Ejercicios en clase: Algoritmos voraces (Greedy)

Análisis y Diseño de Algoritmos

18 de junio de 2020

Ejercicio 1. Describa un algoritmo eficiente que, dado un conjunto $\{a_1, a_2, \ldots, a_n\}$ de puntos en la recta, determine un conjunto mínimo de intervalos de tamaño 1 que contiene a todos los puntos. Justifique que su algoritmo es correcto usando la propiedades de elección voraz y subestructura óptima.

Ejercicio 2. Dados dos conjuntos A y B, cada uno de los cuales tiene n enteros positivos, un *cruce* entre A y B es un conjunto de pares ordenados $\{(a_i, b_j) : a_i \in A, b_j \in B\}$, tales que todo elemento en A aparece exactamente una vez, y todo elemento en B aparece exactamente una vez. La *ganancia* de un cruce X es $\prod_{(a_i,b_j)\in X} a_i^{b_j}$. Diseñe un algoritmo voraz que maximiza la ganancia de un cruce. Analize su algoritmo, justificando que es correcto usando las propiedades de elección voraz y subestructura óptima.

Ejercicio 3. Quiero dirigir un carro de una ciudad a otra a lo largo de una carretera. El tanque de combustible del carro tiene capacidad suficiente para cubrir c kilómetros, El mapa de la carretera indica la localización de los puestos de combustible. Queremos encontrar un algoritmo que garantize el viaje con el menor número de abastecimientos.

Mas formalmente, usted recibe un arreglo A de n números reales, cada uno de los $n-1$ primeros números indica los puntos de posibilidad de recarga y el último elemento indica el lugar de destino. Debe encontrar un subconjunto de los elementos de A con menor tamaño, que cumpla que nunca se le va a agotar la gasolina, es decir, $A[i+1] \leq A[i] + p$ para todo $i < n$.

Diseñe un algoritmo voraz. Analize su algoritmo, justificando que es correcto usando las propiedades de elección voraz y subestructura óptima.

Ejercicio 4. Dado un arreglo A de n números naturales, encontrar un arreglo B de tamaño n, que tenga a los elementos de A permutados y que minimize la suma $\sum_{i=1}^{n} iB[i]$.

Diseñe un algoritmo voraz. Analize su algoritmo, justificando que es correcto usando las propiedades de elección voraz y subestructura óptima.

Ejercicio 5. Escriba una versión recursiva del algoritmo MochilaFraccionaria-Greedy visto en clase

Ejercicio 6. Corra el algoritmo de Huffman para $S = \{1, 2, 3, 4, 5, 6\}$ con hojas unitarias y ponderación $p(1) = p(2) = p(3) = p(4) = p(5) = p(6)$. De tambíen un árbol de Huffman no óptimo para esa misma ponderación.

Ejercicio 7. Muestre que un árbol de Huffman con m hojas tiene $m-1$ nodos internos.

Ejercicio 8. Corra la implementación con fila de prioridades de Huffman Huffman-Fila-Prioridades para $S = \{1, 2, 3, 4, 5, 6\}$ con hojas unitarias y ponderación $p(1) = p(2) = p(3) = p(4) = p(5) = p(6)$.

Ejercicio 9. Corra la implementación con fila de prioridades de Huffman (Huffman-Fila-Prioridades) para $S = \{1, 2, 3, 4, 5, 6, 7, 8\}$ con hojas unitarias y ponderación $p(1) = 10, p(2) = 3, p(3) = 5, p(4) = 7, p(5) = 8, p(6) = 6$.