# Análisis y Diseño de Algoritmos II

## Tema 4: Ramificación y poda

21 de enero de 2013

**1.** Explique el concepto de orden topológico y el algoritmo de ordenación topológica por búsqueda en profundidad.

**2.** Dado un grafo $G = \langle V, A \rangle$ orientado y acíclico, un orden topológico *inverso* es un orden lineal $<$ definido sobre $V$ tal que $i < j \implies \langle i, j \rangle \notin A$.

*a*) Modifique el algoritmo de ordenación topológica por búsqueda en profundidad para que produzca un orden topológico inverso.

*b*) Diseñe un algoritmo que produzca un orden topológico inverso preprocesando la entrada de un algoritmo de ordenación topológica convencional.

*c*) Diseñe un algoritmo que produzca un orden topológico inverso postprocesando la salida de un algoritmo de ordenación topológica convencional.

**3.** Diseñe un algoritmo que resuelva un laberinto tridimensional mediante búsqueda con retroceso. El laberinto está representado por un cubo en el que, originalmente, cada celda aparece libre (' ') o bloqueada ('X'). Se trata de encontrar, si es posible, una salida desde una posición inicial dada. A este efecto, se considera que una salida es cualquier celda libre del borde, es decir, de una de las caras del cubo. El algoritmo recibirá el laberinto y la posición inicial, que debe ser válida y corresponder a una celda libre, y devolverá un booleano que indique si existe o no solución. Además debe marcar las celdas visitadas ('V') y las que forman parte del camino de salida ('S'), si éste existe.

**4.** El algoritmo de coloreado de grafos mediante búsqueda con retroceso expuesto en las transparencias prueba con todos los colores en cada vértice.

¿Se podría reducir el número de colores a probar en cada vértice? Si es así, indique qué cambios se deberían realizar.

**5.** Basándose en el algoritmo de coloreado de grafos mediante búsqueda con retroceso, diseñe un algoritmo para detectar el mínimo número de colores con el que se puede colorear un grafo.

**6.** La versión con acotación superior del algoritmo para la mochila discreta con pesos reales no comprueba explícitamente si el valor de $S$ es superior al de $M$. ¿Por qué no es necesario hacerlo? ¿Dónde nos aseguramos de que el valor vaya aumentando?

**7.** Utilizando la versión con acotación superior del algoritmo para la mochila discreta, implemente la versión con expansión ordenada mediante un montículo.

8. Dado un vector $L$ de $n$ letras distintas y un número natural $m \leq n$, diseñe un algoritmo que calcule el conjunto de todas las palabras con $m$ letras diferentes escogidas entre las dadas.

9. Dado un tablero de ajedrez de $n \times n$ posiciones, y un caballo colocado en una posición inicial $(x, y)$, se pide diseñar un algoritmo que determine (si existe) una secuencia de $n^2 - 1$ movimientos del caballo de forma que se visiten todas las posiciones del tablero exactamente una vez.

   *Pista*: le puede resultar útil diseñar un subalgoritmo que codifique los 8 distintos movimientos posibles del caballo.

10. Modifique el algoritmo anterior, exigiendo que el último movimiento devuelva al caballo a la posición inicial.

11. Imagine que tiene $m$ máquinas y $n$ trabajos y que recibe una matriz $T$ de $m$ filas y $n$ columnas, en la que $T[i, j] \in \mathbb{R}^+$ representa el tiempo que necesita la máquina $i$ para realizar el trabajo $j$. Diseñe los siguientes algoritmos para obtener una asignación de trabajos óptima, minimizando el tiempo total requerido.

    *a*) Por retroceso, sin cota superior ni inferior.

    *b*) Además, con cota inferior (optimista) del coste.

    *c*) Además, con cota superior (pesimista) del coste.

    *d*) Además, usando montículos.

    *Nota*: cada máquina se encargará de exactamente un trabajo, y cada trabajo se asignará exactamente a una máquina.

12. Repita el ejercicio anterior, pero esta vez manejando una matriz $C$ con calidades en vez de con tiempos. En este caso, la asignación óptima será aquella que *maximice* la suma de las calidades.