Problemas de Inteligencia Artificial I **(Curso 2005–2006)**
Cuarto curso de Ingeniería Informática, Universidad de Sevilla

1. Supongamos que, para resolver un problema de espacio de estados, dos alumnos deciden representar de la misma manera los estados y los operadores, y definen de la misma manera la variable `*estado-inicial*` y las funciones `es-estado-final` y `aplica`. Sin embargo, aplicando el procedimiento de búsqueda en profundidad obtienen soluciones distintas ¿Por qué puede ocurrir esto?

2. Consideremos un problema de espacio de estados con factor de ramificación constante b y con una única solución que se encuentra a profundidad d. Calcular, tanto en el mejor como en el peor de los casos, el número de nodos que se necesitan analizar para encontrar la solución, al aplicar un algoritmo de búsqueda en anchura. Análogamente para la búsqueda en profundidad.

3. ¿Son ciertas las siguientes afirmaciones? (justificar la respuesta):

   - La lista de `ABIERTOS` usada en los procedimientos de búsqueda contiene en cada momento todas las hojas del árbol de búsqueda actual.
   - Las listas de `ABIERTOS` y `CERRADOS` no tienen nodos en común.

4. Especificar, razonando la respuesta, un problema de espacio de estados en el que la lista de `cerrados` en los algoritmos de búsqueda en anchura y en profundidad sea superflua.

5. Considérese un problema de espacio de estados en el que las soluciones, en caso de existir, tienen todas un número fijo de operadores igual a d. En caso de tener que usar un método de búsqueda no informada para solucionar el problema ¿qué método sería el más conveniente? Justificar la respuesta.

6. Considérese una modificación del algoritmo A^* en el que en lugar de usar $f(n) = g(n) + h(n)$ para ordenar la cola de abiertos se usa $f'(n) = (1-w)\cdot g(n) + w \cdot h(n)$, siendo w un número real constante, $0 \leq w \leq 1$ ¿Qué algoritmo de búsqueda estaríamos aplicando si tomamos $w = 0$? ¿Y $w = 0.5$? ¿Y $w = 1$?

7. En la misma situación del apartado anterior: suponiendo que h sea una heurística admisible ¿para qué rango de valores de w tendríamos asegurado el encontrar solución óptima? Justificar las respuesta (*Indicación: ¿para qué valores de w la demostración de la optimalidad del algoritmo A^* sigue siendo válida?*)

8. Considérense un par de heurísticas h_1 y h_2, ambas admisibles para un problema representado como espacio de estados. Supuesto que estamos interesados en encontrar una solución de coste mínimo en el menor tiempo posible, ¿qué heurística de las siguientes es la mejor para ser usada junto con el algoritmo A*? (justificar la elección):

   - h_1.
   - h_2.
   - h_3, definida por $h_3(e) = max\{h_1(e), h_2(e)\}$.
   - h_4, definida por $h_4(e) = h_1(e) + h_2(e)$.

9. Sea h una heurística admisible para un problema planteado como espacio de estados y $k > 0$ un número real ¿Qué efecto tiene usar en el algoritmo de búsqueda por primero el mejor, en lugar de h, la heurística que a cada estado e le asigna valor $k \cdot h(e)$? ¿Y en el algoritmo A^*?

10. Considerar el siguiente pseudocódigo de la función `BUSCA-SOLUCION` que implementa una búsqueda de soluciones en un espacio de estados (**Nota:** La función `SUCESORES` es la que se ha visto en clase para la búsqueda óptima).

```
FUNCION BUSCA-SOLUCION()
1. Hacer ABIERTOS la cola formada por el nodo inicial (es decir, el nodo cuyo
   estado es *ESTADO-INICIAL*, cuyo camino es vacío y cuyo coste es 0);
   Hacer CERRADOS vacío
2. Mientras que ABIERTOS no esté vacía,
   2.1 Hacer ACTUAL el primer nodo de ABIERTOS, hacer ABIERTOS el
       resto de ABIERTOS y poner el nodo ACTUAL en CERRADOS.
   2.2 Si ES-ESTADO-FINAL(ESTADO(ACTUAL)),
       2.2.1 devolver el nodo ACTUAL y terminar.
       2.2.2 en caso contrario,
         2.2.2.1 Hacer NUEVOS-SUCESORES la lista de nodos de SUCESORES(ACTUAL)
                 cuyo estado no está ni en ABIERTOS ni en CERRADOS
         2.2.2.2 Hacer ABIERTOS el resultado de incluir NUEVOS-SUCESORES
                 en ABIERTOS y ordenar todo en orden creciente de
                 los costes de los caminos de los nodos
3. Devolver FALLO.
```

¿Se encuentra con este procedimiento una solución de mínimo coste? Si la respuesta es afirmativa, demostrarlo. Si no es así, explicar por qué y dar un ejemplo de espacio de estados en el que este procedimiento no encuentra una solución de mínimo coste.

