Multiplicación Rusa para números grandes

Sergio Salinas, Danilo Abellá

Universidad Santiago de Chile

23 de noviembre, 2017

Introduction

- Sistema de multiplicación escolar es el más habitual a nivel mundial desde que se extendió la numeración arábiga (sistema decimal).
- Existen otros métodos para multiplicar, ya sea de forma más eficiente o accesible.
- Desde hace muchos siglos las matemáticas han sido un punto fuerte en países como Rusia.

Acerca de la multiplicación

- Operación de composición que requiere sumar reiteradamente un número (multiplicando) de acuerdo a la cantidad de veces indicada por otro(multiplicador).
- Factores: Multiplicando y Multiplicador.
- Producto: Resultado.
- Propiedades: conmutativa, asociativa, elemento neutro y distributiva.

$$\underset{\text{multiplicando}}{2} \times \underset{\text{multiplicador}}{4} = 4 \times 2 = \underset{\text{producto}}{8}$$

Multiplicación rusa

- Método de multiplicación basado en la forma en que multiplicaban los campesinos rusos en el siglo XIX.
- También llamada multiplicación binaria debido a que su lógica esta basada la forma binaria de los números.
- Basado en duplicar y reducir números a la mitad.
- El método se puede expresar mediante las dos siguientes formulas.
  Si n es par

$$n \cdot m = \frac{n}{2} \cdot 2m$$

  Si n es impar

$$n \cdot m = \frac{n-1}{2} \cdot 2m + m$$

Ejemplo

n	m	Sumar	n	m	Sumar
121	35	35	1111001	100011	100011
60	70		111100	1000110	
30	140		11110	10001100	
15	280	280	1111	100011000	100011000
7	560	560	111	1000110000	1000110000
3	1120	1120	11	10001100000	10001100000
1	2240	2240	1	100011000000	100011000000
		4235			100001001011

Tabla 1: Multiplicación de $121 \cdot 35$ con su paralelo en forma binaria

El pseucódigo

Algorithm 1: Multiplicación rusa

Data: Dos enteros positivos a y b

Result: El resultado del producto de a y b.

```
1 begin
2     res ← 0
3     while a > 0 do
4         if a is impar then
5             res ← res + b
6         a ← ⌊a/2⌋
7         b ← 2 · b
8     return res
```

¿Por qué funciona? Demostración

Se debe a propiedad distributiva de la suma

$$\begin{aligned} 12 \cdot 9 &= 12 \cdot (1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3) && (1) \\ &= (12 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 12 \cdot 2^3) && (2) \\ &= 12 + 96 && (3) \\ &= 108. && (4) \end{aligned}$$

Implementación

- Implementación en C.
- Números guardados en cadena de caracteres, donde cada elemento es un dígito.
- Uso de memoria dinámica para optimizar memoria.

Data Type	Range	Bytes	Format
signed char	-128 to + 127	1	%c
unsigned char	0 to 255	1	%c
short signed int	-32768 to +32767	2	%d
short unsigned int	0 to 65535	2	%u
signed int	-32768 to +32767	2	%d
unsigned int	0 to 65535	2	%u
long signed int	-2147483648 to +2147483647	4	%ld
long unsigned int	0 to 4294967295	4	%lu
float	-3.4e38 to +3.4e38	4	%f
double	-1.7e308 to +1.7e308	8	%lf
long double	-1.7e4932 to +1.7e4932	10	%Lf
Note: The sizes and ranges of int, short and long are compiler dependent. Sizes in this figure are for 16-bit compiler.			

Algoritmos utilizados

Para trabajar con números grandes se trabajo dígito a dígito con cada número, por se requieren otras operaciones de suma, multiplicación y división.

- Para sumar se uso una modificación de la multiplicación larga.
- Para multiplicar se uso una modificación de la multiplicación larga.
- Para Dividir se uso una modificación de la división larga.
- Para verificar si un número es impar solo se comprueba si su último dígito es un múltiplo de dos.

Suma clásica

- Es la suma a papel y lápiz que se enseña a los niños en el colegio.
- Suma dos dígitos n y m con n $\geq$ m, el número menor lo rellena por la izquierda con ceros hasta igualar en dígitos al mayor.
- Realiza n operaciones, su complejidad es $\mathcal{O}(n)$ con n la cantidad de dígitos del número con más dígitos.

