Análise de Algoritmos
Aula 10

Prof. Murilo V. G. da Silva

DINF/UFPR

(material da disciplina: André Guedes, Renato Carmo, Murilo da Silva)

Queremos multiplicar dois inteiros x e y de precisão arbitrária

Multiplicação Recursiva de Inteiros

Queremos multiplicar dois inteiros x e y de precisão arbitrária

- Inicialmente vamos apresentar a intuição do algoritmo sem muito formalismo

Multiplicação Recursiva de Inteiros

Queremos multiplicar dois inteiros x e y de precisão arbitrária

- Inicialmente vamos apresentar a intuição do algoritmo sem muito formalismo
- Para simplificar (e tornar clara a ideia central do algoritmo), suponha

Multiplicação Recursiva de Inteiros

Queremos multiplicar dois inteiros x e y de precisão arbitrária

- Inicialmente vamos apresentar a intuição do algoritmo sem muito formalismo
- Para simplificar (e tornar clara a ideia central do algoritmo), suponha
  - Os dois números tem n bits, cada bit guardado em um nó de uma lista
  - n é potência de 2

Multiplicação Recursiva de Inteiros

Queremos multiplicar dois inteiros x e y de precisão arbitrária

- Inicialmente vamos apresentar a intuição do algoritmo sem muito formalismo
- Para simplificar (e tornar clara a ideia central do algoritmo), suponha
  - Os dois números tem n bits, cada bit guardado em um nó de uma lista
  - n é potência de 2
- Depois lidamos com as condições de contorno...

Multiplicação Recursiva de Inteiros

Dado $x \in \mathbb{N}$, definimos

- x_{L}: os $\frac{n}{2}$ bits mais à esquerda de x em binário

  ou seja $x_n \ldots x_{\frac{n}{2}}$
- x_{R}: os $\frac{n}{2}$ bits mais à direita de x em binário

  ou seja, $x_{\frac{n}{2}+1} \ldots x_1$

Multiplicação Recursiva de Inteiros

A função *ShiftLeft*(z, k)

Dado z, a representação binária de um inteiro junto com um outro inteiro k

- A função *ShiftLeft*(z, k) retorna a representação:

$$z(\underbrace{0\ldots0}_{k \text{ bits}})$$

Note: *ShiftLeft*(z, k) multiplica o o número representado por z por 2^k

Note: O tempo de execução de *ShiftLeft*(z, k) é $\Theta(k)$

Teorema: Soma em tempo linear

Sejam x, y dois números inteiros tais que a representação binária de ambos os números tem n bits. Em particular, a listas que guardam ambos x e y tem n nós. Então a soma $x + y$ pode ser computada em tempo $\Theta(n)$.

Multiplicação Recursiva de Inteiros

Com isso em mente, observe que:

$$x = 2^{n/2} x_{\mathrm{L}} + x_{\mathrm{R}}, \qquad (1)$$

$$y = 2^{n/2} y_{\mathrm{L}} + y_{\mathrm{R}} \qquad (2)$$

Com isso em mente, observe que:

$$
\begin{align}
x &= 2^{n/2} x_{\mathrm{L}} + x_{\mathrm{R}}, \tag{1}\\
y &= 2^{n/2} y_{\mathrm{L}} + y_{\mathrm{R}} \tag{2}
\end{align}
$$

Usando (1), qual é o valor de xy?

Multiplicação Recursiva de Inteiros

Com isso em mente, observe que:

$$
\begin{aligned}
x &= 2^{n/2} x_{\mathrm{L}} + x_{\mathrm{R}}, \qquad (1) \\
y &= 2^{n/2} y_{\mathrm{L}} + y_{\mathrm{R}} \qquad (2)
\end{aligned}
$$

Usando (1), qual é o valor de xy? Veja abaixo

Multiplicação Recursiva de Inteiros

Com isso em mente, observe que:

$$x = 2^{n/2} x_{\mathrm{L}} + x_{\mathrm{R}}, \tag{1}$$
$$y = 2^{n/2} y_{\mathrm{L}} + y_{\mathrm{R}} \tag{2}$$

Usando (1), qual é o valor de xy? Veja abaixo

$$xy = (2^{n/2} x_{\mathrm{L}} + x_{\mathrm{R}})(2^{n/2} y_{\mathrm{L}} + y_{\mathrm{R}}), \tag{3}$$
$$= 2^{n} x_{\mathrm{L}} y_{\mathrm{L}} + 2^{n/2}(x_{\mathrm{L}} y_{\mathrm{R}} + x_{\mathrm{R}} y_{\mathrm{L}}) + x_{\mathrm{R}} y_{\mathrm{R}} \tag{4}$$

Multiplicação Recursiva de Inteiros

$$xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Pela fórmula acima, para calcular xy, dependemos de quatro produtos:

Multiplicação Recursiva de Inteiros

$$xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Pela fórmula acima, para calcular xy, dependemos de quatro produtos:

- $x_L y_L$
- $x_L y_R$
- $x_R y_L$
- $x_R y_R$

Multiplicação Recursiva de Inteiros

$$xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Pela fórmula acima, para calcular xy, dependemos de quatro produtos:

- $x_L y_L$
- $x_L y_R$
- $x_R y_L$
- $x_R y_R$

Em seguida, combinamos isso somando três termos:

(a) $2^n x_L y_L$

(b) $2^{n/2}(x_L y_R + x_R y_L)$

(c) $x_R y_R$

Multiplicação Recursiva de Inteiros

$$xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Pela fórmula acima, para calcular xy, dependemos de quatro produtos:

- $x_L y_L$
- $x_L y_R$
- $x_R y_L$
- $x_R y_R$

Em seguida, combinamos isso somando três termos:

(a) $2^n x_L y_L$
(b) $2^{n/2}(x_L y_R + x_R y_L)$
(c) $x_R y_R$

Portanto o custo para computar xy se resume ao custo de:

Multiplicação Recursiva de Inteiros

$$xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Pela fórmula acima, para calcular xy, dependemos de quatro produtos:

- $x_L y_L$
- $x_L y_R$
- $x_R y_L$
- $x_R y_R$

Em seguida, combinamos isso somando três termos:

(a) $2^n x_L y_L$
(b) $2^{n/2}(x_L y_R + x_R y_L)$
(c) $x_R y_R$

Portanto o custo para computar xy se resume ao custo de:

- Quatro multiplicações $x_L y_L$, $x_L y_L$, $x_L y_L$, $x_L y_L$ (todas com números de $\frac{n}{2}$ bits)

Multiplicação Recursiva de Inteiros

$$xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Pela fórmula acima, para calcular xy, dependemos de quatro produtos:

- $x_L y_L$
- $x_L y_R$
- $x_R y_L$
- $x_R y_R$

Em seguida, combinamos isso somando três termos:

(a) $2^n x_L y_L$

(b) $2^{n/2}(x_L y_R + x_R y_L)$

(c) $x_R y_R$

Portanto o custo para computar xy se resume ao custo de:

- Quatro multiplicações $x_L y_L$, $x_L y_L$, $x_L y_L$, $x_L y_L$ (todas com números de $\frac{n}{2}$ bits)
- Computar (a) usando *ShiftLeft*$(x_L y_L, n)$

Multiplicação Recursiva de Inteiros

$$xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Pela fórmula acima, para calcular xy, dependemos de quatro produtos:

- $x_L y_L$
- $x_L y_R$
- $x_R y_L$
- $x_R y_R$

Em seguida, combinamos isso somando três termos:

(a) $2^n x_L y_L$

(b) $2^{n/2}(x_L y_R + x_R y_L)$

(c) $x_R y_R$

Portanto o custo para computar xy se resume ao custo de:

- Quatro multiplicações $x_L y_L$, $x_L y_L$, $x_L y_L$, $x_L y_L$ (todas com números de $\frac{n}{2}$ bits)
- Computar (a) usando *ShiftLeft*$(x_L y_L, n)$
- Computar (b) fazendo uma soma usando um *ShiftLeft* no resultado.

Multiplicação Recursiva de Inteiros

Observe que as ideias acima nos fornece diretamente o seguinte algoritmo de divisão e conquista para multiplicar x por y:

Algoritmo 1: *RecursiveMult*(x, y) /*x, y são inteiros de n bits */

Se $n = 1$
    $z = x \wedge y$
Senão
    $z_{\mathrm{LL}} = \mathrm{RecursiveMult}(x_{\mathrm{L}}, y_{\mathrm{L}})$
    $z_{\mathrm{LR}} = \mathrm{RecursiveMult}(x_{\mathrm{L}}, y_{\mathrm{L}})$
    $z_{\mathrm{RL}} = \mathrm{RecursiveMult}(x_{\mathrm{L}}, y_{\mathrm{L}})$
    $z_{\mathrm{RR}} = \mathrm{RecursiveMult}(x_{\mathrm{L}}, y_{\mathrm{L}})$
    $z = \mathrm{Reconstroi}(z_{\mathrm{LL}},\ z_{\mathrm{LR}},\ z_{\mathrm{RL}},\ z_{\mathrm{RR}})$
Devolva z

Multiplicação Recursiva de Inteiros

Observe que as ideias acima nos fornece diretamente o seguinte algoritmo de divisão e conquista para multiplicar x por y:

Algoritmo 3: *RecursiveMult*(x, y) /*x, y são inteiros de n bits */

Se $n = 1$
  $z = x \wedge y$
Senão
  $z_{\mathrm{LL}} = \mathrm{RecursiveMult}(x_{\mathrm{L}}, y_{\mathrm{L}})$
  $z_{\mathrm{LR}} = \mathrm{RecursiveMult}(x_{\mathrm{L}}, y_{\mathrm{L}})$
  $z_{\mathrm{RL}} = \mathrm{RecursiveMult}(x_{\mathrm{L}}, y_{\mathrm{L}})$
  $z_{\mathrm{RR}} = \mathrm{RecursiveMult}(x_{\mathrm{L}}, y_{\mathrm{L}})$
  $z = \mathrm{Reconstroi}(z_{\mathrm{LL}},\, z_{\mathrm{LR}},\, z_{\mathrm{RL}},\, z_{\mathrm{RR}})$
Devolva z

Algoritmo 4: *Reconstroi*$(z_{\mathrm{LL}},\, z_{\mathrm{LR}},\, z_{\mathrm{RL}},\, z_{\mathrm{RR}})$

$a = \mathrm{ShiftLeft}(z_{LL}, n)$
$c = z_{LR} + z_{RL}$
$b = \mathrm{ShifLeft}(c,\ \frac{n}{2})$
$z = a + b + z_{RR}$
Devolva z

Multiplicação Recursiva de Inteiros

O custo total do Algoritmo de Multiplicação Recursiva é $T(n) = 4T(\frac{n}{2}) + \mathcal{O}(n)$.

Multiplicação Recursiva de Inteiros

O custo total do Algoritmo de Multiplicação Recursiva é $T(n) = 4T(\frac{n}{2}) + \mathcal{O}(n)$.

Aplicando o Teorema Mestre:

Multiplicação Recursiva de Inteiros

O custo total do Algoritmo de Multiplicação Recursiva é $T(n) = 4T(\frac{n}{2}) + \mathcal{O}(n)$.

Aplicando o Teorema Mestre:

"Teorema Mestre"

Sejam $a \geq 1,\ b > 1,\ T, f : \mathbb{N} \to \mathbb{R}$ tais que

$$T(n) = aT(n/b) + f(n),$$

então

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{se } f(n) = \mathcal{O}(n^{\log_b a - \epsilon}),\ \text{para algum } \epsilon > 0, \\ \Theta(n^{\log_b a} \log n), & \text{se } f(n) = \Theta(n^{\log_b a}), \\ \Theta(f(n)), & \text{se } f(n) = \Omega(n^{\log_b a + \epsilon}),\ \text{para algum } \epsilon > 0 \text{ e} \\ & af(n/b) < cf(n) \text{ assintoticamente para algum } c < 1. \end{cases}$$

Multiplicação Recursiva de Inteiros

O custo total do Algoritmo de Multiplicação Recursiva é $T(n) = 4T(\frac{n}{2}) + \mathcal{O}(n)$.

Aplicando o Teorema Mestre:

"Teorema Mestre"

Sejam $a \geq 1$, $b > 1$, $T, f : \mathbb{N} \to \mathbb{R}$ tais que

$$T(n) = aT(n/b) + f(n),$$

então

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{se } f(n) = \mathcal{O}(n^{\log_b a - \epsilon}), \text{ para algum } \epsilon > 0, \\ \Theta(n^{\log_b a} \log n), & \text{se } f(n) = \Theta(n^{\log_b a}), \\ \Theta(f(n)), & \text{se } f(n) = \Omega(n^{\log_b a + \epsilon}), \text{ para algum } \epsilon > 0 \text{ e} \\ & af(n/b) < cf(n) \text{ assintoticamente para algum } c < 1. \end{cases}$$

A complexidade de tempo é

Multiplicação Recursiva de Inteiros

O custo total do Algoritmo de Multiplicação Recursiva é $T(n) = 4T(\frac{n}{2}) + \mathcal{O}(n)$.

Aplicando o Teorema Mestre:

"Teorema Mestre"

Sejam $a \geq 1$, $b > 1$, $T, f : \mathbb{N} \to \mathbb{R}$ tais que

$$T(n) = aT(n/b) + f(n),$$

então

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{se } f(n) = \mathcal{O}(n^{\log_b a - \epsilon}), \text{ para algum } \epsilon > 0, \\ \Theta(n^{\log_b a} \log n), & \text{se } f(n) = \Theta(n^{\log_b a}), \\ \Theta(f(n)), & \text{se } f(n) = \Omega(n^{\log_b a + \epsilon}), \text{ para algum } \epsilon > 0 \text{ e} \\ & af(n/b) < cf(n) \text{ assintoticamente para algum } c < 1. \end{cases}$$

A complexidade de tempo é $\Theta(n^2)$

Multiplicação Recursiva de Inteiros

Reescrevendo xy:

$$
\begin{aligned}
xy &= 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R \\
&= 2^n x_L y_L + 2^{n/2}[(x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R)] + x_R y_R \qquad (5)
\end{aligned}
$$

Multiplicação Recursiva de Inteiros

Reescrevendo xy:

$$
\begin{aligned}
xy &= 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R \\
&= 2^n x_L y_L + 2^{n/2}[(x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R)] + x_R y_R \qquad (5)
\end{aligned}
$$

Qual é a vantagem da Equação (5)?

