E.T.S. INGENIERÍA INFORMÁTICA

Análisis y Diseño de Algoritmos
Backtracking
(2º de Grado Ing. Inf., Ing. Sw., Ing. Comp.)
E.T.S.I. INFORMÁTICA

Ejercicios Básicos

1. Considerar un tablero de ajedrez de $N \times N$ escaques y un caballo situado en la esquina inferior izquierda. Dado el movimiento de esta pieza en el juego, encontrar una secuencia de saltos del caballo que pase una única vez por cada casilla del tablero, o determinar que dicha secuencia no existe.

2. Sea $U = \{1...n\}$ y $S = \{S_1, ..., S_m\}$ una colección de subconjuntos de U (i.e. $S_i \subseteq U$ para $1 \leq i \leq m$). Una partición de U es una colección $C \subseteq S$ tal que todos los subconjuntos en C son mutuamente disjuntos y su unión es U.

   - Diseñar un algoritmo de Vuelta Atrás tal que, dados U y S, encuentre una partición o determine si esta no existe. Indica los parámetros de la primera llamada al algoritmo diseñado.
   - Escribe el árbol implícito de expansión para la siguiente instancia del problema: $U = \{1, ..., 12\}$ y $S = \{\{1, 2, 3, 4, 6, 7, 9\}, \{4, 5, 7, 8\}, \{1, 4\}, \{2, 5\}, \{3, 6\}, \{6, 11\}, \{7, 8, 9, 10, 11, 12\}\}$

3. Sea L un tablero cuadrado de tamaño $n \times n$. Algunas posiciones del tablero L presentan obstáculos mientras el resto de casillas están libres (sea l_{ij} un booleano que indique si existe o no obstáculo). Queremos colocar torres en las posiciones libres del tablero de forma que todas las posibles casillas puedan ser atacadas sin que una torre pueda atacar a otra.

   NOTA*: Las torres en el juego de ajedrez pueden ser desplazadas en linea recta cualquier número de casillas; es decir, pueden atacar a cualquier otra pieza en su misma horizontal o vertical. En este problema habrá de tenerse en cuenta la presencia de los obstáculos (es decir, sólo podrá atacar posiciones que estén en alcance directo, o sea, que entre la torre y la posición a atacar NO haya un obstáculo siguiendo una línea recta vertical u horizontal).

   - Diseñar un algoritmo de Vuelta Atrás que encuentre una distribución adecuada para las torres.
   - Modifica el algoritmo anterior para que encuentre el mínimo número de torres posible.

4. Dado un grafo $G(V, E)$, un coloreado del mismo es una asignación de colores (representados como números naturales $1, 2, \cdots$) a vértices, tal que si dos vértices están unidos por una arista entonces tienen distinto color. Se dice que un grafo es k-coloreable si puede colorearse con los colores $1, 2, \cdots, k$.

   a) Implementar un algoritmo que encuentre un k-coloreado de un grafo, o determine que éste no existe.

   b) Implementar un algoritmo que determine el mínimo k tal que el grafo es k-coloreable.

5. Dado un grafo $G(V, E)$, un cliqué es un conjunto de vértices $S \subseteq V$ tal que para todo $v, w \in S$ hay una arista $(v, w) \in E$, i.e., todos los vértices de S están conectados entre sí. Implementar un algoritmo que encuentre el mayor cliqué de un grafo.

6. Dada la matriz de adyacencia de un grafo no dirigido G, encontrar (o determinar que no existe):

   a) un camino hamiltoniano (un camino que pasa una única vez por cada vértice del grafo).

   b) un camino euleriano (un camino que pasa una vez por cada arista del grafo; se puede visitar más de una vez cada vértice).

7. Un constructor dispone de 3 solares (que llamaremos A, B y C) y desea construir 3 edificios distintos: un banco, un hotel y un colegio. El coste de construir cada edificio depende del solar en donde se realice, y viene dado por la siguiente tabla (en millones de euros):

	A	B	C
Banco	1	2	8
Hotel	4	5	3
Colegio	1	7	9

El constructor desea construir los tres edificios, cada uno en un solar, pero con el mínimo desembolso enconómico.

a) Implementar un algoritmo de backtracking para resolver este problema.

b) Dibujar el árbol de búsqueda para la instancia anterior. ¿Cuál es el número de nodos que se visitan para encontrar la solución al problema?

Ejercicios Complementarios

1. Modificar el algoritmo del ejercicio básico 1 de forma que, en lugar de encontrar los movimientos, calcule cuántas soluciones posibles posee el problema, es decir, cuántos recorridos válidos distintos puede hacer el caballo.

