**Práctica1: Aritmética Entera**

**Profesor: Francisco Javier Lobillo Borrero**

**Asignatura: Cálculo Simbólico**

**Alumnos: Francisco Santolalla Quiñonero**

**Francisco Javier Gómez del Olmo**

Contenido

[1. Estructura de datos utilizada. 3](#_Toc371674325)

[Estructura de datos. 3](#_Toc371674326)

[Base y Formato de Entrada de los números. 3](#_Toc371674327)

[Motivación de usar la Base 2^16. 3](#_Toc371674328)

[2. Explicación de cada algoritmo. 4](#_Toc371674329)

[2.1 Multiplicación Escuela 4](#_Toc371674330)

[Explicación de la eficiencia del algoritmo. 4](#_Toc371674331)

[Tiempos aumentando el número de dígitos de ambos operandos. 4](#_Toc371674332)

[Tiempos aumentando el número de dígitos de un solo operando. 5](#_Toc371674333)

[2.2 Multiplicación Karatsuba 6](#_Toc371674334)

[Explicación de la eficiencia del algoritmo. 6](#_Toc371674335)

[Tiempos aumentando el número de dígitos de ambos operandos. 6](#_Toc371674336)

[Tiempos aumentando el número de dígitos de un solo operando. 7](#_Toc371674337)

[2.3 Multiplicación Modular 8](#_Toc371674338)

[Explicación de la eficiencia del algoritmo. 8](#_Toc371674339)

[Tiempos aumentando el número de dígitos de ambos operandos. 8](#_Toc371674340)

[Tiempos aumentando el número de dígitos de un solo operando. 9](#_Toc371674341)

[3. Comparación de los algoritmos. 10](#_Toc371674342)

[3.1 Ambos operandos tienen el mismo número de dígitos. 10](#_Toc371674343)

[Explicación de la eficiencia 10](#_Toc371674344)

[Tabla: aumentando el número de dígitos de ambos operandos. 10](#_Toc371674345)

[3.2 Un operando grande y otro pequeño. 11](#_Toc371674346)

[Explicación de la eficiencia 11](#_Toc371674347)

[Tabla: aumentando el número de dígitos de un solo operando. 11](#_Toc371674348)

[4. Manual de usuario. 12](#_Toc371674349)

# 1. Estructura de datos utilizada.

## Estructura de datos.

La estructura de datos que usamos para operar con los números es la siguiente:

* Base 2^16.

Es decir cada digito de un número tendrá un valor comprendido entre [0-65.535].

* Estructura de datos de un número de ‘n’ dígitos:

Ese número será guardado en un vector de ‘n’ posiciones, cada dígito del número ocupará una posición del vector.

La posición 0 del vector tendrá el dígito menos significativo.

La posición n-1 del vector tendrá el dígito más significativo.

## Base y Formato de Entrada de los números.

Los números se introducirán en base hexadecimal, es decir los números estarán formados por estos dígitos: [0-9,a,b,c,d,e,f].

Una vez introducimos los números en base hexadecimal serán transformados a base 2^16 para operar con ellos.

## Motivación de usar la Base 2^16.

La Microarquitectura de la CPU típica de los ordenadores (y la de nuestro ordenador) es de 32 bits, es decir, trabaja con unidades de datos de 32 bits.

Debido a esto, decidimos operar con una base que tenga la mitad de este tamaño para así aprovechar más el tamaño de los registros, y ganar rapidez al realizar las operaciones.

# 2. Explicación de cada algoritmo.

## 2.1 Multiplicación Escuela

### Explicación de la eficiencia del algoritmo.

El algoritmo de la multiplicación escuela tiene una complejidad de O(n\*m) siendo ‘n’ y ‘m’ el número de dígitos de los operandos. Por lo que en el peor de los casos es de orden cuadrático, se da cuando ambos números tienen el mismo número de dígitos. Y en el mejor de los casos es de orden lineal, cuando uno de los números solo tiene un dígito.

### Tiempos aumentando el número de dígitos de ambos operandos.

|  |  |  |
| --- | --- | --- |
| **Número de Dígitos (oper1)** | **Número de Dígitos (oper2)** | **Escuela (ms)** |
| 1 | 1 | 0.0003 |
| 10 | 10 | 0.0257 |
| 32 | 32 | 0.2893 |
| 64 | 64 | 1.053 |
| 128 | 128 | 4.296 |
| 256 | 256 | 16.796 |
| 512 | 512 | 64.06 |
| 1024 | 1024 | 273.1 |
| 2048 | 2048 | 1073.833 |
| 4096 | 4096 | 4334.8 |