Reescrevendo xy:

$$\begin{aligned} xy &= 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R \\ &= 2^n x_L y_L + 2^{n/2}[(x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R)] + x_R y_R \end{aligned} \tag{5}$$

Qual é a vantagem da Equação (5)?
Precisamos computar apenas três produtos:

Multiplicação Recursiva de Inteiros

Reescrevendo xy:

$$\begin{aligned} xy &= 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R \\ &= 2^n x_L y_L + 2^{n/2}[(x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R)] + x_R y_R \end{aligned} \tag{5}$$

Qual é a vantagem da Equação (5)?
Precisamos computar apenas três produtos:

- $x_L y_L$

Multiplicação Recursiva de Inteiros

Reescrevendo xy:

$$\begin{aligned} xy &= 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R \\ &= 2^n x_L y_L + 2^{n/2}[(x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R)] + x_R y_R \qquad (5) \end{aligned}$$

Qual é a vantagem da Equação (5)?
Precisamos computar apenas três produtos:

- $x_L y_L$
- $(x_L + x_R)(y_L + y_R)$

Reescrevendo xy:

$$
\begin{aligned}
xy &= 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R \\
&= 2^n x_L y_L + 2^{n/2}[(x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R)] + x_R y_R \qquad (5)
\end{aligned}
$$

Qual é a vantagem da Equação (5)?
Precisamos computar apenas três produtos:

- $x_L y_L$
- $(x_L + x_R)(y_L + y_R)$
- $x_R y_R$

E depois de uma operação *R*econstroi que depende apenas de operações de:

- Soma
- Subtração
- *ShifLeft*

Multiplicação Recursiva de Inteiros

Algoritmo 5: *Karatsuba*(x, y) /*x, y são inteiros de n bits */

Se $n = 1$
    $z = x \wedge y$
Senão
    $a' = x_{\mathrm{L}} + x_{\mathrm{R}}$
    $a'' = y_{\mathrm{L}} + y_{\mathrm{R}}$
    $z_{\mathrm{LL}} = \mathrm{RecursiveMult}(x_{\mathrm{L}}, y_{\mathrm{L}})$
    $a = \mathrm{RecursiveMult}(a', a'')$
    $z_{\mathrm{RR}} = \mathrm{RecursiveMult}(x_{\mathrm{L}}, y_{\mathrm{L}})$

    $z = \mathrm{ReconstroiKaratsuba}(z_{\mathrm{LL}}, a, z_{\mathrm{RR}})$
Devolva z

Multiplicação Recursiva de Inteiros

Algoritmo 7: *Karatsuba*(x, y) /*x, y são inteiros de n bits */

Se $n = 1$
    $z = x \wedge y$
Senão
    $a' = x_{\mathrm{L}} + x_{\mathrm{R}}$
    $a'' = y_{\mathrm{L}} + y_{\mathrm{R}}$
    $z_{\mathrm{LL}} = \mathrm{RecursiveMult}(x_{\mathrm{L}}, y_{\mathrm{L}})$
    $a \quad = \mathrm{RecursiveMult}(a', a'')$
    $z_{\mathrm{RR}} = \mathrm{RecursiveMult}(x_{\mathrm{L}}, y_{\mathrm{L}})$

    $z = \mathrm{ReconstroiKaratsuba}(z_{\mathrm{LL}}, a, z_{\mathrm{RR}})$
Devolva z

Algoritmo 8: ReconstroiKaratsuba$(z_{\mathrm{LL}}, a, z_{\mathrm{RR}}))$

$b = a - z_{\mathrm{LL}} - z_{\mathrm{RR}}$
$z' = \mathrm{ShifLeft}(z_{\mathrm{LL}}, n)$
$z'' = \mathrm{ShifLeft}(b, \frac{n}{2})$
Devolva $z' + z'' + z_{\mathrm{RR}}$

O custo total do Algoritmo de Karatsuba é $T(n) = 3T(\frac{n}{2}) + \mathcal{O}(n)$.

Multiplicação Recursiva de Inteiros

O custo total do Algoritmo de Karatsuba é $T(n) = 3T(\frac{n}{2}) + \mathcal{O}(n)$.

Aplicando o Teorema Mestre:

Multiplicação Recursiva de Inteiros

O custo total do Algoritmo de Karatsuba é $T(n) = 3T(\frac{n}{2}) + \mathcal{O}(n)$.