2. Si asignamos a cada casilla del tablero un peso (dado por el producto de sus coordenadas), a cada posible recorrido le podemos asignar un valor que viene dado por la suma de los pesos de las casillas visitadas por el índice del movimiento que nos llevó a esa casilla dentro recorrido. Esto es, si (x_0, y_0) es la casilla inicial y el recorrido R viene dado por los movimientos $[(x_1, y_1), (x_2, y_2), \cdots, (x_k, y_k)]$, con $k = n^2 - 1$, el peso asignado a R vendrá dado por la expresión

$$P(R) = \sum_{i=1}^{k} i x_i y_i$$

a) Modificar el algoritmo del problema anterior para encontrar el camino de coste máximo.

b) Es fácil dar un algoritmo ávido para el apartado anterior. Implementarlo y comparar resultados en cuanto a si encuentra una solución óptima o no, y a la complejidad de ambos algoritmos.

c) Modificar el algoritmo del apartado a) para que encuentre todos los caminos de coste máximo si es que hubiera más de uno.

3. Supongamos que tenemos n hombres y n mujeres y dos matrices P y Q tales que $P[i, j]$ indica la preferencia del hombre i por la mujer j, y $Q[i, j]$ es la preferencia de la mujer i por el hombre j. Diseñar un algoritmo que encuentre un emparejamiento de hombres y mujeres tal que la suma del producto de las preferencias se maximice.

4. El problema de la asignación de trabajos puede ser planteado como sigue: dadas n personas y n trabajos, queremos asignar a cada persona un trabajo. El coste de asignar a la persona i el trabajo j viene dado por la posición $[i, j]$ de la matriz TARIFAS. Diseñar un algoritmo que asigne un trabajo a cada persona minimizando el coste de la asignación.

5. Sea W un conjunto de enteros positivos y M un número entero positivo. Diseñar un algoritmo para encontrar todos los posibles subconjuntos de W cuya suma sea exactamente M. Dibujar el árbol correspondiente para $W = \{5, 7, 10, 12, 15, 18, 20\}$ y $M = 35$.

6. Dar una solución al problema de la Mochila 0-1 utilizando backtracking.

7. Consideremos un país en donde su sistema postal no permite más de n denominaciones distintas de sellos y no más de m sellos en una misma carta. Con estas restricciones, diseñar un algoritmo que calcule, dados los valores m y n y el conjunto de denominaciones $\{d_1, \cdots, d_n\}$, cómo se puede franquear una carta con cualquier valor entre 1 y $P = mn$.

8. Diseñar un algoritmo capaz de transformar un entero inicial n en un entero final m aplicando el menor número posible de transformaciones $f(i) = 3i$ y $g(i) = i/2$. Por ejemplo, si $n = 15$ y $m = 4$, podemos transformar n en m en 4 pasos: $gfgg(15) = 4$. Contemplar también los casos en donde haya múltiples formas de transformar un número en otro, y cuando no exista transformación posible.

9. Dados n tipos de monedas y sabiendo que disponemos de un número limitado K de monedas de cada tipo (el mismo para todas ellas), se desea pagar una cantidad C. Determinar las distintas formas de hacer ese pago sabiendo que puede excederse de la cantidad C pero no puede ser menor. Definir el formato de las soluciones y las restricciones de validez, y construir el árbol de búsqueda para $n = 3$, $d_i \in \{2, 4, 5\}$, $K = 5$, $C = 17$.

10. Disponemos de una tabla T con n filas y m columnas ($n \leqslant m$) que representan las posibilidades de que ciertos trabajadores (n) realicen determinados trabajos (m). Si $T[i, j]$ = `TRUE` entonces el trabajador i-ésimo puede realizar el trabajo j-ésimo. Cada trabajo puede ser realizado por uno o ningún trabajador, y cada trabajador debe tener un trabajo o ninguno. Obtener todas las posibilidades de asignar trabajadores con trabajos, de forma que el número de trabajos realizados sea máximo. construir el árbol de búsqueda para el ejemplo:

	Trabajo1	Trabajo2	Trabajo3	Trabajo4
Operario1	`FALSE`	`TRUE`	`FALSE`	`TRUE`
Operario2	`FALSE`	`TRUE`	`TRUE`	`FALSE`
Operario3	`TRUE`	`TRUE`	`FALSE`	`TRUE`

11. Disponemos de n objetos, cada uno con un volumen v_i , que hay que empaquetar utilizando envases (indistinguibles) de capacidad C. Calcular el empaquetamiento óptimo, es decir, que minimice la cantidad de envases utilizados, teniendo en cuenta que los objetos no se pueden fraccionar. construir el árbol de expansión para el ejemplo dado por 5 objetos de volúmenes $v_i \in \{1, 2, 5, 5, 6\}$ y envases de capacidad $C = 10$.

