ALGORITMIA

Grado en Ingeniería Informática del Software - Curso 2015-2016

GUIÓN DE LA PRÁCTICA 3

OBJETIVO:

- **Divide y Vencerás: modelos recursivos y ejemplos**

Modelos recursivos básicos

El paquete **alg77777777.p3** contiene diez clases siguientes:

Las clases **Sustraccion1.java** y **Sustraccion2.java** tienen un esquema por *SUSTRACCIÓN* con a=1, lo que lleva consigo un **gran gasto de pila**, de hecho se desborda cuando el tamaño del problema crece hasta algunos miles. Afortunadamente los problemas así solucionados, van a tener una mejor solución iterativa (con bucles) que la solución de este tipo (sustracción con a=1).

La clase **Sustraccion3.java** tiene un esquema por *SUSTRACCIÓN* con a>1, lo que lleva consigo un **gran tiempo de ejecución** (exponencial o no polinómico). Esto supone que para un tamaño del problema de algunas decenas el algoritmo no acaba (*tiempo intratable, NP*). La consecuencia es que debemos procurar no plantear soluciones del tipo *SUSTRACCIÓN* con varias llamadas (a>1)

Las clases **Division1.java, Division2.java** y **Division3.java** tienen un esquema por *DIVISIÓN*, siendo la primera del tipo $a<b^k$, la segunda del tipo $a=b^k$ y la última del tipo $a>b^k$ (ver cómo se definen estas constantes en teoría).

SumaVector1.java resuelve de tres formas diferentes el sencillo problema de sumar los elementos de un vector y **SumaVector2.java** mide tiempos de esos tres algoritmos para diferentes tamaños del problema.

Fibonacci1.java resuelve de cuatro formas diferentes el problema de calcular el número de Fibonacci de orden n y **Fibonacci2.java** mide tiempos de esos cuatro algoritmos para diferentes tamaños del problema.

Escribir las siguientes clases recursivas para obtener las siguientes complejidades de ejecución:

- **Sustraccion4.java**, método recursivo POR SUSTRACCION con una complejidad $O(3^{n/2})$
- **Division4.java**, método recursivo POR DIVISIÓN con una complejidad $O(n^2)$ y el número de subproblemas = 4.

TRABAJO PEDIDO

Realizar un análisis de las complejidades y empírico de las 10 clases anteriores. Estudiando el código y ejecutándolo.
Escribir el código para Sustracción4.java y Division4.java

Las clases que programe las incluirá dentro del paquete alg<dnipropio>.p30. Si utiliza Eclipse llamar al proyecto prac03_Recursivo<UOpropio>

Problema Divide y Vencerás

Problema de la moneda falsa

Aunque siempre ha existido, en los últimos años, quizá debido a la crisis y a la globalización, se está detectando cada vez más casos de falsificación de moneda. La policía científica (C.S.I.) tiene unidades que intentan detectar y confiscar las falsificaciones. En este caso la policía ha recibido la información de que una banda quiere introducir al mercado monedas falsas cuya única forma de diferenciarlas de las originales es su peso, aunque no se sabe si este es mayor o menor que el de las auténticas. La banda, para dificultar la detección, ya tiene preparadas gran cantidad de bolsas en las que todas las monedas son originales salvo una que es falsa. Para descubrir que moneda es falsa la unidad de la policía científica dispone de una balanza para comparar el peso de dos conjuntos de monedas. En casa pesada se pueden introducir las monedas que se desee y la información que nos proporciona es si la balanza queda equilibrada, si pesan más los objetos de la derecha o si pesan más los de la izquierda.

TRABAJO PEDIDO

Las clases que programe las incluirá dentro del paquete alg<dnipropio>.p31. Si utiliza Eclipse llamar al proyecto prac03_MonedaFalsa<UOpropio>

Diseñar un algoritmo eficiente para encontrar la moneda falsa y decidir si pesa más o menos que las auténticas.

Implementar dicho algoritmo en Java, utilizando la clase Monedas.java proporcionada.

Calcular la complejidad y medir tiempos de forma empírica para distintos tamaños del problema.

N	*t ordenado*
10000	…..
20000	…..
40000	…..
80000	…..
160000	…..
320000	…..