11. Considérese el problema del *3-puzle*, versión reducida del problema del 8-puzle, en el que en un cuadrado 2×2 se disponen tres bloques (y por tanto un hueco) numerados del 1 al 3. Los estados inicial y final son, respectivamente:

	1
3	2

Estado inicial

1	2
	3

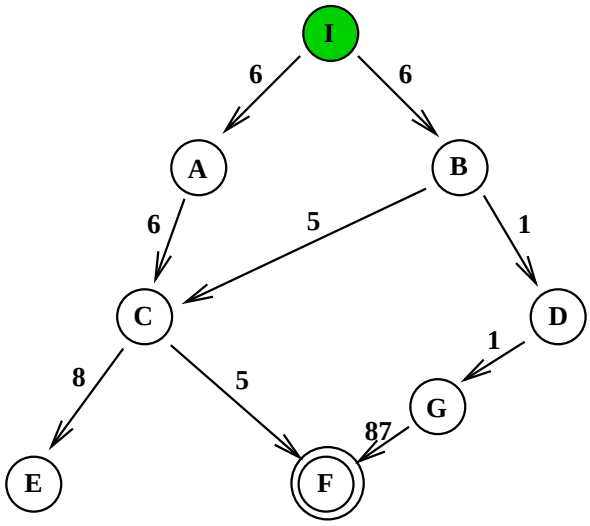
Estado final

Tomamos como operadores del problema el movimiento del hueco arriba, abajo, a la izquierda y a la derecha (*en ese orden*) y como coste de aplicar un operador el valor 1. Representar gráficamente los árboles correspondientes a los siguientes algoritmos de búsqueda.

- Búsqueda en profundidad.
- Búsqueda A* con la heurística h_1 que cuenta la distancia Manhattan desde la posición del hueco a la posición del hueco en el estado final.
- Búsqueda por primero el mejor con la heurística h_2 que cuenta el número de bloques que no están en la posición que deben ocupar en el estado final.

En cada caso, anotar junto a cada nodo el orden en el que se analiza y su heurística o coste más heurística, cuando sea relevante. A igualdad de valoración, resolver los conflictos escogiendo el nodo que más tiempo lleve en la cola de espera. ¿Es h_1 admisible? ¿Es h_2 admisible? ¿Cuál está más informada?

12. El siguiente grafo representa un problema de espacio de estados. Los nodos del grafo son los estados del problema, las aristas conectan estados con sus sucesores y el valor numérico de cada arista representa el coste de pasar de un estado a su sucesor:



El estado inicial del problema es I y el único estado final es F. Se consideran las funciones heurísticas H_1 y H_2 dadas por la siguiente tabla:

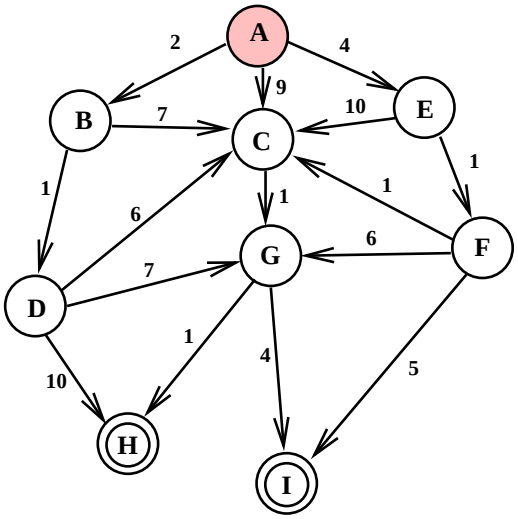
Estado	H_1	H_2
I	16	20
A	8	8
B	10	6
C	3	12
D	2	2
E	9	9
F	0	0
G	1	1

Resolver los siguientes apartados:

- Construir los árboles de búsqueda generados por los algoritmos de búsqueda en profundidad y búsqueda óptima. Indicar, en cada caso, el orden en el que se analizan los nodos y la solución obtenida junto con su coste.
- Análogamente, con el algoritmo de búsqueda A^*, usando las dos heurísticas H_1 y H_2 anteriormente definidas ¿Son admisibles? Justificar las respuestas.

Nota: Considerar que distintos sucesores de un mismo nodo se ordenan alfabéticamente.

13. El siguiente grafo representa un problema de espacio de estados. Los nodos del grafo son los estados del problema, las aristas conectan cada estado con sus sucesores, y el valor numérico de cada arista representa el coste de pasar de un estado al sucesor correspondiente.



El estado inicial del problema es el nodo A, y los estados finales son H e I. Se considera la función heurística h dada por la siguiente tabla:

Nodo	Heurística
A	7
B	7
C	2
D	7
E	4
F	9
G	1
H	0
I	0

Resolver los siguientes apartados:

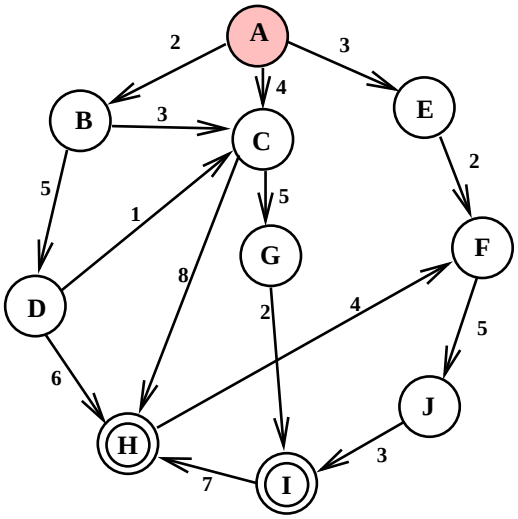
(a) Construir el árbol de búsqueda generado por los algoritmos de búsqueda por primero el mejor y búsqueda óptima.

(b) Para cada uno de ellos decir cual es la solución obtenida, el número de nodos que en cada caso ha sido necesario expandir, y el coste de cada camino solución. ¿Es alguna de las soluciones obtenidas óptima? ¿Cuál? ¿Por qué la otra no lo es?