Suma clásica pseudocódigo

Algorithm 2: Suma clásica

Input: Dos números A[0..la] y B[0..la]
Output: La suma C de A más B

```
1  begin
2      carry ← 0;
3      C[0..la + 1] ← 0;
4      B[0..(la − lb)] ← 0 // Rellena con zeros el inicio de B;
5      while la > 0 do
6          sum ← A[la − 1] + B[la − 1] + carry;
7          carry ← sum/10;
8          C[la] = sum mod 10;
9          la = la − 1;
10     if carry > 0 then
11         C[0] ← 1
12     return C;
```

Multiplicación larga modificada

Complejidad: $\mathcal{O}(n)$
n $\rightarrow$ cantidad de dígitos que tiene el número a multiplicar por dos.
Esto es debido a que en la iteración se hacen tantas operaciones de coste constante como dígitos tenga el número n.

$$
\begin{array}{rr}
 & \downarrow \\
 & 1 \longrightarrow \textit{carry} \\
+ & \\
 & 150 \\
x & 2 \\
\hline
 & 00
\end{array}
$$

$$
\begin{array}{rr}
 & 150 \\
x & 2 \\
\hline
 & 300
\end{array}
$$

Multiplicación larga modificada pseudocódigo

Algorithm 3: Multiplicación por dos

Input: Un número a[0..la]

Output: Un número C con el resultado de multiplciar a por dos

```
1  begin
2      c ← [0, 0, ..,0]
3      k ←i + 1
4      for i ← (la − 1).,0 do
5          n ← a[i] ∗ 2 + c[k]
6          carry ← n div 10
7          c[k] ← n mod 10
8          k ←k - 1
9          c[k] ← c[k] sum carry
10     if c[0] = 0 then
11         Se elimina el primer digito de 'c'
12     return C
```

División larga modificada

- Divide dígito por dígito por dos.
- Realiza n operaciones, su complejidad es $\mathcal{O}(n)$ con n la cantidad de dígitos del número a dividir.

División larga modificada pseudocódigo

Algorithm 4: División por dos

Input: Un número A[0..la]

Output: La división de A por 2.

```
1 begin
2     carry ← 0
3     C[0..la + 1] ← 0
4     for i ← 0..la do
5         n ← (carry · 10 + a[i])/2
6         carry ← (carry · 10 + a[i]) − 2 · n
7         c[i] ← n
8     return C
```

Complejidad asintótica de la multiplicación rusa

- La cantidad de operaciones que hace el algoritmo en cada iteración viene dada por la cantidad de dígitos en binario del número a ser divido por dos, esto se debe a que a que por cada división se quita un bit. Por lo que se repiten $\log_2(n)$ operaciones.
- La suma, la multiplicación por dos y la división por dos tienen coste $\mathcal{O}(n)$.
- Estas operaciones de repiten $\log_2(n)$, con n el multiplicando.

$$\begin{aligned} \log_2(n) \cdot (\mathcal{O}(n) + \mathcal{O}(n) + \mathcal{O}(n)) &= \log_2(n) \cdot (\mathcal{O}(3n)) \\ &= \log_2(n) \cdot (\mathcal{O}(n)) \\ &= \log_2(n) \cdot \mathcal{O}(n) \\ &= \mathcal{O}(\log_2(n) \cdot n) \\ &= \mathcal{O}(n \cdot \log(n)) \end{aligned}$$

- Por lo que la complejidad la multiplicación rusa es $\mathcal{O}(n \cdot \log(n))$.

Experimentos

Para los experimento se consideran los números por cantidad de bits en vez de su largo de digitos en decimal.

- Experimento 1, verifica el tiempo de multiplicar dos números de igual cantidad de bits.
- Experimento 2, Dado dos números de distinta cantidad de bits verifica el coste de multiplicar el menor por el mayor y del mayor por el menor.

Experimento 1

Multiplicación de dos números $n \cdot m$ de igual cantidad de bits.

