INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

ANÁLISIS DE ALGORITMOS

PROFESORA: LUZ MARÍA SÁNCHEZ GARCÍA

INTEGRANTES:

VÁZQUEZ MORENO MARCOS OSWALDO 2016601777

DE LOS SANTOS DÍAZ LUIS ALEJANDRO 2017630451

PRÁCTICA 6 ANÁLISIS DE ALGORITMOS DIVIDE Y VENCERÁS MULTIPLICACIÓN ENTERA

3CM2

9 DE ABRIL DE 2019

Introducción

Multiplicación de enteros grandes

• Coste de realizar las operaciones elementales de suma y multiplicación:

– Es razonable considerarlo constante si los operandos son directamente manipulables por el hardware, es decir, no son muy grandes.

– Si se necesitan enteros muy grandes, hay que implementar por software las operaciones.

• Enteros muy grandes:

– Representación de un entero de n cifras: puede hacerse en un vector con un espacio en O(n) bits.

– Operaciones de suma, resta: pueden hacerse en tiempo lineal.

– Operaciones de multiplicación y división entera por potencias positivas de 10: pueden hacerse en tiempo lineal (desplazamientos de las cifras).

– Operación de multiplicación con el algoritmo clásico (o con el de multiplicación rusa): tiempo cuadrático. (Campos, 2019)

Planteamiento del problema

Sean *u* y *v dos números naturales n* bits donde, por simplicidad, *n* es una potencia de 2. El algoritmo tradicional para multiplicarlos es de complejidad **O(n^2).**

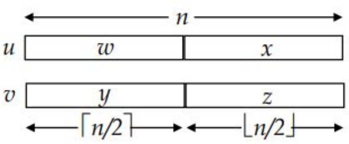
Implementar el algoritmo para lograr reducir una multiplicación de 4 cifras a 4 multiplicaciones de 2 cifras, más tres sumas y varios desplazamientos.

Diseño de la solución

Según el algoritmo de Karatsuba con la técnica de DyV para este problema dice que se descomponen en mitades de igual tamaño:

$$\left.\begin{array}{r} u = 10^s w + x \\ v = 10^s y + z \end{array}\right\} \text{ con } 0 \le x < 10^s,\ \ 0 \le z < 10^s, \text{ y } \quad s = \lfloor n/2 \rfloor$$

– Por tanto, w e y tienen $\lceil n/2 \rceil$ cifras



– El producto es:

$$uv = 10^{2s} wy + 10^{s}(wz + xy) + xz$$

Coste temporal:

– Las sumas, multiplicaciones por potencias de 10 y divisiones por potencias de 10: tiempo lineal.

– Operación módulo una potencia de 10: tiempo lineal (puede hacerse con una división, una multiplicación y una resta).

– Por tanto, si $t(n)$ es el tiempo del algoritmo:

$$t(n) = 3t\left(\lceil n/2 \rceil\right) + t\left(\lfloor n/2 \rfloor\right) + \Theta(n)$$

Si n se supone potencia de 2,

$$t(n) = 4t(n/2) + \Theta(n)$$

La solución de esta recurrencia es:

$$t(n) \in O(n^2)$$

– Teniendo en cuenta que $w+x$ e $y+z$ pueden necesitar $1+\lceil n/2 \rceil$ cifras,

$$t_2(n) \in t_2\left(\lfloor n/2 \rfloor\right) + t_2\left(\lceil n/2 \rceil\right) + t_2\left(1 + \lceil n/2 \rceil\right) + \Theta(n)$$

Y por tanto,

$$t_2(n) \in \Theta\left(n^{\log 3}\right) \in O\left(n^{1,59}\right)$$

– Debido a la constante multiplicativa, el algoritmo sólo es interesante en la práctica para n grande.

– Una buena implementación no debe usar la base 10 sino la base más grande posible que permita multiplicar dos “cifras” (de esa base) directamente en hardware.

A continuación, se muestran el diagrama de flujo de nuestra propuesta de solución para el método de multiplicación de números enteros grandes

Primeramente, se muestra en el diagrama 1.1 el multiplicacion.c de nuestra propuesta de solución.

