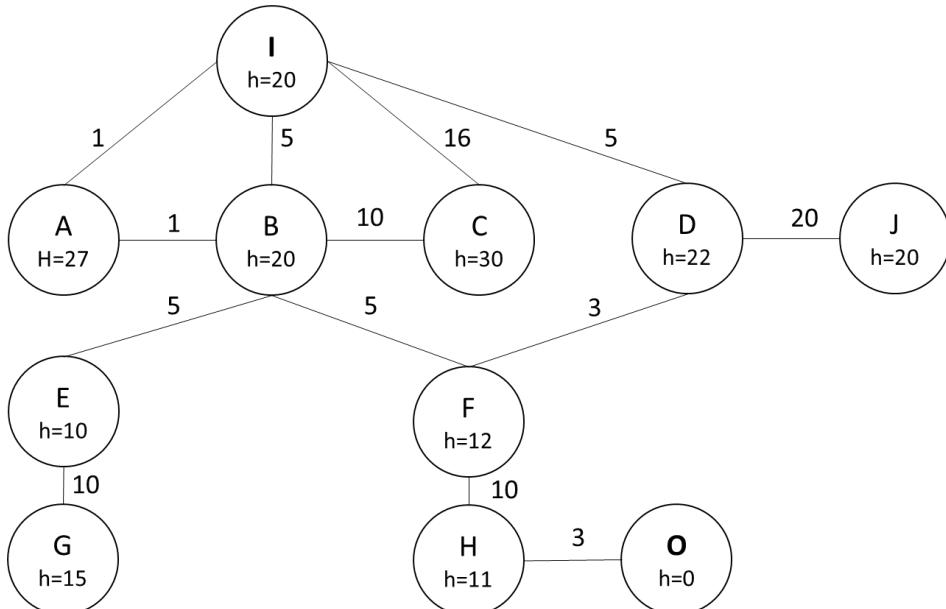


Duración total del examen: 2 horas y media

Problema 16.

Considera el siguiente grafo dibujado que representa el espacio de estados donde **I** es el estado inicial, y **O** es el estado objetivo. Los arcos están etiquetados con el coste de ir de un nodo a otro, y los nodos con  $h$ , la estimación de cuánto cuesta ir del nodo al objetivo.

1. El agente de búsqueda utiliza la búsqueda A\* en grafo para ir del nodo inicial al objetivo. Debes



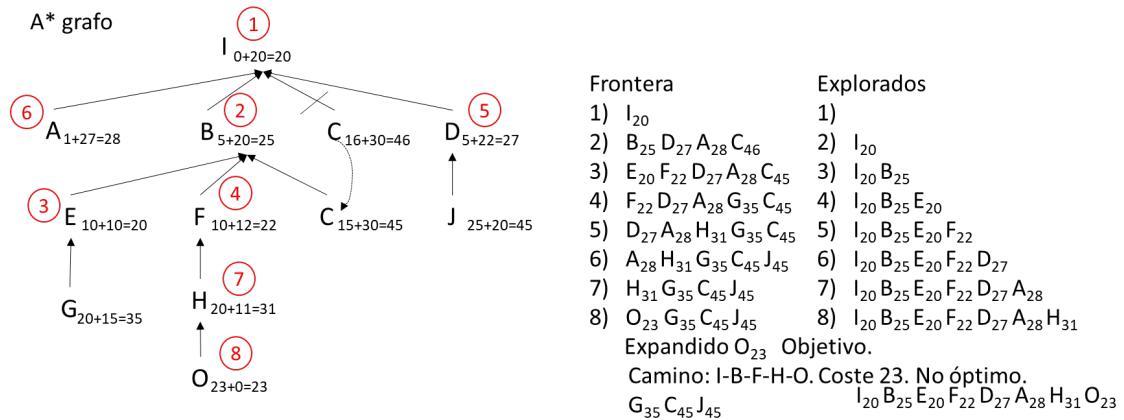
representar el árbol de exploración, etiquetando cada nodo con el valor de la función de evaluación, y marcando el orden de expansión de los nodos. Además, debes mostrar el estado de la FRONTERA y EXPLORADOS, si procede, en cada paso de la búsqueda. En caso de empate expande en orden alfabético. Debes indicar el coste de cada nodo cuando es expandido.

2. ¿Es consistente la heurística reflejada en el grafo? Si no lo es indica algún ejemplo de nodo en el que la heurística no es consistente.
3. Define el valor de la función heurística en cada nodo para que sea consistente.
4. El agente de búsqueda utiliza la búsqueda A\* en árbol para ir del nodo inicial al objetivo con la heurística que has definido en el punto 3. Debes representar el árbol de exploración, etiquetando cada nodo con el valor de la función de evaluación, y marcando el orden de expansión de los nodos. Además, debes mostrar el estado de la FRONTERA y EXPLORADOS, si procede, en cada paso de la búsqueda. En caso de empate expande en orden alfabético.



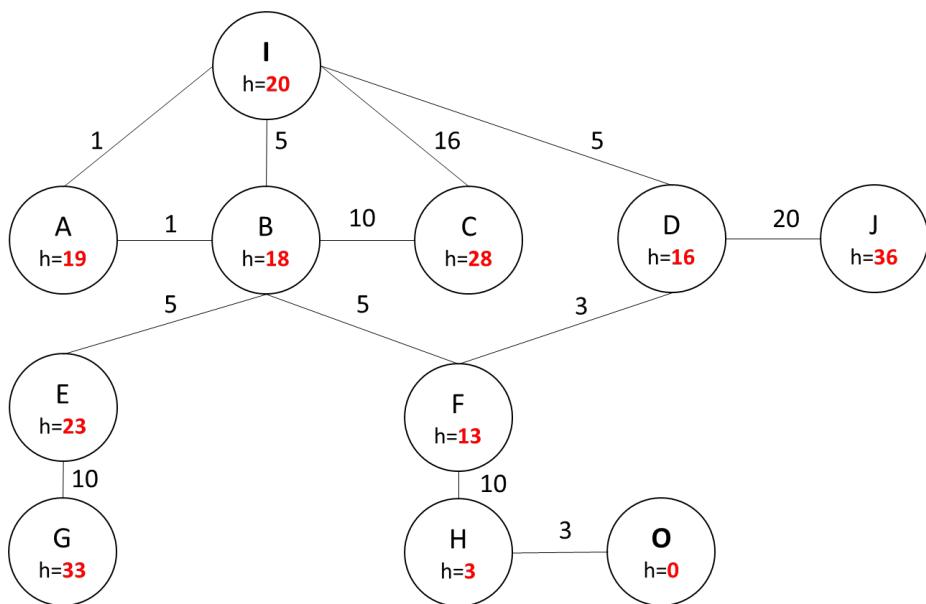
Duración total del examen: 2 horas y media

1)



2) No es consistente. Por ejemplo  $h(D) \leq h(F) + c(D,F)$ .  $22 \leq |12+3=15|$

3)

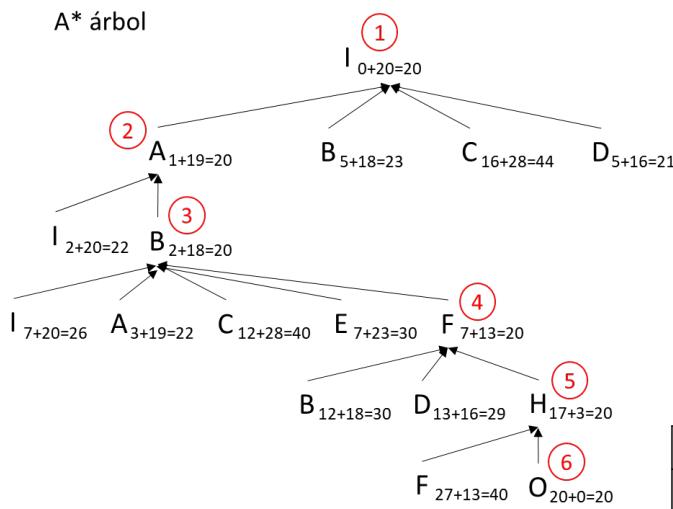


Nodo	A	B	C	D	E	F	G	H	J	I	O
h	19	18	28	16	23	13	33	3	36	20	0



Duración total del examen: 2 horas y media

4)



Frontera

- 1) I<sub>20</sub>
- 2) A<sub>20</sub> D<sub>21</sub> B<sub>23</sub> C<sub>44</sub>
- 3) B<sub>20</sub> D<sub>21</sub> D<sub>22</sub> I<sub>22</sub> B<sub>23</sub> C<sub>44</sub>
- 4) F<sub>20</sub> D<sub>21</sub> A<sub>22</sub> D<sub>22</sub> I<sub>22</sub> B<sub>23</sub> I<sub>26</sub> E<sub>30</sub> C<sub>40</sub> C<sub>44</sub>
- 5) H<sub>20</sub> D<sub>21</sub> A<sub>22</sub> D<sub>22</sub> I<sub>22</sub> B<sub>23</sub> I<sub>26</sub> D<sub>29</sub> B<sub>30</sub> E<sub>30</sub> C<sub>40</sub> C<sub>44</sub>
- 6) O<sub>20</sub> D<sub>21</sub> A<sub>22</sub> D<sub>22</sub> I<sub>22</sub> B<sub>23</sub> I<sub>26</sub> D<sub>29</sub> B<sub>30</sub> E<sub>30</sub> C<sub>40</sub> F<sub>40</sub> C<sub>44</sub>

Expande O<sub>20</sub>, objetivo.

Camino: I-A-B-F-H-O coste 20. Óptimo.

D<sub>21</sub> A<sub>22</sub> D<sub>22</sub> B<sub>23</sub> I<sub>26</sub> D<sub>29</sub> B<sub>30</sub> E<sub>30</sub> C<sub>40</sub> F<sub>40</sub> C<sub>44</sub>

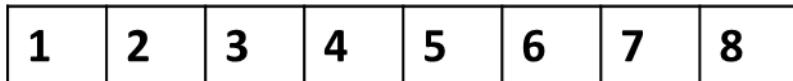
Nodo	A	B	C	D	E	F	G	H	J	I	O
h	19	18	28	16	23	13	33	3	36	20	0



Duración total del examen: 2 horas y media

Problema 17.

Dada la siguiente figura, encontrar un camino de desde la casilla inicial a la casilla final. Se comienza situado en el cuadrado más a la izquierda (casilla 1), y el objetivo es llegar exactamente a la última casilla (casilla 8).



En cada casilla se pueden realizar dos acciones diferentes:

- **Andar** desde la casilla S a la casilla S+1 con **coste 1**.
- **Saltar** de la casilla S a las 2\*S con **coste 2**.

con la **restricción** de que no puede haber más acciones de saltar que de andar para llegar a un estado, y que no se permiten las acciones que mueven más allá de la casilla objetivo.

1. Describe el **problema**:

- ESTADO: Un estado se puede representar por una terna  $(C, P, S)$  en la que:
    - $C \in [1, 8]$  indica la **posición/Casilla en el tablero**.
    - $P \in [1, 8]$  indica el **número de pasos** que se han dado para llegar a la casilla en curso.
    - $S \in [1, 3]$  indica el **número de saltos** que se han dado para llegar a la casilla en curso.
- Nota: Con esta representación es importante descartar que dos estados  $S_i = (C_i, P_i, S_i)$  y  $S_j = (C_j, P_j, S_j)$  son iguales si  $S_i = S_j$  y  $P_i - S_i = P_j - S_j$ .*

Estado inicial : (1,0,0)

Estado objetivo: (8,\*,\*)

Alternativamente podemos representar el estado por una tupla  $(C, D)$ , donde C indica la posición/casilla en el tablero y D representa la diferencia entre el número de pasos que se han dado y el número de saltos, es decir  $C = P - D$ . En este caso los estados son iguales si las tuplas son iguales. Elegimos la primera representación para el resto del problema.

**Estado inicial:** (1,0)

**Test objetivo:** Casilla(S)=8. Que cumplen todas las tuplas (8,\*,\*)

El **modelo de transiciones** debe representar las **precondiciones** de cada acción y sus **efectos**

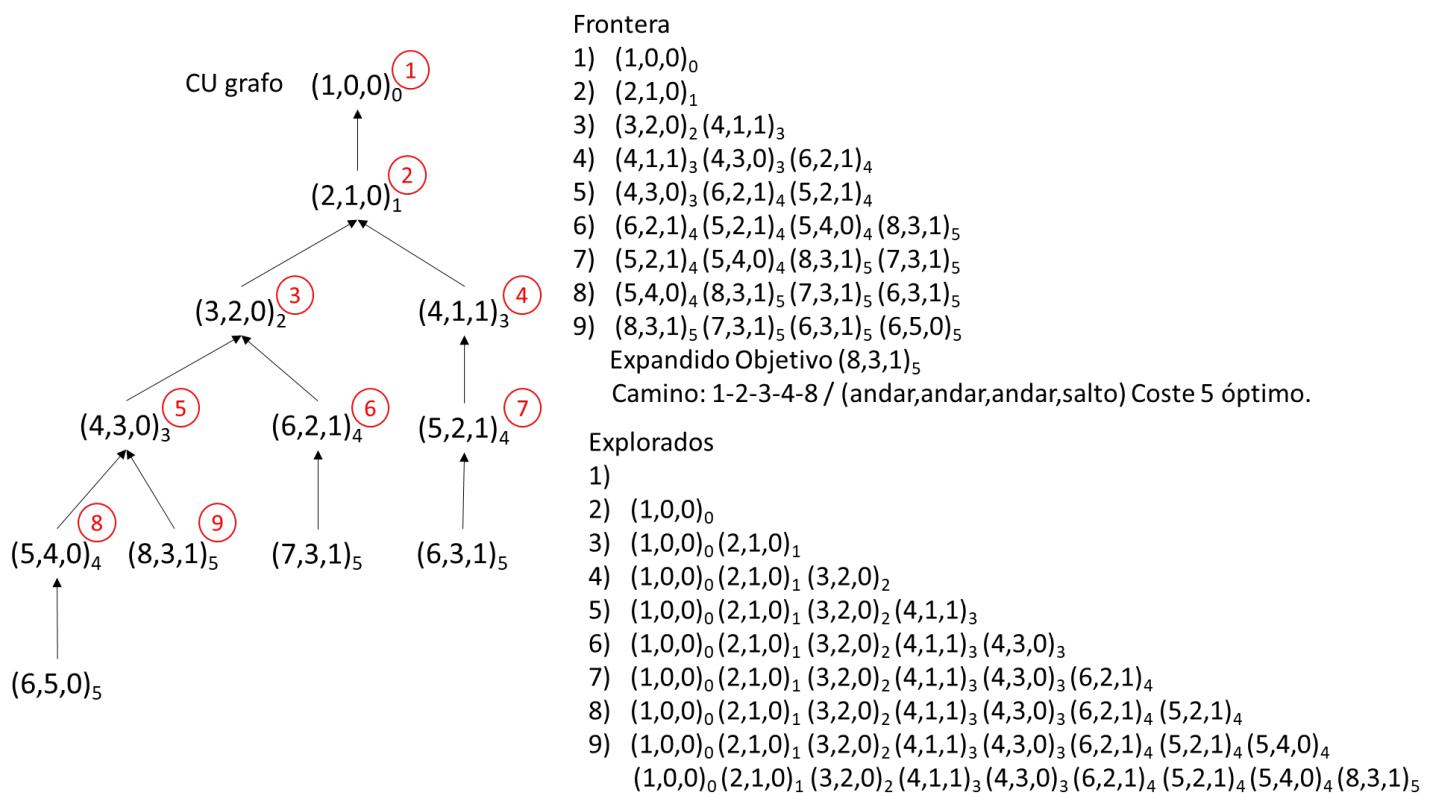
Acciones	Precondiciones	Resultado
Andar ( $C, P, S$ )	$C < 8$	$C = C + 1; P = P + 1$
Saltar ( $C, P, S$ )	$C < 5 \& (S < P)$	$C = C * 2; S = S + 1;$