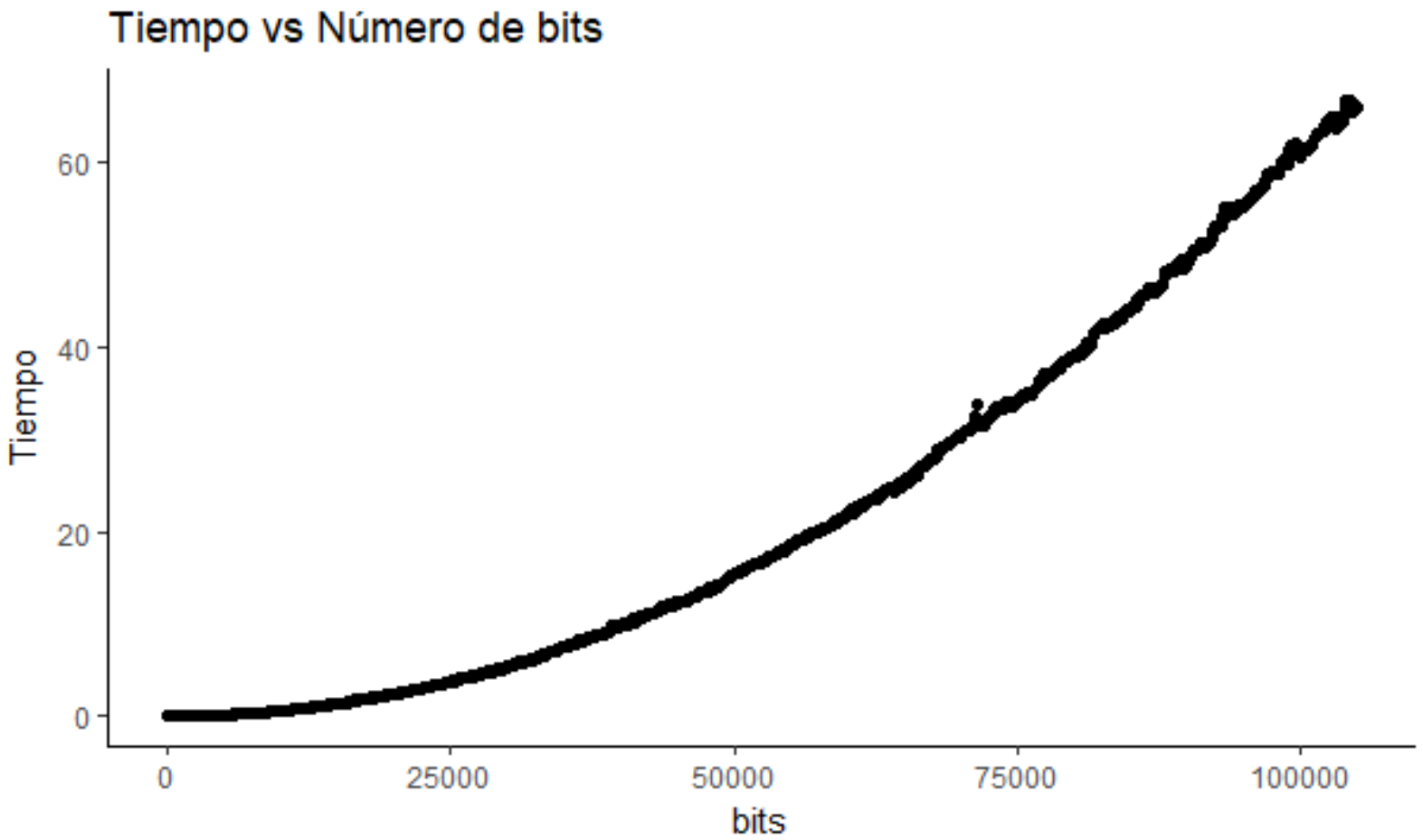

Figura 2: Multiplicación de dos números n y m con la misma cantidad de bits

Experimento 2.a

Multiplicación de dos números $n \cdot m$ con m $>$ n, con n de 2^{100}-1 y m variante, se aumentan 100 bits por punto.

