Multiplicación rápida: Algoritmo de Karatsuba

Un método eficiente para multiplicar grandes números

Seminario I — Gabriela Archila

Universidad del Valle de Guatemala

16 de noviembre de 2024

Introducción a la Multiplicación en Computación

- Importancia de la multiplicación rápida en aplicaciones computacionales.
  - Reducción del tiempo de ejecución.
  - Optimización de Algoritmos Criptográficos.
  - Procesamiento de Señales y Audio.
- Método tradicional de multiplicación (multiplicación de lápiz y papel).
  - Complejidad de $O(n^2)$.
- Necesidad de métodos más rápidos para grandes enteros.

Introducción al Algoritmo de Karatsuba

- **Algoritmos rápidos:** Minimizar operaciones de bits.
- **Sistema binario:** Operaciones básicas en bits (0, 1), suma, resta, multiplicación.
- Andrei Kolmogorov definió que la complejidad de la multiplicación se mide en función del número de operaciones de bits necesarias para calcular el producto de dos números de n dígitos.
- **Conjetura de Kolmogorov (1956):**
  - Complejidad de la multiplicación $M(n) = O(n^2)$.
  - Creencia: no es posible multiplicar más rápido que $O(n^2)$.
- **Descubrimiento de Karatsuba (1960):**
  - Algoritmo de multiplicación con $M(n) = O(n^{\log_2(3)}) \approx O(n^{1.58})$.
  - Refuta la conjetura de $O(n^2)$.

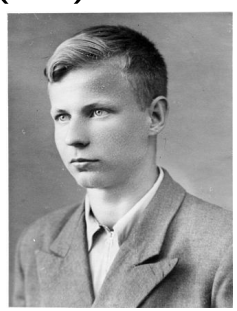

Figura: Anatolii Karatsuba

Anatolii Karatsuba

- **Nombre Completo**: Anatolii Alexeevitch Karatsuba.
- **Nacimiento**: 31 de enero de 1937 en Moscú.
- **Contribución Principal**: Desarrolló el Algoritmo de Karatsuba en 1960, un método para la multiplicación rápida de enteros grandes utilizando la técnica de *divide and conquer*.
- **Legado**: El método de Karatsuba fue un precursor de otros algoritmos rápidos en matemáticas y computación, inspirando técnicas como la Transformada Rápida de Fourier y el algoritmo de Schönhage-Strassen.

Figura: Anatolii Karatsuba

Concepto Clave - *divide and conquer*

- Divide los números en partes para reducir el número de multiplicaciones.
- Estrategia recursiva para descomponer y recomponer el problema.
- Base para algoritmos rápidos como FFT y Schönhage-Strassen.

