Metodología y tecnología de la programación

Ingeniería Informática (UCM)

Hoja de ejercicios 5 Curso 2007/2008

Ejercicios del método devorador

Ejercicio 1 En una cinta magnética hay que grabar n programas de longitudes $l_1, \ldots, l_n$. Se supone que la densidad de información en la cinta y la velocidad de lectura son constantes y que tras cada búsqueda seguida de la lectura de un programa la cinta es automáticamente rebobinada. Si los programas están almacenados en el orden $i_1, \ldots, i_n$, el *tiempo medio de carga de un programa* es proporcional a

$$T = \frac{1}{n} \sum_{j=1}^{n} \sum_{k=1}^{j} l_{i_k}.$$

Determinar y justificar en qué orden hay que grabar los n programas para minimizar el tiempo medio de carga.

Ejercicio 2

1. En el problema de la mochila, demostrar mediante un contraejemplo que si se añade la condición de que los objetos no se pueden partir, es decir, $x_i = 0$ o $x_i = 1$, entonces la estrategia devoradora basada en el precio por unidad no da necesariamente lugar a una solución óptima.
2. Supongamos que en vez de n objetos tenemos n tipos de objetos (que se pueden partir). Formalmente, la restricción $0 \leq x_i \leq 1$ se convierte en $0 \leq x_i$. ¿Funciona correctamente en este contexto el algoritmo devorador basado en el precio por unidad?

Ejercicio 3

1. Demostrar que los caminos construidos por el algoritmo de Dijkstra sobre un grafo conexo no *dirigido* forman un árbol de recubrimiento. ¿Es un árbol de recubrimiento de coste mínimo?
2. ¿Es el camino entre un par de vértices en un árbol de recubrimiento de coste mínimo necesariamente un camino de coste mínimo?

Ejercicio 4 Un viajante de comercio tiene que viajar en coche desde Valencia a Lisboa siguiendo una ruta preestablecida. Con el déposito lleno su coche puede recorrer un máximo de n kilómetros. El viajante dispone de un mapa de carreteras en el cual figuran las distancias entre gasolineras en su ruta y desea utilizar esa información para realizar un número mínimo de paradas para repostar combustible en su recorrido. Para ello hay que desarrollar un método eficiente para determinar en qué gasolineras tiene que parar, implementar el método y (lo más importante y fundamental) demostrar que esa estrategia de lugar a una solución óptima.

Ejercicio 5 ¿Qué ocurre si se ejecutan por error los algoritmos de Kruskal y Prim (vistos en clase) sobre un grafo que no es conexo?

Ejercicio 6 Implementar el algoritmo de Dijkstra utilizando un montículo de mínimos, cuando el grafo viene representado por listas de adyacentes.

Ejercicio 7 El siguiente problema aparece en el análisis automático de programas. Dado un conjunto de variables $x_1, \ldots, x_n$, se tienen algunas restricciones de *igualdad* $x_i = x_j$ y algunas de *desigualdad* $x_i \neq x_j$. ¿Es posible satisfacer todas? Por ejemplo,

$$x_1 = x_2, x_2 = x_3, x_3 = x_4, x_1 \neq x_4$$

no se pueden satisfacer. Escribir un algoritmo eficiente que reciba m restricciones sobre n variables y decida si son satisfactibles.

Ejercicio 8

1. Demostrar que si en un árbol binario algún nodo no tiene dos hijos entonces no puede corresponder a una codificación sin prefijos óptima.

2. Supongamos que un fichero contiene caracteres de 8 bits tales que sus frecuencias son aproximadamente las mismas: la mayor frecuencia es inferior al doble de la frecuencia mínima. Demostrar que la codificación de Huffman en este caso no resulta más eficiente que una codificación ordinaria de longitud fija igual a 8.

Ejercicio 9 1. Desarrollar un algoritmo de coste lineal que tome como entrada un árbol y determine si tiene un *emparejamiento perfecto*: un conjunto de aristas que toca cada nodo exactamente una vez.

2. Un *conjunto de aristas recubridoras* de un grafo no dirigido $G = (V, A)$ es un conjunto de aristas $A' \subseteq A$ que interseca cada ciclo del grafo. Así, si se eliminan las aristas A' se obtiene un grafo acíclico. Escribir un algoritmo eficiente que dado un grafo valorado, no dirigido y conexo, devuelva el conjunto de aristas recubridoras de menor coste.