Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignaturas: Diseño de Algoritmos - Algoritmos II
Primer Cuatrimestre de 2023

Guía Práctica No. 4: Algoritmos Greedy

Esta guía práctica corresponde a la estrategia de diseño de algoritmos Greedy. Encontrará material relacionado en el capítulo 9 de *Introduction to the Design and Analysis of Algorithms* (Levitin 2003), que es lectura recomendada para el desarrollo de la práctica.

Parte de esta práctica cuenta con esquemas de programación y tests de asistencia a su resolución. Éstos pueden accederse a partir del classroom de github Diseno-de-Algoritmos-Algoritmos-II-2023.

Esta práctica no tiene entrega formal. Su resolución es opcional.

1. 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 Java o 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.

2. Considere el problema de los vinos planteado en la práctica anterior, se pide ahora diseñar un algoritmo utilizando la técnica greedy para maximizar las ganancias del dueño de la vinoteca. En caso que el algoritmo no retorne siempre la solución óptima, encuentre un ejemplo que lo muestre.

3. 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 o Java.

4. Dado un conjunto de policías y ladrones con las siguientes restricciones:

   - Cada policía puede capturar solo a un ladrón.
   - Un policía puede capturar solo a ladrones que están a lo sumo a una distancia K de él.

   Diseñe un algoritmo para encontrar el máximo número de ladrones que pueden ser capturados. Su algoritmo debe computar la solución en tiempo polinomial. Su solución puede ser aproximada (no óptima), pero debería mostrar un caso en el cual el valor calculado no es óptimo.

5. 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?

6. Retome la solución al ejercicio de la práctica 1, del problema de coloreo de grafos. Es su solución *Greedy*? En caso que no lo sea, diseñe una solución alternativa, que utilice la técnica Greedy, e impleméntela en Java. Su algoritmo siempre da como resultado una solución óptima al problema?

7. 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.

8. 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.

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