Universidad de La Laguna

Grado en Ingeniería Informática
Computabilidad y Algoritmia
Curso 2014-2015

PRÁCTICA 12

Algoritmos Voraces (*Greedy*): Cambio de monedas

Semana del 1 al 5 de diciembre

1. Objetivo

El objetivo principal de esta práctica consiste en desarrollar un ejemplo de algoritmo voraz (*greedy*) como es el cambio de monedas. Para ello, se detalla a continuación las clases STL necesarias, los requisitos funcionales mínimos y opcionales, ejemplo de prueba, y los criterios de evaluación.

2. Clases STL necesarias

Para poder desarrollar más fácilmente el algoritmo del cambio de monedas, se puede hacer uso de las siguientes clases de la STL (*Standard Template Library*):

```
#include <vector>
#include <list>

std::vector<T>
std::list<T>
```

También se puede hacer uso de cualquier otra clase STL que suponga una mejora significativa en el desarrollo del algoritmo. Para más información de dichas clases, se puede consultar la siguiente web: cppreference.com. Asimismo, se pueden usar las nuevas características del estándar `c++11` del compilador `g++-4.9.1`, que se activan con la orden de compilación:

```
$ g++-4.9.1 -std=c++11 ...
```

3. Requisitos funcionales mínimos

Para superar la práctica, los requisitos funcionales mínimos exigidos son:

1. Dada una cierta cantidad n (número real), el algoritmo debe devolver el conjunto de monedas correcto cuya suma total sea igual al valor de n, así como, el número total de monedas.

2. Dentro del esquema de estrategia voraz, se debe desarrollar de forma independiente la **Función de Selección** que devuelva el mejor candidato (moneda) a ser incluido en la solución.

3. Relacionado con el requisito anterior, el mejor candidato (moneda) será el valor **más alto factible** que queda en el conjunto de candidatos. Es decir, ser deberá explorar el conjunto de candidatos desde la posición de la última moneda seleccionada, y no desde el principio.

4. Requisitos funcionales opcionales

Dentro de la evaluación global de la práctica, se valorarán positivamente los siguientes requisitos optativos:

- Devolver el cambio contando el número de monedas de cada tipo. Por ejemplo, si $n = 7,98$€, el cambio devuelto sería **3** $\times$ 2€, 1€, 50c, **2** $\times$ 20c, 5c, 2c, 1c.
- Considerar tanto monedas como billetes (5€, 10€, 20€, 50€, 100€, 200€, 500€). ¿Cambia en algo el esquema del algoritmo original al incluir los billetes en el conjuno de candidatos?

5. Ejemplos

Se puede probar la aplicación desarrollada usando el siguiente ejemplo:

- Solución básica: $n = 7,43$€ $\rightarrow$ $S =$ { 2€, 2€, 2€, 1€, 20c, 20c, 2c, 1c }, con el número total de monedas igual a 8.
- Solución más elaborada: $n = 5,34$€ $\rightarrow$ $S =$ { 2 $\times$ 2€, 1€, 20c, 10c, 2 $\times$ 2c }, con el número total de monedas igual a 7.

6. Evaluación

Se evaluará positivamente los siguientes aspectos:

- Presentación en el laboratorio: el grado de funcionamiento de la práctica, y si desarrolla requisitos opcionales o mejoras significativas.
- Código subido en la tarea correspodiente a la práctica: buen diseño y limpieza.