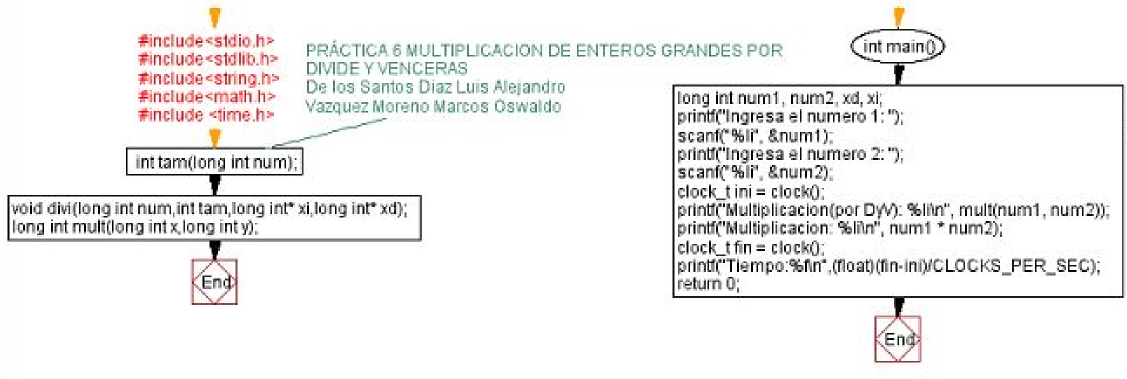


Diagrama 1.1 multiplicacion.c

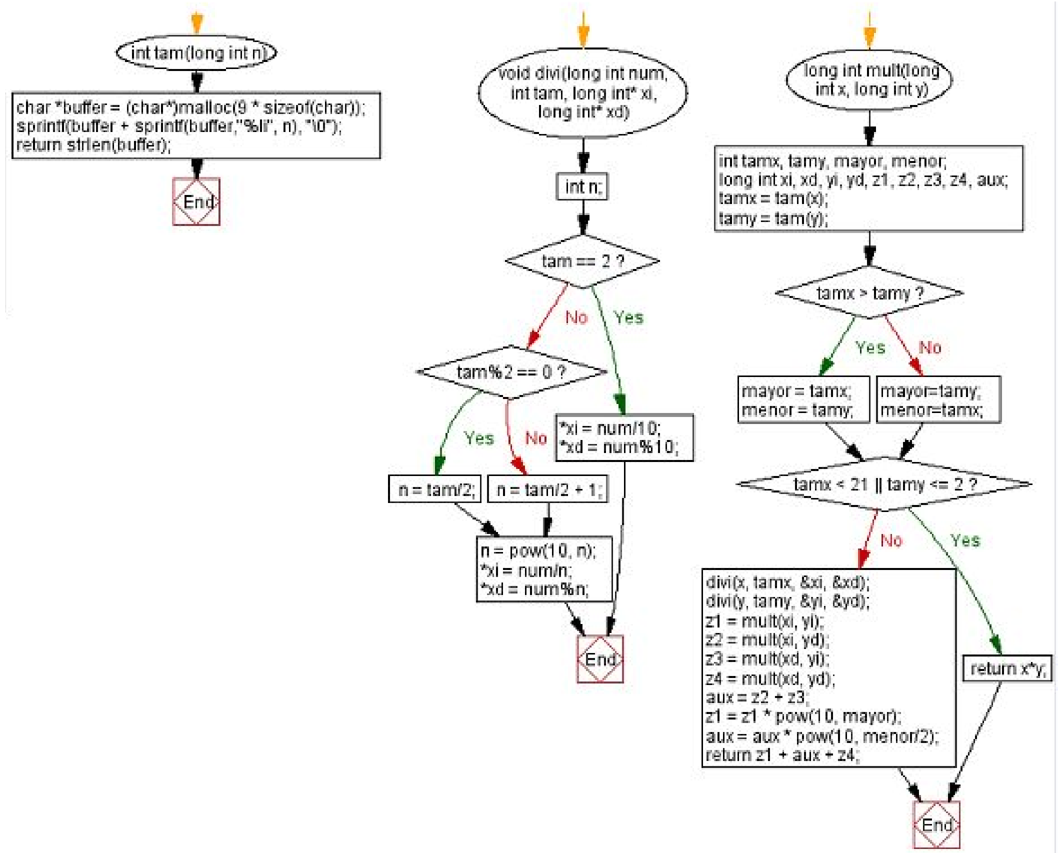


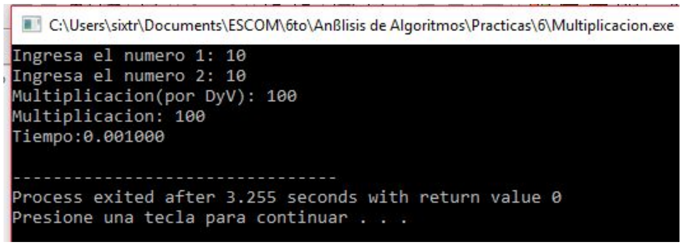
Diagrama 1.2 multiplicacion.c (continuación)