**Coste camino** =  $2 * S + P$ . Suma de número de pasos y el doble del número de saltos.



Duración total del examen: 2 horas y media

2. Desarrolla el árbol de búsqueda que genera el algoritmo de **búsqueda CU** (coste uniforme en grafo). Debes representar el árbol de exploración, etiquetando cada nodo con el estado, el valor de la función de evaluación, y marcando el orden de expansión de los nodos. Además, debes mostrar el estado de la FRONTERA y EXPLORADOS, si procede, en cada paso de la búsqueda. En caso de empate expande primero el nodo que estaba antes en la cola.





Duración total del examen: 2 horas y media

3. Explica razonadamente si la siguiente heurística es admisible:

$$h(S) = 2 * \log_2 \left( \frac{\text{Número Casilla Estado Objetivo}}{\text{Número Casilla Estado } S} \right)$$

**Plantea una heurística admisible** no trivial ( $h=0$  no vale) para el problema. Explica razonadamente la admisibilidad de la heurística propuesta. (0,5 puntos).

La búsqueda realizada con CU nos da el coste óptimo para llegar desde la casilla 1 a la 8. Por lo tanto  $h^*(1)=5$

$h(1) = 2 * \log_2 \left( \frac{8}{1} \right) = 2 * \log_2 (2^3) = 2 * 3 = 6 \geq h^*(1)=5$ . La heurística por lo tanto sobreestima en este estado y no es admisible.

Es fácil ver calcular el coste óptimo para cada estado, y se puede plantear como heurística la siguiente heurística  $h_2$  representada por la tabla:

Estado	(1,*,*)	(2,*,*)	(3,*,*)	(4,*,*)	(5,*,*)	(6,*,*)	(7,*,*)	(8,*,*)
$h_2$	5	4	3	2	3	2	1	0

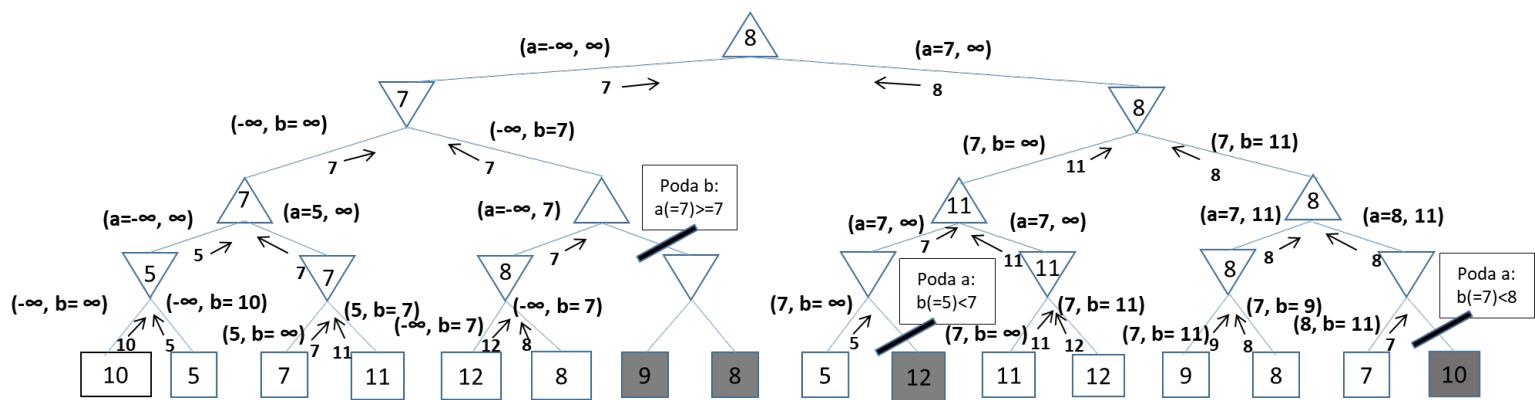
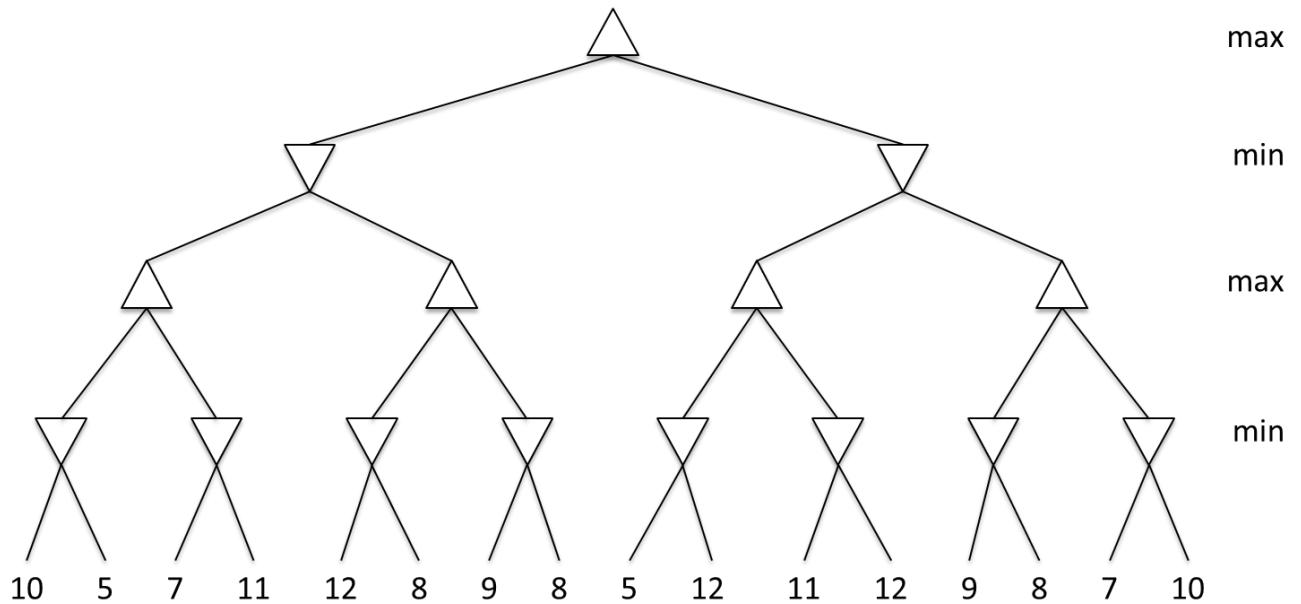
La heurística  $h_3(S) = \log_2 \left( \frac{\text{Número Casilla Estado Objetivo}}{\text{Número Casilla Estado } S} \right)$  también es admisible. Esta heurística surge al relajar las siguientes restricciones: Se permite un número de saltos menor o igual al número de pasos, asume que todas las acciones tienen el mínimo coste 1, y permite dar saltos no enteros.



Duración total del examen: 2 horas y media

Problema 18.

Suponiendo que la raíz del árbol es el jugador MAX, muestra nodos examinados usando la estrategia de poda  $\alpha\text{-}\beta$  y ramas podadas asumiendo recorrido de izquierda a derecha.



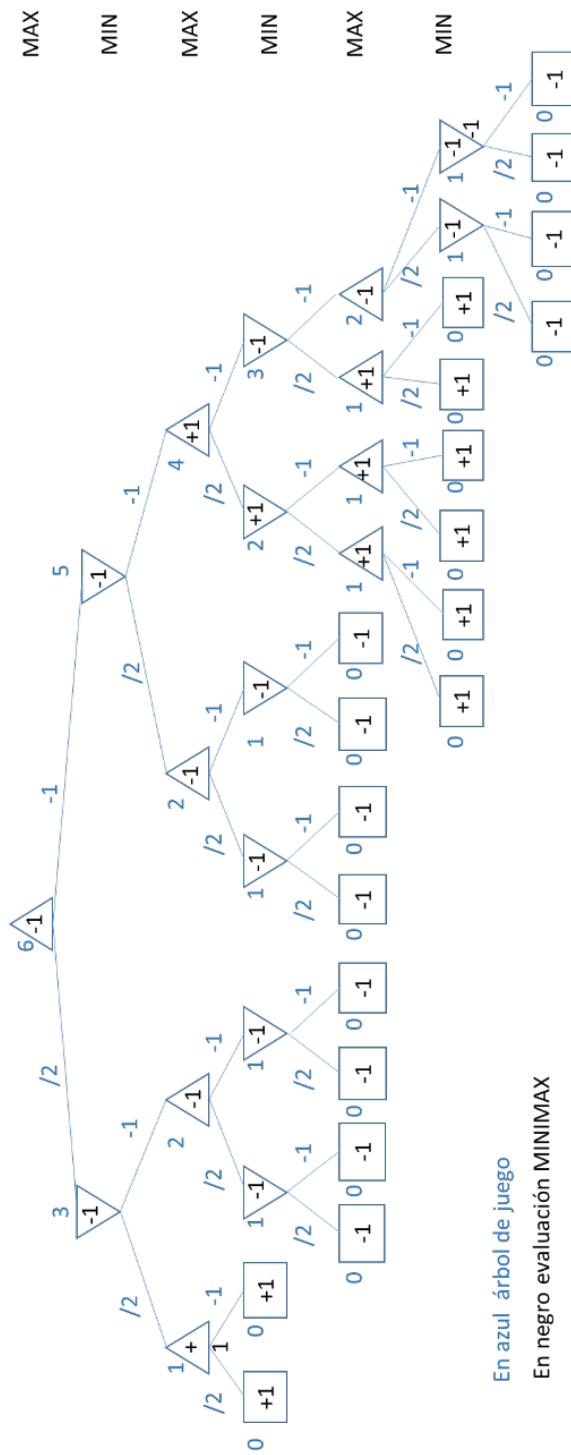


Duración total del examen: 2 horas y media

Problema 19.

El juego de “divide o reduce” consiste en empezar con un número entero  $N$ . Por turnos, cada jugador decide dividir  $N$  por dos (división entera), o decrementar  $N$  en una unidad. Comienza jugando MAX, y el jugador que obtiene el valor 0 al operar gana ( pierde el que en su turno de juego se encuentra con un valor 0). Suponiendo que partimos del valor  $N=6$ , Dibuja el árbol de juego, y sobre el árbol realiza la evaluación MINIMAX, indicando cual es la rama que elegirá el primer jugador para ganar.

Si fueras MAX y pudieras elegir otro número alternativo para empezar a jugar entre 1 y 6 ¿Cuál elegirías?