(c) Construir el árbol de búsqueda generado por el algoritmo A^*. Teniendo en cuenta la solución obtenida responda a la pregunta ¿es h admisible?. Modificar mínimamente la heurística para que A^* encuentre la solución óptima.

Nota: Considerar que los sucesores de un mismo nodo se ordenan alfabéticamente. En caso de empate a la hora de gestionar la cola de abiertos, resolverlo igualmente atendiendo al orden alfabético.

14. El siguiente grafo representa un problema de espacio de estados. Los nodos del grafo son los estados del problema, las aristas conectan estados con sus sucesores y el valor numérico de cada arista representa el coste de pasar de un estado a su sucesor:



El estado incial del problema es A y los estados finales son H e I. Se considera la función heurística dada por la siguiente tabla:

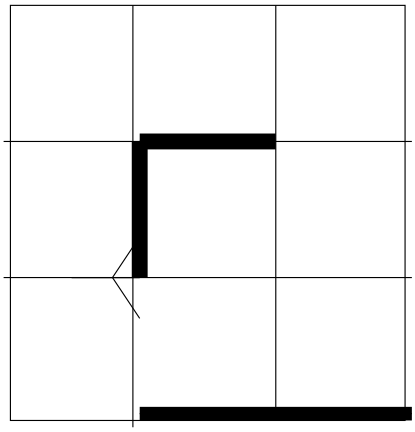
Nodo	Heurística
A	6
B	4
C	4
D	4
E	6
F	7
G	1
H	0
I	0
J	2

Resolver los siguientes apartados:

- Construir los árboles de búsqueda generados por los algoritmos de búsqueda en profundidad y búsqueda en anchura. Indicar, en cada caso, el orden en el que se analizan los nodos y la solución obtenida junto con su coste.
- Análogamente con el algoritmo de búsqueda A^*, usando la heurística anteriormente definida. ¿Es óptima la solución encontrada? Justificar la respuesta.

Nota: Considerar que los sucesores de un mismo nodo se ordenan alfabéticamente.

15. Consideremos un robot móvil que se puede desplazar por la malla siguiente, donde los lados de la malla son de longitud 1 metro.



Cada localización L del robot viene dada por las coordenadas (i, j) del nodo en el que se encuentra, junto con su orientación: **N**orte, **S**ur, **E**ste, **O**este. Por ejemplo, la localización del robot situado en la malla anterior es (1,1), orientación Oeste.

En cada momento, el robot puede:

- desplazarse en el sentido de su orientación actual hasta el siguiente nodo, o
- girar sobre sí mismo 90 grados en un sentido u otro, manteniéndose en las mismas coordenadas.

Además, sabemos que:

- tarda 4 s. en realizar un giro sobre sí mismo.
- se desplaza a 0.5 m/s por los tramos de trazo grueso.
- se desplaza a 1 m/s por los tramos de trazo fino.

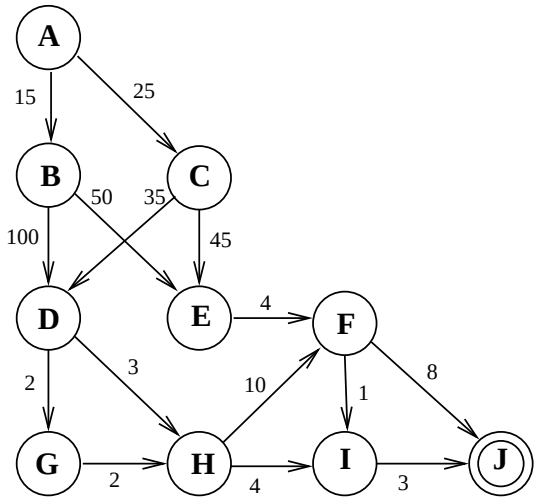
Se pide:

(a) Formalizar el problema de trasladar el robot desde una localización a otra en el mínimo tiempo como un problema de búsqueda en espacio de estados.

(b) En el caso concreto de trasladar el robot desde la localización L1 ($(1, 1)$, orientación Oeste), hasta la localización L2 ($(2, 2)$, orientación Norte):

- Definir una función heurística admisible, lo más informada posible, y justificar su admisibilidad.
- Aplicar el algoritmo A*, utilizando la función heurística definida en el apartado anterior, especificando el orden de generación y el orden de análisis de los nodos. Describir la solución encontrada y el coste de la misma ¿Es la solución encontrada la que tiene tiempo de recorrido mínimo? Justifica la respuesta. Nota: en caso de empate en la valoración de los nodos, ordenar según la orientación dada por la lista (E, S, O, N).

16. El siguiente grafo representa un problema de espacio de estados. Los nodos del grafo son los estados del problema, las aristas conectan cada estado con sus sucesores, y el valor numérico de cada arista representa el coste de pasar de un estado al sucesor correspondiente. El estado J es el único estado final.



(a) Utilizando el algoritmo apropiado, generar el árbol de búsqueda del camino de coste mínimo desde A a J. Indicar algoritmo utilizado, orden de expansión de nodos, nodos que se eliminan, costos parciales obtenidos, número de nodos expandidos, etc... Indicar asímismo el camino encontrado y su coste.

(b) Si $A = X_1, X_2, \ldots, X_i, \ldots, X_n = J$ es el camino de coste mínimo desde A a J, ¿cuál será el camino de coste mínimo desde cualquier nodo intermedio X_i a J? ¿cuál será su coste? Responder razonadamente a estas preguntas sin volver a generar los correspondientes árboles.