Aplicando o Teorema Mestre:

"Teorema Mestre"

Sejam $a \geq 1$, $b > 1$, $T, f : \mathbb{N} \to \mathbb{R}$ tais que

$$T(n) = aT(n/b) + f(n),$$

então

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{se } f(n) = \mathcal{O}(n^{\log_b a - \epsilon}), \text{ para algum } \epsilon > 0, \\ \Theta(n^{\log_b a} \log n), & \text{se } f(n) = \Theta(n^{\log_b a}), \\ \Theta(f(n)), & \text{se } f(n) = \Omega(n^{\log_b a + \epsilon}), \text{ para algum } \epsilon > 0 \text{ e} \\ & af(n/b) < cf(n) \text{ assintoticamente para algum } c < 1. \end{cases}$$

Multiplicação Recursiva de Inteiros

O custo total do Algoritmo de Karatsuba é $T(n) = 3T(\frac{n}{2}) + \mathcal{O}(n)$.

Aplicando o Teorema Mestre:

"Teorema Mestre"

Sejam $a \geq 1,\ b > 1,\ T, f : \mathbb{N} \to \mathbb{R}$ tais que

$$T(n) = aT(n/b) + f(n),$$

então

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{se } f(n) = \mathcal{O}(n^{\log_b a - \epsilon}),\ \text{para algum } \epsilon > 0, \\ \Theta(n^{\log_b a} \log n), & \text{se } f(n) = \Theta(n^{\log_b a}), \\ \Theta(f(n)), & \text{se } f(n) = \Omega(n^{\log_b a + \epsilon}),\ \text{para algum } \epsilon > 0 \text{ e} \\ & af(n/b) < cf(n) \text{ assintoticamente para algum } c < 1. \end{cases}$$

A complexidade de tempo é

Multiplicação Recursiva de Inteiros

O custo total do Algoritmo de Karatsuba é $T(n) = 3T(\frac{n}{2}) + \mathcal{O}(n)$.

Aplicando o Teorema Mestre:

"Teorema Mestre"

Sejam $a \geq 1,\ b > 1,\ T, f : \mathbb{N} \to \mathbb{R}$ tais que

$$T(n) = aT(n/b) + f(n),$$

então

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{se } f(n) = \mathcal{O}(n^{\log_b a - \epsilon}),\ \text{para algum } \epsilon > 0, \\ \Theta(n^{\log_b a} \log n), & \text{se } f(n) = \Theta(n^{\log_b a}), \\ \Theta(f(n)), & \text{se } f(n) = \Omega(n^{\log_b a + \epsilon}),\ \text{para algum } \epsilon > 0 \text{ e} \\ & af(n/b) < cf(n) \text{ assintoticamente para algum } c < 1. \end{cases}$$

A complexidade de tempo é $T(n) = \Theta(n^{\lg 3})$.

Multiplicação Recursiva de Inteiros

O custo total do Algoritmo de Karatsuba é $T(n) = 3T(\frac{n}{2}) + \mathcal{O}(n)$.

Aplicando o Teorema Mestre:

"Teorema Mestre"

Sejam $a \geq 1,\ b > 1,\ T, f : \mathbb{N} \to \mathbb{R}$ tais que

$$T(n) = aT(n/b) + f(n),$$

então

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{se } f(n) = \mathcal{O}(n^{\log_b a - \epsilon}), \text{ para algum } \epsilon > 0, \\ \Theta(n^{\log_b a} \log n), & \text{se } f(n) = \Theta(n^{\log_b a}), \\ \Theta(f(n)), & \text{se } f(n) = \Omega(n^{\log_b a + \epsilon}), \text{ para algum } \epsilon > 0 \text{ e} \\ & af(n/b) < cf(n) \text{ assintoticamente para algum } c < 1. \end{cases}$$

A complexidade de tempo é $T(n) = \Theta(n^{\lg 3})$. Podemos dizer também que:

- $T(n) = \Omega(n^{1.58})$
- $T(n) = \mathcal{O}(n^{1.59})$

Multiplicação Recursiva de Inteiros

O custo total do Algoritmo de Karatsuba é $T(n) = 3T(\frac{n}{2}) + \mathcal{O}(n)$.

