ALGORITMOS DE EXPLORACIÓN DE GRAFOS
RELACIÓN DE EJERCICIOS Y PROBLEMAS

1. 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

2. Resuelva el problema de la mochila 0-1 para los siguientes valores:

   Tamaño de la mochila: M = 61
   Número de objetos: n = 5
   Vector de pesos: W = (1, 11, 21, 23, 33)
   Vector de beneficios: B = (11, 21, 31, 33, 43)

   Represente el árbol de estados que se obtendría al utilizar las técnicas de backtracking y branch&bound. ¿Qué funciones de acotación utilizaría para reducir el espacio de búsqueda?

   NOTA: En ambos casos, numere los nodos según el orden en que son expandidos y comente los criterios que se siguen para la expansión o poda de los nodos.

3. Diseñe un algoritmo B&B para resolver el problema de la asignación con la siguiente matriz de costes:

	f1	**f2**	**f3**	**f4**
d1	94	1	54	68
d2	74	10	88	82
d3	62	88	8	76
d4	11	74	81	21

4. Resuelva el problema del viajante de comercio utilizando la técnica B&B para 5 ciudades utilizando la siguiente matriz de distancias (que es simétrica):

	A	B	C	D	E
A	0	3	10	11	7
B		0	8	12	9
C			0	9	4
D				0	5

5. Una matriz booleana M[1..n][1..n] representa un laberinto cuadrado: si M[i][j] es `true`, se puede pasar por la casilla (i,j); si M[i][j] es `false`, no se puede pasar por la casilla (i,j).

   La siguiente figura muestra un laberinto de ejemplo, en el que la I marca la casilla de inicio, S es la casilla de salida y aparecen con X las casillas por las que no se puede pasar:

I	↓				
X	↓		X		X
←	↓	X	X		
↓	X	X	X		X
↓	X	↑	→	↓	X
↓	→	→	X	S	X

   a. Indique las restricciones implícitas y explícitas que nos aseguren un árbol de estados finito para el problema anterior.

   b. Represente el árbol de estados asociado al laberinto de la figura.

   c. Diseñe un algoritmo backtracking que, partiendo de la casilla de inicio, encuentre un camino en el laberinto hasta la casilla de salida.

   d. Indique cómo se podría resolver este problema utilizando branch&bound con una estrategia LC y represente el orden en que se expandirían los nodos en el ejemplo anterior.

Recorridos sobre grafos

6. Diseñe un algoritmo eficiente que compruebe si un grafo es bipartido o no.

   NOTA: Un grafo es bipartido si su conjunto de vértices V puede dividirse en dos conjuntos disjuntos X e Y de tal forma que todas las aristas del grafo van de un vértice en X a un vértice en Y.

   [Kleinberg&Tardos, 2006, sección 3.4: "Testing Bipartiteness"]

7. Diseñe un algoritmo eficiente que compruebe si un grafo no dirigido es conexo.

   NOTA: Un grafo no dirigido es conexo si se puede encontrar un camino que conexte cualquier par de vértices del grafo.

   [Kleinberg&Tardos, 2006, sección 3.2: "Graph Connectivity and Graph Traversal"]

8. Diseñe un algoritmo eficiente que compruebe si un grafo dirigido tiene ciclos o es un grafo dirigido acíclico (DAG).

9. Diseñe un algoritmo eficiente que ordene topológicamente un grafo dirigido acíclico (un problema con numerosas aplicaciones prácticas).

   NOTA: Una ordenación topológica de un grafo dirigido es una ordenación de sus nodos $(v_1, v_2, \ldots, v_n)$ de tal forma que $i<j$ para todos los arcos (v_i,v_j) del grafo.

   [Kleinberg&Tardos, 2006, sección 3.6: "Directed Acyclic Graphs and Topological Ordering"]

10. Dados dos nodos, v y w, de un grafo no dirigido G(V,E), diseñe un algoritmo de orden O(|V|+|E|) que permita calcular el número de caminos de longitud mínima que existen entre los nodos v y w.

    ¡Ojo! Se pide el número de caminos diferentes entre v y w que pasan por un número menor de nodos intermedios, no un único camino ni la longitud de éste (que sería lo que nos devolvería el algoritmo de Dijkstra).

    [Kleinberg&Tardos, 2006, problema 3.10 sobre redes sociales]

11. Un punto de articulación es un vértice de un grafo conexo tal que, si retiramos el vértice del grafo junto con todas las aristas que inciden en él, el grafo deja de ser un grafo conexo. Implemente el siguiente algoritmo para detectar todos los puntos de articulación de un grafo G(V,E) y calcule su orden de eficiencia:

    a. Se realiza un recorrido en profundidad del grafo numerando los vértices v en orden ascendente num(v).

    b. Se recorre el árbol de exploración en postorden para calcular el vértice de menor numeración low(v) alcanzable desde cada vértice v. Ese valor low(w) se calcula como el mínimo de

        i. num(v),
        ii. num(w) de los vértices w para los que existe la arista (v,w) en G pero no en el árbol de exploración en profundidad,
        iii. low(w) para los vértices w hijos de v en el árbol de exploración en profundidad (esto es, cuando la arista (v,w) se recorre al realizar la exploración en profundidad).

    c. La raíz del árbol es un punto de articulación de G si, y sólo si, tiene más de un hijo en el árbol de exploración en profundidad.

    d. Los vértices v distintos a la raíz del árbol son puntos de articulación de G si, y sólo si, tienen un hijo w tal que low(w)≥num(w).

    NOTA: Si x es hijo de v y low(v)<num(v), entonces existe una cadena de aristas que une x con el resto de vértices de G aunque se suprima v.

    [Brasssard&Bratley, 1997, sección 9.3.1: "Puntos de articulación"]