Duración total del examen: 2 horas y media

Problema 20.

```
; -----
; MODULO MAIN
;-----
(defmodule MAIN
  (export deftemplate nodo))
```

```
(deftemplate MAIN::nodo
  (slot casilla (default 1))
  (slot pasos (default 0))
  (slot saltos (default 0))
  (multislot camino)
  (slot coste (default 0))
  (slot clase (default abierto)))
```

```
(deffacts MAIN::estado-inicial (nodo (casilla 1)(pasos 0)
                                         (saltos 0)(camino "1 0 0")))
```

```
(defrule MAIN::pasa-el-mejor-a-cerrado-CU
  ?nodo <- (nodo (coste ?c1)(clase abierto))
  (not (nodo (clase abierto) (coste ?c2&:(< ?c2 ?c1))))
  =>
  (modify ?nodo (clase cerrado))
  (focus OPERADORES))
```

```
; -----
; MODULO OPERADORES
;-----
(defmodule OPERADORES
  (import MAIN deftemplate nodo))
```

```
(defrule OPERADORES::Andar
  (nodo (casilla ?casilla&:(<= ?casilla 7))
        (pasos ?pasos) (saltos ?saltos)
        (camino $?movimientos)(coste ?coste)(clase cerrado))
=>
  (assert (nodo (casilla (+ ?casilla 1))
                (pasos (+ ?pasos 1)) (saltos ?saltos)
                (camino ?movimientos
                  (implode$ (create$ (+ ?casilla 1)
                                    (+ ?pasos 1) ?saltos)))
                (coste (+ ?coste 1)))))
```

```
(defrule OPERADORES::Saltar
  (nodo (casilla ?casilla&:(<= 8 (* 2 ?casilla)))
        (pasos ?pasos) (saltos ?saltos&:(<= ?saltos ?pasos))
        (camino $?movimientos)(coste ?coste)(clase cerrado))
=>
  (assert (nodo (casilla (* ?casilla 2))
                (pasos ?pasos) (saltos (+ ?saltos 1))
                (camino ?movimientos
                  (implode$ (create$
                                (* ?casilla 2)?pasos (+ ?saltos 1) )))
                (coste (+ ?coste 2))))))
```



Duración total del examen: 2 horas y media

```
;-----  
; MODULO RESTRICCIONES  
;-----  
(defmodule RESTRICCIONES  
  (import MAIN deftemplate nodo))  
  
; eliminamos nodos repetidos  
(defrule RESTRICCIONES::repeticiones-de-nodo  
  (declare (auto-focus TRUE))  
  ?nodo1 <- (nodo (casilla ?casilla)(saltos ?saltos1)  
                    (pasos ?pasos1) (coste ?coste1))  
  ?nodo2 <- (nodo (casilla ?casilla)(saltos ?saltos2)  
                    (pasos ?pasos2&:(= (- ?pasos1 ?saltos1)  
                                         (- ?pasos2 ?saltos2)))  
                    (coste ?coste2&:(>= ?coste1 ?coste2)))  
  (test (neq ?nodo1 ?nodo2))  
=>  
  (retract ?nodo1))  
  
;-----  
; MODULO SOLUCION  
;-----  
;definimos el modulo solucion  
(defmodule SOLUCION  
  (import MAIN deftemplate nodo))  
  
;miramos si hemos encontrado la solucion  
(defrule SOLUCION::encuentra-solucion  
  (declare (auto-focus TRUE))  
  ?nodo1 <- (nodo (casilla 8) (camino $?camino)  
                    (clase cerrado))  
=>  
  (retract ?nodo1)  
  (assert (solucion ?camino)))  
  
;escribimos la solucion por pantalla  
(defrule SOLUCION::escribe-solucion  
  (solucion $?movimientos)  
=>  
  (printout t "La solucion tiene " (- (length ?movimientos) 1) " pasos" crlf)  
  (loop-for-count (?i 1 (length ?movimientos))  
    (printout t "(" (nth ?i $?movimientos) ")" " "))  
  (printout t crlf)  
  (halt))
```



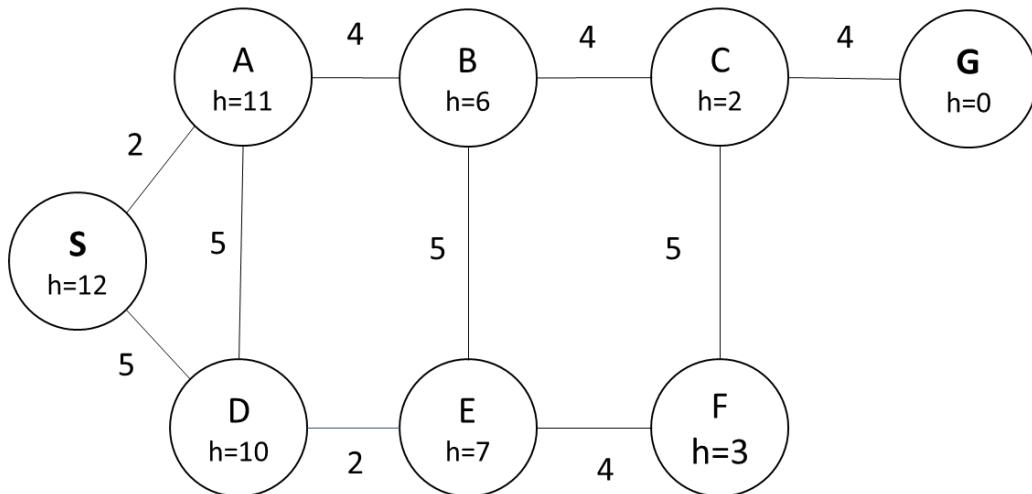
Duración total del examen: 2 horas y media

Problema 21.

Considera el grafo dibujado abajo que representa el espacio de estados donde S es el estado inicial, y G es el estado objetivo. Todos los arcos son bidireccionales.

Para cada una de las siguientes estrategias, muestra el camino devuelto, o escribe ninguno si no devuelve ningún camino solución. En caso de empates, asume que se expanden por orden alfabético. Pinta los árboles de búsqueda, representando el orden de expansión de nodos y muestra el estado de la FRONTERA y EXPLORADOS, si procede, en cada ciclo de la búsqueda. Indica el coste de la solución y el camino encontrado.

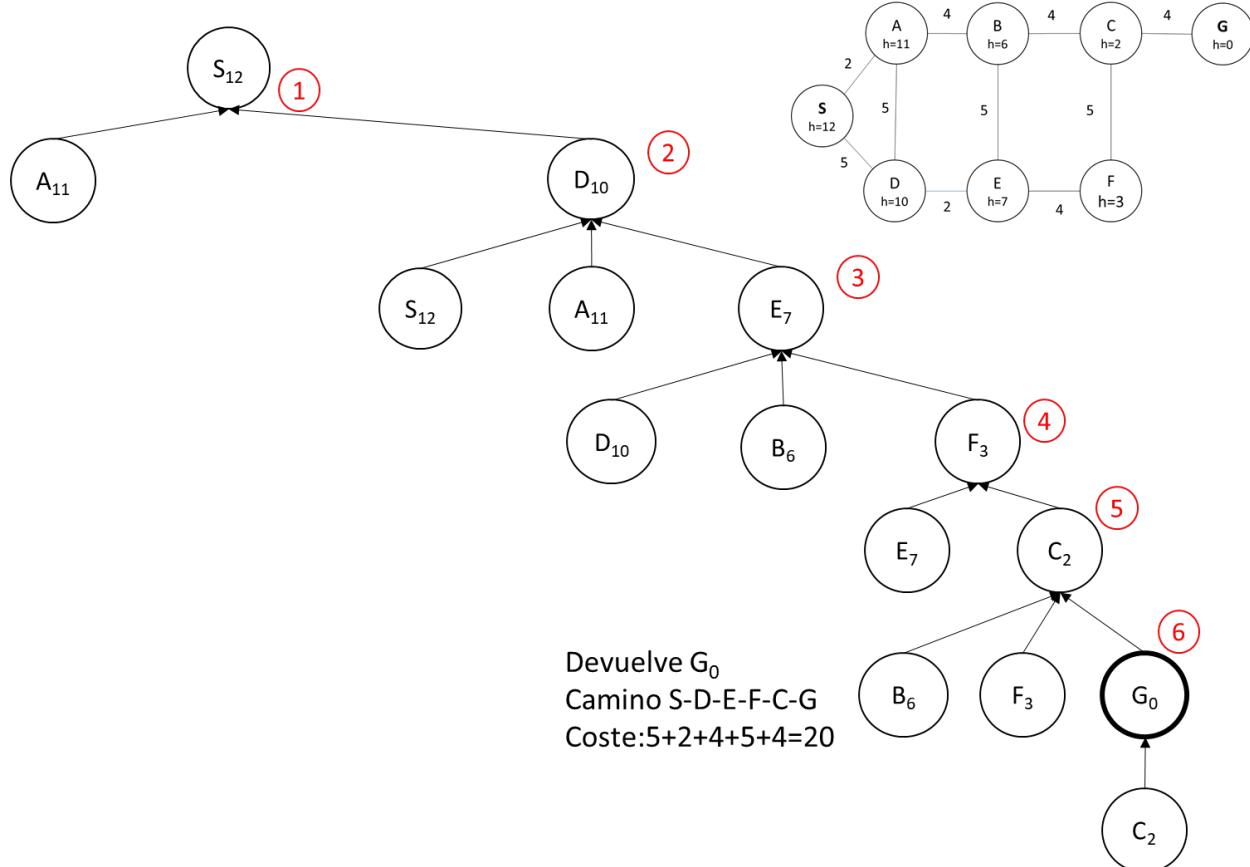
- (a) [0,5 ptos] Búsqueda local (Escalada, buscando el mínimo).
- (b) [0,5 ptos] Búsqueda en grafo A\*.
- (c) [0,5 ptos] ¿Es la heurística representada admisible?  
¿Es la heurística representada consistente?  
Explica tus respuestas.





Duración total del examen: 2 horas y media

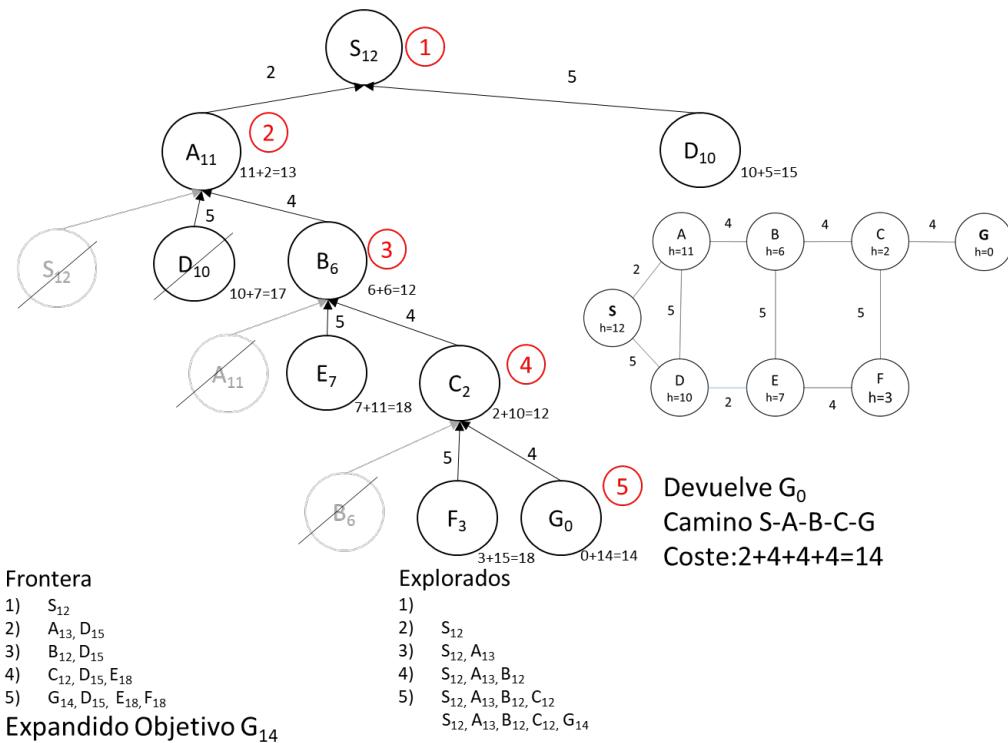
(a) [0,5 ptos] Búsqueda local (Escalada, buscando el mínimo).





Duración total del examen: 2 horas y media

(b) [0,5 ptos] Búsqueda en grafo A\*.



(c) [0,5 ptos]

¿Es la heurística representada admisible? Explica tu respuesta.

*Si. Todos los nodos tienen un valor de la función heurística menor que el coste óptimo para llegar al objetivo desde ese nodo, es decir  $h(n) \leq h^*(n)$*

¿Es la heurística representada consistente? Explica tu respuesta.

*No. Si observamos la búsqueda A\* en grafo realizada, en el paso 2 expandimos un nodo A de valor  $f=h+c = 13$ , y en el paso 3 el nodo B de valor  $f= 12$ . Si la heurística fuera consistente se garantizaría que no se expande un nodo si quedan nodos con un valor f menor o igual. Esto nos da pistas de que no es consistente.*

*La heurística es consistente si para todos los nodos  $h(n) \leq c(n,n') + h(n')$ . En los siguientes casos no se cumple:*

$$h(A) = 11 \leq h(B)+c(A,B)= 6 +4 = 10$$

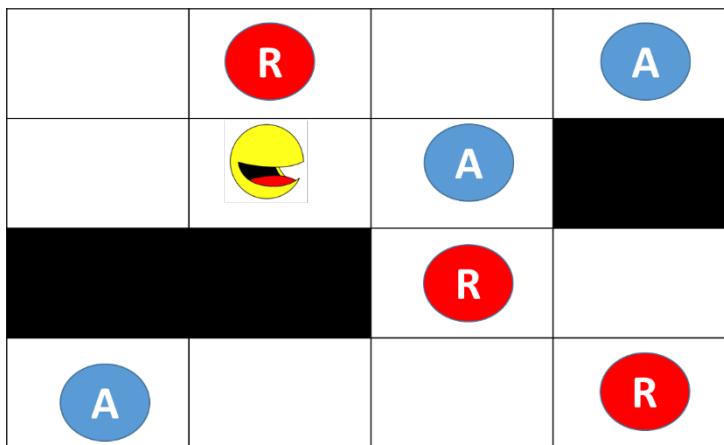
$$h(D) = 10 \leq h(E)+c(D,E)= 7 +2 = 9$$



Duración total del examen: 2 horas y media

Problema 22.

El comecocos representado en el tablero puede comer dos clases de piezas de fruta representadas por los colores rojo (R) y azul (A). El objetivo del comecocos es saborear los dos tipos de frutas: Una vez que ha comido una fruta de cada tipo el juego finaliza (aunque el comecocos podría comer más de una pieza de cada fruta antes de que el juego acabe). Los cuatro movimientos posibles son arriba, abajo, izquierda y derecha. Debe moverse en todos los turnos. No es posible cruzar obstáculos (casillas negras). Se considera que la fruta que ocupa una casilla es comida una vez que el comecocos alcanza la casilla.



Se pide:

1. Define una **representación del problema**, especificando una representación del estado, estado inicial, estados finales, así como operadores(es) y su coste.

