Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignaturas: Diseño de Algoritmos - Algoritmos II
Primer Cuatrimestre de 2017

Guía Práctica No. 6: Algoritmos Greedy

Ej. 1. Lea el capítulo 9 de *Introduction to the Design and Analysis of Algorithms* (Levitin 2003).

Ej. 2. Considere el problema de dar cambio por C centavos, utilizando monedas de valores $d_1 > d_2 > \cdots > d_k$, y sabiendo que se cuenta con M_i monedas de valor d_i. Diseñe un algoritmo utilizando la técnica Greedy e impleméntelo en Haskell, que resuelva el problema de dar cambio, aproximando a una solución óptima (es decir, que de cambio con el menor número de monedas). Su programa debe dar como resultado cuáles son las monedas que componen el cambio, y debe respetar la restricción de que no pueden darse en el cambio más de M_i monedas de valor d_i.

Analice además su programa de acuerdo al tiempo de ejecución en el peor caso. De un ejemplo en el cual su solución no daría una solución óptima.

Ej. 3. Considere el probema de asignar la sala 101 de la Facultad de Cs. Exáctas de la UNRC para el dictado de clases de las carreras de la facultad. Cada clase i que se solicita tiene asociado un horaio s_i de comienzo y un horario f_i de finalización, tales que, $s_i \leq f_i$, de forma que la clase$i$, si se realiza, debe hacerse en el intervalo $[s_i, f_i)$. Dos clases i, j son *compatibles* si los intervalos $[s_i, f_i)$ y $[s_j, f_j)$ no se superponen, es decir si $(s_i \geq f_j) \vee (s_j \geq f_i)$.

Diseñe un algoritmo greedy que seleccione el conjunto de clases mutuamente compatibles que tenga cardinal máximo.

Ej. 4. Considere el siguiente problema. Se cuenta con un software para el manejo de una agenda personal. La agenda permite la definición de *citas*, donde cada cita cuenta con una descripción, una fecha, un horario de inicio, un horario de finalización y una prioridad. En un día determinado, una persona puede tener citas $C = C_1 \dots C_k$, algunas de las cuales pueden solaparse. En estos casos, es útil elegir el subconjunto de C óptimo de citas compatibles (es decir, que no se solapan entre sí). Tal subconjunto es óptimo si la suma de las prioridades de las citas elegidas es maximal entre todos los subconjuntos de C compatibles.

Diseñe un algoritmo Greedy para, dado un conjunto de citas correspondientes a una fecha particular, encontrar un subconjunto de tareas compatibles de prioridad maximal (o una aproximación a esto). Implemente su algoritmo en Haskell.

Ej. 5. Implemente en Java los algoritmos de Prim y Kruskal para la construcción de árboles abarcadores de costo mínimo. Realice el análisis de tiempo de ejecución en peor caso para estos algoritmos.

Ej. 6. Diseñe una solución *Greedy* e impleméntela en Java para el problema de la mochila. Su algoritmo siempre da como resultado una solución óptima al problema?

Ej. 7. Diseñe una solución *Greedy* e impleméntela en Java para el problema de coloreo de grafos. El problema consiste en colorear todos los nodos de un grafo no dirigido, de manera tal que no existan en el grafo dos nodos adyacentes con el mismo color, y con la menor cantidad posible de colores. Su algoritmo siempre da como resultado una solución óptima al problema?

Ej. 8. Escriba un programa Java o Haskell que tome una secuencia de símbolos con sus correspondientes frecuencias de ocurrencia, y produzca la tabla de Huffman (tabla conteniendo los códigos binarios para cada símbolo de acuerdo a la codificación de Huffman) para los símbolos.

Ej. 9. Explique cuál es la relación entre el concepto de *matroide* y el de algoritmo greedy.

Ej. 10. Qué es un *matroide de grafo*? Guarda este concepto alguna relación con algoritmos greedy sobre grafos con costos, como los algoritmos de Prim y Kruskal?