UNIVERSIDAD DE GRANADA

/ UGR / decsai

Departamento de Ciencias de la Computación e Inteligencia Artificial

Asignatura: Algorítmica
Grado en Ingeniería Informática
Relación de problemas: Tema 5

1. Introducción

Para elaborar los ejercicios es recomendable utilizar la siguiente bibliografía:

- G. Brassard, P. Bratley, "Fundamentos de Algoritmia", Prentice Hall, 1997
- J.L. Verdegay: Lecciones de Algorítmica. Editorial Técnica AVICAM (2017)

Antes de realizar los ejercicios prácticos, se recomienda al estudiante revisar y comprender las diapositivas estudiadas en clase y complementarlas con la información procedente de la bibliografía básica y recomendada en la guía docente de la asignatura.

En la relación de problemas se presentarán, para cada temática, un conjunto de preguntas sobre conceptos teóricos que el alumno debe dominar antes de abordar los ejercicios prácticos.

2. Algoritmos para la exploración en grafos: Conceptos

1. ¿Qué tipos de recorridos sobre grafos conoces? ¿Qué diferencias hay entre ellos? Escribe el pseudo-código de cada tipo de recorrido expuesto.
2. Expón un ejemplo de grafo (inventado por ti) y explica, paso a paso, cómo realizar cada recorrido sobre grafos explicado en el ejercicio anterior en el grafo de ejemplo.
3. ¿Qué tipos de recorridos sobre árboles conoces? ¿Qué diferencias hay entre ellos? Escribe el pseudo-código de cada tipo de recorrido expuesto.
4. Expón un ejemplo de grafo (n-ario e inventado por ti) y explica, paso a paso, cómo realizar cada recorrido sobre grafos explicado en el ejercicio anterior en el grafo de ejemplo.
5. Defina los siguientes conceptos, indicando para cada descripción un ejemplo (diferente para cada apartado) del concepto descrito.

    1. Juegos n-personales.
    2. Juegos de suma cero.
    3. Juegos de información perfecta.
    4. Estrategia.
    5. Pagos.
    6. Juego en forma extensiva.
6. ¿Qué es un árbol de juego? Explique cada una de sus componentes. Finalmente, describa un juego simple y dibuje su árbol de juego.
7. ¿En qué consiste la técnica minimax? ¿Cuándo puede aplicarse? Describa el procedimiento general de la técnica, incluyendo las páginas del libro de referencia donde se encuentre esta información.
8. Exponga un juego que pueda resolverse mediante la técnica de minimax (o maximin), y resuélvalo, detallando cada paso a seguir hasta alcanzar la solución.
9. ¿En qué consiste el método de poda α-β? Describa el procedimiento general. Escriba las páginas del libro de referencia donde aparezca esta información.
10. Describa un juego que pueda resolverse mediante la técnica de poda α-β, y aplíquela para resolverlo, indicando paso a paso los pasos seguidos por el procedimiento.

11. ¿Qué similitudes y diferencias existen entre la técnica min-max (max-min) y la poda α-β? Si tuvieses que implementar una para resolver un juego, ¿cuál escogerías? ¿porqué?
12. ¿Cuál es la idea general de la técnica de algoritmos Backtracking? ¿Cuándo para un algoritmo backtracking?
13. Escribe cuáles son las componentes que hay que diseñar para elaborar un algoritmo backtracking.
14. Escribe el procedimiento general de un algoritmo backtracking.
15. ¿En qué consiste el problema de las N reinas? Diseña las componentes de un algoritmo Backtracking para solucionar este problema. Escribe el pseudo-código del algoritmo.
16. ¿En qué consiste la técnica de Ramificación y Poda (Branch&Bound)? Comenta las similitudes y diferencias, ventajas e inconvenientes, con respecto a los algoritmos Backtracking.
17. Suponiendo que estamos trabajando con la técnica de Branch&Bound, ¿qué es un nodo vivo? ¿Y un nodo muerto? ¿Y un nodo en expansión o en curso?
18. Escribe las componentes que hay que diseñar para desarrollar un algoritmo Branch&Bound y el procedimiento general del mismo.
19. Escribe el procedimiento general para resolver un problema con la técnica de Branch&Bound.
20. ¿En qué consiste el problema de la asignación de tareas? Diseña las componentes de un algoritmo Branch&Bound para solucionarlo y escribe el procedimiento general que lo resuelve. Por último, pon un ejemplo de cómo funcionaría el algoritmo explicando claramente cada paso.
21. Diseña las componentes de un algoritmo Branch&Bound que resuelva el problema de la mochila 0/1 mediante la técnica de Branch&Bound. Escribe el pseudo-código del algoritmo que lo resuelve y pon un ejemplo de su funcionamiento, explicando claramente cada paso.