*El estado viene dado por una tupla  $(x,y, ComR, ComA)$  donde  $x \in [1,4]$ ,  $y \in [1,4]$ ,  $ComR \in \{T,F\}$  y  $ComA \in \{T,F\}$*

*El **estado inicial** viene dado por  $(2,2, F, F)$*

*Se define el **test objetivo** como  $(ComR=T \& ComA = T)$  que cumplen todas las tuplas  $(x,y, T, T)$ .*

El **modelo de transiciones** debe representar las **precondiciones** de cada acción y sus **efectos**

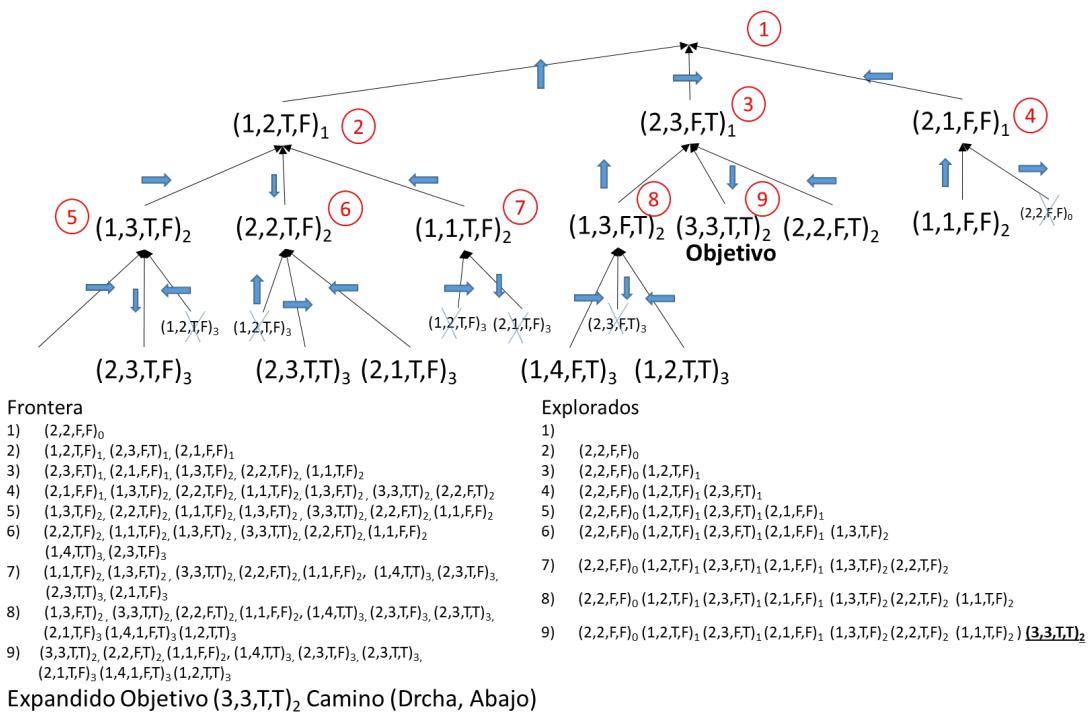


Duración total del examen: 2 horas y media

Acciones	Precondiciones	Resultado
Arriba (x,y,ComR,ComA)	y > 2, tablero[x,y-1] != negro	si (tablero[x,y-1]=A) comA=T, si (tablero[x,y-1]=R) comR=T, y=y-1
Abajo (x,y,ComR,ComA)	y < 3, tablero[x,y+1] != negro	si (tablero[x,y+1]=A) comA=T, si (tablero[x,y+1]=R) comR=T, y=y+1
Izq (x,y,ComR,ComA)	x > 2, tablero[x-1,y] != negro	si (tablero[x-1,y]=A) comA = T, si (tablero[x-1,y]=R) comR = T, x=x-1
Dcha (x,y,ComR,ComA)	x < 3, tablero[x+1,y] != negro	si (tablero[x+1,y]=A) comA = T, si (tablero[x+1,y]=R) comR = T, x=x+1

**Coste camino** = Suma de número de pasos.

2. Dibuja el árbol de búsqueda, representando el orden de expansión de nodos y muestra el estado de la FRONTERA y EXPLORADOS, si procede, en cada ciclo de la búsqueda de Coste Uniforme en grafo. [0,25 ptos]



3. Para cada una de las siguientes heurísticas indica si es admisible o no [0,25 ptos]



*Duración total del examen: 2 horas y media*

- a) El número de frutas sin comer que hay en el tablero **NO**
  - b) La menor distancia de Manhattan a una de las frutas que hay sin comer en el tablero. **NO**
  - c) La máxima distancia de Manhattan entre cualquiera dos frutas que permanecen sin comer en el tablero. **NO**
  - d) La mínima distancia de Manhattan entre cualesquiera dos frutas de color opuesto. **NO**
4. Define una **función heurística diferente de las anteriores** que sea admisible. [0,5 ptos]

*El error de las heurísticas anteriores es que puedes haber alcanzado el objetivo (haber comido cada una de las clases de fruta), y quedar frutas sin comer en el tablero.*

*Una heurística admisible sería:*

*La menor distancia de Manhattan a una de las clases de fruta que quedan sin comer más la menor distancia de Manhattan de esta a la otra clase que queda sin comer si es el caso. En caso de que no queden clases de fruta sin comer la heurística será cero.*



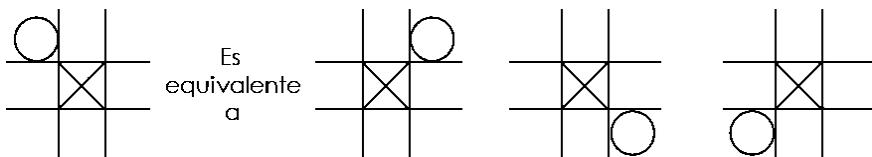
Duración total del examen: 2 horas y media

**Problema 23.**

El juego del TRES EN LÍNEA, es un juego de dos jugadores (X e Y) que marcan alternadamente los espacios de un tablero de 3x3, empezando primero el jugador X. Gana el que logra colocar sus tres fichas en línea primera, y se empata si se rellenan todas las casillas sin que haya tres en línea. Se define  $X_n$  como el número de filas, columnas o diagonales con exactamente  $n$  Xs, y sin ninguna O. De la misma forma se define  $O_n$ , como el número de filas, columnas o diagonales con exactamente  $n$  Os, y ninguna X. La función de utilidad asigna +1 a cualquier posición con  $X_3 = 1$  y -1 a cualquier posición con  $O_3 = -1$ . Todas las demás posiciones terminales tienen el valor 0. *Para las posiciones no terminales*, utilizaremos la siguiente *función de evaluación*:  $\text{Eval}(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$ .

1. **Muestra el árbol de juego desde la posición inicial con el tablero vacío hasta la profundidad 2 (Es decir, una X y una O en el tablero), teniendo en cuenta la simetría. [0,5 ptos].**

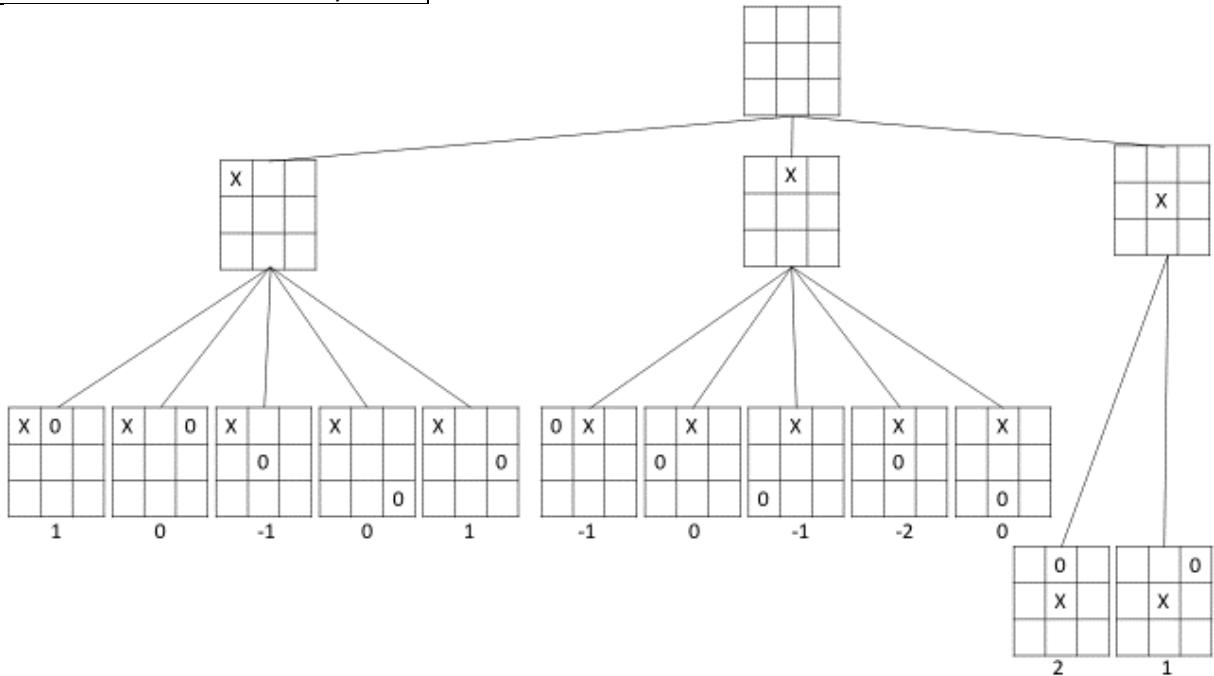
Por ejemplo, las siguientes posiciones son equivalentes y sólo aparecerá la primera representándolas en el árbol de juego.



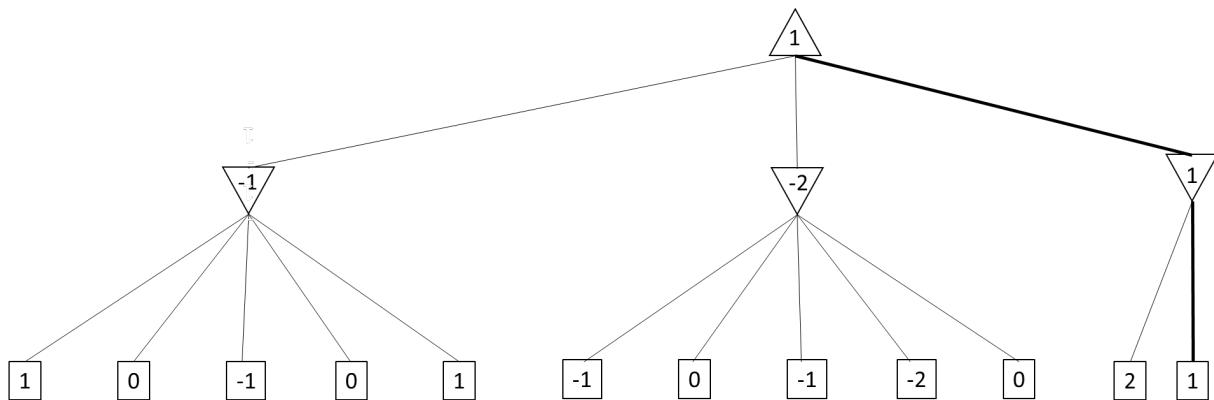
2. **Marca en el árbol la evaluación de todas las posiciones en profundidad 2 con la función de evaluación dada. [0,5 ptos]**



Duración total del examen: 2 horas y media



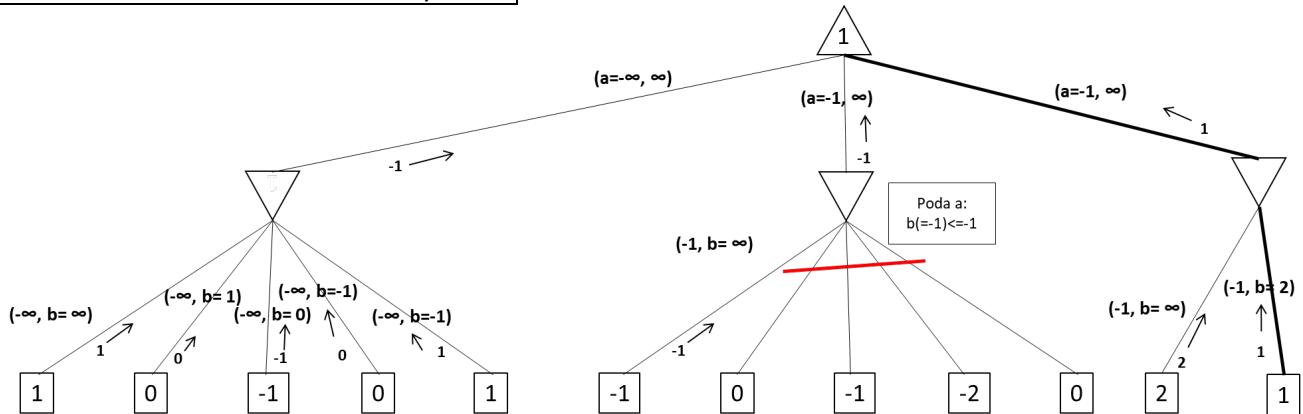
3. Realiza sobre el árbol la evaluación **MINIMAX**, indicando que rama elegirá el primer jugador X (MAX), para ganar. [0,5 ptos]



4. Realiza la poda alfa-beta sobre el árbol de juego, y marca rodeando con un círculo los nodos que no serán evaluados en tu árbol de juego con la poda. [0,5 ptos]



Duración total del examen: 2 horas y media



#### Problema 24.

La compañía “*Magacén*” utiliza empresas locales para transportar mercancías a su destino. Para una mercancía concreta cuenta con tres tramos que pueden realizarse por 3 distintas empresas. El reparto debe realizarse antes de 48 horas, y como política de la compañía nunca se contrata a la misma empresa de transporte para realizar más de un tramo. En el caso de la mercancía a transportar hay 3 empresas que pueden realizar el transporte en cada uno de los tres tramos necesarios para cubrir el transporte. La tabla siguiente representa los tramos a cubrir por las empresas candidatas, mostrando el coste en euros, y el tiempo empleado en horas.

	Tramo Tr <sub>1</sub>	Tramo Tr <sub>2</sub>	Tramo Tr <sub>3</sub>
Empresa E <sub>1</sub>	2 €, 35 h	3 €, 16 h	2 €, 5 h
Empresa E <sub>2</sub>	5 €, 10 h	5 €, 35 h	4 €, 25 h
Empresa E <sub>3</sub>	10 €, 6 h	8 €, 8 h	6 €, 4 h

El problema consiste en decidir qué tramo se asignará a cada empresa, de modo que se **minimice el coste total respetando las restricciones de tiempo y la política de empresa**. Los técnicos deciden utilizar el algoritmo A\* para resolver el problema.

