# Diseño de Algoritmos

## Tema 1: Algoritmos devoradores

2013-14

**1.** En el esquema general de los *algoritmos devoradores*, no aparece explícitamente la función objetivo. ¿En qué parte del esquema es tenida en cuenta? ¿Cómo?

**2.** Un pintor dispone de varias latas de pintura de diferentes capacidades, expresadas en litros. Cada litro cubre $1m^2$ de superficie. El pintor debe pintar una serie de habitaciones, siendo cada una de ellas de diferente tamaño. Diseñe un algoritmo que, siguiendo una estrategia voraz, minimice el número de latas que le queden al pintor al finalizar el trabajo. Analice la complejidad del algoritmo. Describa los elementos que lo identifican como perteneciente al esquema general de los algoritmos voraces.

**3.** La versión entera (ó 0/1) del *problema de la mochila* consiste en llenar una mochila con *objetos indivisibles* maximizando el valor total de éstos. La mochila puede contener objetos cuyo peso total no exceda de su capacidad, $c$, y se conocen los pesos, $p_1, \ldots, p_n$, y valores, $v_1, \ldots, v_n$, de los objetos.

Demuestre, mediante contraejemplos apropiados, que ninguna de las siguientes *estrategias devoradoras* permiten, por sí mismas, resolver exactamente el problema:

*a*) Primero los objetos más valiosos.

*b*) Primero los objetos menos pesados.

*c*) Primero los objetos de mayor relación valor-peso.

Diseñe un algoritmo aproximado para resolver el problema basado en cada una de ellas. Intente primero obtener un algoritmo que realice $O(n^2)$ operaciones elementales y estudie luego cómo podría mejorarse.

**4.** Tenemos que almacenar $n$ programas de $t_1, \ldots, t_n$ KiB, respectivamente, en un CD-ROM de $c$ KiB de capacidad. Desgraciadamente, $\sum_{i=1}^{n} t_i > c$, es decir, no caben todos. Diga cuál sería la *estrategia devoradora* trivial que se seguiría en los siguientes supuestos:

*a*) Se desea maximizar el número de programas.

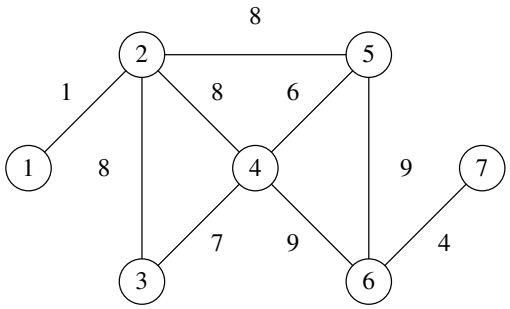
*b*) Se desea aprovechar al máximo la capacidad del disco.

¿Es la estrategia devoradora del segundo supuesto óptima? Si lo es, demuéstrelo. Si no lo es, ponga un contraejemplo.

**5.** ¿Qué orden temporal tiene, en el peor caso, el algoritmo de *Kruskal* si las estructuras de partición se representan mediante *vectores de pertenencia*? Compare el resultado con el del algoritmo de *Prim*.

**6.** Calcule, mediante los algoritmos de *Kruskal* y *Prim*, un árbol de expansión mínimo del siguiente grafo mostrando paso a paso la evolución de los algoritmos.

¿Cuántos árboles de expansión mínimos tiene este grafo? ¿Pueden todos encontrarse con cada uno de los algoritmos? ¿Dónde se reflejan las distintas posibilidades en cada algoritmo?



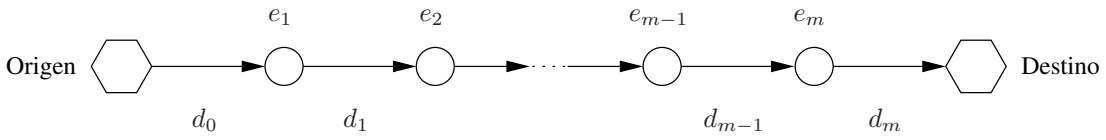

**7.** ¿Qué ocurre exactamente al aplicar los algoritmos de *Prim* y *Kruskal* a un grafo no conexo? Si es necesario, adáptelos para que calculen un bosque de expansión mínimo del grafo.

**8.** Describa cómo se pueden emplear montículos en el algoritmo de *Kruskal* en lugar de preordenación y qué ventaja se obtendría.