Divide and conquer

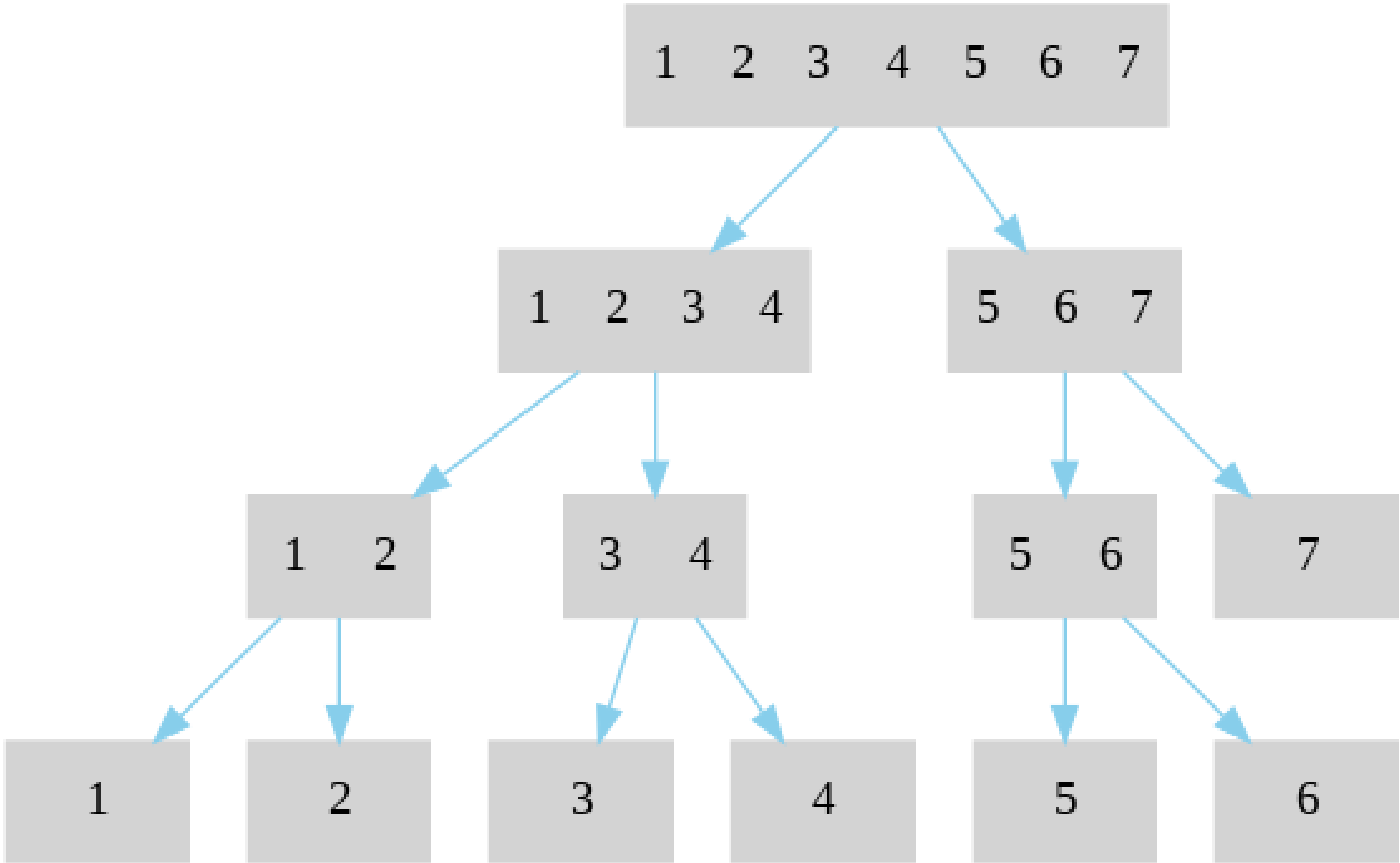

Figura: *Divide and conquer*

Teorema Karatsuba

Teorema

La multiplicación de enteros de n dígitos puede realizarse con $O(n^{\log_2(3)})$.

Demostración. (Esquema)

- Dado X y Y números de n dígitos:
  - Divide X en $X_1 \cdot 10^{n/2} + X_2$ y Y en $Y_1 \cdot 10^{n/2} + Y_2$.
- Calcula los productos recursivos:
  - $U = X_1 \cdot Y_1$.
  - $V = X_2 \cdot Y_2$.
  - $W = (X_1 + X_2)(Y_1 + Y_2) - U - V$.
- Usando U, V, y W:
  - $P = 10^n \cdot U + 10^{n/2} \cdot W + V$

Teorema Karatsuba

- La recurrencia que describe el costo total del algoritmo es:

$$M(n) = 3M(n/2) + O(n)$$

Donde:

- $3M(n/2)$ proviene de las tres llamadas recursivas para calcular U, V, y W.
- $O(n)$ representa el costo de combinar los resultados U, V, y W para calcular P.

Antes de avanzar es importante conocer lo que establece el siguiente teorema, ya que ayudara a obtener la complejidad del algoritmo.

Teorema Karatsuba (teorema maestro)

Teorema (teorema maestro)

El teorema maestro se aplica a relaciones de recurrencia de la forma:

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

donde:

- $a \geq 1$ *es el número de subproblemas en los que se divide el problema,*
- $b > 1$ *es el factor por el cual se reduce el tamaño del problema en cada subproblema,*
- $f(n)$ *es una función que describe el costo del trabajo adicional realizado fuera de las llamadas recursivas.*

Teorema Karatsuba (teorema maestro)

El teorema maestro establece tres casos para determinar la solución asintótica $T(n)$:

1. **Caso 1**: Si $f(n) = O(n^{\log_b a - \epsilon})$ para alguna constante $\epsilon > 0$, entonces:

$$T(n) = \Theta(n^{\log_b a})$$

2. **Caso 2**: Si $f(n) = \Theta(n^{\log_b a})$, entonces:

$$T(n) = \Theta(n^{\log_b a} \log n)$$

3. **Caso 3**: Si $f(n) = \Omega(n^{\log_b a + \epsilon})$ para alguna constante $\epsilon > 0$ y si $af(n/b) \leq cf(n)$ para alguna constante $c < 1$ y suficientemente grande n, entonces:

$$T(n) = \Theta(f(n))$$

Teorema Karatsuba

- Dada la recurrencia:

$$M(n) = 3M\left(\frac{n}{2}\right) + O(n)$$

Comparémosla con la forma general del Teorema Maestro:

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

Aquí, identificamos:

  - $a = 3$.
  - $b = 2$.
  - $f(n) = O(n)$).

