Cálculo Simbólico
Práctica 1

José Ignacio Carmona Villegas
Juan Hernández García

Estructura de datos

Se ha creado una clase (BigInt) que encapsule la representación de la estructura de datos y las operaciones.

Se ha optado por guardar una representación decimal del número, para facilitar la E/S. En Java la representación de un entero ocupa 32b (4294967296), pero debido a que pretendemos garantizar el mayor número de cifras decimales, se opta por aprovechar 30b de dichos 32.

Dicha asignación nos garantiza al menos 9 cifras decimales significativas completas, es decir, el número más grande que se puede representar en una cifra del tipo BigInt es 999999999.

Gracias a esto, dadas a,b,c cifras de BigInt, el número representado por el BigInt abc sería xxxxxxxxxxxxxxxxxxxxxxxxxxx (siendo x una cifra decimal), el cual es directamente imprimible simplemente imprimiendo las cifras a,b,c de forma secuencial.

Además, se ha añadido un booleano (1b) para indicar si el BigInt es negativo.

Las operaciones se plantean siguiendo los algoritmos explicados en clase, suponiendo números siempre positivos, y luego se envuelven (patrón decorator) para añadir el álgebra de signos.

En todo el proceso, cada vez que hay que realizar un operación aritmética, se precalcula si el tipo de dato resultante puede ser contenido en un int32, o necesita un int64(long). Los resultados parciales se van almacenando en el contenedor que mejor se ajusta a su tamaño. De esta manera se hace un uso inteligente de datos temporales.

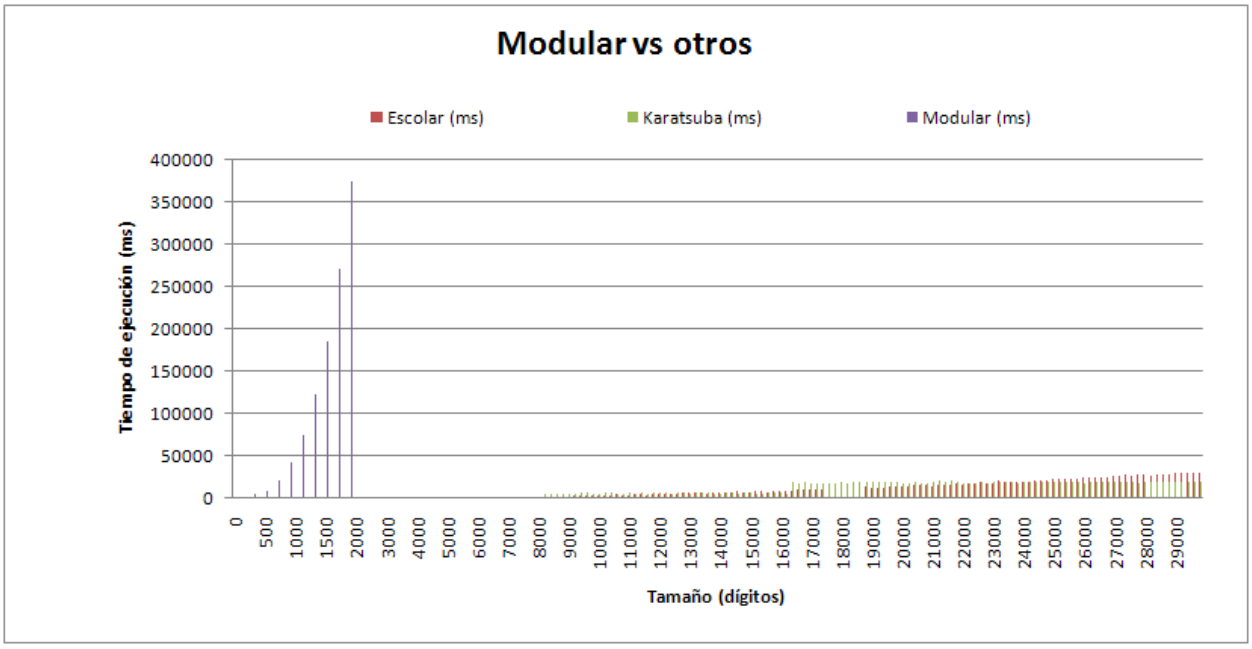
En el algoritmo modular, nos encontramos con un bug bastante interesante. Puesto que las ecuaciones de yi del algoritmo crecen en tamaño (número de operaciones) conforme aumenta i, no basta con acumular los resultados en un campo de doble longitud (64b), y para i suficientemente alto, se desborda. La solución pasa por aplicar el módulo final en cada operación, y así mantener siempre los cálculos intermedios dentro de un campo de longitud simple (32b).

Obtención de Tiempos

Para obtener los tiempos se ha creado una clase especial que permite:

- Comenzar a contabilizar tiempo.
- Pausar la cuenta del tiempo.
- Reanudar la cuenta de tiempo.
- Finalizar la cuenta del tiempo.