- (a) Define una **representación “eficiente” del problema**, especificando el conjunto de posibles estados, estado inicial, estados finales, así como operador(es) y su coste



Duración total del examen: 2 horas y media

- ESTADO: Un estado se puede representar por una tupla de tres elementos, en la que el primer elemento representa a que empresa se asigna el primer tramo, el segundo elemento la empresa a la que se asigna el segundo tramo, y el tercero la obra a la que se asigna el tercer tramo. Los elementos no asignados podemos representarlos en blanco, o de forma incremental ir representando tuplas de 1, 2 y 3 elementos. Las restricciones del estado son que no se pueden repetir empresas, y el tiempo no puede superar 48 horas, por lo que incluiremos también el tiempo acumulado por las obras asignadas en el estado. Representamos el estado con la tupla  $([E_i, E_j, E_k], t)$  con  $i \neq j, j \neq k, i \neq k$ , donde:
  - $E_i \in \{E_1, E_2, E_3, \text{null}\}$  donde null indica no asignado. Utilizaremos una representación incremental, asignando en orden los tramos. Representaremos  $[\text{null}, \text{null}, \text{null}]$  con  $[]$ ,  $[E_i, \text{null}, \text{null}]$  con  $[E_i]$  y  $[E_i, E_j, \text{null}]$  con  $[E_i, E_j]$ .
  - $t \in [0, 48]$  indica el tiempo acumulado por las empresas en hacer los tramos asignados.

Estado inicial :  $([], )$

Estado objetivo:  $([E_i, E_j, E_k], t)$

El **modelo de transiciones** debe representar las **precondiciones** de cada acción y sus **efectos**

Acciones	Precondiciones	Resultado
Asignar1Tramo	$([], 0)$ , Duración( $E_i, Tr_1$ ) $\leq 48$	$([E_i], \text{Duración}(E_i, Tr_i))$
Asignar2Tramo	$([E_i], t)$ , $t + \text{Duración}(E_j, Tr_2) \leq 48$ , $i \neq j$	$([E_i, E_j], t + \text{Duración}(E_j, Tr_2))$
Asignar3Tramos	$([E_i, E_j], t)$ $t + \text{Duración}(E_k, Tr_3) \leq 48$ , $i \neq k$ , $j \neq k$	$([E_i, E_j, E_k],$ $t + \text{Duración}(E_k, Tr_3))$ ,

**Coste camino** = Suma de coste en euros de las obras asignadas.



Duración total del examen: 2 horas y media

- (b) Define dos buenas **funciones heurísticas h** optimistas (admisibles) para el problema. Comenta como has llegado a definir cada una de las heurísticas.

Define dos heurísticas: Las heurísticas las obtenemos de relajar las restricciones del problema que son **las restricciones de tiempo y la política de empresa**.

**h<sub>1</sub>**: La primera heurística relaja las dos restricciones, permitiendo que la suma de tiempos de las obras asignadas sea mayor que 48 horas, y repetir empresas para estimar el coste de los tramos sin asignar. Se permite repetir empresas entre las pendientes de asignar. Permitir repetir utilizando las ya asignadas es admisible pero no aporta nada de información a la heurística.

**h<sub>2</sub>**: La segunda heurística, más informada, sólo relaja la opción de repetir las empresas para estimar el coste de los tramos sin asignar.

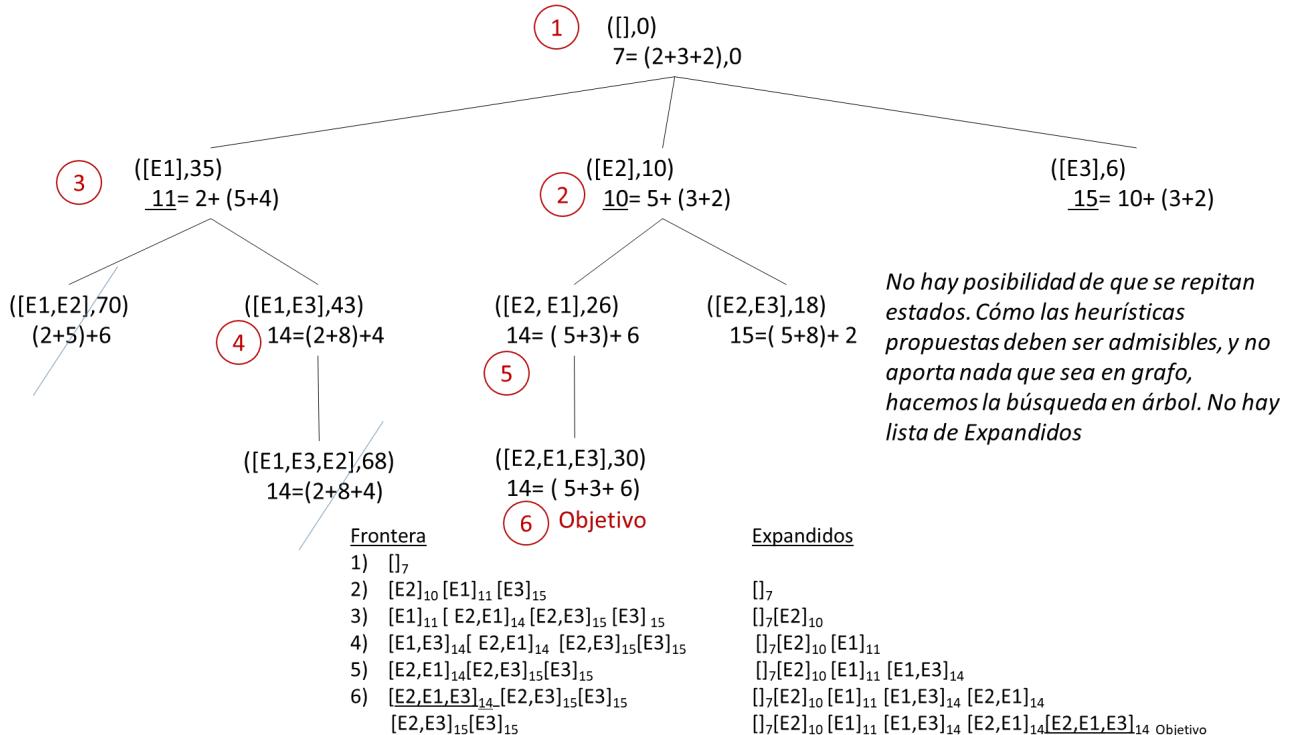
Ambas heurísticas son optimistas al relajar las restricciones del problema.

- (c) **Desarrolla el árbol de búsqueda que genera el algoritmo A\* para las dos heurísticas propuestas.** Indica el orden en el que se expanden los nodos y los valores de f, g y h para cada nodo del árbol de búsqueda. Muestra el estado de la FRONTERA y EXPLORADOS, si procede, en cada ciclo de la búsqueda. En caso de empate en la función heurística, considera que se expanden primero los nodos que has dibujado más a la izquierda en tu árbol de búsqueda.

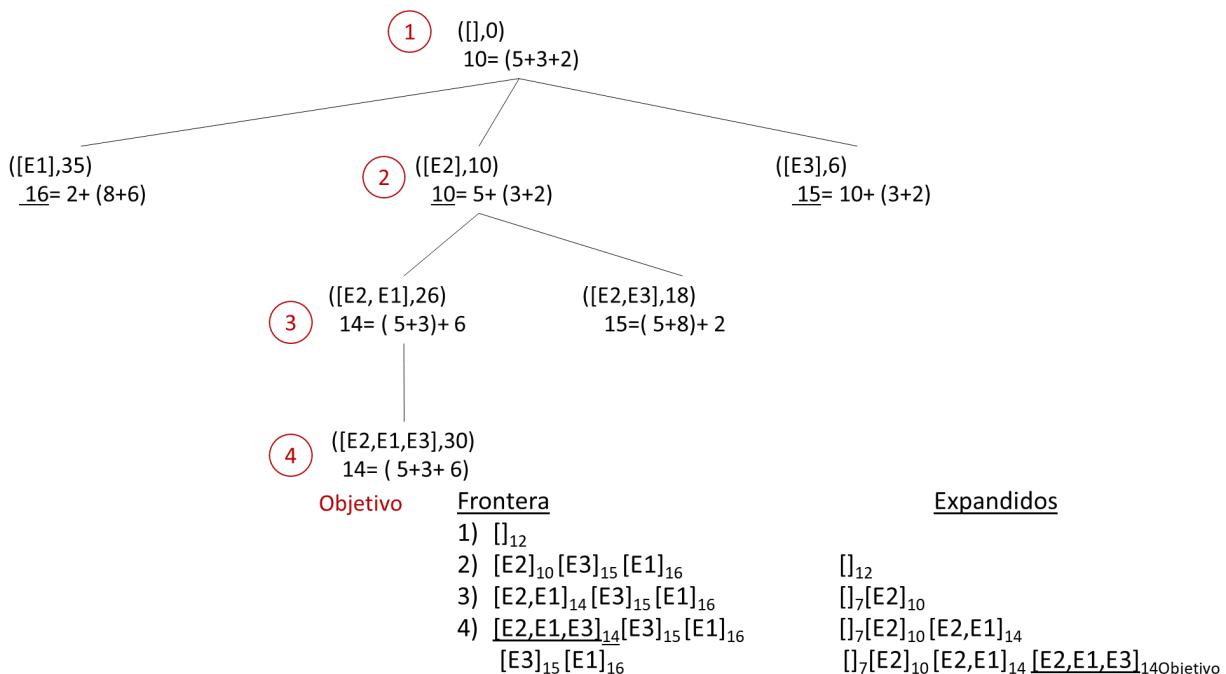


Duración total del examen: 2 horas y media

Árbol de búsqueda A\* con  $h_1$ :



Árbol de búsqueda A\* con  $h_2$ :





Duración total del examen: 2 horas y media

### Problema 25.

El juego Otelo4 se juega en un tablero de 4x4 como el mostrado en la figura y consiste en cubrir todo el tablero con fichas blancas (B) y negras (N), colocándolas alternativamente, de manera que gane el que más fichas de su color consiga. Las reglas para jugar al Oterlo4 son las siguientes: **Para colocar una ficha en el tablero hay que tener en esa fila, columna o diagonal otra ficha propia que encierre fichas contrarias con la que colocamos.** Si no se encierra ficha contraria no es posible colocar ficha. Al colocar la ficha, todas las fichas de color contrario que queden atrapadas entre dos fichas propias cambian su color al nuestro. El tablero comienza con las fichas que aparecen en la **figura 1**.

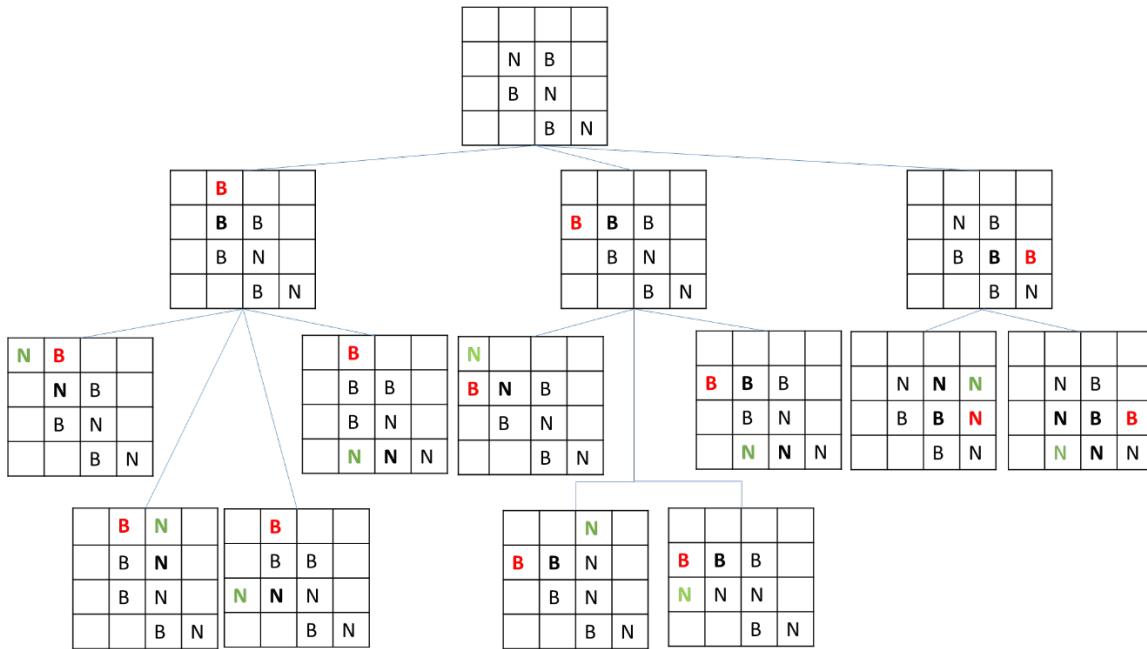
	N	B	
	B	N	
		B	N

Fig. 1 Estado inicial del juego

4	2	2	4
2	1	1	2
2	1	1	2
4	2	2	4

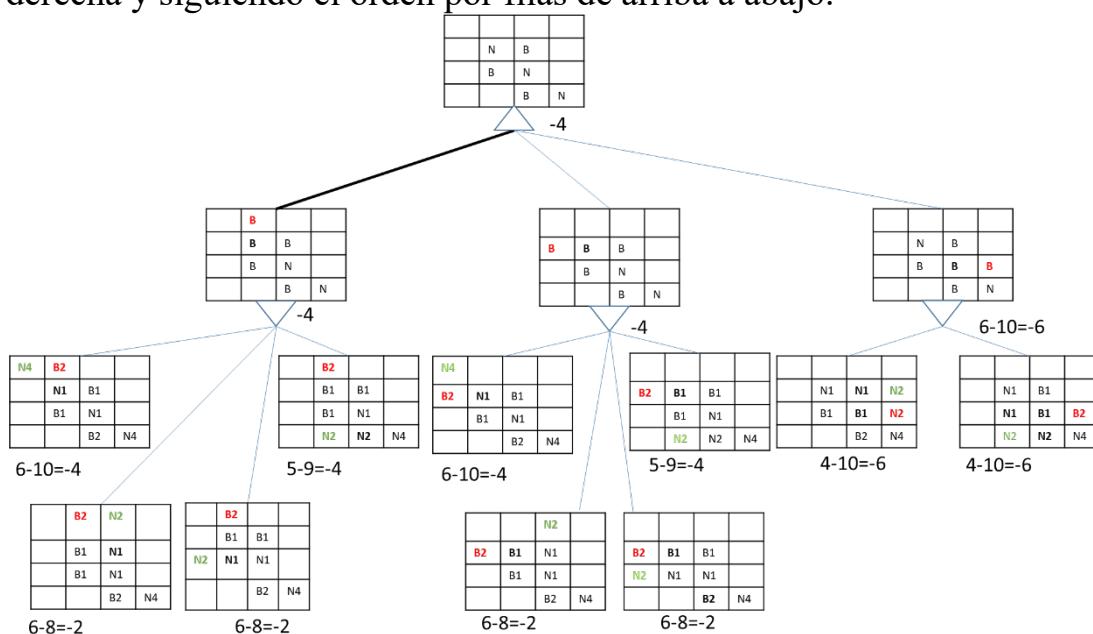
Fig.2 Valoración de las piezas según posición

- 1) Representa el árbol de juego explorando hasta profundidad 2 (un movimiento de las blancas y otro de las negras).