(c) Para cada nodo X del camino óptimo obtenido en el apartado 1, decir razonadamente cuánto vale $h^*(X)$.

(d) Siguiendo la metodología seguida en los apartados (a), (b) y (c), rellenar la siguiente tabla con los valores h^*, de cada nodo del grafo:

nodo	A	B	C	D	E	F	G	H	I	J
h^*										

(e) Dar tres ejemplos, h_1, h_2, y h_3, de heurísticas tales que h_1 y h_2 sean admisibles, h_3 no, y h_2 "informe mejor" al algoritmo que h_1.

(f) En general, y para problemas reales, ¿es ésta una técnica razonable para encontrar heurísticas admisibles? Razonar la respuesta.

17. Describir en pseudocódigo un algoritmo para encontrar *todas* las soluciones a un problema de satisfacción de restricciones, combinando AC3 con búsqueda.

18. Supongamos que tenemos una solución a un problema de satisfacción de restricciones dado y que algunas de las restricciones (no todas) se modifican ¿Cuál sería el método más adecuado (de los vistos en clase) para obtener una solución al nuevo problema planteado? Justificar la respuesta.

19. En el caso de que un PSR (problema de satisfacción de restricciones) tenga más de una solución, ¿las distintas ejecuciones del algoritmo de reparación heurística (con la heurística de mínimos conflictos) encuentran siempre la misma solución? ¿y el algoritmo de búsqueda–AC3? Razonar las respuestas.

20. Utilizando el algoritmo de consistencia de arcos, determinar todas las soluciones del siguiente problema de satisfacción de restricciones:

- Variables: A, B, C y D
- Dominios: El dominio de las cuatro variables es $\{1, 2, 3, 4\}$.

- Restricciones: $R_1 : A < B$
  $R_2 : D < C$
  $R_3 : A \neq C$
  $R_4 : D < A$
  $R_5 : B \neq C$

21. Una persona va a celebrar una fiesta y tiene que decidir a cuáles de sus cuatro amigos (Pedro, Carlos, Rosa y Teresa) va a invitar. La lista de invitados tiene que estar sujeta a las siguientes restricciones:

    - Pedro o Carlos (al menos uno de los dos) debe estar invitado.
    - Rosa o Carlos (al menos uno de los dos) debe estar invitado.
    - Rosa o Teresa (al menos una de los dos) debe estar invitada.
    - No se puede invitar simultáneamente a Rosa y a Pedro.
    - No se puede invitar simultáneamente a Teresa y a Carlos.

    Obtener una posible lista de invitados planteando el problema como un problema de satisfacción de restricciones y resolviéndolo aplicando el algoritmo de reparación heurística, con la heurística de mínimos conflictos y comenzando con la asignación según la cual no se invita a ninguno de los amigos.

22. En una fiesta infantil se desea repartir regalos sorpresa entre tres niños, María, Juan y Ana. Para ello, se dispone de cinco tipos de regalos: libros (li), lápices (la), discos compactos (cd), carpetas (car) y calculadoras (cal). Además, se conocen las preferencias de los niños, que son las siguientes:

    - María prefiere libros o lápices a discos compactos, carpetas y calculadoras.
    - Juan prefiere lápices, carpetas o calculadoras a libros y discos compactos.
    - Ana prefiera discos compactos o calculadoras a libros, lápices y carpetas.

    Se trata de que cada niño quede satisfecho. Es decir, que le guste más su regalo que el recibido por los demás. Para este problema de satisfacción de restricciones, se pide:

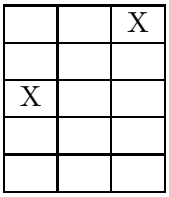
    - Indicar las variables del mismo y el dominio de cada una.
    - Expresar las restricciones para cada par de variables.
    - Dibujar el grafo con la red de restricciones del problema.
    - Resolverlo usando consistencia de arcos. Mostrar mediante una tabla la restricción considerada y los valores eliminados en cada paso.

23. Una empresa quiere saber si puede comprometerse a realizar las tareas T_1, T_2, T_3 y T_4. La realización de dichas tareas depende de la utilización de los recursos **a**, **b** y **c** en el siguiente sentido: T_1 necesita el recurso **a** o el **c**, T_2 necesita **a** o **b**, T_3 necesita **a**, **b** o **c**, T_4 necesita **b**. Hay que tener en cuenta que un mismo recurso no puede ser utilizado para dos tareas simultáneamente, que el tiempo de realización de cada tarea es de dos unidades de tiempo, y que el comienzo de cada tarea ha de ser como sigue: T_2 ha de comenzar en el instante 1, T_1 y T_4 han de comenzar en el instante 2, y T_3 ha de comenzar en el instante 3.

    Nota: Una tarea puede utilizar recursos diferentes en cada unidad de tiempo, durante su realización.

    - Plantearlo como un problema de satisfacción de restricciones.
    - Encontrar una solución, utilizando el algoritmo de *backtracking* (se aconseja usar también *forward checking* en cada paso).
    - ¿Tiene solución? ¿Es única? ¿Debe comprometerse la empresa a llevar a cabo las tareas en esas condiciones?

24. Un alumno de 4º tiene que elaborar un pasatiempo para el periódico de la ETSII. El pasatiempo consiste en un crucigrama de la forma:



donde:

- Las palabras que pueden aparecer están contenidas en el conjunto siguiente: {ES EL LA SI SO LO LAS LOS SAL SOL OSA OSO SOLA SOLO SALAS SOLOS}.
- Una palabra puede aparecer una sóla vez, en horizontal o en vertical.