- Calculamos el exponente crítico:

$$p = \log_b a = \log_2 3$$

Donde $\log_2 3 \approx 1.585$.

Teorema Karatsuba

- Según el Caso 1 del Teorema Maestro:

$$n = O\left(n^{\log_2 3-\epsilon}\right) \quad \text{para cualquier } \epsilon > 0.$$

- Por lo tanto, se satisface la condición del Caso 1.
- Por lo tanto, la complejidad es $T(n) = O(n^{\log_2 3})$.

Resumen del Algoritmo

1. Sean X y Y representados como cadenas de n dígitos en alguna base B, donde $m < n$. Entonces,

$$X = x_1 B^m + x_0, \quad Y = y_1 B^m + y_0, \text{ donde } x_0, y_0 < B^m$$

2. Entonces, se tiene los productos parciales

$$P_1 = x_1 \cdot y_1 \text{ (Partes altas)}$$

$$P_2 = x_0 \cdot y_0 \text{ (Partes bajas)}$$

$$P_3 = (x_1 + x_0)(y_1 + y_0) - P_1 - P2 \text{ (Cruzado)}$$

3. Por lo tanto, para $P = XY$:

$$P = P_1 \cdot B^{2m} + P_3 \cdot B^m + P_2,$$

Ejemplo del Algoritmo de Karatsuba

Ejemplo simple: 12 y 34

Sea $12 = 1 \cdot 10^1 + 2$ y $34 = 3 \cdot 10^1 + 4$. Entonces,

$$X1 = 1, \quad X0 = 2, \quad Y1 = 3, \quad Y0 = 4.$$

Ahora, al calcular los productos parciales:

$$\begin{aligned} P1 &= X1 \cdot Y1 = 1 \cdot 3 = 3 \\ P2 &= X0 \cdot Y0 = 2 \cdot 4 = 8 \\ P3 &= (X1 + X0) \cdot (Y1 + Y0) - P1 - P2 \\ &= (1 + 2) \cdot (3 + 4) - 3 - 8 \\ &= 3 \cdot 7 - 3 - 8 = 21 - 3 - 8 = 10 \end{aligned}$$

Ejemplo del Algoritmo de Karatsuba

Combinando los resultados:

$$12 \times 34 = P1 \cdot 10^2 + P3 \cdot 10^1 + P2 = 3 \cdot 100 + 10 \cdot 10 + 8 = 408.$$

Ejemplo algoritmo Karatsuba

Ejemplo : 1234 y 5678

Sea $1234 = 12 \cdot 10^2 + 34$ y $5678 = 56 \cdot 10^2 + 78$. Entonces,

$$X1 = 12, \quad X0 = 34. \quad Y1 = 56, \quad Y0 = 78.$$

Ahora,

$$\begin{aligned} P1 &= X1 \cdot Y1 = 12 \times 56 \\ P2 &= X0 \cdot Y0 = 34 \times 78 \\ P3 &= (X1 + X0) \cdot (Y1 + Y0) - P1 - P2 \\ &\Rightarrow (12 + 34) \times (56 + 78) - P1 - P2 \end{aligned}$$

Expansión de la recursión en los productos parciales:
- $P1 = 12 \times 56$

Ejemplo algoritmo Karatsuba

- $12 = 1 \cdot 10^1 + 2$ y $56 = 5 \cdot 10^1 + 6$
- $P_1 1 = 1 \cdot 5 = 5, \quad P_1 2 = 2 \cdot 6 = 12$
- $P_1 3 = (1+2) \cdot (5+6) - 5 - 12 = 18$
- Combinar: $12 \times 56 = 5 \cdot 10^2 + 18 \cdot 10^1 + 12 = 672$

- Repetir el proceso para $P2 = 34 \times 78$ y $P3 = (46) \times (134)$. Por lo tanto,

$$1234 \times 5678 = 672 \cdot 10^4 + 2840 \cdot 10^2 + 2652$$

La recursividad en los productos parciales continúa hasta que los números sean lo suficientemente pequeños para ser multiplicados directamente.