3. Algoritmos para la exploración en grafos: Ejercicios prácticos

22. Las 3 en raya. En un tablero vacío de 3x3, el primer jugador selecciona una casilla vacía y dibuja una “X”. El segundo jugador hace lo propio, colocando una “O”. Se repiten los movimientos hasta que: **a)** No haya casillas libres (empate), o **b)** Un jugador consiga dibujar 3 casillas consecutivas horizontales, diagonales o verticales (gana +1 y el otro jugador pierde -1). ¿Es posible encontrar una estrategia pura para el primer jugador que le permita no perder en cualquier partida? Desarrolle el árbol de juego, teniendo en cuenta las jugadas que son simétricas para simplificar el mismo. Por ejemplo: Las 4 matrices dadas a continuación pueden asumirse simétricas.

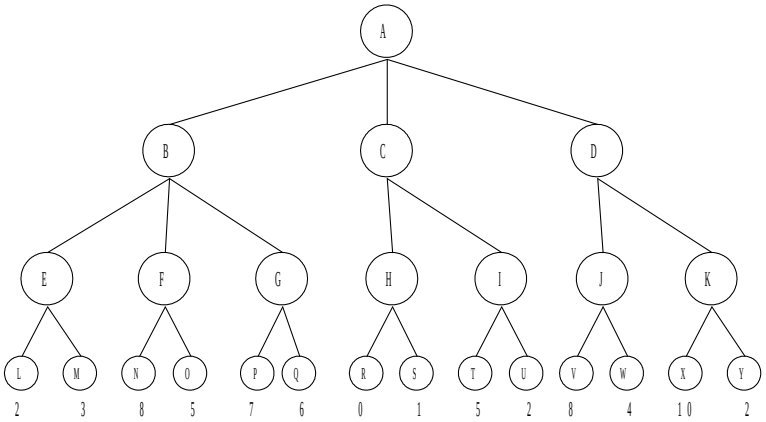
O		
	X	

	X	
O		

		O
	X	

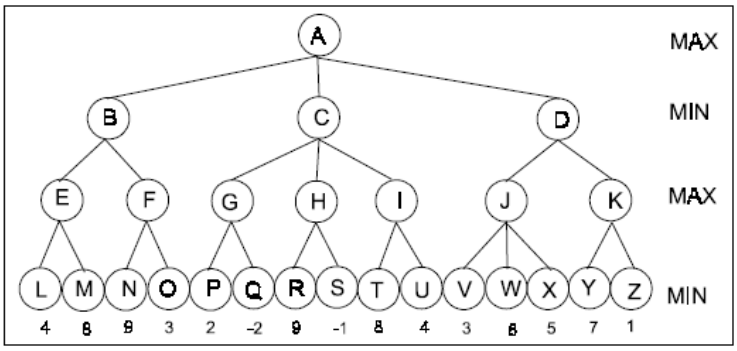
	X	
		O

23. Sea el árbol de la figura, representando un juego de 2 jugadores donde el jugador 1 (el que comienza el movimiento) debe maximizar su beneficio. ¿Cuál debería ser el primer movimiento a realizar para asegurarse un mínimo de beneficio? Justifique la respuesta con argumentos que hagan uso del procedimiento max-min.



