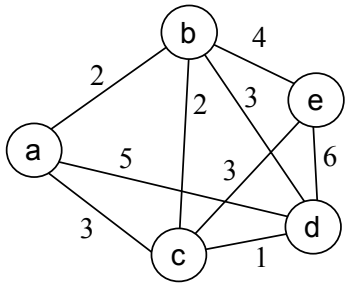
10.1. En un problema determinado, una solución está dada por una tupla de **n** elementos (x_1, x_2, ..., x_n). Para cada elemento existen en total **m** posibles valores. Comparar el número de nodos generados para resolver el problema usando un algoritmo de backtracking (suponiendo que no se realiza ninguna poda), un algoritmo de avance rápido (se supondrá que cada posibilidad probada es un nodo), y un algoritmo de ramificación y poda. En el último caso, ¿podemos predecir el número exacto de nodos generados o debemos dar un mejor y peor caso?

10.2. Aplicar la técnica de ramificación y poda al problema del viajante. Dado un grafo no dirigido y ponderado, encontrar un ciclo simple de coste mínimo que pase por todos los nodos una sola vez. Definir la representación de la solución y una forma de obtener la cota superior, inferior y el coste estimado a partir de una solución parcial. Escribir el algoritmo para resolver el problema. ¿Cuál es el orden de complejidad del algoritmo?

Muestra la ejecución sobre el siguiente grafo.



10.3. Aplicar el algoritmo de ramificación y poda para el problema de la mochila 0/1, al siguiente ejemplo: n= 5, M= 20, b= (10, 7, 6, 4, 2), p= (30, 15, 11, 8, 2). Elige el tipo de representación de la solución y las estrategias de ramificación y de poda que, previsiblemente, den lugar a un número mínimo de nodos generados.

10.4. Suponer un problema de satisfacción de restricciones como el de las **n** reinas, donde queremos obtener todas las posibles soluciones para un **n** dado. En esta situación, ¿se obtiene algún beneficio utilizando ramificación y poda en lugar de backtracking? Justifica la respuesta. Resuelve la pregunta para el caso general de problemas de este tipo.

10.5. Resolver por backtracking el problema de obtener de una serie de números dada {x_1, x_2, ..., x_n} los que suman una cantidad S, utilizando el menor número de elementos x_i posible. Utilizar un esquema no recursivo, como el visto en clase, con funciones **Generar**, **MasHermanos**, **Solución** y **Criterio**, indicando también cómo se representa la solución y cuando finaliza el algoritmo.

Decir cómo se prodrían obtener cotas inferiores y superiores y cómo se podría estimar el coste, para resolverlo utilizando ramificación y poda. Definir un estrategia de ramificación y de poda adecuadas a este problema concreto.

Aplicar sobre el siguiente ejemplo: x= {4, 12, 6, 3, 1}, S = 11.

10.6. Suponer un juego de tablero como el ajedrez, donde el número de posibilidades es muy elevado y el árbol de juego completo es inabordable para cualquier computador. Para poder resolver el juego de forma eficiente, ¿es necesario variar las definiciones de la cota superior e inferior del beneficio, y los criterios para realizar

la poda del árbol? ¿Qué problemas puede tener una estrategia de ramificación en profundidad? ¿Cómo será la estrategia de ramificación más adecuada?

10.7. Para el problema del cambio de monedas, especificar una buena forma de realizar el cálculo de las cotas y la estimación de beneficio que se puede obtener a partir de un nodo. Definir los demás aspectos necesarios para poder aplicar el esquema de ramificación y poda y escribir el algoritmo.

Suponer que a partir de un nodo obtenemos una solución mediante un algoritmo de avance rápido. ¿Qué información nos aporta esto, sobre la solución óptima que se puede alcanzar a partir de ese nodo?

10.8. (TG 13.2) En un problema de ramificación y poda podemos utilizar dos técnicas distintas para calcular las cotas y para la estimación del beneficio a partir de un nodo.

a) Para el cálculo de las cotas podemos usar un método con un tiempo **t(n) = 2n** (cada cota tarda t(n) = n), que producirá una poda del **p*50%** de los nodos, o bien podemos usar otro método que tarda **t(n) = $2n^2$**, que produce una poda del **p*75%** de los nodos. El valor de **p** (entre 0 y 1) hace referencia a lo rápido que la estimación del beneficio nos dirige hacia la solución óptima (se supone que usamos una estrategia LC).
b) Para la estimación del beneficio se puede usar un procedimiento de tiempo **t(n) = n**, que da lugar a un valor de **p=0.5**, o se puede usar otro procedimiento con **t(n) = n^2**, que tiene un **p=0.25**.