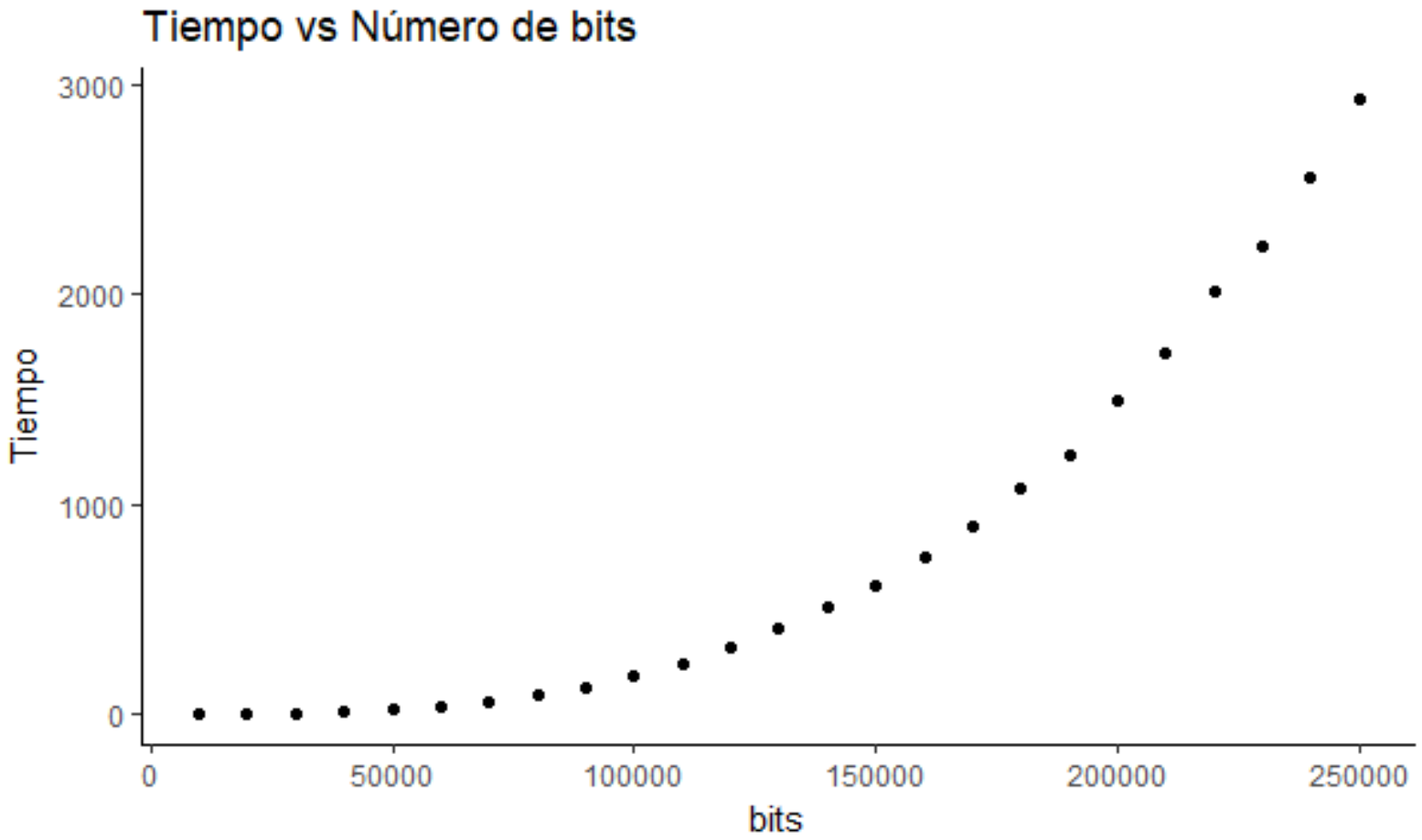

Figura 3: Multiplicación de dos números n·m con n $>$ m, n es fijo y m variante

Experimento 2.b

Multiplicación de dos números $n \cdot m$ con m $>$ n, con n variante y m de $2^{100} - 1$ bits, se aumentan 10000 bits por puntos, se aumentan 10000 bits por puntos.