Se pide:

- Plantearlo como un problema de satisfacción de restricciones.
- Reducir los dominios de las variables, utilizando el algoritmo de consistencia de arcos.
- Encontrar una solución.

25. Consideremos un "sudoku" de tamaño n. Es decir, un cuadrado de tamaño n, formado por n cuadrados de tamaño n, de forma que:

- Cada uno de los cuadrados interiores está formado por los números naturales de 1 a n, sin repeticiones.
- Cada fila (y cada columna) de el cuadrado exterior también está formada por los números naturales de 1 a n, sin repeticiones.

  Se pide: describir una representación del sudoku de tamaño n como un PSR, explicitando claramente las variables, los dominios y las restricciones que componen dicha representación.

Como caso particular, resolver el siguiente sudoku de tamaño 4 aplicando el algoritmo de búsqueda–AC3:

1			
	3	1	
3	4	2	1

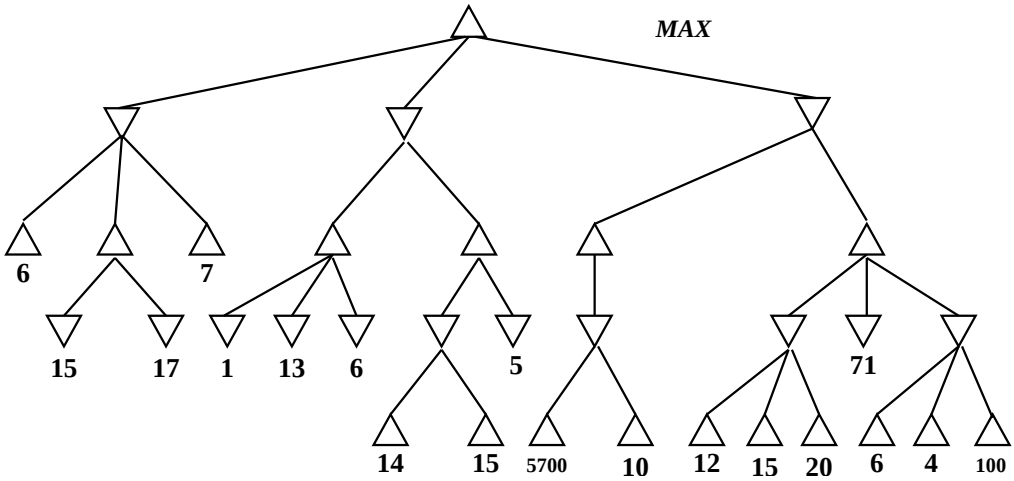
26. ¿Influye el orden en el que se analizan los sucesores de un nodo en la eficiencia del algoritmo minimax con poda alfa-beta? Justificar la respuesta.

27. Usando la misma profundidad, ¿pueden ser distintas las decisiones que tome la máquina en cada jugada usando minimax con poda alfa-beta de las decisiones que tome usando minimax sin poda alfa-beta?

28. En el algoritmo de minimax con poda alfa-beta ¿Es posible podar TODOS los sucesores de un nodo de juego que se está analizando? ¿Por qué?

29. Supongamos que aplicando el algoritmo minimax con poda alfa-beta se comienza a analizar un nodo MAX con el intervalo inicial (α_0, β_0) y una vez terminado el análisis del nodo no se ha podado ninguno de sus sucesores inmediatos ¿Qué podemos deducir del valor minimax de dichos sucesores? ¿Qué podríamos deducir si se tratara de un nodo MIN?

30. Consideremos el árbol de juego siguiente, en el que los valores de la función de evaluación estática se muestran debajo de cada hoja. Supongamos que el nodo raiz le corresponde al jugador MAX.



Si aplicamos la estrategia alfa–beta con cotas iniciales $\alpha = -\infty$ y $\beta = +\infty$, se pide:

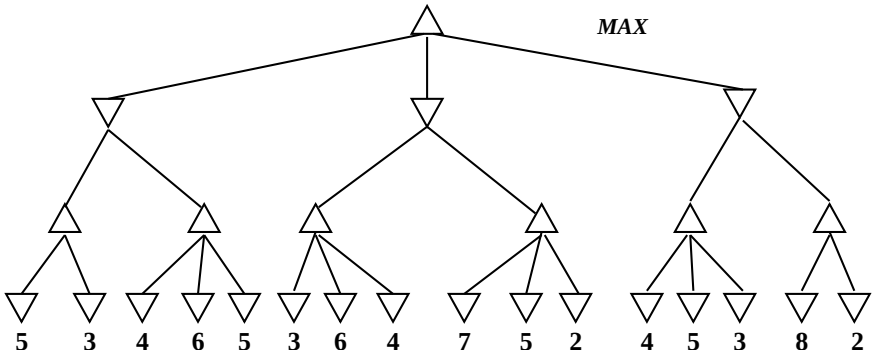
(a) Determinar el movimiento finalmente elegido por el jugador MAX.

(b) Detallar las podas que se han realizado, explicando razonadamente la última de ellas.

(c) Marcar con "(no)" los nodos no examinados.

(d) Especificar (marcando con el número correspondiente) el orden en el que se han explorado los nodos, y subrayar el valor de la función de evaluación estática en aquellos nodos en los que se ha hecho uso de ella.