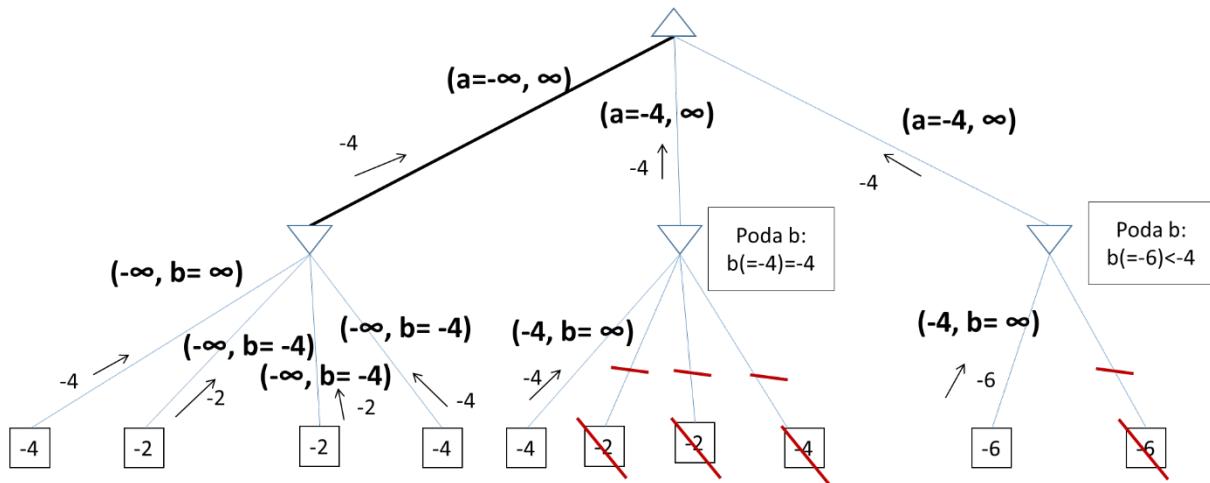
2) Dibuja el árbol de juego, y sobre el árbol realiza la evaluación **MINIMAX**, indicando cual es la rama que elegirá el primer jugador para ganar.

Para la evaluación de los estados utiliza la tabla de la **figura 2** para asignar valor a cada ficha en función de la posición que ocupa en el tablero. El valor de un estado se obtiene sumando los valores de las fichas propias y restando las del contrario. **Orden de generación de sucesores:** Para expandir los nodos debes intentar los movimientos en el tablero desde la esquina superior izquierda hacia la derecha y siguiendo el orden por filas de arriba a abajo.





3) Utiliza el algoritmo de **poda alfa-beta sobre el árbol de juego** para averiguar qué movimiento deberían hacer las blancas desde la posición inicial (marca en el árbol de juego los valores alfa y beta, dónde se produce poda, cruza una raya cortando las ramas no exploradas y tacha los nodos terminales no visitados).





### Problema 26.

Dada una lista de números, escribe un programa CLIPS que ordene dicha lista. Se pide completar las reglas necesarias para que funcione la siguiente función CLIPS:

```
(deffunction ordena($?lista)
  (reset)
  (assert (lista ?lista))
  (run))
```

Ejemplo de uso:

```
CLIPS>(ordena 6 4 9 3 23 44 5 3)
VALORES ORDENADOS:(3 3 4 5 6 9 23 44)
CLIPS>
```

```
(defrule intercambia
  ?l<- (lista $?b ?valor1 ?valor2&:(> ?valor1 ?valor2) $?e)
  =>
  (retract ?l)
  (assert (lista $?b ?valor2 ?valor1 $?e)))

(defrule ordenado
  ?lista <- (lista $?valores)
  (not (lista $?b ?valor1 ?valor2&:(> ?valor1 ?valor2) $?e))
  =>
  (printout t "VALORES ORDENADOS:" $?valores crlf))

(deffunction ordena($?lista)
  (reset)
  (assert (lista ?lista))
  (run))
```



Escuela de  
Ingeniería y Arquitectura  
**Universidad Zaragoza**

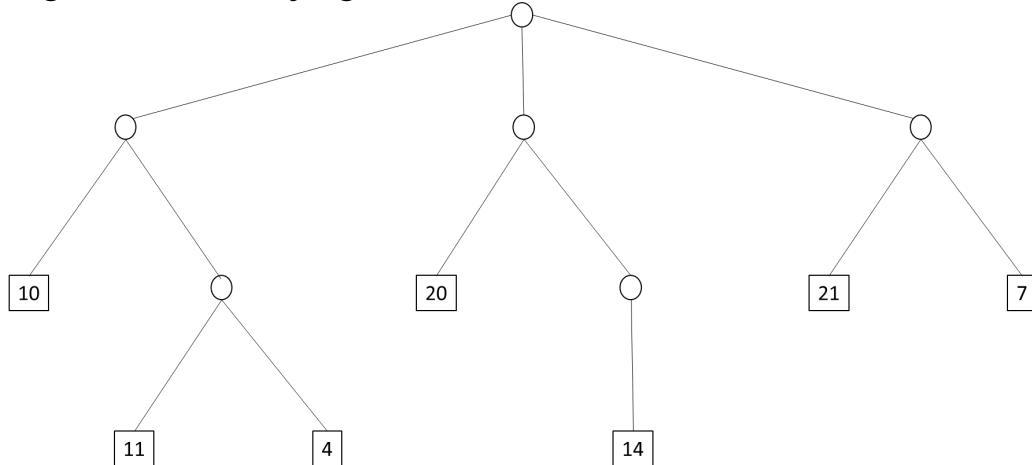
**Ejercicios de  
Inteligencia Artificial.  
Curso 20-21**





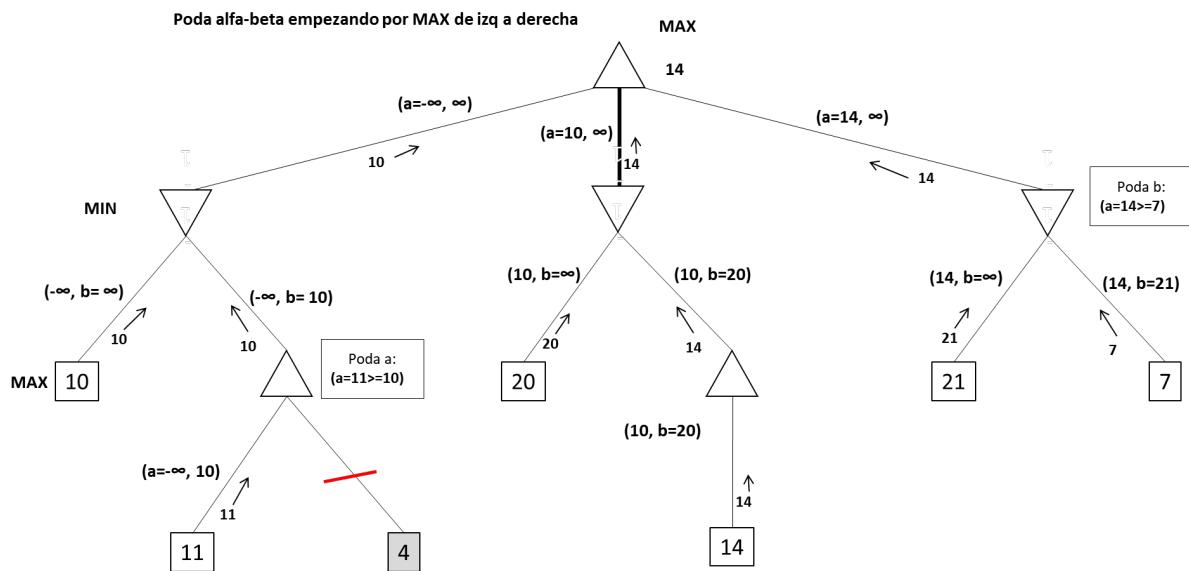
Problema 27.

Dado el siguiente árbol de juego:



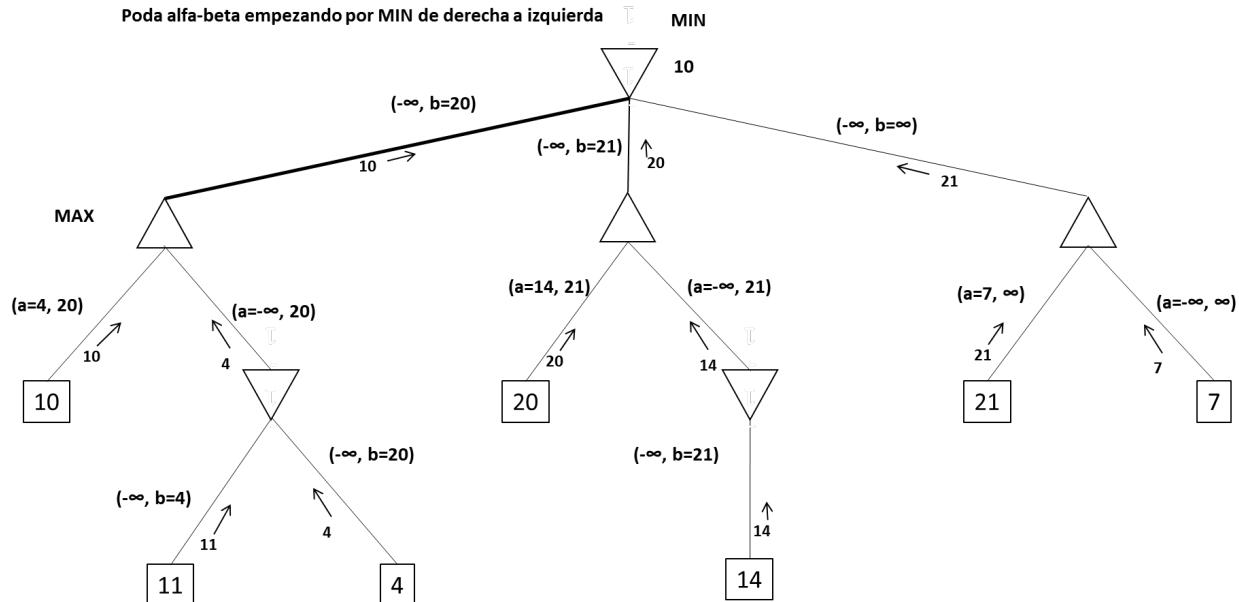
donde los valores numéricos que aparecen en los nodos hoja corresponden a estimaciones de lo prometedoras que son para el jugador MAX las situaciones de la partida representadas por dichos nodos. Aplicar el método de poda alfa-beta al árbol anterior para los siguientes casos:

- a) Suponiendo que el nodo raíz es un nodo MAX y el recorrido se realiza de izquierda a derecha. ¿Cuál es la decisión o jugada más acertada para MAX en éste caso? ¿Cuál es el valor obtenido por MAX?



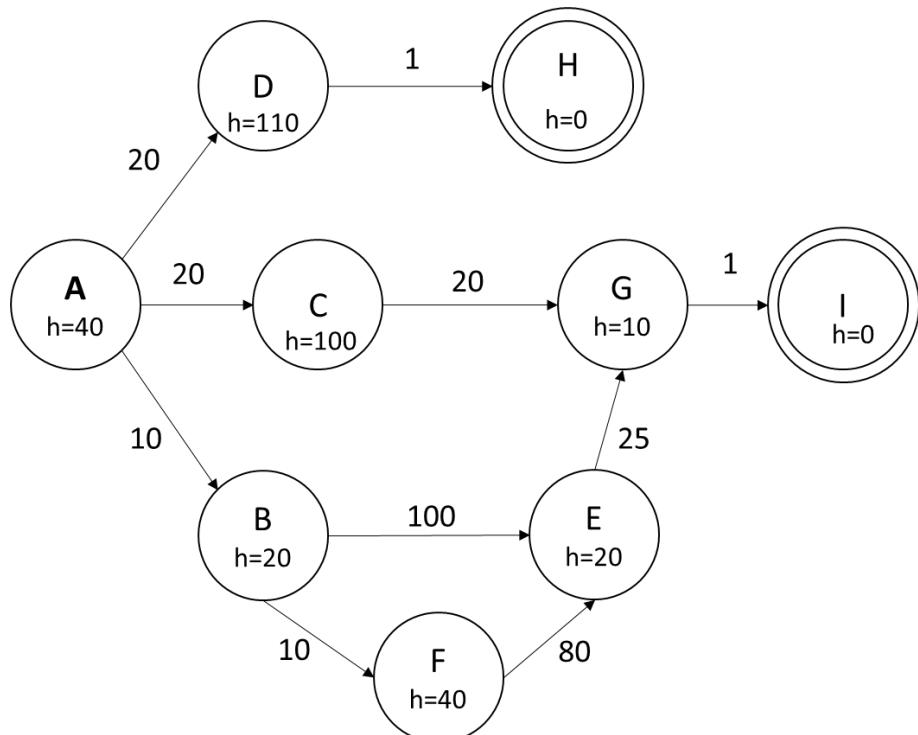


b) Suponiendo que el nodo raíz es un nodo MIN y el recorrido se realiza de derecha a izquierda. ¿Cuál es la decisión o jugada más acertada para MIN en éste caso? ¿Cuál es el valor obtenido por MIN?