Implementación de la solución

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include <time.h>

/*
PRÁCTICA 6 MULTIPLICACION DE ENTEROS GRANDES POR
DIVIDE Y VENCERAS

De los Santos Diaz Luis Alejandro
Vazquez Moreno Marcos Oswaldo


*/

int tam(long int num);
void divi(long int num,int tam,long int* xi,long int* xd);
long int mult(long int x,long int y);

int main()
{
    long int num1, num2, xd, xi;
    printf("Ingresa el numero 1: ");
    scanf("%li", &num1);
    printf("Ingresa el numero 2: ");
    scanf("%li", &num2);
    clock_t ini = clock();
    printf("Multiplicacion(por DyV): %li\n", mult(num1, num2));
    printf("Multiplicacion: %li\n", num1 * num2);
    clock_t fin = clock();
    printf("Tiempo:%f\n",(float)(fin-ini)/CLOCKS_PER_SEC);
    return 0;
}

int tam(long int n)
{
    char *buffer = (char*)malloc(9 * sizeof(char));
    sprintf(buffer + sprintf(buffer,"%li", n), "\0");
    return strlen(buffer);
}

void divi(long int num, int tam, long int* xi, long int* xd)
{
    int n;
    if(tam == 2)
    {
        *xi = num/10;
        *xd = num%10;
```

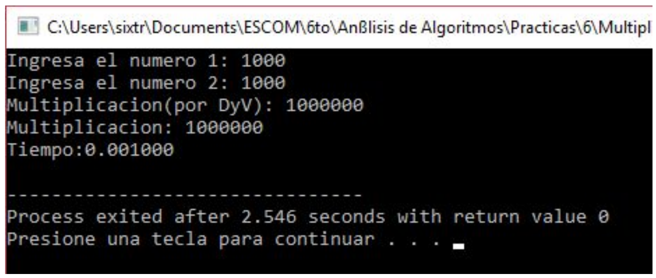
```
        return;
    }
    if(tam%2 == 0) n = tam/2;
    else n = tam/2 + 1;
    n = pow(10, n);
    *xi = num/n;
    *xd = num%n;
    return;
}

long int mult(long int x, long int y)
{
    int tamx, tamy, mayor, menor;
    long int xi, xd, yi, yd, z1, z2, z3, z4, aux;
    tamx = tam(x);
    tamy = tam(y);
    if(tamx > tamy)
    {
        mayor = tamx;
        menor = tamy;
    }
    else
    {
        mayor=tamy;
        menor=tamx;
    }
    if(tamx < 21 || tamy <= 2) return x*y;
    divi(x, tamx, &xi, &xd);
    divi(y, tamy, &yi, &yd);
    z1 = mult(xi, yi);
    z2 = mult(xi, yd);
    z3 = mult(xd, yi);
    z4 = mult(xd, yd);
    aux = z2 + z3;
    z1 = z1 * pow(10, mayor);
    aux = aux * pow(10, menor/2);
    return z1 + aux + z4;
}
```

Funcionamiento



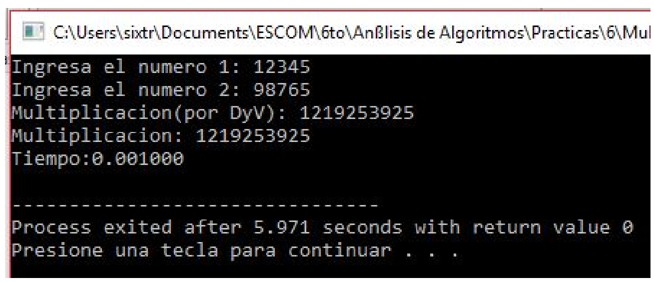
```
C:\Users\sixtr\Documents\ESCOM\6to\Anßlisis de Algoritmos\Practicas\6\Multiplicacion.exe
Ingresa el numero 1: 10
Ingresa el numero 2: 10
Multiplicacion(por DyV): 100
Multiplicacion: 100
Tiempo:0.001000