24. En el árbol de juego del ejercicio anterior, ¿cuáles serían los nodos que no visitaría el algoritmo de poda α-β? Aplique este algoritmo, indicando en cada paso el valor de los parámetros α y β para cada nodo visitado, y cuándo y porqué se realiza una poda.
25. Realice una búsqueda en Internet y escriba las reglas del juego de 2 jugadores **Backgammon**. Indique, también, la dirección web desde donde ha obtenido esta información. ¿Se puede utilizar la técnica Max-min para resolver el juego? ¿Porqué? ¿Qué otras técnicas de IA se han utilizado para resolver este juego? ¿Hay alguna que sea campeona del mundo? ¿Podría explicar cómo funciona, de forma simple, esta campeona?
26. ¿Podría indicar cómo podría extenderse la técnica max-min (min-max) para juegos de 3 jugadores? ¿Y de n jugadores? Exponga un ejemplo de juego n-personal (3, 4, 5 jugadores, etc., los que prefiera) y describa el árbol de juego, de forma simple e ilustrativa. Puede hacer un dibujo del árbol si lo desea.
27. Realice una búsqueda en la bibliografía recomendada o en Internet para encontrar las reglas del juego **BlackJack**. Escríbalas. Asuma un número de 4 jugadores en el juego, ¿cómo resolvería este problema utilizando las técnicas de búsqueda para juegos estudiadas en este tema?. No hace falta que diseñe e implemente completamente un jugador inteligente de BlackJack: Simplemente, dé ideas generales (simplifique las reglas del juego a su gusto si le es más sencillo) sobre cómo se podría plantear el juego y resolverlo con alguna técnica del tema.

28. Sea el árbol de juego de la siguiente figura, donde los pagos se han calculado mediante estimaciones heurísticas:



Se pide:

a) Establecer los valores que serán asignados por el algoritmo minimax al resto de los nodos del árbol.

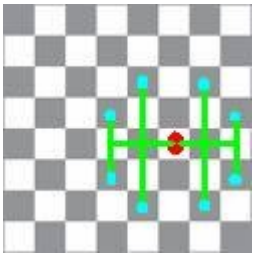
b) ¿Qué movimiento será seleccionado por el jugador MAX?

c) ¿Qué nodos no serán generados si se aplica la poda alfa-beta? (se asume que en un mismo nivel los nodos se van generando de izquierda a derecha según aparecen en la figura).

d) Si se generaran los nodos de derecha a izquierda ¿cuál sería el valor minimax asignado a la raiz? ¿cuántos nodos se podarían con alfa-beta?

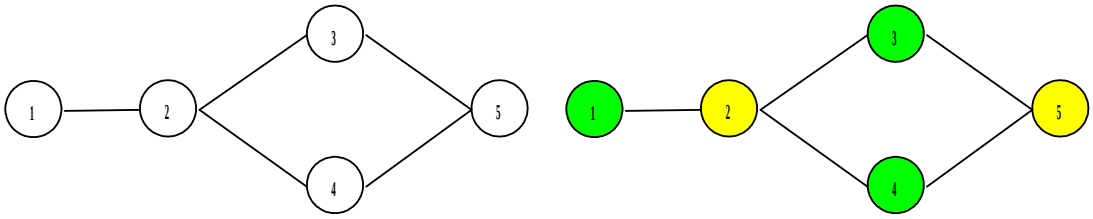
29. A fin de determinar la jugada a realizar por un jugador, se va a utilizar el algoritmo minimax aplicando una función de estimación a los nodos situados 3 capas por delante del estado actual en el árbol del juego. Se supone que dicho árbol es binario y completo, y que a los nodos de la tercera capa, en el orden en que se generan, les corresponden los siguientes valores de la función de estimación: -0.4, -0.4, 0, 0.4, 0.6, 0.8, 0.4, 0.6. Indicar qué nodos no se generarían si utilizáramos la poda alfa-beta. Si los nodos se generaran justamente en el orden opuesto, ¿cómo se comportaría el algoritmo y por qué?

30. El problema del recorrido del caballo. En un tablero de ajedrez, un caballo debe pasar por todas las casillas del tablero sin pisar dos veces la misma casilla y sin hacer ningún movimiento que lo saque del tablero. Sabiendo que un caballo se puede mover en movimientos en L, diseñar las componentes de un algoritmo Backtracking que resuelva el problema. Por último, dar el pseudo-código de la solución. Para mayor ayuda, se proporciona la siguiente figura mostrando las casillas que puede visitar el caballo para su posición actual dada:



31. El problema del coloreo de un grafo consiste en: Dado un grafo G=(V, A), no dirigido, y un conjunto *K* de colores, se pide dibujar todos los vértices del grafo con el mínimo número de colores posible, con la restricción de que no puede haber dos vértices adyacentes (unidos por una arista) que tengan el mismo color. Se pide: Diseñar un algoritmo Backtracking que resuelva este problema.