#### Problema 28

Considera el grafo dibujado que representa el espacio de estados donde A es el estado inicial, y los nodos H e I son los nodos objetivo. Cada arco está etiquetado con su coste y en cada nodo aparece la estimación de la menor distancia desde ese nodo al objetivo.





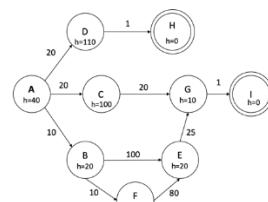
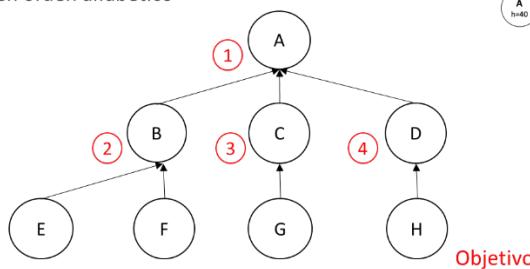
Para cada una de las siguientes estrategias de búsqueda, muestra el camino devuelto y su coste, o escribe ninguno si no devuelve ningún camino solución. Pinta los árboles de búsqueda, representando el orden de expansión de nodos y muestra el estado de la FRONTERA y EXPLORADOS, si procede, en cada ciclo de la búsqueda.

(a) [0,5 ptos] Búsqueda en **anchura en grafo**. En caso de empate expandir en orden alfabético. Test objetivo al crear nodo.

Búsqueda en Anchura en Grafo

Test objetivo al crear nodo

En caso de empate, colocar en frontera en orden alfabético



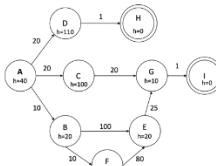
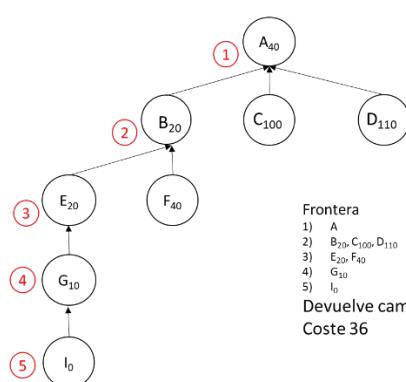
Devuelve  $G_0$   
Camino A-D-H  
Coste:  $20+1=21$

Frontera  
1) A  
2) B, C, D  
3) C, D, E, F  
4) D, E, F, G  
5) E, F, G, H  
Test Objetivo H encontrado

Explorados  
1)  
2) A  
3) A, B  
4) A, B, C  
5) A, B, C, D

(b) [0,5 ptos] Búsqueda local (Escalada, buscando el mínimo).

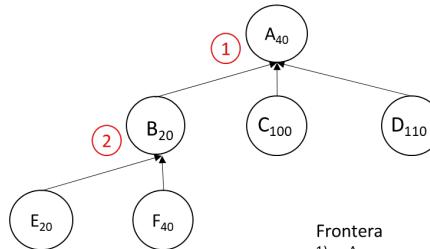
Búsqueda Local (Escalada, buscando mínimo)  
Si permitimos progresar en llano



Frontera  
1) A  
2) B<sub>20</sub>, C<sub>100</sub>, D<sub>110</sub>  
3) E<sub>20</sub>, F<sub>40</sub>  
4) G<sub>10</sub>  
5) I<sub>0</sub>  
Devuelve camino A-B-E-G-I  
Coste 36



Búsqueda Local (Escalada, buscando mínimo)  
Si No permitimos progresar en llano



Frontera  
1) A  
2) B<sub>20</sub>, C<sub>100</sub>, D<sub>110</sub>  
3) E<sub>20</sub>, F<sub>40</sub>

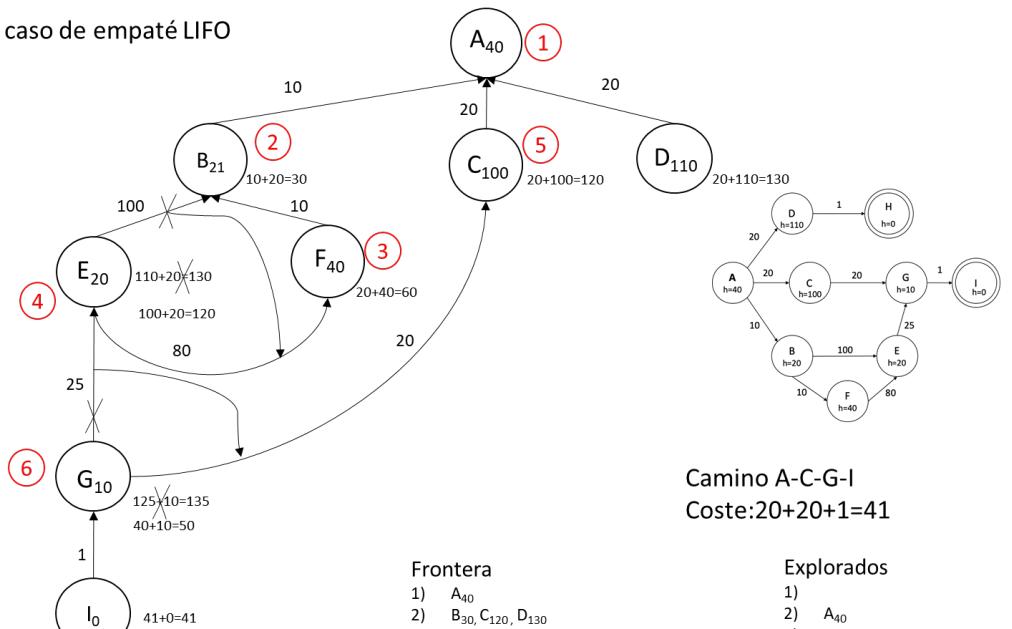
Devuelve camino A-B  
Coste 10. No encuentra solución.  
Mínimo local (llano).

(c)

[1  
ptos]

Búsqueda en grafo A\*. En caso de empates, asume que se expande primero el último en entrar en la cola, o primero el último que actualiza el coste, para resolver el empate.

A\*, En caso de empate LIFO



Camino A-C-G-I  
Coste: 20+20+1=41

Frontera  
1) A<sub>40</sub>  
2) B<sub>21</sub>, C<sub>100</sub>, D<sub>130</sub>  
3) F<sub>60</sub>, C<sub>120</sub>, E<sub>130</sub>, D<sub>130</sub>  
4) E<sub>120</sub>, C<sub>120</sub>, D<sub>130</sub>  
5) C<sub>120</sub>, D<sub>130</sub>, G<sub>135</sub>  
6) G<sub>50</sub>, D<sub>130</sub>  
7) I<sub>41</sub>, D<sub>130</sub>  
8) D<sub>130</sub>

Explorados  
1)  
2) A<sub>40</sub>  
3) A<sub>40</sub>, B<sub>30</sub>  
4) F<sub>60</sub>, A<sub>40</sub>, B<sub>30</sub>  
5) E<sub>120</sub>, F<sub>60</sub>, A<sub>40</sub>, B<sub>30</sub>  
6) C<sub>120</sub>, E<sub>120</sub>, F<sub>60</sub>, A<sub>40</sub>, B<sub>30</sub>  
7) G<sub>50</sub>, C<sub>120</sub>, E<sub>120</sub>, F<sub>60</sub>, A<sub>40</sub>, B<sub>30</sub>  
8) I<sub>0</sub>, G<sub>50</sub>, C<sub>120</sub>, E<sub>120</sub>, F<sub>60</sub>, A<sub>40</sub>, B<sub>30</sub>

Expandido Objetivo I<sub>0</sub>

(d) [0,5 ptos] ¿Es la heurística representada admisible?  
¿Es la heurística representada consistente?  
Explica tu respuesta.

No es admisible  $h(D)=110$  es mayor que el coste para llegar al objetivo desde D que es 1. Ni consistente  $h(D) \neq 1+h(H)$

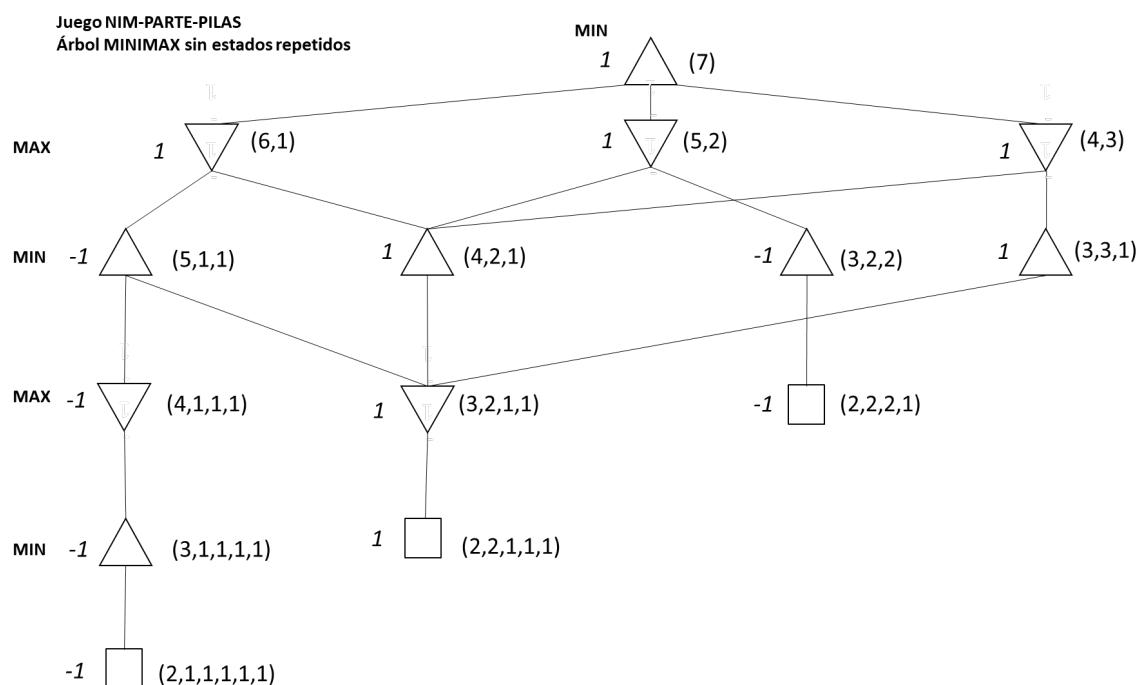


### Problema 29.

El NIM-PARTE-PILAS es un juego con dos jugadores con las siguientes reglas: El juego comienza con una simple pila de monedas iguales. El movimiento de un jugador consiste en dividir una de las pilas **en dos pilas que contengan diferentes números de monedas**. Por ejemplo, si una pila contiene seis monedas, esta se puede dividir en pilas de 5 y 1, o 4 y 2, pero no en pilas de 3 y 3. En el siguiente movimiento de este ejemplo, partiendo de las pilas de 5 y 1 puedes dividir la pila de 5 en pilas de 4 y 1 obteniendo tres pilas (4,1,1), o dividir la pila de 5 en pilas de 3 y 2 obteniendo tres pilas (3,2,1). Partiendo de las pilas de 4 y 2, puede partir la pila de 4 en dos pilas de 3 y 1 obteniendo tres pilas (3,1,2). (No está permitido partir la pila de 4 en dos pilas de 2 monedas).

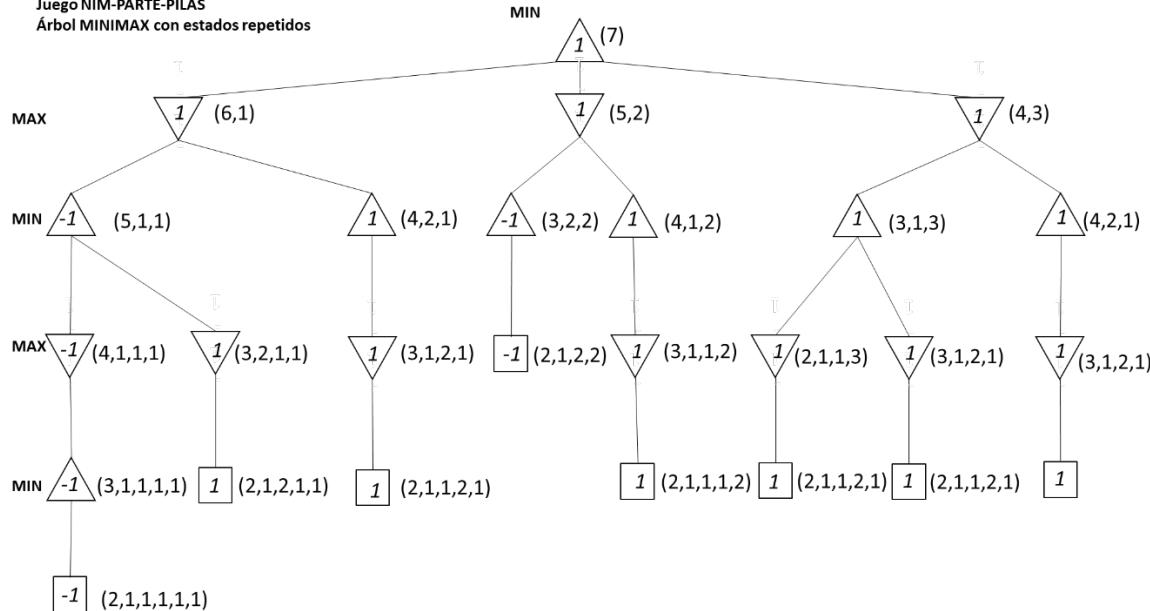
**El primer jugador que no puede realizar movimiento (porque no puede partir ninguna de las pilas en pilas de valor diferente) pierde.**

Dibujar el árbol de juego completo para esta versión del NIM si el juego **comienza por una pila de 7 monedas y comienza a jugar MIN**. Realiza la evaluación MINIMAX sobre este árbol de juego evaluando 1 un nodo terminal si MAX gana, y -1 si MIN gana. ¿Quién gana y qué movimiento debe realizar para ganar?