Aplicando o Teorema Mestre:

"Teorema Mestre"

Sejam $a \geq 1,\ b > 1,\ T, f : \mathbb{N} \to \mathbb{R}$ tais que

$$T(n) = aT(n/b) + f(n),$$

então

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{se } f(n) = \mathcal{O}(n^{\log_b a - \epsilon}),\ \text{para algum } \epsilon > 0, \\ \Theta(n^{\log_b a} \log n), & \text{se } f(n) = \Theta(n^{\log_b a}), \\ \Theta(f(n)), & \text{se } f(n) = \Omega(n^{\log_b a + \epsilon}),\ \text{para algum } \epsilon > 0 \text{ e} \\ & af(n/b) < cf(n) \text{ assintoticamente para algum } c < 1. \end{cases}$$

A complexidade de tempo é $T(n) = \Theta(n^{\lg 3})$. Podemos dizer também que:

- $T(n) = \Omega(n^{1.58})$
- $T(n) = \mathcal{O}(n^{1.59})$

pois $1.58 < \lg 3 < 1.59$

Tornando as coisas mais formais

Dada uma sequência $x = (x_{n-1}, \ldots, x_0) \in \{0,1\}^n$,

Tornando as coisas mais formais

Dada uma sequência $x = (x_{n-1}, \ldots, x_0) \in \{0,1\}^n$,

- $\overline{x}$ o *inteiro representado por* x *em base* 2

Tornando as coisas mais formais

Dada uma sequência $x = (x_{n-1}, \ldots, x_0) \in \{0,1\}^n$,

- $\overline{x}$ o *inteiro representado por* x *em base* 2

Isto é,

Tornando as coisas mais formais

Dada uma sequência $x = (x_{n-1}, \ldots, x_0) \in \{0,1\}^n$,

- $\overline{x}$ o *inteiro representado por* x *em base* 2

$$\text{Isto é,} \qquad \overline{x} = \sum_{i=0}^{n-1} 2^i x_i.$$

Tornando as coisas mais formais

Dada uma sequência $x = (x_{n-1}, \ldots, x_0) \in \{0,1\}^n$,

- $\overline{x}$ o *inteiro representado por x em base* 2

$$\text{Isto é,} \qquad \overline{x} = \sum_{i=0}^{n-1} 2^i x_i.$$

Notação para tamanho de x:

Tornando as coisas mais formais

Dada uma sequência $x = (x_{n-1}, \ldots, x_0) \in \{0,1\}^n$,

- $\overline{x}$ o *inteiro representado por* x *em base* 2

$$\text{Isto é,} \qquad \overline{x} = \sum_{i=0}^{n-1} 2^i x_i.$$

Notação para tamanho de x:

$$|(x_{n-1}, \ldots, x_0)| = n.$$

Tornando as coisas mais formais

Dada uma sequência $x = (x_{n-1}, \ldots, x_0) \in \{0,1\}^n$,

- $\overline{x}$ o *inteiro representado por* x *em base* 2

$$\text{Isto é,} \qquad \overline{x} = \sum_{i=0}^{n-1} 2^i x_i.$$

Notação para tamanho de x:

$$|(x_{n-1}, \ldots, x_0)| = n.$$

Dado $x \in \mathbb{N}$, denotamos por $\langle x \rangle$ qualquer sequência sobre $\{0,1\}$ satisfazendo

$$\overline{\langle x \rangle} = x.$$

Tornando as coisas mais formais

Dada uma sequência $x = (x_{n-1}, \ldots, x_0) \in \{0,1\}^n$,

- $\overline{x}$ o *inteiro representado por* x *em base* 2

$$\text{Isto é,} \qquad \overline{x} = \sum_{i=0}^{n-1} 2^i x_i.$$

Notação para tamanho de x:

$$|(x_{n-1}, \ldots, x_0)| = n.$$

Dado $x \in \mathbb{N}$, denotamos por $\langle x \rangle$ qualquer sequência sobre $\{0,1\}$ satisfazendo

$$\overline{\langle x \rangle} = x.$$

Concatenação:

Tornando as coisas mais formais

Dada uma sequência $x = (x_{n-1}, \dots, x_0) \in \{0,1\}^n$,

- $\overline{x}$ o *inteiro representado por* x *em base* 2

$$\text{Isto é,} \qquad \overline{x} = \sum_{i=0}^{n-1} 2^i x_i.$$

Notação para tamanho de x:

$$|(x_{n-1}, \dots, x_0)| = n.$$

Dado $x \in \mathbb{N}$, denotamos por $\langle x \rangle$ qualquer sequência sobre $\{0,1\}$ satisfazendo

$$\overline{\langle x \rangle} = x.$$

Concatenação: Dadas as sequências $x = (x_{n-1}, \dots, x_0)$ e $y = (y_{m-1}, \dots, y_0)$:

Tornando as coisas mais formais

Dada uma sequência $x = (x_{n-1}, \ldots, x_0) \in \{0,1\}^n$,

- $\overline{x}$ o *inteiro representado por* x *em base* 2

$$\text{Isto é,} \qquad \overline{x} = \sum_{i=0}^{n-1} 2^i x_i.$$

Notação para tamanho de x:

$$|(x_{n-1}, \ldots, x_0)| = n.$$

Dado $x \in \mathbb{N}$, denotamos por $\langle x \rangle$ qualquer sequência sobre $\{0,1\}$ satisfazendo

$$\overline{\langle x \rangle} = x.$$

Concatenação: Dadas as sequências $x = (x_{n-1}, \ldots, x_0)$ e $y = (y_{m-1}, \ldots, y_0)$:

$$x \cdot y = (x_{n-1}, \ldots, x_0, y_{m-1}, \ldots, y_0).$$

Tornando as coisas mais formais

Dados $x, y \in \mathbb{N}$, denotamos por $x \times y$ o produto de x e y.