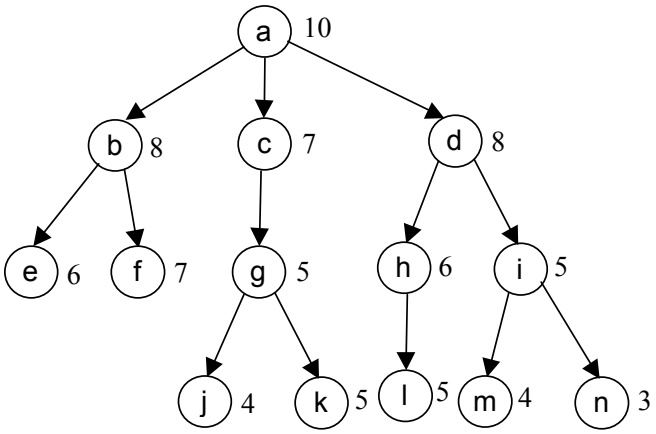
Sea **M** el número total de nodos del árbol completo, suponer que se puede usar cualquier combinación de los métodos anteriores y que el tiempo de manejar la lista de nodos vivos es despreciable. Calcular para cada combinación el tiempo necesario para ejecutar el algoritmo. ¿De qué forma se consigue un mejor tiempo de ejecución? ¿Te parece que los datos del problema son realistas (tiempos de ejecución, valores de **p**, ...) o existe alguna incoherencia?

10.9. Suponer el problema de encontrar todos los subconjuntos de un conjunto dado de enteros positivos $\{x_1, x_2, ..., x_n\}$ que sumen una cantidad M. Para resolverlo utilizamos ramificación y poda, con una representación binaria de la solución. La cota inferior usada es la suma de los elementos de la solución actual, y la cota superior es la cota inferior más los elementos que faltan por tratar.

¿Por qué no tiene ningún sentido dar una estimación del beneficio y utilizar una estrategia LC? Dar una estrategia de ramificación y de poda adecuadas para este problema, que sean óptimas en cuanto a tiempo de ejecución y memoria requerida.

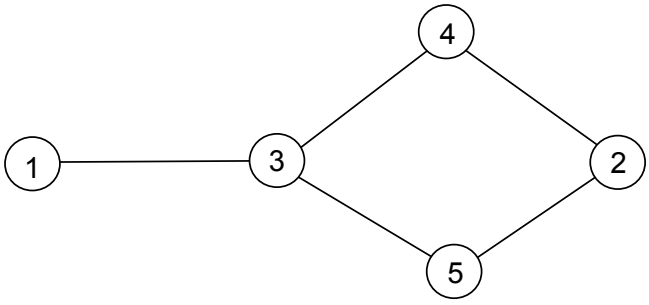
10.10. En muchos problemas, un procedimiento de avance rápido es usado como una manera de estimar una cota o el beneficio que se puede alcanzar a partir de un nodo, para usarlo después en un algoritmo de ramificación y poda. Del mismo modo, podríamos pensar en usar un algoritmo de programación dinámica para hacer una estimación más precisa de esos mismos valores. Usando el ejemplo del problema de la mochila, muestra por lo menos dos razones por las que esta forma de realizar las estimaciones no tiene ninguna utilidad.

10.11. (TG 13.3) El siguiente árbol de soluciones corresponde a un problema de maximización:



El número que acompaña a cada nodo es la cota superior del beneficio que se puede alcanzar a partir de ese nodo, que será igual al beneficio estimado para ese nodo. Para los nodos hoja, será el valor de una solución final. No existe una cota inferior, puesto que a partir de un nodo puede no existir una solución. Enumerar el orden en que se recorren los nodos con los métodos: backtracking, ramificación y poda con FIFO, LIFO, LC-FIFO y LC-FIFO. Suponer que los hijos se generan por orden alfabético.

10.12. (EX) Suponer que queremos resolver el problema de coloración de grafos usando ramificación y poda. Dar fórmulas para calcular las cotas y el beneficio estimado para cada nodo, y establecer una estrategia de poda, según las características del problema. Dar un esquema en pseudocódigo del algoritmo y mostrar la ejecución sobre el siguiente ejemplo sencillo.



