UADE

Exposición de experto

Temas a desarrollar

1. Problema del cambio
2. Estrategia de solución
3. Algoritmo
4. Complejidad temporal
5. Síntesis

Problema del cambio

El problema del cambio. Las reglas del juego.

El problema del cambio consiste en encontrar la forma de pagar un monto de valor ***v*** con monedas, utilizando la mínima cantidad de estas.
Las monedas son de denominaciones $\mathbf{d_1}$, $\mathbf{d_2}$, … , $\mathbf{d_k}$, con una cantidad ilimitada de cada denominación.

Ejemplo:

Monedas: **{5, 2, 1, 0.50, 0.20, 0.10, 0.05}**

Hay varias opciones para pagar **9.25**; por ejemplo:

2 + 2 + 2 + 2 + 1 + 0.20+ 0.05

5 + 2 + 2 + 0.10 + 0.10 + 0.05

&c

Intuición: conviene usar siempre la mayor moneda que sea posible.

Ejemplo: una picada frente al lago

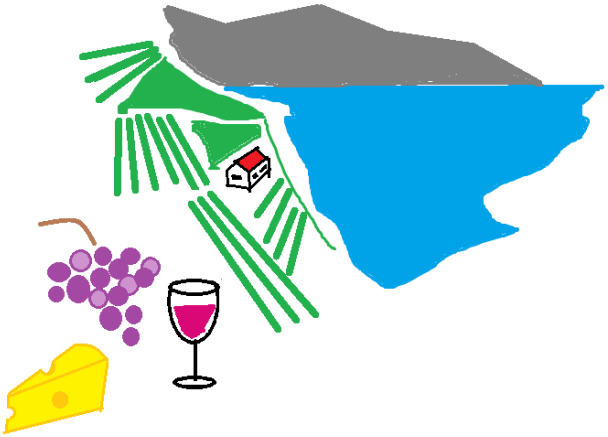
Monedas

5	2	1	0.50	0.20	0.10	0. 05
✓✓✓	✓	✓	✓	✓✓		✓

Monto: 18.95

Pago: 5 5 5 2 1 0.50 0.20 0.20 0.05

A pagar: 0.00 **Monedas usadas: 9**



UADE

Problema del Cambio *Greedy* - Estrategia

Conjunto candidatos	Las diferentes denominaciones de monedas
Función selección	Intentar seleccionar una moneda de mayor valor
Función factibilidad	Validar que la monedad seleccionada no supere el monto que debe pagarse
Función solución	Verificar que no se haya alcanzado el valor que queremos pagar
Función objetivo	Minimizar la cantidad de monedas a devolver para un valor dado

El algoritmo *Greedy* para el problema del cambio

ALGORITMO CAMBIO
Entrada: v entero
Salida: n entero

```
entero n ← 0
entero s ← 0
entero i ← 0
vector monedas = [500, 200, 100, 50, 20, 10, 5]
mientras (s < v) Y (i < longitud(monedas))
    si (s + monedas[i] ≤ v)
        s ← s + monedas[i]
        n ← n + 1
    sino
        i ← i + 1
    fin si
fin mientras
si (i < longitud(monedas))
    devolver n
sino
    devolver -1
fin si
```

Suponemos que el vector de denominaciones está ordenado de manera decreciente

Comenzamos usando la moneda de mayor valor mientras se pueda; cuando no sea ya posible porque excede el monto por pagarse, pasamos a la siguiente

El algoritmo devuelve la cantidad de monedas usadas. Se lo puede adaptar para que informe cuáles se usaron

Problema del cambio *Greedy* – Complejidad temporal

```
ALGORITMO CAMBIO
Entrada: v: entero
Salida: n: entero
  entero n ← 0
  entero s ← 0
  entero i ← 0
  Vector monedas = [100, 50, 25, 10, 5, 1]
  mientras (s < v) Y (i < longitud(monedas))
    si s + monedas[i] < v
      s ← s + monedas[i]
      n ← n + 1
    sino
      i ← i + 1
    fin si
  fin mientras
  si i < longitud(monedas)
    devolver n
  sino
    devolver No hay solución
  fin si
```

Si es necesario ordenar el repertorio de *k* valores de monedas, esto debe ser tenido en cuenta

Θ (*k* log *k*)

Θ(*v*)

La complejidad está dada por el ciclo. Aquí se itera en el peor de los casos *v* veces por un factor constante.

Ejemplo: whisky en Loch Ness, antes de 1971

Monedas

libra (240)

corona (60)

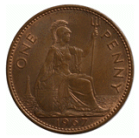

½ corona (30)


florín (24)


chelín (12)


6p (6)

3p (3)



p (1)

Monto: 1£ 48 p

Pago: 1£ 48 p

A pagar: 0 **Monedas usadas:** 4

Solución óptima: 1 libra y dos florines (3 monedas)

UADE

¿Es el algoritmo *Greedy* correcto?

- La solución al problema del cambio dada por el algoritmo *greedy* para el primer conjunto de monedas dado {5F, 2F, 1F, 50¢, 20¢, 10¢, 5¢} es correcto por las denominaciones particulares de monedas. La condición de que cada moneda sea por lo menos el doble de la inmediata menor y las denominaciones cubran todas las posibilidades de pago es suficiente para garantizar que el algoritmo *greedy* da una solución óptima.

- En el caso del sistema monetario imperial británico (antes del pasaje al sistema decimal) esa condición no se cumple y el algoritmo no siempre da una solución óptima, como vimos en el segundo ejemplo.

- Se verán técnicas para encontrar soluciones óptimas a este problema en todos los casos en este mismo curso.

BRASSARD, Gilles. *Fundamentals of algorithmics.* Prentice Hall, 1996. ISBN: 9780133350685

Síntesis

Síntesis

- El problema del cambio consiste en encontrar la forma de pagar un monto de valor ***v*** con monedas utilizando la mínima cantidad de estas. Las monedas son de denominaciones **d1, d2, … , dn,** con una cantidad ilimitada de cada denominación.

- La función de selección intenta seleccionar siempre una moneda de mayor valor.

- La complejidad temporal es Θ(*v*).

- El algoritmo *Greedy* no garantiza una solución correcta. Una condición suficiente es que cada moneda sea al menos el doble de la inmediata inferior. Esta condición es satisfecha por casi todos los sistemas monetarios razonables.

- Pero para algunos conjuntos de denominaciones, el algoritmo *greedy* no funciona y es necesario recurrir a otras técnicas.

Bibliografía

BRASSARD, Gilles. *Fundamentals of algorithmics.* Upper Saddle River: Prentice Hall, 1996. ISBN: 9780133350685

¡Muchas gracias!

UADE