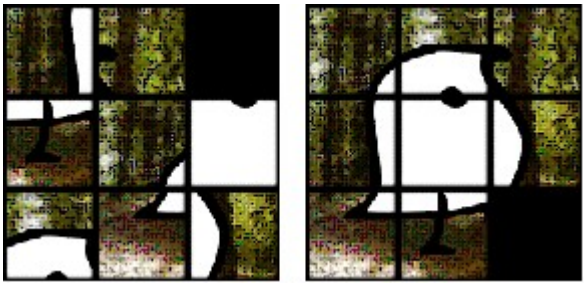
Como ejemplo, se proporcionan dos figuras con el coloreo de un grafo. A la izquierda, el grafo original. A la derecha, el grafo coloreado con 2 colores:



32. Enuncia el problema del Viajante de Comercio. Diseña un algoritmo Backtracking para solucionarlo y proporcionar una solución óptima a este problema, dando un pseudo-código del mismo y exponiendo un ejemplo de su funcionamiento, explicando paso a paso cómo el pseudo-código resolvería el problema de ejemplo.
33. El problema de la suma de subconjuntos. Dado un número entero *M* y un conjunto *A* conteniendo *n* números enteros, el problema consiste en identificar todos los subconjuntos de *A* cuya suma sea *M*. Por ejemplo, para *A=(11, 13, 14, 7)* y *M=31* hay 2 soluciones: Los subconjuntos (11, 13, 7) y (24, 7). Se pide desarrollar un algoritmo Backtracking que resuelva este problema, diseñando claramente las componentes Backtracking y proporcionando un pseudo-código que dé la solución. Aplicar el pseudo-código al ejemplo dado en el ejercicio y explicar cada paso que realizaría para solucionar este problema.
34. Un laberinto se ha representado por una matriz 2-D de *f* filas y *c* columnas. Se utiliza el valor 1 para representar un obstáculo en una casilla del laberinto, y el valor 0 para indicar que la casilla está libre y se puede visitar. Además, la entrada al laberinto se identifica con el valor 2, y la salida con el valor 3. Se pide desarrollar un algoritmo Backtracking que devuelva el camino desde la entrada hacia la salida de un laberinto dado como entrada, sabiendo que cada movimiento consiste en avanzar/retroceder una casilla bien en horizontal o bien en vertical (nunca en diagonal). Diseñe las componentes del algoritmo Backtracking que resuelve el problema y proporcione el pseudo-código del mismo. Por último, aplique el algoritmo sobre el laberinto siguiente explicando paso a paso las acciones que se realizan para devolver la solución:

1	3	1	1	1	1	1	1
1	0	1	0	0	0	0	1
1	0	0	1	1	1	0	1
1	1	0	0	0	0	0	1
1	0	1	1	1	0	1	1
1	0	1	0	0	0	0	1
1	0	0	0	1	1	0	1
1	1	1	1	1	1	2	1

35. El problema del N-puzzle. Supongamos un puzzle como el que se muestra en la siguiente figura (izquierda: Puzzle sin resolver; derecha: puzzle resuelto):



Este puzzle es de tamaño N=8 y se puede representar como una matriz en la que cada ficha se asigna a un número entero entre 1 y 8 (el valor 0 se corresponde con la casilla vacía). Resolver este puzzle implicar partir de una posición desordenada de las fichas (matriz de la izquierda) y llegar a la posición final ordenada (matriz de la derecha). Se pide diseñar un algoritmo Backtracking que resuelva este problema para valores de $N=2^k$. Exponga claramente cada componente del diseño del algoritmo y su pseudocódigo, sabiendo que la casilla vacía puede ser ocupada en cada movimiento intercambiando ésta por cualquiera de las fichas adyacentes en vertical u horizontal (nunca en diagonal). Explique, paso a paso, qué tareas realiza el algoritmo para solucionar el puzzle anterior, dado por las matrices siguientes:

4	3	0
8	1	5
2	7	6

(*Matriz original)*

1	2	3
4	5	6
7	8	0

(*Matriz del puzzle solucionado)*

36. Resolver el problema del N-puzzle planteado en el ejercicio anterior, utilizando la técnica de Branch&Bound. Para ello, establezca la cota en función de la distancia de Manhattan de cada ficha del puzzle hasta su posición. La distancia de Manhattan de una pieza situada en la casilla (x_i, y_i), cuya posición correcta es (x_j, y_j), se define como $|x_j-x_i| + |y_j-y_i|$. Por tanto, se seleccionará como nodo en expansión aquel que minimice la distancia de Manhattan para todas las fichas. Diseñe cada componente Branch&Bound del algoritmo y proporcione el pseudo-código del programa que lo resolvería. Exponga, como ejemplo, la solución al puzzle del ejercicio anterior, explicando paso a paso cada tarea realizada por el algoritmo.
37. Basándote en la solución Branch&Bound para solucionar el problema de asignación de tareas, proponga una solución Branch&Bound para solucionar el problema del viajante de comercio. Diseñe cada componente del algoritmo y exponga el pseudo-código. Por último, ponga un caso de ejemplo y explique paso a paso qué haría el algoritmo y cómo evolucionaría el árbol de estados explorado.
38. Un Sudoku es un pasatiempo matemático que consiste en rellenar con números del 1 al 9 una tabla de 9x9 elementos, dividida en subcuadrículas de 3x3, con casillas ya rellenadas previamente. Las normas del pasatiempo son:
    - No puede haber dos casillas en la misma fila con el mismo número.

- No puede haber dos casillas en la misma columna con el mismo número.
- No puede haber dos casillas, en la misma subcuadrícula de 3x3, con el mismo número.

La siguiente figura muestra un ejemplo de Sudoku sin resolver:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Se pide: Elaborar un algoritmo Backtracking que resuelva el Sudoku. Exponga el diseño de de cada una de las componentes Backtracking para resolver el problema y escriba el pseudo-código del algoritmo diseñado.

39. El Problema de Asignación Cuadrática consiste en asignar un conjunto de *n* localizaciones a *n* instalaciones, atendiendo a que entre cada par de instalaciones habrá un flujo de transporte de elementos y que cada par de localizaciones está a una distancia conocida. Un ejemplo de aplicación real es la construcción de hospitales: Entre cada instalación (UCI, quirófano, consultas, Servicio de Radiología, etc.) y otra instalación habrá un flujo de elementos (pacientes, material, médicos, etc.). Además, entre cada dependencia del edificio donde se construirá el hospital hay una distancia que recorrer. En este caso, hay que asignar el servicio del hospital a las dependencias del edificio tal que se minimice el coste de viajar de una dependencia a otra en función del flujo entre instalaciones. Siendo $\mathbf{F}_{n,n}$ y $\mathbf{D}_{n,n}$ las matrices de flujo entre instalaciones y de distancias entre dependencias, respectivamente, y $\mathbf{p}_n$ un vector con la asignación de instalaciones a dependencias, la función de coste se define como:

$$\min_p \{\sum_{i=1}^{n}\sum_{j=1}^{n} F_{ij} D_{p(i)p(j)}\}$$

Se pide: Diseñar una solución Branch&Bound para solucionar este problema. Diseñe completamente las componentes del algoritmo y escriba el pseudo-código que solucione el problema. Ponga un ejemplo pequeño (n=4) y explique cómo actúa el algoritmo paso a paso para solucionar el problema.

40. Un instructor de esquí dispone de n pares de esquís para sus n alumnos. Obligatoriamente, cada alumno debe recibir un par de esquís, que han de adecuarse lo máximo posible a su altura. El problema del instructor es asignar los esquís a los alumnos de forma que se minimice la suma de los valores absolutos de las diferencias entre las alturas de los alumnos y las longitudes de los respectivos esquís asignados. Proponga un algoritmo que resuelva este problema de forma óptima y aplíquelo sobre el siguiente ejemplo:

Altura	178	168	190	170
Longitud	183	188	168	175