10.13. En el esquema de algoritmo minimax con poda alfa-beta visto en clase, puesto que existe propagación de valores de los hijos a los padres, se utilizó un procedimiento recursivo. Debido a esto, ¿cómo es (implícitamente) la estrategia de ramificación? Propón una estructura de representación que nos permita utilizar un algoritmo no recursivo. Determinar qué datos se deben almacenar para cada nodo, qué nodos se encontrarán en la lista de nodos vivos, cómo se realiza la propagación de valores y cuándo se saca un nodo de la lista.

10.14. (EX) En Windows, cuando se quieren copiar varios archivos en disquetes, se empiezan a copiar por el orden seleccionado. Cuando un disquete se llena, se mete otro y se sigue copiando. Con este método, puede que necesitemos más disquetes de los necesarios. Por ejemplo, suponiendo que en un disquete caben 1.4 Mbytes, y los archivos son de tamaño: 400 Kb, 400 Kb, 800 Kb, 800 Kb, necesitaríamos 3 discos, cuando podríamos hacerlo con sólo 2. Se supone que los archivos no se pueden partir y que son de menor tamaño que la capacidad del disquete.

Diseñar una solución para el problema anterior utilizando ramificación y poda. Dado un array **T[1..n]**, con los tamaños de los archivos, y una cantidad **M** que indica el espacio libre de los disquetes, el problema consiste en encontrar el número mínimo de disquetes necesarios para copiar los archivos, y la asignación **S[1..n]**, indicando el número de disquete en el que se debe copiar cada archivo.

a) Exponer qué forma tendrá el árbol de búsqueda, con un ejemplo sencillo de árbol. ¿Qué representa cada nivel del árbol? ¿Qué descendientes son generados para cada nodo? ¿Cuándo un nodo es una solución final?
b) Escribir el esquema del algoritmo de ramificación y poda. Especificar cómo son las funciones para: generar los hijos de un nodo, comprobar si un nodo es solución final. Dar fórmulas (lo más ajustadas posible) para calcular cotas y el beneficio estimado para cada nodo.
c) Establecer una estrategia de ramificación y de poda adecuadas, y muestra la ejecución del algoritmo para el ejemplo anterior.

10.15. (EX) Suponer que resolvemos el problema del ejercicio 7.17 utilizando ramificación y poda. Si en un instante tenemos una cierta solución parcial **s**, en la que hemos planificado las **k** primeras tareas, dar una manera de calcular las cotas y el beneficio estimado. ¿Cómo es la estrategia de ramificación y de poda? Una vez con esto, escribe el algoritmo para resolver el problema y muestra la ejecución para el siguiente ejemplo: m= 3, n= 6, t= (35, 40, 20, 25, 10, 50), v= (5, 1, 10).

10.16. (EX) Supón que el problema de los votantes del ejercicio 7.18 se resuelve usando ramificación y poda. Una solución viene dada por una tupla $\mathbf{s} = (\mathbf{s_1}, \mathbf{s_2}, ..., \mathbf{s_m})$, donde $\mathbf{s_i}$ vale 1 ó 0, dependiendo de si el votante **i** es coaccionado o no. Definir una estrategia de ramificación y de poda adecuadas para este problema. Especificar una forma de calcular las cotas, a partir de cada nodo, usando el algoritmo diseñado en el ejercicio 7.18. ¿Cuándo acaba el algoritmo y qué solución es la que devuelve? Finalmente, escribe el algoritmo de ramificación y poda para resolver el problema, y muestra la ejecución sobre el ejemplo del ejercicio 7.18, para **p** = 1

10.17. (EX S01) Considera el esquema de ramificación y poda visto en clase, con la condición de poda para un problema de minimización y suponiendo que a partir de un nodo siempre existe alguna solución. Supón que hacemos un cálculo de las cotas muy preciso, de manera que para ciertos nodos interiores ocurre que CI(i) = CS(i). ¿Puede esta situación causar algún problema en la ejecución del algoritmo (sin tener en cuenta el tiempo de ejecución)? En caso afirmativo, explica brevemente cuál es el problema y cómo se podría solucionar.

10.18. (EX) La compañía *Eoloeléctrica Española* ha estado estudiando el problema de los cortes de suministro eléctrico durante este verano. Se ha comprobado que la potencia que producen las centrales es suficiente para cubrir la demanda. El

problema se encuentra en la falta de capacidad de las líneas de alta tensión, que llevan la electricidad desde las centrales eléctricas hasta las ciudades. Se supone que hay **m** centrales productoras **P**= (p_1, ..., p_m), cada una de las cuales produce **PP**[i] Mwatios, con i= 1.. m. Por otro lado, existen **n** ciudades **C**= (c_1, ..., c_n), cuya potencia máxima consumida es **PC**[j], con j= 1..n.

