Método voraz: pago óptimo con monedas

(Cortesía de los autores del libro *Estructuras de Datos y Métodos Algorítmicos, 213 ejercicios resueltos*, 2ª edición, N. Martí Oliet, Y. Ortega Mallén, y A. Verdejo, Garceta 2013. Las notas correponden al Ejercicio 12.6 de dicho libro)

Asignatura: Métodos Algorítmicos de Resolución de Problemas. **Profesor:** Ricardo Peña

Curso 2014/15

1. Enunciado

Se tiene un conjunto finito $M = \{m_0, m_1, \ldots, m_n\}$ de tipos de monedas, donde cada m_i es un número natural que indica el valor de las monedas de tipo i, y se cumple $m_0 < m_1 < \cdots < m_n$. Suponiendo que la cantidad disponible de monedas de cada tipo es ilimitada, se quiere pagar de forma exacta una cantidad $C > 0$ utilizando un número mínimo de monedas. En general, el problema puede no tener solución, porque C puede ser inalcanzable de forma exacta, pero si $m_0 = 1$ la existencia de solución está claramente garantizada.

1. Una estrategia voraz para encontrar una solución consiste en considerar los tipos de monedas de mayor a menor valor e ir cogiendo tantas monedas de cada tipo considerado como sea posible. Escribir un algoritmo que implemente esta estrategia. Demostrar que, en general, esta estrategia no siempre da lugar a una solución óptima que minimice el número total de monedas.

2. Demostrar que si $M = \{v^0, v^1, \ldots, v^n\}$ para algún entero $v > 1$, la estrategia voraz propuesta en el apartado anterior da lugar a una solución óptima.

3. Demostrar que la estrategia voraz también es correcta en el caso más general en que cada tipo de moneda tiene un valor múltiplo del anterior: es decir, $m_0 = 1$ y para todo i entre 1 y n se tiene que $m_i = v_{i-1} m_{i-1}$ para algún $v_{i-1} > 1$.

4. Hasta ahora se suponía ilimitada la cantidad disponible de monedas de cada tipo. Demostrar que cuando cada tipo de moneda tiene un valor múltiplo del anterior, la estrategia voraz sigue siendo correcta aunque se limite la cantidad de monedas disponibles de cada tipo.

2. Solucion

Apartado 1

El siguiente algoritmo implementa la estrategia propuesta:

```
fun monedas-v (M[0..n] de nat+, C:nat) dev sol[0..n] de nat
    sol[0..n] := [0]
    falta := C;  i := n
    mientras falta != 0 && i >= 0 hacer
       sol[i] := falta div M[i]
       falta  := falta mod M[i]
       i := i - 1
    fmientras
ffun
```

La condición $i \geq 0$, para no comprobar más tipos de monedas de los que tenemos, no es necesaria cuando $m_0 = 1$, pues si i llega a valer -1, tendremos $falta = 0$, ya que la última asignación habrá sido $falta := falta \ mod \ 1$.

Sin embargo, el algoritmo no siempre encuentra una solución óptima. Consideremos un conjunto de monedas de valores $M = \{1, 4, 6\}$ y una cantidad a pagar $C = 8$. El algoritmo anterior devolverá como solución pagar con una moneda de valor 6 y dos monedas de valor 1, lo que hace un total de tres monedas; mientras que la solución óptima es pagar con dos monedas de valor 4.

En el capítulo dedicado a la técnica de programación dinámica se verá cómo resolver este problema en el caso general.

Apartado 2

Supongamos que tenemos un conjunto de tipos de monedas de la forma $M = \{v^0, v^1, \ldots, v^n\}$ para un cierto $v > 1$. La demostración de la corrección de la estrategia voraz se apoya en el siguiente resultado:

Lema: Para cualquier solución óptima $Y = (y_0, y_1, \ldots, y_n)$ se cumple que

$$\forall i : 0 \leq i < n : y_i < v. \tag{1}$$

Demostración: Si existiera algún i tal que $y_i \geq v$, entonces se cumpliría que $y_i = y_i' + v$ (con $y_i' \geq 0$) y, por tanto, $y_i v^i = y_i' v^i + v^{i+1}$, con lo que se podría pagar la cantidad $y_i v^i$ con y_i' monedas de valor v^i y una moneda de valor v^{i+1} y, ya que $y_i' + 1 < y_i$, mejoraríamos la solución, lo cual contradice la optimalidad de Y.