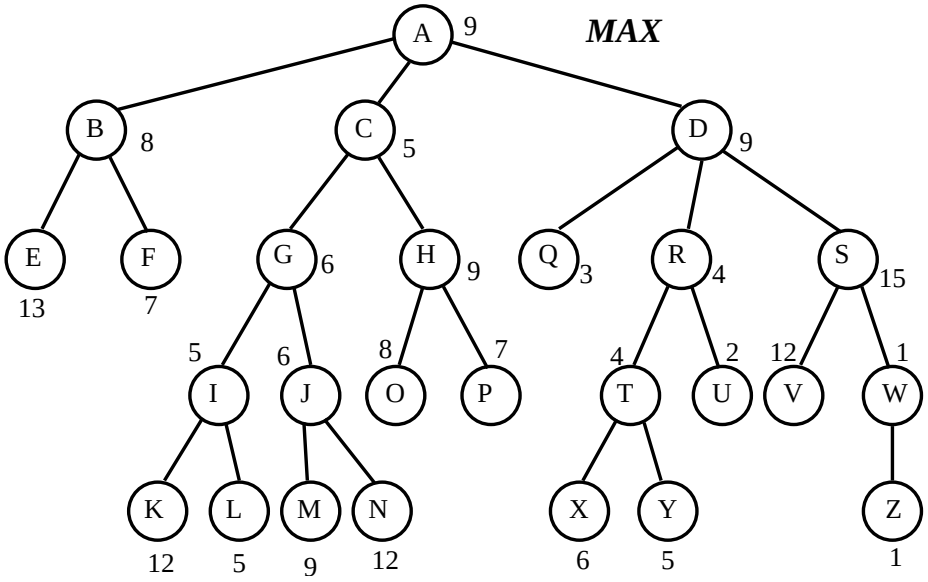
31. Consideremos el árbol de juego siguiente, en el que los valores de la función de evaluación estática se muestran debajo de cada hoja.



a) Si aplicamos la estrategia alfa–beta con cotas iniciales $\alpha = -\infty$ y $\beta = +\infty$, determinar razonadamente cuál es el movimiento elegido. Especificar, además, qué nodos no se han examinado a causa de las podas realizadas. Explicar razonadamente, en la última poda, por qué se pueden dejar de examinar los nodos podados.

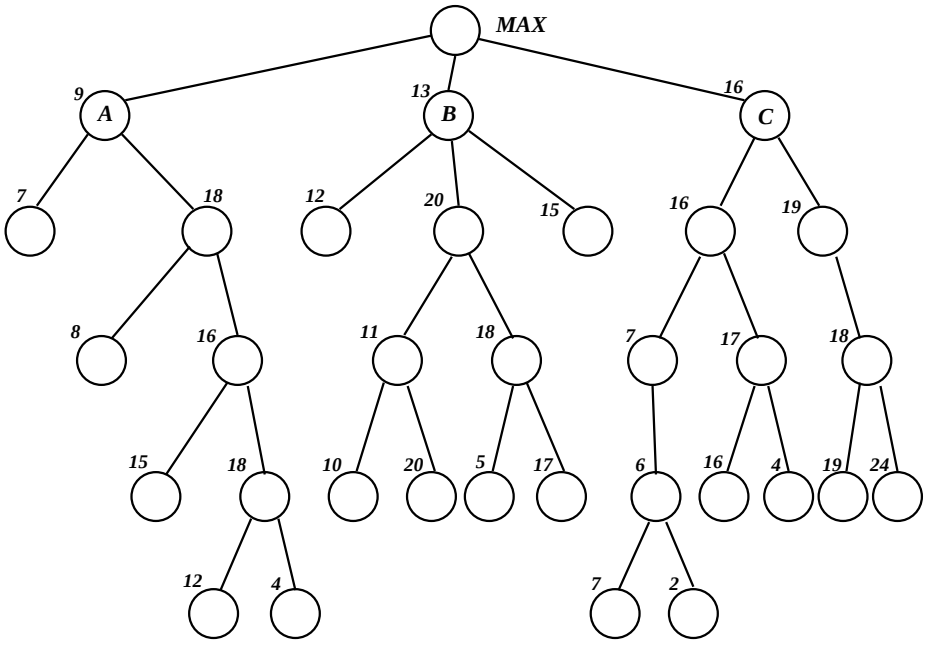
b) Supongamos que, usando la poda alfa–beta en un árbol de juego, se decide podar una serie de sucesores de un determinado nodo N. Y que, más adelante, encontramos el mismo estado (es decir, misma situación de juego y mismo turno de jugador) en otro nodo M que está en el mismo nivel que N. ¿Podemos podar, directamente, los mismos sucesores de M que se podaron en N? Razónese la respuesta, dando un contraejemplo si la respuesta es negativa.

32. Dado un juego de estrategia, el árbol siguiente es el árbol de juego *completo* correspondiente al estado A con turno para MAX. Las cantidades que aparecen junto a cada nodo del árbol son los valores de la función de evaluación estática de cada uno de los nodos.



- ¿Qué movimiento se decidiría realizar si se usara el algoritmo minimax con profundidad 1?
- Dibujar el árbol que se generaría si se tomara como método de decisión el algoritmo minimax con poda alfa-beta (con cotas iniciales $\alpha = +\infty$ y $\beta = -\infty$) y hasta profundidad 3. En el dibujo deben aparecer exclusivamente los nodos generados, detallando además el orden en el que se analizan. Especificar claramente dónde se producen las podas y de qué tipo son ¿Qué movimiento se elegiría finalmente? ¿En qué nodos se ha necesitado el valor de la función de evaluación estática? ¿Qué nodos del árbol completo se dejan de analizar?
- Si se aplicara el procedimiento de decisión minimax al árbol de juego completo ¿qué movimiento se elegiría? ¿es posible afirmar en general que la decisión que se tomaría analizando el árbol completo sería la misma que si se analiza hasta una determinada profundidad? Justificar la respuesta.