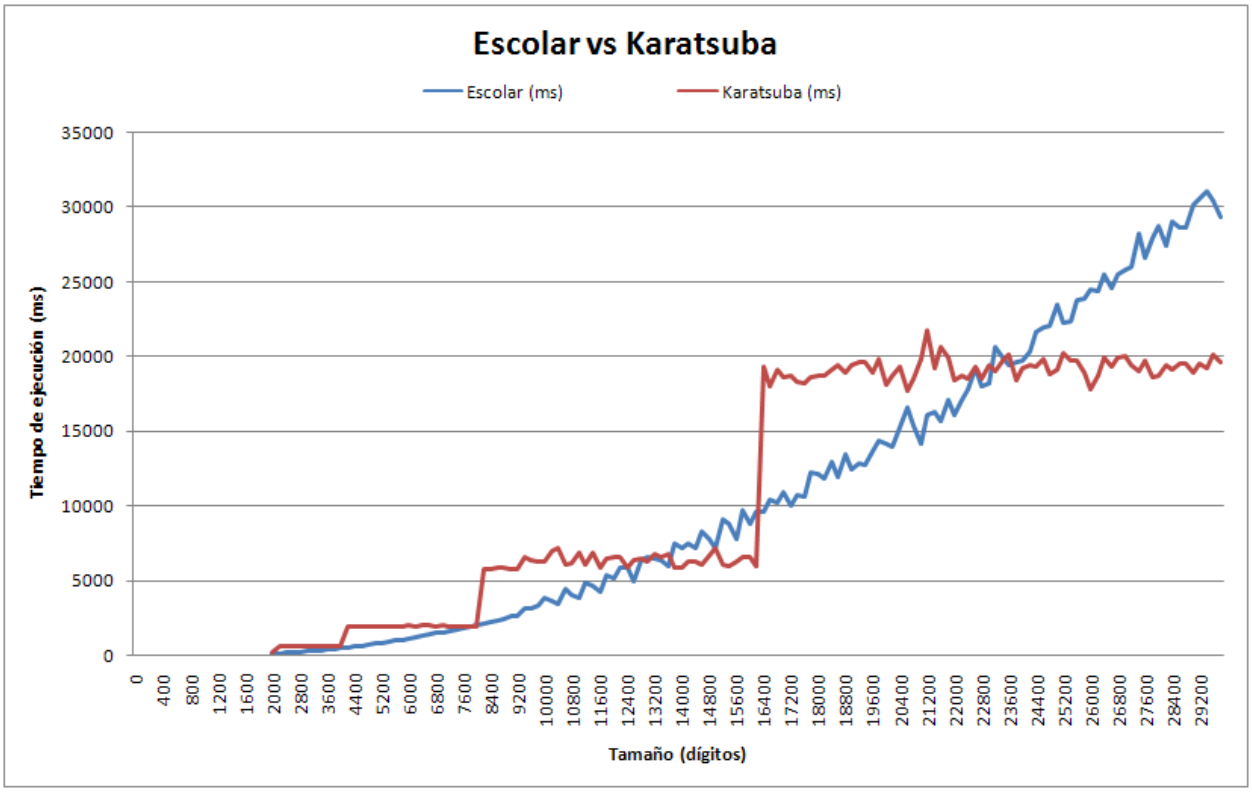
Haciendo uso de esta clase auxiliar, ubicada en el paquete “Util”, se han obtenido los tiempos siguientes:



Podemos observar que la multiplicación modular es peor en todos los casos. Se han calculado tiempos para ella para casos de menor número de dígitos involucrados porque la tendencia exponencial es clara desde un principio.

Esta gráfica se centra en mostrar la diferencia entre la multiplicación modular y el resto. Aunque sabíamos que la multiplicación modular sería más lenta desde el principio, la implementación de la misma es interesante debido a que algunos de los conceptos utilizados para ello nos serán útiles en el futuro.

Pasemos ahora a evaluar Karatsuba y Escolar.



Se puede observar perfectamente el comportamiento y tendencia de los dos algoritmos. Karatsuba muestra un compotamiento logarítmico, que cuadra con su eficiencia teórica. Se obtienen picos en las mediciones a pesar de haber realizado medias. Estos son debidos a la dedicación del procesador a otras tareas. No obstante, son picos locales, que no afectan a la hora de ver la clara tendencia.

A partir de cierto momento, Karatsuba siempre será mejor.

Extras

De forma adicional a lo pedido en la práctica se han implementado las operaciones:

- División
- Mod

Manual de instalación y uso

Prerrequisitos:

- Tener instalado java jre7: http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html
- Descargar IDE eclipse Java: http://www.eclipse.org/downloads/moreinfo/java.php
- Asegurarse de que el path de donde toma el jre es correcto.

Datos de entrada:

- Los números con los que se opera deben estar contenidos en el fichero cuyo path es: **/IntsArithmetic/IntArithmetic/data.txt**
- Los dos números con los que se desea operar deben estar separados por un enter. ('\n').

Datos de salida:

- Se suministrarán por consola (integrada en eclipse) y en un log cuyo path es: **/IntsArithmetic/IntArithmetic/Resultados.txt**

Proyecto: puede abrirse como un proyecto convencional. Adicionalmente puede descargarse desde el repositorio directamente.
https://github.com/juanhg/IntArithmetic

Ejecución

- Cuando se tengan los datos de entrada correctamente ubicados, bastará con colocarse encima del fichero Main.class, pulsar botón derecho/Run As/ Java Application.
- En ese momento en consola aparecerá un menú con la siguiente estructura:

1 - Suma.
2 - Resta.
3 - Multiplicacion Escolar.
4 - Multiplicacion Karatsuba.
5 - Multiplicacion Modular.
6 - Division
7 - Salir.
Elija una opción:

- Por consola podrá introducir un número y ejecutar una operación.
- Las opciones son autoexplicativas.
- Los resultados deben mostrarse por consola y almacenarse en el log de salida.
- Cuando desee terminar la ejecución, pulse la opción número 6.