## 

### Tiempos aumentando el número de dígitos de un solo operando.

|  |  |  |
| --- | --- | --- |
| **Número de Dígitos (oper1)** | **Número de Dígitos (oper2)** | **Escuela (ms)** |
| 1 | 1 | 0.0003 |
| 10 | 1 | 0.003 |
| 32 | 1 | 0.0173 |
| 64 | 1 | 0.0370 |
| 128 | 1 | 0.044 |
| 256 | 1 | 0.0796 |
| 512 | 1 | 0.159 |
| 1024 | 1 | 0.297 |
| 2048 | 1 | 0.5916 |
| 4096 | 1 | 1.167 |

## 2.2 Multiplicación Karatsuba

### Explicación de la eficiencia del algoritmo.

El algoritmo de la multiplicación karatsuba está basado en la técnica divide y vencerás. Tiene una complejidad de O(n^1’39) siendo ‘n’ el número de dígitos del operando de mayor tamaño. Este es el principal inconveniente del algoritmo, si un operando es mucho mayor que el otro la eficiencia del algoritmo se verá marcada por el número de dígitos del operando mayor, independientemente de que el operando menor sea un número muy pequeño. La ventaja que tiene es que cuando el número de dígitos de ambos operandos es aproximadamente el mismo y se aproxima por debajo a algún valor potencia de 2, la eficiencia del algoritmo mejora bastante.

### Tiempos aumentando el número de dígitos de ambos operandos.

|  |  |  |
| --- | --- | --- |
| **Número de Dígitos (oper1)** | **Número de Dígitos (oper2)** | **Karatsuba (ms)** |
| 1 | 1 | 0.0370 |
| 10 | 10 | 0.3023 |
| 32 | 32 | 0.9409 |
| 64 | 64 | 3.245 |
| 128 | 128 | 9.645 |
| 256 | 256 | 28.369 |
| 512 | 512 | 84.090 |
| 1024 | 1024 | 257.13 |
| 2048 | 2048 | 784.380 |
| 4096 | 4096 | 2392.3 |

### Tiempos aumentando el número de dígitos de un solo operando.

|  |  |  |
| --- | --- | --- |
| **Número de Dígitos (oper1)** | **Número de Dígitos (oper2)** | **Karatsuba (ms)** |
| 1 | 1 | 0.0370 |
| 10 | 1 | 0.306 |
| 32 | 1 | 0.9703 |
| 64 | 1 | 2.953 |
| 128 | 1, | 9.1483 |
| 256 | 1 | 27.116 |
| 512 | 1 | 82.32 |
| 1024 | 1 | 254.766 |
| 2048 | 1 | 762.533 |
| 4096 | 1 | 2281.6 |

## 2.3 Multiplicación Modular

### Explicación de la eficiencia del algoritmo.

El algoritmo de la multiplicación modular debe cambiar los números a base modular, multiplicarlos modularmente y devolverlos a las base original 2^16. La multiplicación de dos números en base modular es muy eficiente, es de orden lineal. Pasar los números de su base original a base modular también tampoco lleva demasiado tiempo. El problema de la ineficiencia de este algoritmo reside en volver a pasar el número de base modular a su base original.

Los primos los escogemos a partir del primer primo pi<2^16 para así no tener que usar tantos primos y garantizarnos que tengamos primos suficientes para poder multiplicar los dos números.

### Tiempos aumentando el número de dígitos de ambos operandos.

|  |  |  |
| --- | --- | --- |
| **Número de Dígitos (oper1)** | **Número de Dígitos (oper2)** | **Modular (ms)** |
| 1 | 1 | 0.0666 |
| 10 | 10 | 1.0666 |
| 32 | 32 | 20.8 |
| 64 | 64 | 138.566 |
| 128 | 128 | 1014.1 |
| 256 | 256 | 7804.0 |
| 512 | 512 | 60243.0 |
| 1024 | 1024 | 471513.0 |

### Tiempos aumentando el número de dígitos de un solo operando.

|  |  |  |
| --- | --- | --- |
| **Número de Dígitos (oper1)** | **Número de Dígitos (oper2)** | **Modular (ms)** |
| 1 | 1 | 0.0666 |
| 10 | 1 | 0.26 |
| 32 | 1 | 4.34 |
| 64 | 1 | 20.04 |
| 128 | 1 | 140.2 |
| 256 | 1 | 1046.1 |
| 512 | 1 | 7846.8 |
| 1024 | 1 | 61048.0 |
| 2048 | 1 | 474914 |

### 

# 3. Comparación de los algoritmos.

## 3.1 Ambos operandos tienen el mismo número de dígitos.

### Explicación de la eficiencia

La multiplicación escuela empieza teniendo los mejores tiempos cuando los números tienen menos de mil dígitos.

La multiplicación Karatsuba tiene la peor eficiencia en todos los casos, la ineficiencia de este algoritmo reside en tener que pasar el número de base modular a su base original.