Pseudocódigo del Algoritmo de Karatsuba

```
def karatsuba(x, y)
  if (x < 10) or (y < 10):
    return x * y
  n = max(tamaño(x), tamaño(y))
  m = n // 2
  (alto1, bajo1) = dividir(x, 10**m)
  (alto2, bajo2) = dividir(y, 10**m)
  P2 = karatsuba(bajo1, bajo2)
  P3 = karatsuba((bajo1 + alto1), (bajo2 + alto2))
  P1 = karatsuba(alto1, alto2)
  return (P1 * 10**(2 * m2)) + ((P3 - P1 - P2) * 10**m2) +
  P2
```

Comparación de la complejidad

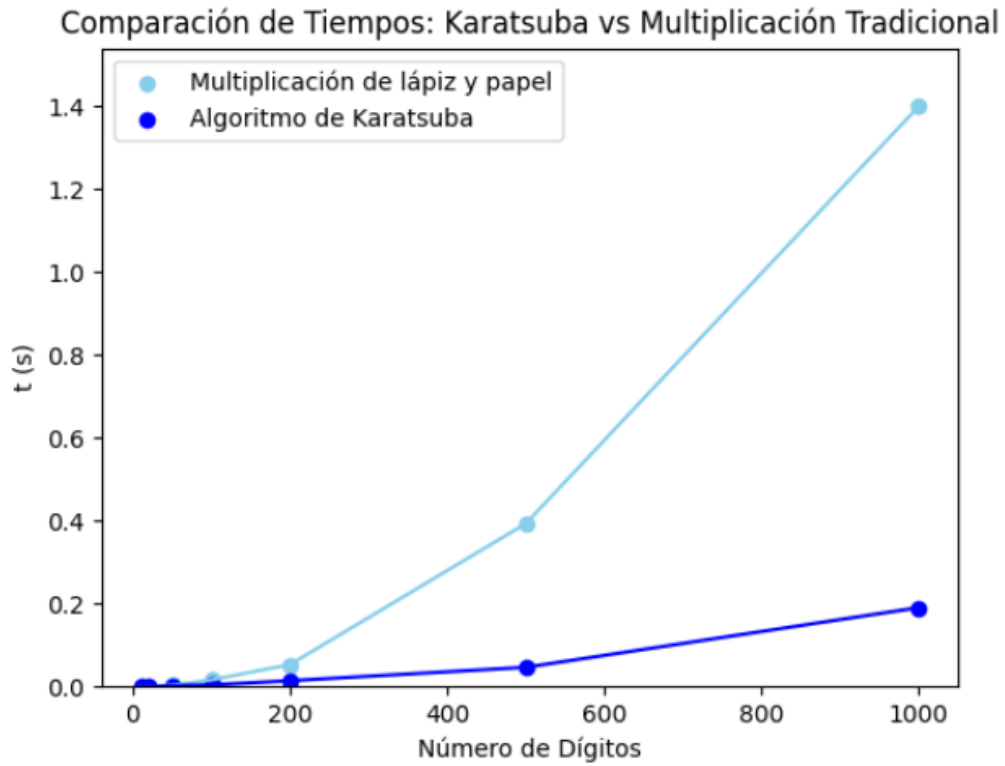

Figura: Comparación de tiempo

Recordar:

- Multiplicación tradicional: $M(n) = O(n^2)$
- Algoritmo Karatsuba: $M(n) = O(n^{\log_2(3)})$

Aplicaciones

- Procesamiento de Señales y Audio con Fast Fourier Transform (FFT).
- Criptografía.
- Álgebra Computacional.

Limitaciones

- Eficiencia para Números Pequeños.
- Uso de Memoria.
- Dependencia en la División de los Números.
  - Karatsuba funciona bien cuando los números pueden dividirse en partes iguales como base 2 o 10.
  - Limita su aplicabilidad en sistemas que operan en bases numéricas distintas.
- **Extra:** Existe como una generalización del algoritmo de Karatsuba llamado el algoritmoo de Toom-Cook, también conocido como Toom-3.

Muchas gracias