ALGORITMIA

Grado en Ingeniería Informática del Software - Curso 2019-2020

GUION DE LA PRÁCTICA 3

OBJETIVO:

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

Modelos recursivos básicos

El paquete alg77777777.p**3** 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).

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 6 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>.p31 , si utiliza Eclipse se incluirá en el proyecto conjunto con el nombre: prac03_Gilito<UOpropio>

Problema Divide y Vencerás

El tío Gilito

El Tío Gilito (tío del Pato Donald) es sin duda el más acaudalado de Disney, ya que a base de operaciones financieras (sus detractores las clasifican dentro de la usura), ha acumulado un ingente patrimonio, del que no pierde oportunidad de presumir.
La realidad es que sus compañeros de Disney no tienen a quién pedirle dinero, si no es a él. Por eso, le piden créditos (monedas de oro iguales), a los que por supuesto Gilito les aplica un interés leonino. Los acreedores se confabularon para cuando le devuelven un crédito (n monedas de oro), **siempre** le meten **una y solo una moneda** falsa, que pesa **menos** que las verdaderas.
Ante tal provocación y para detectar la moneda falsa, el tío Gilito se hizo con una balanza electrónica de última generación (nos dice si pesa menos lo del platillo izquierdo, menos lo del platillo derecho, o igual).
Cada vez que hace un pesaje gasta una energía 1 WatioHora, independientemente del peso que ponga en cada platillo (1 pesaje=1 WatioHora).
Actualmente tiene el algoritmo escenificado en la clase Gilito1, que le supone un gasto en la factura de luz superior a la estafa, que le maquinaron, de la moneda falsa.

Por ello, **NOS PIDE**:

A) Estudiar bien el sistema actual de las clases comentadas **Gilito1.java** y **Gilito1Tiempos1.java**.

B) Diseñar una clase **Gilito2.java,** con un algoritmo que **minimice** el gasto energético, y por ende la factura de la luz.

Para el algoritmo diseñado, se debe calcular la cantidad media (media de los n casos que suponen la moneda falsa en cada una de las posiciones) de WatiosHora consumidos por la balanza, para un tamaño de monedas n=100, 200, 400, 800, … (hasta n>10000).
Rellenar una tabla con esos valores de gasto energético antes obtenidos.
Desde el punto de vista teórico, razonar: ¿qué función modela el gasto energético antes calculado?

C) Diseñar una clase **Gilito2Tiempos.java,** que calcule el tiempo de ejecución en el **caso peor** del algoritmo diseñado. Está claro que el caso peor supone la colocación de la moneda falsa en una posición que suponga que el algoritmo se comporte lo peor posible.

El tiempo se ha de calcular para un tamaño de monedas n=100, 200, 400, 800, … (hasta n>10000).
Rellenar una tabla con esos valores de tiempos antes obtenidos.
Desde el punto de vista teórico, razonar: ¿qué función modela el tiempo de ejecución antes calculado?

Las clases que programe las incluirá dentro del paquete alg<dnipropio>.p32 Si utiliza Eclipse llamar al proyecto prac03_Gilito<UOpropio>
La respuesta a las preguntas planteadas en el enunciado y las tablas con los tiempos se incluirán en un documento.
En el campus virtual se entregarán dos ficheros:

1. *ZIP con el proyecto / código y*
2. *PDF documento de tiempos, según el calendario previsto.*