A partir de mil dígitos puede verse como karatsuba empieza a tener mejores tiempos. Esto es debido a varias razones:

* Karatsuba está teniendo la eficiencia del mejor de sus casos O(n^1’39). Ya que el número de dígitos de ambos operandos es el mismo y su número de dígitos es potencia de dos.
* Multiplicación escuela está teniendo la eficiencia del peor de sus casos O(n^2). Esto sucede cuando ambos números tienen el mismo número de dígitos.

### Tabla: aumentando el número de dígitos de ambos operandos.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Número de Dígitos (oper1)** | **Número de Dígitos (oper2)** | **Escuela (ms)** | **Karatsuba (ms)** | **Modular (ms)** |
| 1 | 1 | 0.0003 | 0.0370 | 0.0666 |
| 10 | 10 | 0.0257 | 0.3023 | 1.0666 |
| 32 | 32 | 0.2893 | 0.9409 | 20.8 |
| 64 | 64 | 1.053 | 3.245 | 138.566 |
| 128 | 128 | 4.296 | 9.645 | 1014.1 |
| 256 | 256 | 16.796 | 28.369 | 7804.0 |
| 512 | 512 | 64.06 | 84.090 | 60243.0 |
| 1024 | 1024 | 273.1 | 257.13 | 471513 |
| 2048 | 2048 | 1073.833 | 784.380 |  |
| 4096 | 4096 | 4334.8 | 2392.3 |  |

## 3.2 Un operando grande y otro pequeño.

### Explicación de la eficiencia

En estos casos, en los que un operando es un número muy grande y el otro un número muy pequeño, la multiplicación escuela tiene la eficiencia del mejor de sus casos. Además como el operando pequeño es de 1 dígito la eficiencia que tiene es de orden lineal O(n).

La multiplicación Karatsuba está en el peor de sus casos. Su complejidad es de O(n^1’39) siendo ‘n’ el número de dígitos del operando de mayor tamaño. Este es el principal inconveniente del algoritmo, si un operando es mucho mayor que el otro la eficiencia del algoritmo se verá marcada por el número de dígitos del operando mayor, independientemente de que el operando menor sea un número muy pequeño.

Aún que llevemos al algoritmo de Karatsuba al peor de sus casos, la multiplicación modular sigue siendo la más ineficiente de todas, debido a el coste de volver a pasar el número de base modular a su base original.

### Tabla: aumentando el número de dígitos de un solo operando.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Número de Dígitos (oper1)** | **Número de Dígitos (oper2)** | **Escuela (ms)** | **Karatsuba (ms)** | **Modular (ms)** |
| 1 | 1 | 0.0003 | 0.0370 | 0.0666 |
| 10 | 1 | 0.003 | 0.306 | 0.26 |
| 32 | 1 | 0.0173 | 0.9703 | 4.34 |
| 64 | 1 | 0.0370 | 2.953 | 20.04 |
| 128 | 1 | 0.044 | 9.1483 | 140.2 |
| 256 | 1 | 0.0796 | 27.116 | 1046.1 |
| 512 | 1 | 0.159 | 82.32 | 7846.8 |
| 1024 | 1 | 0.297 | 254.766 | 61048.0 |
| 2048 | 1 | 0.5916 | 762.533 | 474914 |
| 4096 | 1 | 1.167 | 2281.6 |  |

Ç

# 4. Manual de usuario.

Haremos una breve descripción de cómo se ejecuta el programa para poder hacer las multiplicaciones con los distintos algoritmos.

1. El programa nos preguntará como queremos introducir los números, recordemos que los números se introducirán en base hexadecimal, es decir los números estarán formados por estos dígitos: [0-9,a,b,c,d,e,f]:

*Desea introducir los numeros:*

*1. A mano*

*2. Aleatoriamente*

1. Si elegimos aleatoriamente tendremos que decir el número de dígitos que va a tener cada número:

*Cuantos digitos quiere que tenga el primer numero:*

*200*

*Cuantos digitos quiere que tenga el segundo numero:*

*200*

1. Nos preguntará que algoritmo de multiplicación deseamos usar:

*Diga que multiplicacion quiere usar:*

*1. Escuela*

*2. Karatsuba*

*3. Modular*

*4. Las tres anteriores para comparar los tiempos*

1. Mostrará los resultados del algoritmo que elijamos, y el tiempo empleado:

*Ejemplo caso 4:*

*Resultado de multiplicar los dos numeros ESCUELA: 2880 7161 64560 55013*

*Tiempo empleado en la multiplicacion escuela: 0 ms*

*Resultado de multiplicar los dos numeros KARATSUBA: 2880 7161 64560 55013*

*Tiempo empleado en la multiplicacion Karatsuba: 0 ms*

*Resultado de la multiplicacion modular: 2880 7161 64560 55013*

*Tiempo empleado en la multiplicacion modular: 0 ms*

1. Finalmente nos preguntará si deseamos hacer más multiplicaciones o terminar:

*¿Desea hacer mas multiplicaciones?*

*1. Si*

*2. No*