Problemas: Tema 5

Problema 1
Se tienen N elementos distintos almacenados en una estructura de acceso directo (por ejemplo, un vector con los números 1, 2, 3, 4 y 5, o la cadena abcdefg) y se quiere obtener todas las formas distintas de colocar esos elementos, es decir, hay que conseguir todas las permutaciones de los N elementos. Diseñar un algoritmo que use Backtracking para resolver el problema.

Problema 2
Resolver el problema anterior considerando la posibilidad de que los elementos se repitan entre sí (por ejemplo, el vector 1, 2, 3, 1 o la cadena acabada).

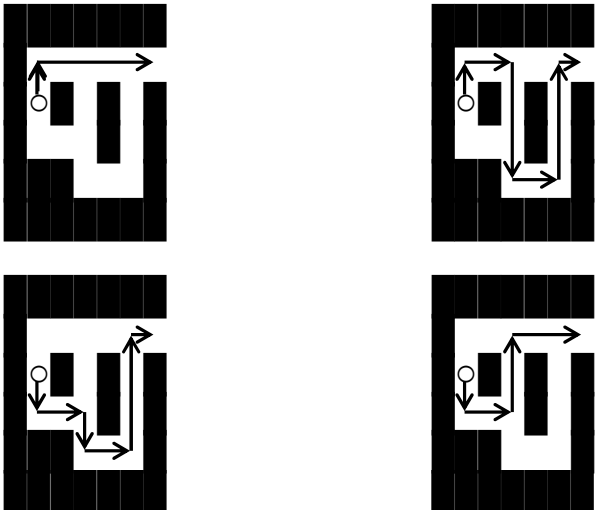
Problema 3
Se tiene un número de Tam cifras almacenado en una cadena de texto; por ejemplo, la cadena dato = 1151451.
Diseñar un algoritmo que mediante técnicas de Backtracking encuentre, de la manera más eficiente posible, todos los números distintos de N cifras que puedan formarse con los números de la cadena sin alterar su orden relativo dentro de la misma.
Por ejemplo, si N = 4, son números válidos 1151, 1511 y 1541, pero no 4551 o 5411 que aunque pueden formarse con los dígitos de la cadena dato implican una reordenación.

Problema 4
Se dispone de un tablero M de tamaño FxC (F es la cantidad de filas y C la cantidad de columnas) y se pone en una casilla inicial (posx, posy) un caballo de ajedrez. El objetivo es encontrar, si es posible, la forma en la que el caballo debe moverse para recorrer todo el tablero de manera que cada casilla se utilice una única vez en el recorrido (el tablero 8x8 siempre tiene solución independientemente de dónde comience el caballo). El caballo puede terminar en cualquier posición del tablero.
Un caballo tiene ocho posibles movimientos (suponiendo, claro está, que no se sale del tablero). Un movimiento entre las casillas M_{ij} y M_{pq} es válido solamente si:

- (|p–i|=1)&&(|q–j|=2), o bien si
- (|p–i|=2)&&(|q–j|=1),

es decir, una coordenada cambia dos unidades y la otra una única unidad.

Problema 5
Se dispone de una tabla laberinto[1..n,1..m] con valores lógicos que representa un laberinto.
El valor TRUE indica la existencia de una pared (no se puede atravesar), mientras que FALSE representa una casilla recorrible.
Para moverse por el laberinto, a partir de una casilla se puede desplazar horizontal o verticalmente, pero solo a una casilla vacía (FALSE). Los bordes de la tabla están completamente a TRUE excepto una casilla, que es la salida del laberinto. Diseñar un algoritmo Backtracking que encuentre todos los caminos posibles que llevan a la salida desde una casilla inicial determinada, si es posible salir del laberinto.



Diseñar un algoritmo Backtracking que encuentre el mejor camino posible que lleve a la salida desde una casilla inicial determinada, si es posible salir del laberinto.

Problema 6
Se tiene la tabla de sustitución que aparece a continuación

	a	**b**	**c**	**d**
a	b	b	a	d
b	c	a	d	a
c	b	a	c	c
d	d	c	d	b

que se usa de la manera siguiente: en una cadena cualquiera, dos caracteres consecutivos se pueden sustituir por el valor que aparece en la tabla, utilizando el primer carácter como fila y el segundo carácter como columna. Por ejemplo, se puede cambiar la secuencia ca por una b, ya que M[c,a]=b.
Implementar un algoritmo Backtracking que, a partir de una cadena no vacía texto y utilizando la información almacenada en una tabla de sustitución M, sea capaz de encontrar la forma de realizar las sustituciones que permite reducir la cadena texto a un carácter final, si es posible.
Ejemplo: Con la cadena texto=acabada y el carácter final=d, una posible forma de sustitución es la siguiente (las secuencias que se sustituyen se marcan para mayor claridad): aca*ba*da → a*ca*cda → abc*da* → *ab*cd → b*cd* → *bc* → d.

Entregable: un problema a elegir entre los problemas 3 y 4 y el problema 6.