12. Sea F una fórmula lógica en forma normal conjuntiva, i.e., $F = C_1 \wedge C_2 \wedge \cdots C_m$, donde $C_i = (l_{i,1} \vee \cdots l_{i,k})$, siendo $l_{i,k}$ una variable $v \in \{v_1, \cdots, v_n\}$ o su negación. Por ejemplo,

$$F = (v_1 \vee \neg v_2 \vee v_3) \wedge (v_2 \vee \neg v_3 \vee \neg v_4) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_4)$$

    Consideremos una representación de una fórmula lógica mediante la siguiente estructura de datos:

```
TArrayLiterales = ARRAY [1..MAXLITERAL] DE ℤ
TCláusula = REGISTRO
                numLiterales: ℕ
                literal: TArrayLiterales
            FINREGISTRO
TArrayCláusulas = ARRAY [1..MAXCLAUSULAS] DE TCláusula
TFormulaFNC = REGISTRO
                numCláusulas: ℕ
                cláusula: TArrayClausulas
            FINREGISTRO
```

    donde los literales v_i y $\neg v_i$ ($1 \leqslant i \leqslant n$) se representan respectivamente como i y $-i$ dentro del array correspondiente. Implementar un algoritmo de bactracking que determine si una fórmula F es satisfacible o no (i.e., si hay una asignación de valores de verdad a las variables $v_1, \cdots, v_n$ que hace que F sea cierta).

13. Un cuadrado mágico normal de orden n es una matriz de tamaño $n \times n$ que contiene los números del 1 al n^2 de manera que cada fila, cada columna y la diagonal principal suman lo mismo (matemáticamente, puede demostrarse que dicha suma es $M = (n^3 + n)/2$). Definir un algoritmo de backtracking para hallar un cuadrado mágico de un cierto orden n.

14. Se desea encontrar una serie de fracciones x_i/y_iz_i, donde x_i, y_i, z_i son dígitos de 1 a 9 e y_iz_i es el número $10y_i + z_i$, tal que

$$\sum_{i=1}^{n} \frac{x_i}{y_i z_i} = 1$$

y donde cada dígito $1 \cdots 9$ aparezca entre 1 y $\lceil n/3 \rceil$ veces. Confeccionar un algoritmo de backtracking para resolver este problema.

15. La compañía aérea Colibrí opera vuelos entre n ciudades. Sea $V = \{v_{ij}\}_{n \times n}$ con $v_{ij} =$ `CIERTO` si, y sólo si, hay un vuelo directo de esta aerolínea entre las ciudades i y j. Con objeto de mejorar su servicio, la compañía decide poner supervisores en algunas ciudades, de manera que toda ciudad tenga un supervisor o esté conectada por vuelo directo con una ciudad con supervisor.

    a) Diseñar un algoritmo de backtracking que encuentre una solución con k supervisores como máximo, o determine que ésta no existe.

    b) Ídem para encontrar la solución que requiera menos supervisores.

16. Un fabricante de camisetas hawaianas recibe n pedidos en su taller. Cada uno de estos incluye una cantidad de camisetas c_i y un rango de fechas s_i y f_i, con $s_i \leqslant f_i$, dentro de las cuales hay que servir el pedido. Un pedido de c camisetas requiere c días de confección y se entrega inmediatamente cuando está listo, ya que no se dispone de almacén. Se desea determinar si hay un orden de atención de los pedidos que permita satisfacerlos dentro de las respectivas fechas de entrega.

    a) Construir un algoritmo de *backtracking* para resolver el problema.

    b) Modificar el algoritmo anterior para que devuelva el orden de atención que maximiza el número de pedidos satisfechos dentro de plazo.

17. Sea $M = \{M_{ij}\}_{j=1\cdots n}^{i=1\cdots m}$ una matriz de números enteros. Se desea encontrar un camino (empleando movimientos arriba, abajo, izquierda y derecha) desde M_{11} hasta M_{mn} de manera las posiciones visitadas tengan valores estrictamente decrecientes.

    a) Construir un algoritmo de *backtracking* para resolver el problema.

    b) Modificar el algoritmo anterior para que devuelva el camino más corto entre las posiciones indicadas.

18. Un fabricante de camisetas hawaianas recibe n pedidos en su taller. Cada uno de estos incluye una cantidad de camisetas c_i y un rango de fechas s_i y f_i, con $s_i \leqslant f_i$, dentro de las cuales hay que servir el pedido. Un pedido de c camisetas requiere c días de confección y se entrega inmediatamente cuando está listo, ya que no se dispone de almacén. Se desea determinar si hay un orden de atención de los pedidos que permita satisfacerlos dentro de las respectivas fechas de entrega.

    a) Establecer la estructura solución, y las restricciones del problema.

    b) Escribir un algoritmo de Backtracking que resuelva el problema. ¿El algoritmo debe encontrar todas las soluciones o parar cuando encuentra una?