Análisis algorítmico: Divide y vencerás

Siro Sánchez del Amo

Febrero 2018

FACULTAD DE
INFORMÁTICA

UNIVERSIDAD DE
MURCIA

Índice

1. Introducción

Este documento consiste en un análisis algorítmico de la técnicas "Multiplicación directa", "Divide y vencerás" y "Karatsuba y Ofman". Toda teoría, conocimiento adquirido y algoritmos vienen dados en el libro de la asignatura de AED2 [1]

2. Algoritmo para "Enteros largos"

2.1. Definición del problema

El problema que se plantea es la multiplicación de números muy largos, lo cual a la larga puede dar problemas ya sea por llegar al límite o por lentitud de las operaciones, para poder realizar operaciones muy largas hemos usado listas y en cada posición de estas habrá un caracter, esto nos dará mas juego que usando simplemente enteros o doble precisión. Con esto en mano, procederemos al análisis de los algoritmos que hemos adaptado al problema.

2.2. Solución al problema

2.2.1. Multiplicación directa

La multiplicación directa fue programada con el siguiente código. Las funciones elementales se han omitido.

```
list<char> multiplicacionDirecta(list<char> EL1,list<char> EL2){
  list<char> resultado;
  list<char> auxiliar;
  int negativo=0;
  list<char>::iterator it1=EL1.begin();
  list<char>::iterator it2=EL2.begin();
  if((*it1)=='-') {
    EL1.erase(it1);
    negativo++;
  }
  if((*it2)=='-') {
    EL2.erase(it2);
    negativo++;
  }
  it2=EL2.end();
  it2--;
  int desplazamiento=1;
  resultado=multiplicacionSimple(EL1,(*it2));
  for(unsigned int i=0;i<EL2.size()-1;i++){
    it2--;
    auxiliar=multiplicacionSimple(EL1,(*it2));
    for(int j=0;j<desplazamiento;j++) auxiliar.insert(auxiliar.end(),'0');
    resultado=suma(resultado,auxiliar);
    desplazamiento++;
  }
  list<char> uno;
  uno.insert(uno.begin(),'1');
  if(negativo==1 && !menor(resultado,uno)) resultado.insert(resultado.begin(),'-');
  return resultado;
}
```

2.2.2. Divide y vencerás

Divide y vencerás fue programado con el siguiente código. Las funciones elementales se han omitido.

```
list<char> DYV(list<char> EL1,list<char> EL2,int n,int base){
  list<char> resultado;
  if(n==base) return multiplicacionDirecta(EL1,EL2);
  else{
    list<char> w = divisionSuperior(EL1);
    list<char> y = divisionSuperior(EL2);
    list<char> x = divisionInferior(EL1);
    list<char> z = divisionInferior(EL2);
    list<char> m1 = DYV(w,y,n/2, base);
    list<char> m2 = DYV(w,z,n/2, base);
    list<char> m3 = DYV(x,y,n/2, base);
    list<char> m4 = DYV(x,z,n/2, base);
    resultado=suma(desplazar(m1,n),suma(desplazar(suma(m2,m3),n/2),m4));
    return resultado;
  }
}
```

2.2.3. Karatsuba y Ofman

Karatsuba fue programado con el siguiente código. Las funciones elementales se han omitido.

```
list<char> KYO(list<char> EL1, list<char> EL2, int n, int base){
  list<char> resultado;
  if (n==base) return multiplicacionDirecta(EL1,EL2);
  else{
    list<char> w = divisionSuperior(EL1);
    list<char> y = divisionSuperior(EL2);
    list<char> x = divisionInferior(EL1);
    list<char> z = divisionInferior(EL2);
    list<char> m1 = KYO(w,y,n/2, base);
    list<char> m2 = KYO(x,z,n/2, base);
    list<char> m3 = KYO(restar(w,x),restar(z,y),n/2,base);
    resultado=suma(desplazar(m1,n),suma(desplazar(suma(suma(m3,m2),m1),n/2),m2));
    return resultado;
  };
}
```