33. Dado un juego de estrategia, el árbol siguiente es el árbol de juego completo correspondiente a un estado con turno para MAX. Las cantidades que aparecen junto a cada nodo del árbol son los valores de la función de evaluación estática de cada uno de los nodos.



(a) ¿Qué movimiento se decidiría realizar si se usara el algoritmo minimax con profundidad 2?

(b) Dibujar el árbol que se generaría si se tomara como método de decisión el algoritmo minimax con poda alfa-beta (con cotas iniciales $\alpha = -\infty$ y $\beta = +\infty$) y hasta profundidad 4. En el dibujo deben aparecer exclusivamente los nodos generados, detallando además el orden en el que se analizan. Especificar claramente, cómo evolucionan en cada nodo los valores de α y β, y dónde se producen las podas. En el dibujo que se realice, sombrear los nodos en los que se haya necesitado el valor de la función de evaluación estática ¿Qué movimiento se elegiría finalmente? ¿Cuántos nodos del árbol completo se dejan de analizar? ¿Se elegiría un movimiento distinto si se aplicara minimax sin poda hasta profundidad 4? ¿Cómo se explica que el movimiento elegido sea distinto al del apartado anterior?

(c) ¿Qué valor podríamos poner en el último nodo/hoja evaluado para que no se produzca la última poda? ¿Qué repercusión tendría esto en la decisión sobre el mejor movimiento a realizar por MAX?

34. Dos jugadores **A** y **B**, tienen delante un conjunto de $2 * N$ fichas alineadas. Hay 2 clases de fichas A y B, N fichas de cada jugador, y en la situación inicial pueden estar en cualquier orden. Ejemplo para $N = 4$:

B	A	B	A	A	B	B	A

Los dos jugadores juegan por turnos.

- Turno de **A**: Una ficha A puede "comerse" todas las fichas del tipo B que se encuentren entre dos fichas A, o entre una una ficha A y el extremo del tablero. Los huecos resultantes se eliminan.
- Turno de **B**: Recíproco del anterior. Las fichas B se "comen" a las fichas A.

Por ejemplo a partir de la situación inicial, las posibles jugadas para **A** serían:

B	A	B	A	A	B	B	A

↙

A	B	A	A	B	B	A

↓

B	A	A	A	B	B	A

↘

B	A	B	A	A	A

El juego termina cuando algún jugador se queda sin fichas. Gana el jugador que conserve fichas al terminar el juego, y su ganancia será el número de fichas conservadas.

Resolver los siguientes apartados (suponemos que empieza el jugador **A**).

(a) Desarrollar el árbol del juego hasta la profundidad 2.

(b) Proponer una función de evaluación estática para las hojas del árbol generado en el punto anterior, y concluir la mejor jugada para **A** que se obtendría con la estrategia minimax. Si **A** hiciera esa jugada ¿tiene garantizado ganar el juego? Explicar la respuesta.

(c) Con la misma función de evaluación estática, desarrollar el árbol del juego que generaría la estrategia **minimax** con la poda **alfa-beta**. En este caso el árbol debe ser generado hasta los estados finales.(*Nota: Sólo deben aparecer en el árbol los nodos que necesariamente se hayan generado*) ¿Cuál sería ahora la mejor jugada para **A**?

(d) Si **A** se equivoca, y no elige la mejor jugada ¿garantizaría este error que **B** ganara el juego? Explicar la respuesta.

Nota: Desarrollar los árboles pedidos de tal forma que los sucesores de un nodo se ordenen entre sí con el siguiente criterio: *el hijo más a la izquierda será el que corresponda a la eliminación de las fichas enemigas que estén más a la izquierda.*

35. Describir un programa de enfriamiento para el algoritmo de enfriamiento simulado, de manera que el comportamiento del algoritmo con dicho programa sea análogo al de la búsqueda en escalada aleatoria.

36. Supongamos que n trabajadores tienen que realizar n tareas, y que conocemos el tiempo q_{ij} de realización por parte del trabajador i-ésimo (t_i) de la tarea j-ésima (T_j). El problema es cómo asignar a cada trabajador una y sólo una tarea, de manera que se realicen todas las tareas en un tiempo total mínimo.

    A continuación presentamos como ejemplo para $n = 4$ una tabla $Q = (q_{ij})$ con los tiempos que cada trabajador necesita para realizar cada una de las tareas:

		Tareas			
	Q	T_1	T_2	T_3	T_4
Trabajador	t_1	12	43	15	7
	t_2	9	10	6	4
	t_3	5	13	29	2
	t_4	4	11	17	9

    Dos posibles asignaciones (la segunda de ellas óptima) son:

    - $t_1 \to T_2$, $t_2 \to T_3$, $t_3 \to T_1$, $t_4 \to T_4$, con tiempo total igual a $43 + 6 + 5 + 9 = 63$.
    - $t_1 \to T_4$, $t_2 \to T_3$, $t_3 \to T_1$, $t_4 \to T_2$, con tiempo total igual a $7 + 6 + 5 + 11 = 29$.

    Se pide:

    - Representar el problema adecuadamente para que pueda ser resuelto mediante un algoritmo de búsqueda local. Esto es: describir una representación para los estados, definir un estado inicial, una función que genere un sucesor y una función objetivo.