Juego NIM-PARTE-PILAS  
Árbol MINIMAX con estados repetidos



### Problema 30.

La siguiente figura representa el estado inicial con cinco monedas colocadas en línea donde la O indica CARA y la C CRUZ:

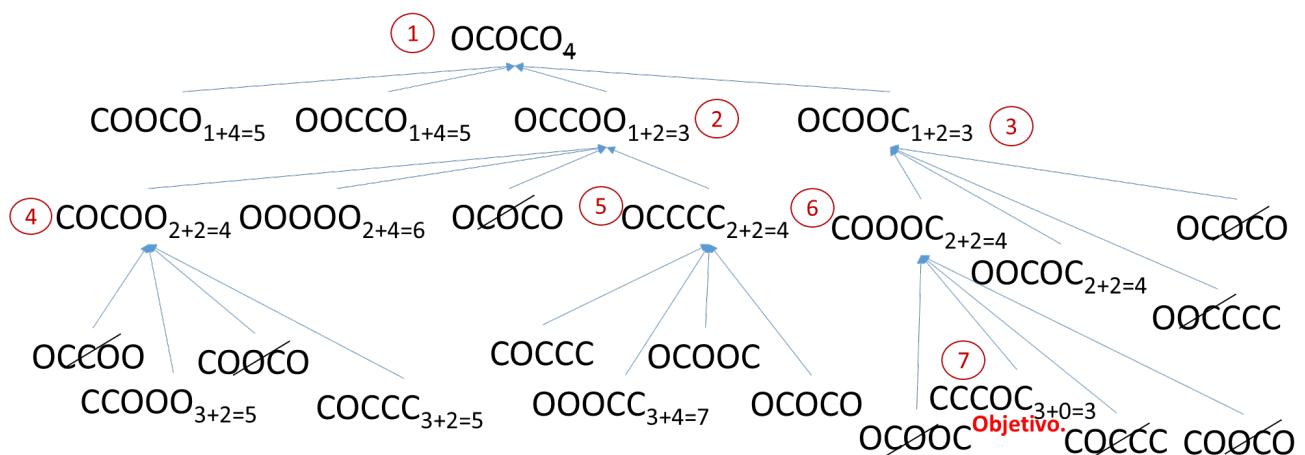
O C O C O

Los **movimientos posibles de coste 1** son dar la vuelta a dos monedas contiguas. El **objetivo** del problema es llegar al estado siguiente:

C C C O C

Dada la **función heurística**  $h(n) = \text{número de monedas mal colocadas}$

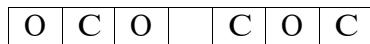
Desarrolla el árbol de búsqueda que genera el algoritmo A\* en grafo para la heurística propuestas. Indica claramente sobre el árbol de búsqueda el orden en el que se expanden los nodos, los valores de f, g y h para cada nodo, el tratamiento de nodos duplicados, el camino obtenido, y su coste. En caso de empate en la función heurística, considera que se expanden primero los nodos que has dibujado más a la izquierda en tu árbol de búsqueda.





### Problema 31.

La siguiente figura representa un tablero de juego con siete casillas que contienen seis monedas colocadas inicialmente como se muestra, donde O indica CARA y la C CRUZ. El jugador **MAX** tiene las fichas marcadas como O y el **MIN** las fichas marcadas como C. Los jugadores pueden mover sus fichas y las del contrario en cada turno



Los movimientos posibles son dos:

- **Contigua:** Una moneda se puede desplazar a la casilla libre contigua
- **Salto:** Una moneda puede saltar sobre otra moneda contraria (sólo una) cambiando de signo la ficha contraria, si a continuación hay una casilla vacía. Ejemplo:  
 $\text{OCO\_COC} \Rightarrow \text{O\_CCCOC}$

**Restricción** sobre las acciones: una jugada de desplazamiento no es válida si deshace la jugada inmediatamente anterior. Por ejemplo, si un jugador pasa de OC\_COC al estado OCC\_COC, el otro jugador no puede volver a pasar a OC\_COC.

El **objetivo de cada jugador** es tener 4 fichas del mismo tipo estrictamente consecutivas

Para la evaluación de los estados se utilizará la siguiente función:

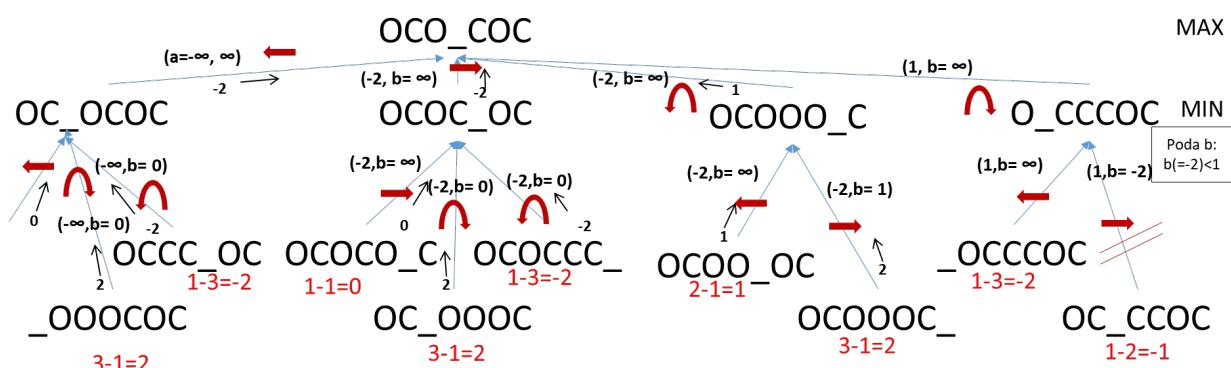
$f'(n)$  = tamaño del mayor grupo de Os consecutivas - tamaño del mayor grupo de Cs consecutivas.

Ejemplos:

$$f'(\text{OCO\_COC}) = 1 - 1 = 0$$

$$f'(\text{OCC\_COC}) = 1 - 2 = -1$$

Utiliza el algoritmo de **poda alfa-beta** para averiguar cual debería ser el primer movimiento del jugador MAX desde la posición inicial. Haz la exploración hasta el nivel 2 (una jugada de MAX y una de MIN). Aplica siempre el mismo orden: movimientos posibles recorriendo el tablero de izquierda a derecha. Para cada nodo muestra claramente la evolución de los valores alfa y beta. ¿Cuál es el movimiento que debería escoger MAX?





### Problema 31.

Dos algoritmos de búsqueda son equivalentes si y sólo si expanden los mismos nodos en el mismo orden y devuelven el mismo camino como solución. En este ejercicio se propone el algoritmo de **coste uniforme (CU)** utilizando como coste de las acciones para ir del nodo i al nodo j un valor, al que denominamos  $d_{ij}$ , que puede ser diferente del coste real de las acciones  $c_{ij}$ . Tienes que analizar si el algoritmo de coste uniforme CU utilizando las opciones propuestas de función de coste  $d_{ij}$  en las distintas cuestiones es equivalente a otro algoritmo de búsqueda. Para todas las cuestiones que siguen se supone: 1) los algoritmos de **búsqueda son en grafo**, 2) el valor del coste real  $c_{ij} > 0$  para ir del nodo i al j, 3) sólo hay un nodo objetivo en el problema, 4) en caso de empate se resolverán alfabéticamente asumiendo que los nodos tienen como nombre una letra o cadena de caracteres, y 5) las heurísticas son consistentes.

1. Marca todas las opciones de coste  $d_{ij}$  que hacen que ejecutar el algoritmo de **CU** con los costes  $d_{ij}$  propuestos lo hacen equivalente a ejecutar el algoritmo **búsqueda primero en anchura** [0,5 pts]

- $d_{ij} = 0$   
  $d_{ij} = 1$   
  $d_{ij} = -1$

- $d_{ij} = a, a > 0$   
  $d_{ij} = a, a < 0$   
Ninguna de las anteriores

2. Marca todas las opciones de coste  $d_{ij}$  que hacen que ejecutar el algoritmo de **CU** con los costes  $d_{ij}$  propuestos lo hacen equivalente a ejecutar el algoritmo **búsqueda primero en profundidad** [0,5 pts]

- $d_{ij} = 0$   
  $d_{ij} = 1$   
  $d_{ij} = -1$

- $d_{ij} = a, a > 0$   
  $d_{ij} = a, a < 0$   
Ninguna de las anteriores

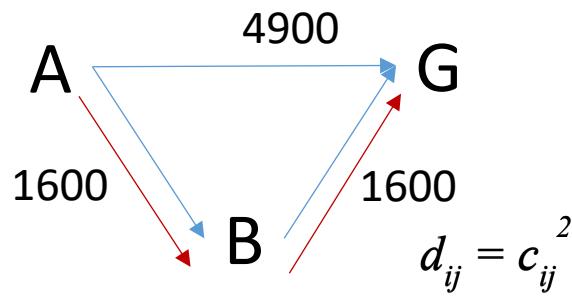
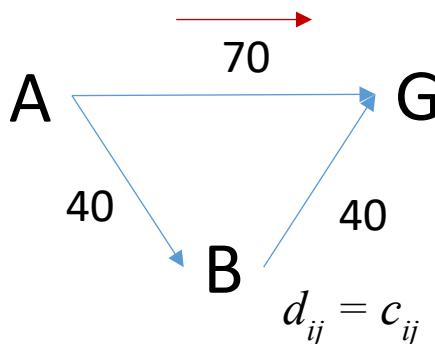


3. Marca todas las opciones de coste  $d_{ij}$  que hacen que ejecutar el algoritmo de CU con los costes  $d_{ij}$  propuestos lo hacen equivalente a ejecutar el algoritmo búsquedas **Coste Uniforme** con el coste real  $c_{ij} > 0$  [0,5 pts]

- $d_{ij} = c_{ij}^2$         $d_{ij} = c_{ij} + a, a > 0$   
  $d_{ij} = 1/c_{ij}$         $d_{ij} = a \cdot c_{ij} + b, a > 0, b > 0$   
  $d_{ij} = a \cdot c_{ij}, a > 0$        Ninguna de las anteriores

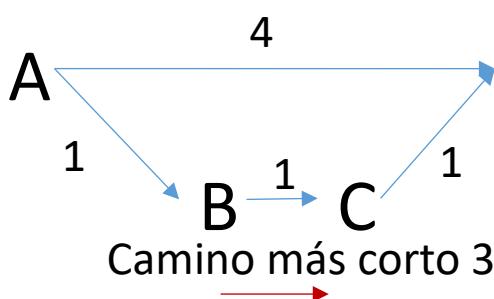
- No.  $d_{ij} = c_{ij}^2$  porque supone diferentes escalas

Camino más corto 70

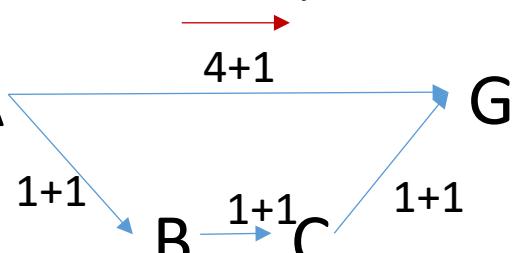


Menor  $d_{ij}$  3200

- No.  $d_{ij} = 1/c_{ij}$  porque son los valores inversos
- Si.  $\sum a \cdot c_{ij} = a \cdot \sum c_{ij}$ . El coste de todos los caminos queda multiplicado por el mismo factor  $a > 0$
- No.  $d_{ij} = c_{ij} + a, a > 0 \Rightarrow \sum c_{ij} + n(\text{pasos}_{\text{camino}}) \cdot a$



Menor  $d_{ij} = 5$



$d_{ii} = c_{ii} + 1, 1 > 0$

- No.  $d_{ij} = a \cdot c_{ij} + b, a > 0, b > 0$  por ser el la  $d_{ij}$  dependiente del número de pasos como el caso anterior.



4. Marca todas las opciones de coste  $d_{ij}$  que hacen que ejecutar el algoritmo de CU con los costes  $d_{ij}$  propuestos lo hacen equivalente a ejecutar el algoritmo búsquedas  $A^*$  con el coste real  $c_{ij} > 0$  y la función heurística  $h$  [0,5 pts]

- $d_{ij} = a.h(i), a > 0$         $d_{ij} = c_{ij} + h(i) - h(j)$   
  $d_{ij} = a.h(j), a > 0$         $d_{ij} = c_{ij} + h(j) - h(i)$   
  $d_{ij} = c_{ij} + h(i)$        Ninguna de las anteriores  
  $d_{ij} = c_{ij} + h(j)$

$A^* = \sum c_{ij} + h(n)$  Suma de los costes hasta el nodo mas la heurística del nodo.

- NO.  $d_{ij} = a.h(i), a > 0$  Supone la suma de heurísticas
- NO.  $d_{ij} = a.h(j), a > 0$  Supone la suma de heurísticas
- NO.  $d_{ij} = c_{ij} + h(i)$  Supone la suma de heurísticas
- NO.  $d_{ij} = c_{ij} + h(j)$  Supone la suma de heurísticas
- NO.  $d_{ij} = c_{ij} + h(i) - h(j)$   
 $\Rightarrow \sum c_{ij} + h(\text{inicio}) - h(2) + h(2) - h(3) + \dots h(n) =$   
 $\Rightarrow \sum c_{ij} + h(\text{inicio}) - h(n)$
  
- Si.  $d_{ij} = c_{ij} + h(j) - h(i)$   
 $\Rightarrow \sum c_{ij} + h(2) - h(\text{inicio}) + h(3) - h(2) + \dots h(n) =$   
 $\Rightarrow \sum c_{ij} + h(n) - h(\text{inicio})$  siendo  $h(\text{inicio})$  una constante