--------------------------------
Process exited after 3.255 seconds with return value 0
Presione una tecla para continuar . . .
```

Imagen 1.1



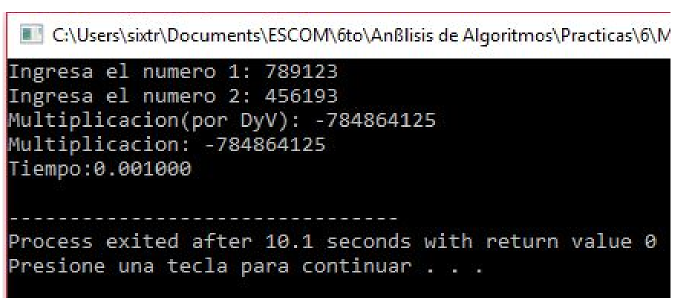
```
C:\Users\sixtr\Documents\ESCOM\6to\Anßlisis de Algoritmos\Practicas\6\Multipl
Ingresa el numero 1: 1000
Ingresa el numero 2: 1000
Multiplicacion(por DyV): 1000000
Multiplicacion: 1000000
Tiempo:0.001000

--------------------------------
Process exited after 2.546 seconds with return value 0
Presione una tecla para continuar . . .
```

Imagen 1.2



```
C:\Users\sixtr\Documents\ESCOM\6to\Anßlisis de Algoritmos\Practicas\6\Mul
Ingresa el numero 1: 12345
Ingresa el numero 2: 98765
Multiplicacion(por DyV): 1219253925
Multiplicacion: 1219253925
Tiempo:0.001000

--------------------------------
Process exited after 5.971 seconds with return value 0
Presione una tecla para continuar . . .
```

Imagen 1.3



```
C:\Users\sixtr\Documents\ESCOM\6to\Anßlisis de Algoritmos\Practicas\6\M
Ingresa el numero 1: 789123
Ingresa el numero 2: 456193
Multiplicacion(por DyV): -784864125
Multiplicacion: -784864125
Tiempo:0.001000