- Representar el problema adecuadamente para que pueda ser resuelto mediante un algoritmo genético. Esto es, definir quiénes son los genes, la longitud de los cromosomas, la función de decodificación, la función objetivo.

37. Describir los elementos necesarios para una representación adecuada del problema de encontrar el mínimo de la función $f(x) = x^m$ en un dominio de la forma $[0, 2^n) \cap N$, siendo n y m números naturales, usando un algoritmo genético.

38. Considérese la siguiente variante simplificada del "problema de la mochila". Se tienen n objetos, tal que cada objeto i ($1 \leq i \leq n$) tiene un volumen v_i. Se trata de seleccionar un subconjunto de estos objetos para colocar dentro de una mochila que admite un volumen total máximo V, de manera que se minimice el espacio libre. Codificar el problema para resolverlo mediante un algoritmo genético. Esto es, describir para este problema: los genes, la longitud de los cromosomas, la función de decodificación y la función objetivo.

39. Codificar el problema de las n-reinas para resolverlo mediante un algoritmo genético. Esto es, describir para este problema: los genes, la longitud de los cromosomas, la función de decodificación y una función objetivo.

40. Un ganadero tiene un rebaño de n ovejas. Cada oveja i tiene un peso p_i y la vende por un precio v_i. Dispone de un camión que es capaz de cargar un peso total T. Su problema es seleccionar una serie de ovejas para llevarlas al mercado de ganado en el camión, de manera que se maximize el precio total de las ovejas transportadas, sin superar el peso total soportado por el camión. Codificar este problema para resolverlo con un algoritmo genético. Esto es, describir para este problema: genes, longitud de los cromosomas, función de decodificación y función objetivo.

41. Sea P una población con siete cromosomas, $C_i, i = 1, \ldots, 7$ con valores de función objetivo: $F(C_1) = 5$, $F(C_2) = 3$, $F(C_3) = 7$, $F(C_4) = 1$, $F(C_5) = 4$, $F(C_6) = 2$ y $F(C_7) = 1$. Supóngase que se desean seleccionar cinco cromosomas *por el método de ruleta*, y que con tal finalidad se obtiene la siguiente secuencia de cinco números entre 0 y 23, de manera aleatoria: $[5, 3, 14, 4, 17]$ ¿Qué cinco cromosomas serían seleccionados? Explicar el procedimiento de selección.

42. ¿Qué funciones y variables necesitamos definir para representar un problema de optimización de manera que pueda ser abordado por un algoritmo genético? Explicar con claridad qué representa cada una de ellas.

43. ¿Por qué un algoritmo genético que use el método de selección por ruleta probabilística no puede emplearse directamente para resolver un problema en el que se trata de minimizar el valor de una función objetivo? ¿Cómo podríamos modificar el problema para que sí se pudiera usar?

44. Considérese el siguiente conjunto de predicados que describen el mundo en un problema de planificación de acciones de un camión `T` que transporta paquetes entre ciudades:

    - `paquete(x)`: el objeto `x` es un paquete.
    - `ciudad(x)`: el objeto `x` es una ciudad.
    - `autovía(c1,c2)`: las ciudades `c1` y `c2` están conectadas por autovía.
    - `en(x,c)`: el objeto `x` (el camión o un paquete) está en la ciudad `c`.
    - `dentro-camion(x)`: el paquete `x` está cargado en el camión.
    - `descargado()`: el camión está descargado.

    Las acciones que se pueden realizar son las siguientes:

- `carga(p,c)`: el camión (que debe estar descargado) carga el paquete `p` en la ciudad `c`. Una vez cargado, el paquete ya no se considera que esté en la ciudad `c`.
- `descarga(p,c)`: el camión descarga el paquete `p` en la ciudad `c`.
- `ir(c1,c2)`: el camión se desplaza por autovía desde la ciudad `c1` a la ciudad `c2`.

Supongamos que deseamos encontrar la secuencia de acciones que a partir de un estado inicial en el que un paquete `P1` está en Barcelona, un paquete `P2` está en Madrid, y el camión `T` está en Sevilla, deja finalmente el paquete `P1` en Sevilla, el paquete `P2` en Barcelona y el camión descargado. Supondremos que existe una autovía entre Barcelona y Madrid y otra entre Madrid y Sevilla.

Representar el problema en el formalismo STRIPS. Es decir, describir el estado inicial, el objetivo y las acciones. **Nota:** Realizar este ejercicio en la misma hoja del enunciado.

45. Considérese el siguiente problema de planificación en el mundo de los bloques:

- **Estado inicial**: `SOBRELAMESA(A), SOBRELAMESA(B), SOBRELAMESA(C), BRAZOLIBRE, DESPEJADO(A), DESPEJADO(B), DESPEJADO(C)`
- **Objetivo final**: `SOBRE(A,B),SOBRE(B,C)`

El algoritmo STRIPS, aplicado a este problema, es capaz de encontrar la solución **AGARRAR(B), APILAR(B,C), AGARRAR(A), APILAR(A,B)**. Escribir una secuencia de situaciones por las que pasa el estado actual y la pila de objetivos y operadores, justo hasta el momento en que el operador **AGARRAR(B)** se aplica al estado actual. Especificar si en algún punto de dicha secuencia existirían otras alternativas para la pila ¿Cómo gestiona el algoritmo STRIPS la existencia de alternativas a la hora de ir construyendo la pila?