Tornando as coisas mais formais

Dados $x, y \in \mathbb{N}$, denotamos por $x \times y$ o produto de x e y.

Multiplicação de Inteiros (MI)

Tornando as coisas mais formais

Dados $x, y \in \mathbb{N}$, denotamos por $x \times y$ o produto de x e y.

Multiplicação de Inteiros (MI)

- Entrada: Um par (x, y), onde $x = (x_{n-1}, \ldots, x_0)$ e $y = (y_{m-1}, \ldots, y_0)$ são sequências sobre $\{0, 1\}$.

Tornando as coisas mais formais

Dados $x, y \in \mathbb{N}$, denotamos por $x \times y$ o produto de x e y.

Multiplicação de Inteiros (MI)

- Entrada: Um par (x, y), onde $x = (x_{n-1}, \ldots, x_0)$ e $y = (y_{m-1}, \ldots, y_0)$ são sequências sobre $\{0, 1\}$.
- Saída: Uma sequência z sobre $\{0, 1\}$ tal que $\overline{z} = \overline{x} \times \overline{y}$

O Algoritmo do Ensino Fundamental

Multiplica$_F(x, y)$

$n \leftarrow \max\{|x|, |y|\}$
acrescente 0s ao início de x ou y até que ambas tenham tamanho n
$r \leftarrow (0)$
Para i *de* 0 *até* $n - 1$
    Se $y_i \neq 0$
        $r \leftarrow$ Soma(r, x)
    acrescente 0 ao final de x
Devolva r

O Algoritmo do Ensino Fundamental

$\text{Multiplica}_F(x, y)$

$n \leftarrow \max\{|x|, |y|\}$
acrescente 0s ao início de x ou y até que ambas tenham tamanho n
$r \leftarrow (0)$
Para i *de* 0 *até* $n - 1$
    Se $y_i \neq 0$
        $r \leftarrow \text{Soma}(r, x)$
    acrescente 0 ao final de x
Devolva r

$\text{Soma}(x, y)$

Entrada : duas sequências x e y sobre $\{0, 1\}$.
Saída : uma sequência r sobre $\{0, 1\}$ satisfazendo $\overline{r} = \overline{x} + \overline{y}$

$n \leftarrow \max\{|x|, |y|\}$
acrescente 0s ao início de x ou y até que ambas tenham tamanho n
$c \leftarrow 0$
$r \leftarrow ()$
Para i *de* 0 *até* $n - 1$
    acrescente $(x_i + y_i + c) \bmod 2$ ao início de r
    $c \leftarrow x_i y_i$
Se $c \neq 0$
    acrescente c ao início de r
Se $r = ()$
    acrescente 0 ao início de r
Devolva r

Quanto custa?

Teorema 44

O tempo de execução de Soma(x, y) é $\Theta(n)$, onde $n = \max\{|x|, |y|\}$.

Prova

A instrução da linha 1 pode ser implementada com um algoritmo que percorre as sequências x e y para descobrir qual sequência é mais longa. Isso toma tempo menor or igual a cn, para alguma constante c. Portanto esta linha executa em tempo $\Theta(n)$.

O laço principal (linhas 5 até 7) contém duas instruções e cada uma delas é executada em tempo constante. Como o laço é executado n vezes, o custo deste laço é $2n = \Theta(n)$.

Cada uma das demais instruções leva tempo constante, portanto as demais linhas do algoritmo executam em tempo $\Theta(1)$. Portanto o tempo de execução total do algoritmo é $\Theta(n) + \Theta(n) + \Theta(1) = \Theta(n)$.

Quanto custa?

Corolário 45

O tempo de execução de pior caso de $\text{Multiplica}_F(x, y)$ é $\Theta(n^2)$, onde $n = \max\{|x|, |y|\}$.

Prova

O tempo de execução do algoritmo é dominado assintoticamente pelo laço principal (linhas 4 até 7). Neste laço a instrução assintoticamente dominante é a instrução da linha 6, que tem custo $\Theta(n)$.

Como o laço executa n vezes, temos que o custo do algoritmo é $n\Theta(n) = \Theta(n^2)$.

É possível fazer melhor?

Teorema 46

Todo algoritmo para MI consome tempo $\Omega(n)$, onde $n = \max\{|x|, |y|\}$ para toda instância (x, y) onde $x \neq 0 \neq y$.