Para solucionar el problema, se pueden construir nuevas líneas de alta tensión, desde una central excedentaria **i** hasta una ciudad deficitaria **j**. La construcción de esa línea tiene un coste específico **D[i, j]**. La capacidad de la nueva línea es fija, ya que por ley no se puede superar un tope **T**. La capacidad de las líneas existentes en la actualidad está almacenada en **L[i, j]**. Entre una central y una ciudad puede existir más de una línea.

Se pide:

a) Diseñar un algoritmo, con alguna de las técnicas vistas en clase, para resolver el problema de satisfacer la demanda de todas las ciudades, minimizando el coste requerido. Se entiende que la solución está formada por las nuevas líneas que se deben construir y el coste total de su construcción.

b) Hacer una estimación aproximada del orden de complejidad del algoritmo diseñado. Aplicar el algoritmo sobre el siguiente ejemplo:

m= 3 centrales; **n** = 3 ciudades

Central	p1	p2	P3
PP (Mwatios)	16	20	13

Ciudad	c1	c2	c3
PC (Mwatios)	14	16	13

Capacidad de cada línea nueva **T** = 5 Mwatios

Líneas existentes

L (Mwatios)	**c1**	**c2**	**c3**
p1	5	0	0
p2	0	4	0
p3	0	0	5

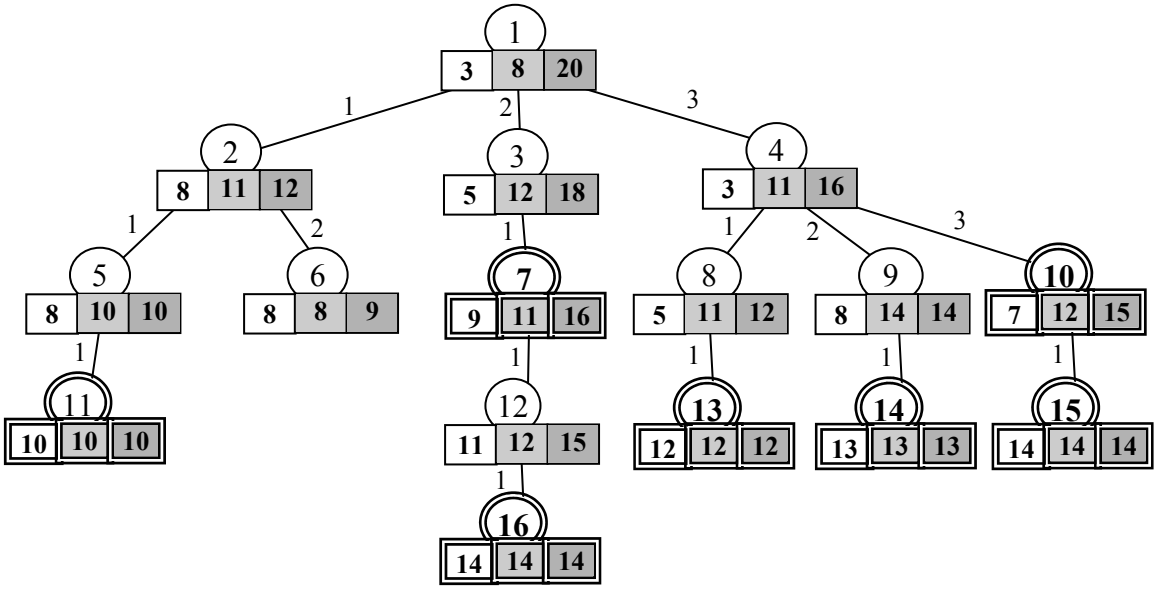
Coste de las líneas nuevas

D (mill. euros)	**c1**	**c2**	**c3**
p1	2	5	4
p2	3	7	3
p3	5	6	8

10.19. (TG 13.6) a) Consideramos el problema de la mochila modificado consistente en dado un damero de dimensiones M1 × M2 y n piezas de dimensiones i1 × j1, i2 × j2, ..., in × jn, cada una de ellas con beneficio b1, b2, ..., bn, maximizar el beneficio de poner las piezas en el tablero teniendo en cuenta que las piezas se pueden separar en cuadrículas para ponerlas en el tablero y que el beneficio de poner una cuadrícula de una pieza de dimensión i × j con beneficio b es b/ij. Resolver el problema con avance rápido y explicar cómo funcionaría para el caso de un tablero 3×4 y piezas 2×3, 1×4, 3×1 y 2×2, con beneficios 6, 3, 4 y 2, respectivamente.

b) Explicar cómo se podría utilizar el avance rápido anterior en la resolución por ramificación y poda del mismo problema pero sin poder dividirse en cuadrados las piezas.