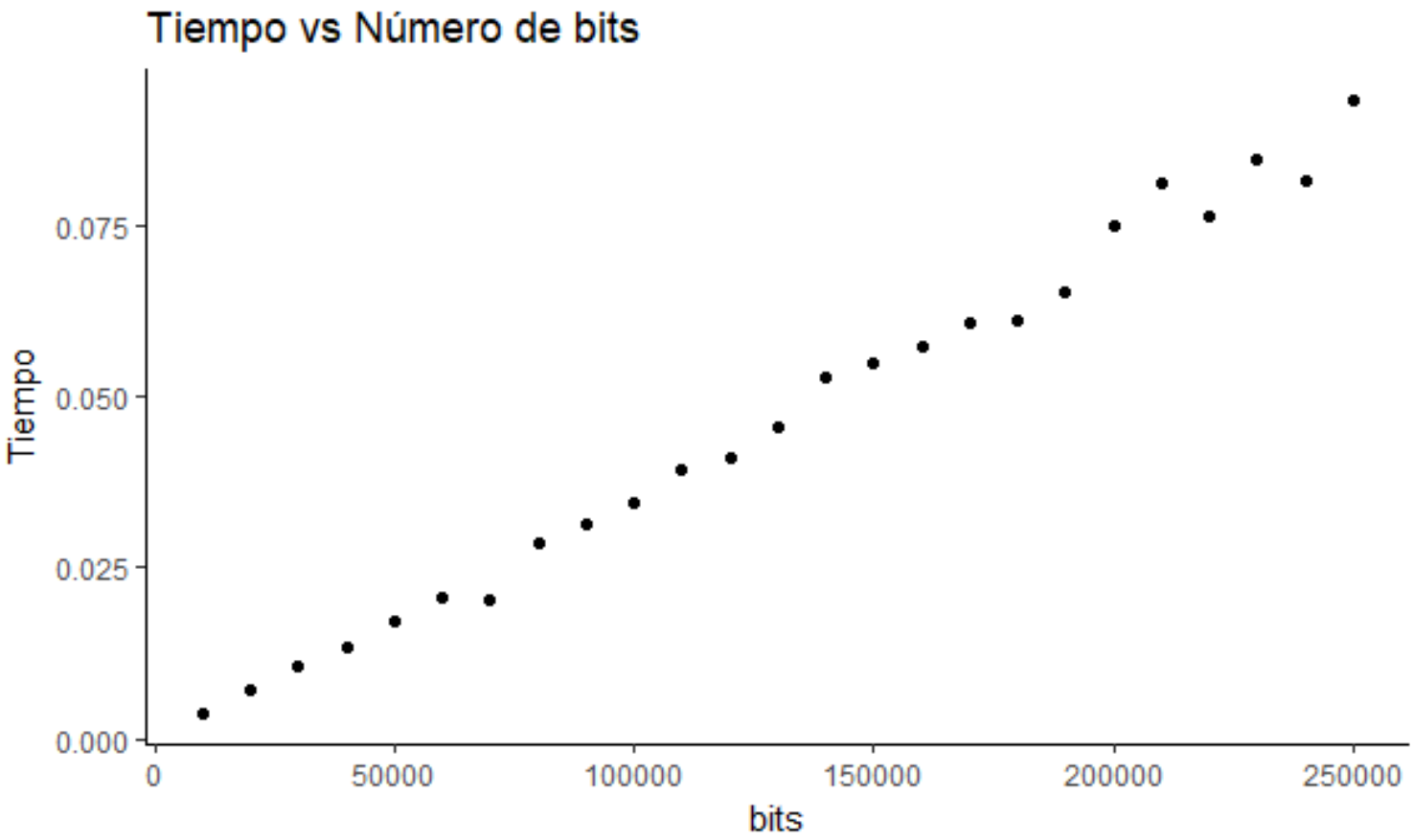

Figura 4: Multiplicación de dos números dos números n·m con n $>$ m, n variante

Análisis de resultados

1. Curva real aproximada a la curva esperada por su complejidad de $\mathcal{O}(n \cdot \log(n))$.
2. Tiene mucho menor costo dividir por 2 el número con menor cantidad de bits ya que hace menos cantidad de iteraciones.

Conclusión

- Algoritmo simple, fácil de utilizar en papel y lápiz.
- Ineficiente en coste para multiplicar números grandes.