Sea entonces $X = (x_0, x_1, \ldots, x_n)$, donde x_i es el número de monedas de valor v^i, la solución del algoritmo voraz. Sea $Y = (y_0, y_1, \ldots, y_n)$ una solución óptima, que existe porque $v^0 = 1$ y el número de posibles soluciones es finito. Como X e Y son soluciones, se cumple que

$$\sum_{i=0}^{n} x_i v^i = C = \sum_{i=0}^{n} y_i v^i.$$

Comparamos X con Y empezando por las monedas de mayor valor, siendo j el primer tipo de moneda en el que difieran; entonces se tiene:

$$x_j v^j + \sum_{i=0}^{j-1} x_i v^i = C' = y_j v^j + \sum_{i=0}^{j-1} y_i v^i.$$

No puede ocurrir que $x_j < y_j$, porque el algoritmo voraz coge en cada paso el máximo posible de monedas del tipo correspondiente. Tampoco puede ocurrir que $x_j > y_j$, porque entonces no se puede compensar en el resto de Y la cantidad no pagada con monedas de valor v^j, como demostramos a continuación. Formalmente, se tiene que

$$x_j > y_j \Rightarrow \sum_{i=0}^{j-1} y_i v^i \geq \sum_{i=0}^{j-1} y_i v^i - \sum_{i=0}^{j-1} x_i v^i = (x_j - y_j) v^j \geq v^j.$$

Por otra parte, por el resultado (1), se tiene que cada $y_i \leq v - 1$, entonces

$$\sum_{i=0}^{j-1} y_i v^i \leq \sum_{i=0}^{j-1} (v-1) v^i = (v-1) \sum_{i=0}^{j-1} v^i = (v-1) \frac{v^{j-1} v - 1}{v - 1} = v^j - 1 < v^j,$$

con lo que se llega a una contradicción.

Concluimos entonces que $\forall j : 0 \leq j \leq n : x_j = y_j$, por lo que X e Y resultan ser la misma solución. Así pues, la solución generada por el algoritmo propuesto es óptima.

Apartado 3

La demostración sigue exactamente el mismo esquema que en el apartado anterior, donde los valores eran potencias de un valor base. En este caso se verifica una propiedad análoga al resultado (1) del apartado anterior.

Lema: Para cualquier solución óptima $Y = (y_0, y_1, \ldots, y_n)$ se cumple que

$$\forall i : 0 \leq i < n : y_i < v_i. \tag{2}$$

Demostración: Si $y_i \geq v_i$, entonces podemos escribir $y_i = v_i + y'_i$, con $y'_i \geq 0$, de donde $y_i m_i = v_i m_i + y'_i m_i = m_{i+1} + y'_i m_i$, por lo que se podrían sustituir $v_i > 1$ monedas de valor m_i por una moneda de valor m_{i+1}, contradiciendo la optimalidad de Y.

A continuación, verificamos el siguiente resultado:

Lema:

$$\forall k : 1 \leq k < n : \sum_{i=0}^{k} (v_i - 1) m_i < m_{k+1} \tag{3}$$

Demostración: Por inducción sobre k. Como caso básico, para $k = 0$ tenemos $(v_0 - 1)m_0 < v_0 m_0 = m_1$. Supongamos que $\sum_{i=0}^{k-1} (v_i - 1)m_i < m_k$; entonces

$$\sum_{i=0}^{k} (v_i - 1)m_i = \sum_{i=0}^{k-1} (v_i - 1)m_i + (v_k - 1)m_k <^{h.i.} m_k + (v_k - 1)m_k = v_k m_k = m_{k+1}.$$

Sea $X = (x_0, x_1, \ldots, x_n)$ la solución del algoritmo voraz para cambiar una cantidad C, que comparamos con una solución óptima $Y = (y_0, y_1, \ldots, y_n)$ (dicha solución existe). Sea $j \leq n$ el tipo de moneda de mayor valor tal que $x_j \neq y_j$. Como el algoritmo sigue la estrategia de coger el máximo número posible de monedas de cada tipo, tenemos $y_j < x_j$, lo que significa que en el resto de la solución óptima Y hay que compensar una cantidad $(x_j - y_j)m_j \geq m_j$ utilizando monedas de valores $m_0, \ldots, m_{j-1}$. Pero ya sabemos por (2) que $y_i < v_i$ (equivalentemente $y_i \leq v_i - 1$) para i entre 0 y $j - 1$, por lo que, utilizando el resultado (3),

$$\sum_{i=0}^{j-1} y_i m_i \leq \sum_{i=0}^{j-1} (v_i - 1)m_i < m_j,$$

de modo que la compensación es imposible, y por tanto X debe ser igual a Y.

