GUION DE LA PRÁCTICA 3 (directorio p3)

0)INTRODUCCIÓN

Se abordará el estudio de la complejidad temporal y el tiempo de ejecución de algoritmos recursivos Divide y Vencerás (**DyV**), tanto para el esquema de **Sustracción** como para el de **División**.

La segunda parte de la práctica consiste en dar solución a un problema concreto mediante la técnica DyV, medir los tiempos de ejecución del algoritmo que se implemente y proceder al análisis final de los tiempos obtenidos.

Seguiremos en esta práctica midiendo los tiempos SIN_OPTIMIZACIÓN, ya que incluso en este caso de modelos recursivos, el optimizador JIT parece que produce mayor distorsión de tiempos que el caso de modelos iterativos (vistos en las sesiones anteriores). En las tablas de tiempos que nos pidan, si cualquier tiempo se alarga más de 1 minuto pondremos Fuera de Tiempo (FdT).

1)DyV por SUSTRACCIÓN

Las clases **Sustraccion1.java** y **Sustraccion2.java** tienen un esquema DyV por *SUSTRACCIÓN* con ***a*=1**, lo que lleva consigo un **gran gasto de pila**. Afortunadamente, los problemas así solucionados, van a tener en la práctica una mejor solución iterativa (con bucles) que la solución recursiva de este tipo (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).

Procede:

```
javac *.java
dir
java -Djava.compiler=NONE Sustraccion1 1 //10,100,…
java -Djava.compiler=NONE Sustraccion2 1 //10,100,…
java -Djava.compiler=NONE Sustraccion3 1 //10,100,…
```

SE LE PIDE:

Tras analizar la complejidad de las tres clases anteriores, no se le pide hacer sus tablas de tiempos, pero sí razonar si los tiempos coinciden o no con la complejidad temporal de cada algoritmo.

Contestar: ¿para qué valor de n dejan de dar tiempo (abortan) las clases Sustraccion1 y Sustraccion2? ¿por qué sucede eso?

Calcular: ¿cuántos años tardaría en finalizar la ejecución Sustracción3 para n=80?

Implementar una clase Sustraccion4.java con una complejidad $O(n^3)$ (con su pertinente toma de tiempos) y después rellenar una tabla en la que se muestre el tiempo (en msg.) para n=100, 200, 400, 800, … (hasta FdT).

Implementar una clase Sustraccion5.java con una complejidad $O(3^{n/2})$ (con su pertinente toma de tiempos) y después rellenar una tabla en la que se muestre el tiempo (en msg.) para n=30, 32, 34, 36, … (hasta FdT).

Calcular: ¿cuántos años tardaría en finalizar la ejecución Sustracción5 para n=80?

2)DyV por DIVISIÓN

Las clases **Division1.java, Division2.java y Division3.java** tienen un esquema por DyV por División, respectivamente del tipo $a<b^k$, $a=b^k$ y $a>b^k$ (ver cómo se definen estas constantes en teoría).

Procede:

```
dir
java -Djava.compiler=NONE Division1 1 //10,100,…
java -Djava.compiler=NONE Division2 1 //10,100,…
java -Djava.compiler=NONE Division3 1 //10,100,…
```

SE LE PIDE:

Tras analizar la complejidad de las tres clases anteriores, no se le pide hacer sus tablas de tiempos, pero sí razonar si los tiempos coinciden o no la complejidad temporal de cada algoritmo.

Implementar una clase Division4.java con una complejidad $O(n^2)$ (con $a<b^k$) y la pertinente toma de tiempos. Después rellenar una tabla en la que se muestre el tiempo (en msg.) para n=1000, 2000, 4000, 8000, … (hasta FdT).

Implementar una clase Division5.java con una complejidad $O(n^2)$ (con $a>b^k$) y la pertinente toma de tiempos. Después rellenar una tabla en la que se muestre el tiempo (en msg.) para n=1000, 2000, 4000, 8000, … (hasta FdT).

3)DOS EJEMPLOS BÁSICOS

Cada una de las clases **VectorSuma.java y Fibonacci.java** resuelven un problema básico de diversas formas, la primera sumar los elementos de un vector de n componentes y la segunda calcular el número de Fibonacci de un orden n dado.

SE LE PIDE:

Tras analizar la complejidad de los diversos algoritmos que hay dentro de las dos clases, ejecutarlas y tras poner en una tabla los tiempos obtenidos, comparar la eficiencia de cada algoritmo.

4)PRÁCTICA DyV

El problema que se pide resolver nos es muy conocido, ordenar n elementos ordenables (p.e. de tipo entero). Este problema tiene tres buenos algoritmos casi lineales **O(nlogn)**: **Quicksort** (Rápido), **MergeSort** (Mezcla) y **HeapSort** (Montículo). Pues bien, de esos tres algoritmos, dos de ellos son DyV (Rápido y Mezcla) y serán el objetivo del estudio posterior que se pide.

SE LE PIDE:

Implementar una clase Mezcla.java que ordene los elementos de un vector.

Implementar una clase MezclaTiempos.java que tras ser ejecutada sirva para rellenar la siguiente tabla (análoga a las vistas en la práctica 2):

TABLA ALGORITMO MEZCLA

(tiempos en milisegundos y SIN_OPTIMIZACIÓN):

Pondremos "FdT" para tiempos superiores al minuto y "FdF" los menores a 50 msg.

N	*t ordenado*	*t inverso*	*t aleatorio*
31250			
62500			
125000			
250000			
500000			
1000000			
2000000			
4000000			
8000000			
… (hasta FdT)			

Rellene la tabla siguiente, trasladando a ella los tiempos obtenidos para el Rápido en el caso aleatorio en la práctica 2 y los obtenidos aquí para el Mezcla en el caso aleatorio. ¿Qué constante le sale como comparación de ambos algoritmos?

TABLA ALGORITMO MEZCLA vs. RÁPIDO (caso aleatorio)

(tiempos en milisegundos y SIN_OPTIMIZACIÓN):

Pondremos "FdT" para tiempos superiores al minuto y "FdF" los menores a 50 msg.

n	*t mezcla(t1)*	*t rapido(t2)*	*t1/t2*
250000			
500000			
1000000			
2000000			
4000000			
1000000			
2000000			
4000000			
8000000			

Se ha de entregar en un **.pdf** el trabajo que se le pide y además las clases **.java** que ha programado. Todo ello lo organizará en una carpeta, que es la que entregará comprimida en un fichero **practica3ApellidosNombre.zip**.

Esta práctica es para dos semanas y el **plazo límite** de entrega de esta práctica es **un día antes** de que comience la siguiente práctica, en la tarea que se creará a tal efecto en el Campus Virtual.