**9.** Un camión ha de cubrir una ruta en la que existen $m$ estaciones de servicio $e_1, \ldots, e_m$. El camionero desea parar en el mínimo número posible de estaciones de servicio, teniendo en cuenta que llena el depósito cada vez que para, lo que le permite recorrer $n$ kilómetros. El camionero parte con el depósito lleno y conoce las distancias entre las distintas estaciones, su lugar de origen y su destino como se muestra en la figura.

*a*) Diseñe un *algoritmo devorador* para resolver este problema.

*b*) Realice un análisis de su eficiencia temporal.




**10.** Deseamos atravesar una autopista de $m$ carriles en la que la policía ha establecido $n$ *puntos de control*. En cada uno de ellos ha colocado *puestos de control* ocupando uno o más de los carriles, pero no todos.

Conocemos, para cada punto de control, en qué carriles están situados los puestos de control. Para atravesar la autopista sin ser interceptados debemos cruzar cada punto de control por un carril en el que no esté la policía. El problema es que debemos realizar el mínimo número posible de cambios de carril (un único cambio de carril puede implicar cruzar varios carriles) a fin de no levantar sospechas.

Diseñe un algoritmo que calcule el camino óptimo, es decir, que diga por qué carril hay que pasar en cada punto de control para minimizar el número de cambios de carril.

El algoritmo recibirá una matriz binaria de $m$ filas y $n$ columnas en la que cada elemento informará sobre la existencia de un puesto de control en un carril y un punto de control dados. Como resultado devolverá un vector de $n$ elementos indicando el carril elegido en cada punto de control.

**11.** Un camarero trabaja en un restaurante que dispone de mesas para dos, cuatro y seis comensales. Suponiendo un aforo limitado y que no es posible reservar una mesa con antelación, diseñe un algoritmo voraz que asigne las mesas a los diferentes grupos de comensales que desean acceder al recinto. Suponga que todos desean acceder al mismo tiempo y que el número de grupos y de comensales en cada grupo es desconocido hasta el momento en el que entran en el restaurante y el camarero les tiene que asignar una mesa (los grupos entran de forma ordenada). ¿Es la estrategia devoradora diseñada óptima? Describa los elementos que lo identifican como perteneciente al esquema general de los algoritmos voraces.

**12.** Resuelva el ejercicio anterior suponiendo que se conoce previamente el tamaño de los grupos de comensales interesados en acceder al restaurante. ¿Es la estrategia devoradora diseñada óptima?

**13.** Diseñe un algoritmo que, siguiendo una estrategia voraz, obtenga el camino más corto entre dos nodos en un grafo. Suponga que dispone de las coordenadas $x$ e $y$ de cada uno de los nodos del grafo. Analice la complejidad del algoritmo. Describa los elementos que lo identifican como perteneciente al esquema general de los algoritmos voraces.

**14.** Un fontanero dispone de varios trozos de tuberías de dieferentes longitudes. Diseñe un algoritmo, basado en un estrategia devoradora, que minimice el número de uniones necesarias para construir una única tubería de longitud $x$. El fontanero puede realizar los cortes en la tubería que considere oportunos. Analice la complejidad del algoritmo. Describa los elementos que lo identifican como perteneciente al esquema general de los algoritmos voraces.

**15.** Considere el típico juego donde el jugador debe evitar que asteroides que caen de la parte superior de la pantalla colisionen con la nave espacial que controla. Suponga que la nave espacial solo puede desplazarse horizontalmente y que la pantalla puede dividirse en celdas de igual tamaño. Cada una de las filas representa un instante $t(i)$. En dicho instante la nave espacial solo puede ocupar una columna de la pantalla. Suponga también que la nave solo puede desplazarse a las dos columnas vecinas de un instante a otro.

**16.** Diseñe un algoritmo voraz que encuentre la mejor ubicación para una piedra en el juego Go-Moku[1], dado un estado concreto del tablero. Describa los elementos que lo identifican como perteneciente al esquema general de los algoritmos voraces. Analice su complejidad. Suponga que dispone de una función *enlinea* : $c \times i \times j \rightarrow n$, de orden $O(n^2)$, que determina el número de piedras en línea que se conectarían en caso de colocar una piedra de un determinado color $c$ en la celda $< i, j >$.

[1] http://wikipedia.org/Go-moku