10.20. (EX D01) En cierto problema de maximización representamos la solución mediante una tupla $\mathbf{S} = (s_1, s_2, ..., s_k)$. Cada s_j puede tomar ciertos valores enteros. El valor de **k** no es conocido a priori. Es más, a partir de una solución pueden existir otras soluciones. El problema es resuelto con ramificación y poda, de manera que el espacio de soluciones tiene la siguiente forma.

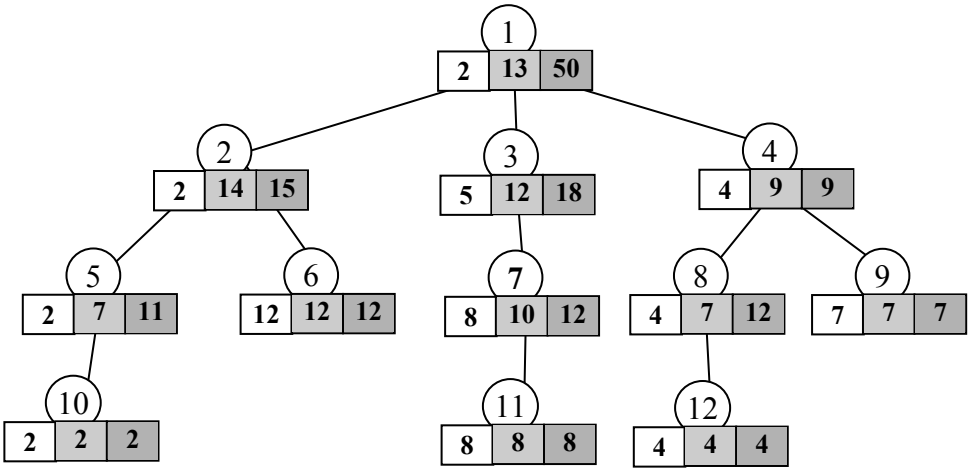


En cada nivel **q** se prueban las posibilidades para s_q. Para cada nodo se muestran (de izquierda a derecha) la cota inferior, el beneficio estimado y la cota superior. Los nodos que son una posible solución final aparecen con una doble línea. El beneficio final de estos nodos coincide con la cota inferior.

a) Modificando el esquema de ramificación y poda visto en clase, diseña una estructura general del algoritmo para resolver un problema como el anterior. Obviamente, las funciones básicas (*Generar*, *Solución*, *CI*, *CS*, etc.) deben quedar sin especificar.
b) Mostrar la ejecución del algoritmo (orden en que son recorridos los nodos, lista de nodos vivos, variable de poda, solución final obtenida) utilizando las estrategias de ramificación LC-FIFO y LC-LIFO. ¿Cuál consideras más adecuada? ¿Por qué?

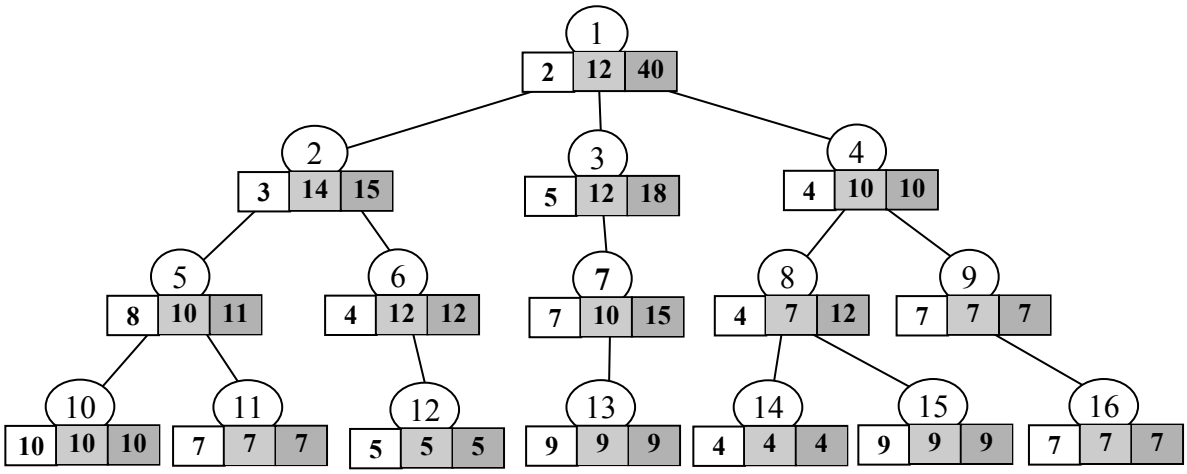
10.21. (EX D02) El árbol de abajo representa el espacio de soluciones en un problema de minimización. Para cada nodo se muestra (de izquierda a derecha) la cota inferior, el coste estimado y la cota superior. Todos los nodos hoja son soluciones válidas. El problema se resuelve con ramificación y poda, usando la estrategia LC-LIFO.