--------------------------------
Process exited after 10.1 seconds with return value 0
Presione una tecla para continuar . . .
```

Imagen 1.4

Plataforma experimental

La ejecución de los algoritmos anteriores se llevó a cabo en una computadora personal que se describe en la siguiente imagen.

Especificaciones del dispositivo

HP Laptop 15-bs0xx

Nombre del dispositivo	LAPTOP-13V7QO38
Procesador	Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 2.00 GHz
RAM instalada	8.00 GB
Id. del dispositivo	ACEA8FAF-1F85-49D4-BC5D-A7059ECE254D
Id. del producto	00327-30000-00000-AAOEM
Tipo de sistema	Sistema operativo de 64 bits, procesador x64
Lápiz y entrada táctil	La entrada táctil o manuscrita no está disponible para esta pantalla

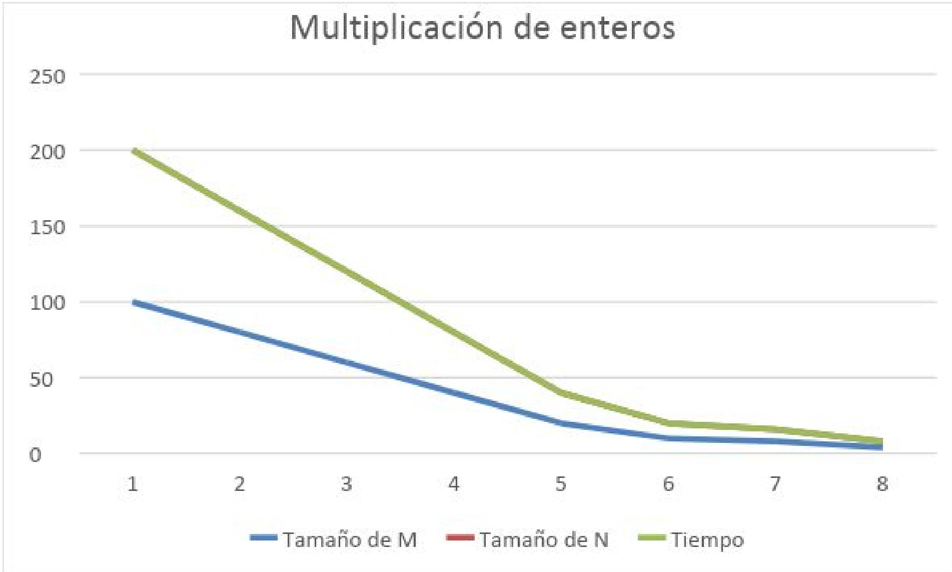
Imagen 1.5 Plataforma Experimental

El compilador utilizado fue gcc integrado dentro del IDE DevC en un sistema operativo de 64 bits Windows 10.

Gráfica del comportamiento temporal.

Tamaño de M	Tamaño de N	Tiempo
100	100	0.000077
80	80	0.000076
60	60	0.0000732
40	40	0.000025
20	20	0.000006
10	10	0.000003
8	8	0.000001
4	4	0.000001

Tabla 1.1



Gráfica 1.1

Cuestionario

1.- ¿Cuál de los 2 algoritmos es más útil en la práctica?

Se ha decidido en conjunto que el caso de la multiplicación de números grandes es más útil si tomamos en cuenta que no logramos programar la transformada rápida de Fourier por indicación de la profesora debido al corto tiempo y las vacaciones de semana santa.

2.-¿En qué aplicaciones se pueden implementar?

La transformada rápida de Fourier FFT es un algoritmo que reduce el tiempo de cálculo de n2 pasos a n·log2(n). El único requisito es que el número de puntos en la serie tiene que ser una potencia de 2 (2n puntos), por ejemplo 32, 1024, 4096, etc.

3.- ¿Considera que el algoritmo de Fourier es eficiente? ¿Por qué?

Si porque debido al tipo de complejidad en el algoritmo provoca un cambio brusco siendo la Serie Rápida de Fourier una función exponencial.

4.- Describa brevemente otros 3 algoritmos que se implementen con la técnica DyV.

1. *Máxima suma en cualquier subarreglo contiguo:*
   Nos dan un arreglo A[0,1,2,3….n-1] de enteros, y queremos encontrar la máxima suma en cualquier subarreglo contiguo,

mismo problema que se puede solucionar con la estrategia greedy para hallar la máxima subsuma.

2. *Amigos y regalos*:

   Tenemos dos números naturales C1 y C2 y dos primos P1 y P2 tal que p1<P2. Queremos hallar el mínimo valor de n perteneciente a los naturales tal que podamos particionar el conjunto 1{1, 2, 3… n} en dos conjuntos A y B que cumplan:
   - |A| >= C1
   - |B| >= C2
   - Ningún elemento de A es divisible entre P1.
   - Ningún elemento de B es divisible entre P2.
3. *Cúmulo:*

   Dados n puntos en el plano cartesiano XY, hay que hallar la mínima distancia entre cualquier par de puntos.

5.- ¿Qué tipo de problemas tuvo al realizar esta práctica?

Se tuvieron complicaciones en cuestión de tiempos, ya que el algoritmo es demasiado rápido en números sumamente pequeños, sin embargo con números ya más grandes comienza a notarse la diferencia, al principio creímos que lo estábamos implementando mal ya que no notamos diferencias de ningún tipo.

Conclusiones

En conclusión, podemos decir que lo que se vio en este problema es sin duda alguna el claro ejemplo de cómo el algoritmo de implementación llamado Divide y Vencerás o conocido en inglés como *Divide and Conquer* ya que el hecho de llevar a cabo más fácilmente la multiplicación de números grandes facilita al compilador, así como al lenguaje mismo que cuentan con precisión bastante alta volverlo más rápido, eficiente y capaz. Sin duda se han visto algunas otras formas y métodos, a veces cuando las multiplicaciones son más largas no se puede comprobar si son correctas las operaciones que se llevan a cabo.

Bibliografía

Bibliografía

Academy, K. (15 de 6 de 2006). *Algoritmo Divide y Vencerás .* Obtenido de https://es.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms

Campos, J. (07 de Abril de 2019). *Algoritmia básica.* Obtenido de Algoritmia básica (Universidad de Zaragoza): http://webdiis.unizar.es/asignaturas/AB/material/3-Divide%20y%20venceras.pdf

García, L. M. (07 de 04 de 2019). *Classroom.* Obtenido de Práctica 6 Ánalisis de Algoritmos: https://classroom.google.com/c/NzEzNTUxMDU1NVpa/a/NzgxNTE2NTA3MVpa/details