Apartado 4

En primer lugar, hay que hacer notar que cuando limitamos el número de monedas disponible de cada tipo ($l_i \geq 0$ para todo i entre 0 y n), puede que no exista solución, incluso cuando haya monedas de valor 1. Para demostrar que la estrategia voraz propuesta sigue siendo óptima, demostraremos el siguiente resultado:

> Si existe solución y hay una moneda que pudiendo ser utilizada no se utiliza, entonces es posible modificar la solución para utilizar dicha moneda, de forma que además se reduzca el número total de monedas usadas.

Esto es una consecuencia de la siguiente afirmación:

Lema: Sea $M = \{m_0, m_1, \ldots, m_n\}$ donde para todo i entre 1 y n se tiene que $m_i = v_{i-1} m_{i-1}$ para algún entero $v_{i-1} > 1$. Sean $x_i \geq 0$, para todo i entre 0 y n, y sea k con $0 \leq k \leq n$. Entonces

$$\forall l : l > 0 : \sum_{i=0}^{k} x_i m_i \geq l m_{k+1} \Rightarrow (\forall i : 0 \leq i \leq k : (\exists y_i : 0 \leq y_i \leq x_i : \sum_{i=0}^{k} y_i m_i = l m_{k+1})).$$

Demostración: Por inducción sobre k. Como caso básico tenemos $k = 0$. Si $x_0 m_0 \geq l m_1 = l v_0 m_0$, se sigue que $x_0 \geq l v_0$. Basta entonces tomar $y_0 = l v_0$ para obtener el resultado $y_0 m_0 = l v_0 m_0 = l m_1$.

Supongamos que es cierto para $k-1$, y sea $l > 0$ tal que $x_k m_k + \sum_{i=0}^{k-1} x_i m_i \geq l m_{k+1}$. Distinguimos dos posibilidades:

1. $x_k \geq l v_k$.
   Nos basta tomar $y_k = l v_k \leq x_k$ e $y_i = 0 \leq x_i$, para todo i entre 0 y $k-1$, obteniendo

$$y_k m_k + \sum_{i=0}^{k-1} y_i m_i = l v_k m_k + 0 = l m_{k+1}.$$

2. $x_k < l v_k$.
   A partir de la hipótesis $x_k m_k + \sum_{i=0}^{k-1} x_i m_i \geq l m_{k+1}$ se tiene

$$\sum_{i=0}^{k-1} x_i m_i \geq l m_{k+1} - x_k m_k = l v_k m_k - x_k m_k = (l v_k - x_k) m_k,$$

   con $l v_k - x_k > 0$. Por hipótesis de inducción para $k-1$, existen $0 \leq y'_i \leq x_i$, para todo i entre 0 y $k-1$, tales que

$$\sum_{i=0}^{k-1} y'_i m_i = (l v_k - x_k) m_k.$$

   Tomamos $y_k = x_k$ e $y_i = y'_i$, para todo i entre 0 y $k-1$, con los cuales obtenemos

$$y_k m_k + \sum_{i=0}^{k-1} y_i m_i = x_k m_k + \sum_{i=0}^{k-1} y'_i m_i = x_k m_k + (l v_k - x_k) m_k = l v_k m_k = l m_{k+1}.$$

Supongamos ahora que existe una solución $X = (x_0, x_1, \ldots, x_n)$ tal que

$$\exists k : 0 \leq k \leq n-1 : l_{k+1} > x_{k+1} \wedge \sum_{i=0}^{k} x_i m_i > m_{k+1},$$

es decir, existe alguna moneda disponible del tipo m_{k+1} que, pudiendo haber sido utilizada en la solución, no se ha utilizado. Tomando $l = 1$ el último lema nos permite afirmar que es posible quitar $\sum_{i=0}^{k} y_i > 1$ monedas de los tipos más pequeños y coger en su lugar una moneda del tipo m_{k+1}. Por tanto, si la solución no sigue la estrategia, entonces no es óptima, pues se puede ir mejorando a base de ir aplicando este resultado.