ALGORITMIA

Grado en Ingeniería Informática del Software - Curso 2018-2019

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>.p30 , si utiliza Eclipse llamar al proyecto prac03_EnteroGrande<UOpropio>
Se entregará en el campus virtual según el calendario previsto.

Problema Divide y Vencerás

Multiplicación de Enteros Grandes

La operación de multiplicar dos números es uno de los algoritmos elementales que aprendemos de pequeños. Esta operación se encuentra implementada y optimizada para números binarios en el firmware de todos los procesadores. Dependiendo de la arquitectura del procesador se pueden realizar operaciones de forma directa con enteros de 32 bits o de 64 bits, aunque dependiendo del tamaño de estos se puede producir un desbordamiento, con lo que si no se controla se podrían obtener resultados incorrectos. Esto nos lleva a poder manejar productos de enteros que como mucho alcanzan 19 dígitos sin signo o 18 dígitos con signo (para el caso de 64 bits, un long de Java).

¿Qué podemos hacer si queremos realizar operaciones con enteros que contengan más dígitos? Una opción sería utilizar reales con la notación en coma flotante que permite representar números más grandes, pero sólo conservaríamos los primeros dígitos significativos, con lo cual si queremos conservar toda la precisión de un entero con muchos dígitos no podríamos. Así que, la otra alternativa sería inventarnos un nuevo tipo de datos (realmente en orientación a objetos será una nueva clase) que contenga una estructura de datos para almacenar todos los dígitos de los enteros y que implemente todas las operaciones, utilizando esa estructura de datos, mediante los algoritmos a nivel de nuestro lenguaje de programación.

En esta práctica vamos a partir de una clase *`EnteroGrande`* parcialmente implementada: utiliza un String para almacenar los dígitos de los enteros y ya dispone de ciertas operaciones básicas. Vamos a centrarnos en la operación de multiplicar que puede darnos cierto juego con varios algoritmos alternativos:

- Disponemos del algoritmo “clásico” que aprendemos todos en la escuela y que podríamos retomar para aplicar en este caso.
- Podríamos hacer un enfoque Divide y Vencerás en el que dividamos los números en dos partes una con la mitad de los dígitos superiores y otra con la mitad de los dígitos inferiores y a partir de aquí realizar cuatro multiplicaciones y componer los resultados.
- Podríamos utilizar un enfoque Divide y Vencerás optimizado en un algoritmo ampliamente tratado por los textos de algoritmia clásicos (Knuth, Brassard) cuya solución fue publicada por primera vez por Karatsuba en 1962[1] (utilizando la transformada de Fourier hay un método que mejora la complejidad). Este algoritmo se parece bastante al anterior, pero logra resolver el proceso con tres multiplicaciones por nivel.

SE LE PIDE:

PRIMERA PARTE

Completar la clase EnteroGrande proporcionada con la operación de multiplicar, proporcionando al menos dos implementaciones: algoritmo “clásico” y algoritmo Divide y Vencerás optimizado (algoritmo de Karatsuba). Se propone un proceso iterativo de desarrollo en el que primero y como mínimo se resuelva el caso básico con enteros positivos tratados como si tuvieran un número de cifras potencia de dos, y a partir de este se vayan refinando los casos para trabajar con enteros de cualquier número de cifras y en el que cada factor pueda tener diferente número y poder llegar al caso general (esto último puede conllevar la modificación de alguno de los métodos proporcionados).

[1] Vease para un primer apunte: https://es.wikipedia.org/wiki/Algoritmo_de_Karatsuba

Implementar en Java la solución teniendo en cuenta que la información sobre los participantes estará contenida en un fichero de texto como el que se muestra a continuación de ejemplo:

```
3
24745340*68458198
7141668043491380*3843921569889377
484583795355828918498086640273 74*352330362990886908300844467321 51
```

En el ejemplo anterior se está especificando que hay 3 parejas de Enteros Grandes, separados por el carácter * y que debemos multiplicar. Todos los ficheros de ejemplo tendrán el mismo formato.

Se deben implementar una clase convencional (o clase JUnit) para poder realizar pruebas exhaustivas de que todas las operaciones implementadas y en concreto las operaciones de multiplicar funcionan correctamente.

SEGUNDA PARTE

Queremos comparar la eficiencia de los dos algoritmos pedidos como obligatorios. Por tanto, se debe:

- Calcular la complejidad analítica que tienen estos algoritmos e incluirlo posteriormente en las tablas.
- Medir empíricamente el tiempo que nuestros algoritmos tardarían en realizar multiplicaciones de enteros con distintos números de cifras, empezando desde un tamaño **n=16** hasta que la memoria del ordenador del que disponemos "aguante" o el tiempo para un tamaño sea demasiado grande.
  Para realizar esas pruebas crear una nueva clase EnteroGrandePrueba y utilizar el método que se proporciona para generar enteros con distintos números de cifras aleatorias. Así podremos ir aumentando el tamaño siguiendo una progresión en forma de potencia de 2. Obviamente debemos medir los tiempos para cada tamaño sin tener en cuenta salidas de texto por consola que nos desvirtúen los resultados.

n	*t Mult. «Clásico»*	*t Mult. DV*
Complejidad		
16		
32		
64		
128		
…		
…		
Oveflow heap		

- Compruebe si los tiempos obtenidos en el apartado anterior cumplen o no la complejidad teórica de cada algoritmo.
- Compare los tiempos medidos de un algoritmo y otro. Explicar y justificar que ocurre en la comparativa de tiempos.

TRABAJO OPTATIVO:

Se proponen varias ampliaciones posibles en esta práctica:

1. Como se indicó en la primera parte se podría intentar ir más allá de la multiplicación de enteros de un número de cifras potencia de dos y generalizar la operación de multiplicar para cualquier entero. Indicar las modificaciones que se han tenido que realizar.

2. Una vez realizadas la implementación optimizada de DV, afinar la implementación para intentar mejorar sus tiempos de ejecución al menos para determinados tamaños del problema. Indicar las modificaciones que se han tenido que realizar. Y medir tiempos para mostrar que mejora la implementación anterior. Si se ha intentado y no se ha conseguido indicarlo de la misma manera, explicando el razonamiento seguido y por qué se considera que no ha tenido éxito.
3. ¿Es posible paralelizar esta práctica con el framework Fork-Join visto en clase de teoría? Justificar la respuesta y proporcionar una implementación si es posible.

Las clases que programe las incluirá dentro del paquete alg<dnipropio>.p31 Si utiliza Eclipse llamar al proyecto prac03_EnteroGrande<UOpropio>
La respuesta a las preguntas planteadas en el enunciado y las tablas con los tiempos se incluirán en un fichero aparte.
Todo ello se entregará en el campus virtual según el calendario previsto.