3. Análisis de los algoritmos

3.1. Multiplicación directa

Calculamos la cota inferior:

$$t_m(n) = 14 + 4 + 5n[MultSimple] + \sum_{i=0}^{n-1}(2 + 4 + 5n[MultSimple] + 12 + 4n[Suma] + \sum_{j=0}^{m}(2))$$

$$t_m(n) = 18 + 5n + 18n - 18 + 9n^2 - 9n + 2mn - 2m$$
$$t_m(n) = 9n^2 - 14n + 2mn - 2m$$
$$t_m(n) \in \Omega(n^2)$$

Calculamos cota superior:

$$t_M(n) = 5 + 6 + 3 + 7 + 5n[MultSimple] + \sum_{i=0}^{n-1}(2 + 7n + 5[MultSimple] + \sum_{j=0}^{m}(2) + 17 + 8n[Suma] + 1) + 4$$

$$t_M(n) = 25 + 5n - 2n - 2 + 7n^2 7n + 5n - 5 + 2mn - 2m + 17n - 17 + 8n^2 - 8n + n - 1$$
$$t_M(n) = 15n^2 + 15n + 2mn - 2m$$
$$t_M(n) \in O(n^2)$$

Lo que al final deriva en:

$$\left.\begin{array}{l} t_M(n) \in O(n^2) \\ t_m(n) \in \Omega(n^2) \end{array}\right\} t_p(n) \in \theta(n^2).$$

En la siguiente tabla se pueden ver los distintos experimentos realizados para este algoritmo:

Dígitos	T.experimental	Memoria
1	∼0.00025s	Less than 1KB
2	∼0.00031s	Less than 1KB
4	∼0.00038s	Less than 1KB
8	∼0.00049s	Less than 1KB
16	∼0.00125s	Less than 1KB
32	∼0.00450s	Less than 1KB
64	∼0.01426s	Less than 1KB
128	∼0.05532s	Less than 1KB
256	∼0.14827s	∼33KB
512	∼0.36122s	∼66KB
1024	∼1.22063s	∼132KB

3.2. Divide y vencerás

La función `DYV` es una función recursiva, así pues se calculará de manera distinta:

$$t(n) = \begin{cases} t(n) = n^2 & n = base \\ t(n) = 8n + 4t(\frac{n}{2}) & n > base \end{cases}$$

$$n = 2^k \rightarrow t(2^k) - 4t(2^{k-1}) = 0$$

$$\frac{x^{2^k} - 4x^{2^{k-1}}}{x^{2^{k-1}}} = x - 4$$

$$(x-4) = 8 \times 2^k$$

$$(x-4) = (x-2)$$

$$c_1 \times 2^k + c_2 \times 4^k = 0$$

$$c_1 \times 2^{\log n} + c_2 \times 4^{\log n}$$

$$c_1 \times n^{log2} + c_2 \times n^{log4}$$

$$c_1 \times n + c_2 \times n^2$$

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

En la siguiente tabla se pueden ver los distintos experimentos realizados para este algoritmo:

Dígitos	T.experimental	Memoria
1	∼0.00002s	Less than 1KB
2	∼0.00005s	Less than 1KB
4	∼0.00012s	Less than 1KB
8	∼0.00051s	Less than 1KB
16	∼0.00481s	Less than 1KB
32	∼0.01741s	Less than 1KB
64	∼0.02497s	Less than 1KB
128	∼0.05821s	Less than 1KB
256	∼0.08921s	∼79KB
512	∼0.16236s	∼104KB
1024	∼1.18317s	∼200KB

3.3. Karatsuba y Ofman

La función `KYO` es una función recursiva, así pues se calculará de manera distinta:

$$t(n) = \begin{cases} t(n) = n^2 & n = base \\ t(n) = 10n + 3t(\frac{n}{2}) & n > base \end{cases}$$

$$n = 2^k \rightarrow t(2^k) - 3t(2^{k-1}) = 0$$

$$\frac{x^{2^k} - 3x^{2^{k-1}}}{x^{2^{k-1}}} = x - 3$$

$$(x-3) = 10 \times 2^k$$

$$(x-3) = (x-2)$$

$$c_1 \times 2^k + c_2 \times 3^k = 0$$

$$c_1 \times 2^{\log n} + c_2 \times 3^{\log n}$$

$$c_1 \times n^{log2} + c_2 \times n^{log3}$$

$$c_1 \times n + c_2 \times n^{1,59}$$

$$t(n) \in O(n^{1,59})$$

En la siguiente tabla se pueden ver los distintos experimentos realizados para este algoritmo:

Dígitos	T.experimental	Memoria
1	∼0.00002s	Less than 1KB
2	∼0.00012s	Less than 1KB
4	∼0.00049s	Less than 1KB
8	∼0.00165s	Less than 1KB
16	∼0.00422s	Less than 1KB
32	∼0.00826s	Less than 1KB
64	∼0.01724s	Less than 1KB
128	∼0.02942s	Less than 1KB
256	∼0.08425s	Less than 10KB
512	∼0.09029s	Less than 20KB
1024	∼1.09802s	Less than 30KB

4. Conclusiones experimentales

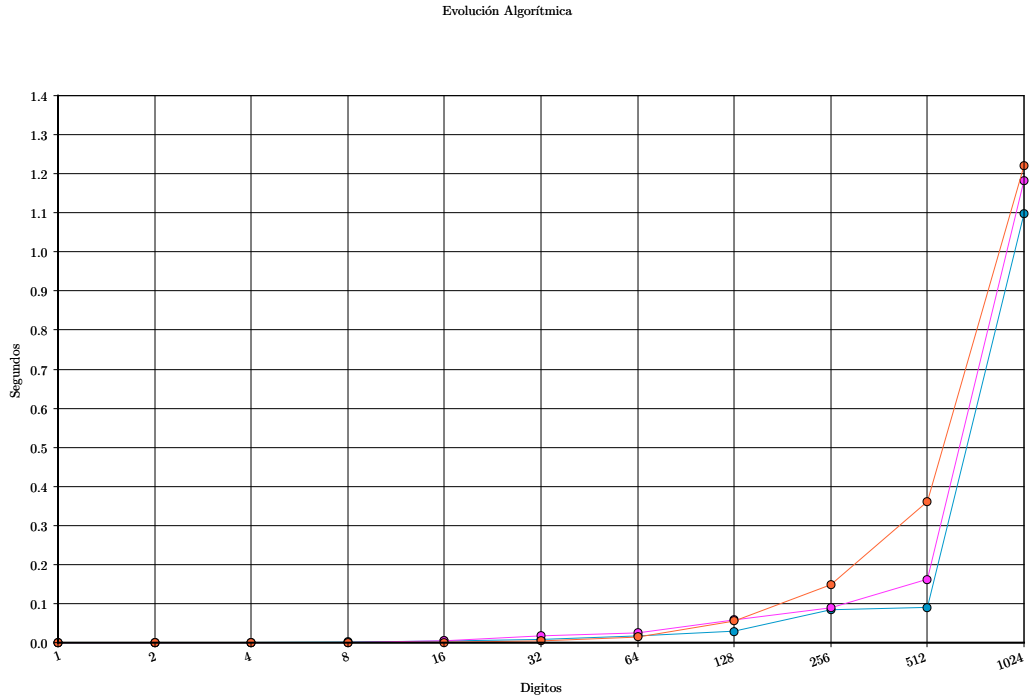

Figura 1: Gráfica de los tiempos de ejecución

Como conclusiones podemos sacar varias:

- El algoritmo preferido es sin duda `Karatsuba`. Al principio no se diferencia mucho de `Multiplicación directa` y `Divide y vencerás` pero conforme van aumentando la cantidad de números a multiplicar se nota.
- En cuanto a la partición base de los algoritmos, cuanto menos base, más recursividad tendrá que realizar, eso quiere decir más tiempo. Entonces si sólo realizáramos una única recursividad y las particiones por método directo, conseguiremos tiempos muy rápidos.

Referencias

[1] Garcia Mateos Gines, Cervera Lopez Joaquin, Marin Perez Norberto, and Gimenez Canovas Domingo. *Algoritmos y estructuras de datos volumen 2*. Diego Marin Librero Editor SL, 1st edition, 2003.