Mostrar, para cada paso del algoritmo, la lista de nodos vivos, el valor de la variable de poda **C**, los nodos podados y la solución óptima.



10.22. (EX S04) El árbol de abajo representa el espacio de soluciones en un problema de maximización. Para cada nodo se muestra (de izquierda a derecha) la cota inferior, el beneficio estimado y la cota superior. Son soluciones válidas del problema los nodos hoja de nivel 3. El problema se resuelve con ramificación y poda, usando la estrategia LC-LIFO.

Mostrar, para cada paso del algoritmo, la lista de nodos vivos, el valor de la variable de poda **C**, los nodos podados y la solución óptima actual. Justifica brevemente los pasos más destacados de la ejecución del algoritmo.



10.23. (EX J05) Cierto problema de maximización se resuelve utilizando ramificación y poda, aplicando una estrategia LC-LIFO. La solución se representa mediante una tupla de tamaño 4, **s**= (s_1, s_2, s_3, s_4), donde cada s_i puede tomar valor 0 ó 1. Son soluciones todas las tuplas de 4 elementos en las que el número de unos sea menor o igual que 3. Mostrar el árbol generado en la ejecución del algoritmo, indicando también la evolución de la lista de nodos vivos, la variable de poda C, así como el orden de recorrido de los nodos, los nodos podados y la solución final. Suponer que para cada tupla se calculan las cotas y estimaciones indicadas abajo. ¿Cambiaría el resultado, en este ejemplo, si se utilizara la estrategia LC-FIFO? Justifica la respuesta de forma muy breve.

Tupla	CI	BE	CS
Raíz	2	6	13
0	2	7	10
1	3	8	12
00	8	8	8
01	4	7	10
10	5	7	10
11	6	6	12
000	4	6	7
001	8	8	8
010	6	7	8
011	3	4	7

Tupla	CI	BE	CS
100	6	7	10
101	4	8	8
110	4	5	9
111	5	5	11
0000	6	6	6
0001	5	5	5
0010	2	2	2
0011	8	8	8
0100	1	1	1
0101	7	7	7
0110	4	4	4

Tupla	CI	BE	CS
0111	3	3	3
1000	3	3	3
1001	10	10	10
1010	5	5	5
1011	7	7	7
1100	8	8	8
1101	9	9	9
1110	11	11	11
1111	9	9	9

10.24. (EX J05) Resolver el problema del ejercicio 8.24 mediante un algoritmo de backtracking o mediante ramificación y poda. Se deben utilizar obligatoriamente los esquemas algorítmicos vistos en clase. Explicar cómo es la representación de la solución, la forma del árbol, y las funciones genéricas del esquema correspondiente. Se pueden dar por supuestos los esquemas vistos en clase, sólo es necesario indicar aquellas partes que se consideren ambiguas.

10.25. (EX) Resolver el problema del ejercicio 7.25 de forma óptima por backtracking o por ramificación y poda. Se deberán utilizar los esquemas vistos en clase, que se pueden dar por supuestos.

10.26. (EX) Resolver el problema del corral del ejercicio 7.26 por backtracking o por ramificación y poda, suponiendo que los tablones no se pueden partir. Se deberán utilizar los esquemas vistos en clase, que se pueden dar por supuestos. Definir la forma de representar la solución y las funciones genéricas del esquema.

10.27. (EX J06) Resolver el problema del juego de la oca del ejercicio 7.27 de forma óptima por ramificación y poda. Se deberá utilizar el esquema visto en clase, que se puede dar por supuesto. Definir la forma de representar los nodos, la manera de calcular las cotas, la generación de los descendientes de un nodo y las demás funciones genéricas del esquema.