É possível fazer melhor?

Teorema 46

Todo algoritmo para MI consome tempo $\Omega(n)$, onde $n = \max\{|x|, |y|\}$ para toda instância (x, y) onde $x \neq 0 \neq y$.

Prova

Dada uma instância (x, y) do MI com $x \neq 0 \neq y$, a resposta desta instância será uma sequência de tamanho pelo menos $n = \max\{|x|, |y|\}$.

É possível fazer melhor?

Teorema 46

Todo algoritmo para MI consome tempo $\Omega(n)$, onde $n = \max\{|x|, |y|\}$ para toda instância (x, y) onde $x \neq 0 \neq y$.

Prova

Dada uma instância (x, y) do MI com $x \neq 0 \neq y$, a resposta desta instância será uma sequência de tamanho pelo menos $n = \max\{|x|, |y|\}$.

Corolário 47

O tempo de execução de pior caso de qualquer algoritmo para MI é $\Omega(n)$, onde $n = \max\{|x|, |y|\}$.

Algoritmo Recursivo

Algoritmo Recursivo

Teorema 48

Dados $x, y \in \{0,1\}^*$, então

$$\overline{x \cdot y} = 2^{|y|} \times \overline{x} + \overline{y},$$

Algoritmo Recursivo

Teorema 48

Dados $x, y \in \{0,1\}^*$, então
$$\overline{x \cdot y} = 2^{|y|} \times \overline{x} + \overline{y},$$

Corolário 49

Dados $x \in \{0,1\}^*$ e $n \in \mathbb{N}$,
$$\overline{x \cdot 0^n} = 2^n \times \overline{x}.$$

Algoritmo Recursivo

Teorema 48

Dados $x, y \in \{0,1\}^*$, então
$$\overline{x \cdot y} = 2^{|y|} \times \overline{x} + \overline{y},$$

Corolário 49

Dados $x \in \{0,1\}^*$ e $n \in \mathbb{N}$,
$$\overline{x \cdot 0^n} = 2^n \times \overline{x}.$$

Corolário 50

Dados $x, y \in \{0,1\}^*$ e $n \in \mathbb{N}$, é possível computar

- $\langle 2^{|y|} \times \overline{x} + \overline{y} \rangle$ em tempo $\mathcal{O}(1)$, e
- $\langle 2^n \times \overline{x} \rangle$ em tempo $\Theta(n)$.

Algoritmo Recursivo

Dado $x \in \{0,1\}^n$, com n par, definimos

$$x_{\mathrm{L}} = (x_{n-1}, \ldots, x_{n/2})$$
$$x_{\mathrm{R}} = (x_{n/2-1}, \ldots, x_0),$$

Algoritmo Recursivo

Dado $x \in \{0,1\}^n$, com n par, definimos

$$x_{\mathrm{L}} = (x_{n-1}, \ldots, x_{n/2})$$
$$x_{\mathrm{R}} = (x_{n/2-1}, \ldots, x_0),$$

de forma que

Dado $x \in \{0,1\}^n$, com n par, definimos

$$\begin{aligned} x_{\mathrm{L}} &= (x_{n-1}, \ldots, x_{n/2}) \\ x_{\mathrm{R}} &= (x_{n/2-1}, \ldots, x_0), \end{aligned}$$

de forma que

$$x = x_{\mathrm{L}} \cdot x_{\mathrm{R}},$$

Algoritmo Recursivo

Dado $x \in \{0,1\}^n$, com n par, definimos

$$x_{\mathrm{L}} = (x_{n-1}, \ldots, x_{n/2})$$
$$x_{\mathrm{R}} = (x_{n/2-1}, \ldots, x_0),$$

de forma que

$$x = x_{\mathrm{L}} \cdot x_{\mathrm{R}},$$

Pelo T.48, temos que

Algoritmo Recursivo

Dado $x \in \{0,1\}^n$, com n par, definimos

$$x_{\mathrm{L}} = (x_{n-1}, \dots, x_{n/2})$$
$$x_{\mathrm{R}} = (x_{n/2-1}, \dots, x_0),$$

de forma que

$$x = x_{\mathrm{L}} \cdot x_{\mathrm{R}},$$

Pelo T.48, temos que

$$\overline{x} = 2^{n/2} \times \overline{x_{\mathrm{L}}} + \overline{x_{\mathrm{R}}}.$$

Algoritmo Recursivo

$$
\begin{aligned}
\overline{x} \times \overline{y} &= \overline{x_{\mathrm{L}} \cdot x_{\mathrm{R}}} \times \overline{y_{\mathrm{L}} \cdot y_{\mathrm{R}}} \\
&\overset{\text{T.48}}{=} (2^{n/2} \times \overline{x_{\mathrm{L}}} + \overline{x_{\mathrm{R}}})(2^{n/2} \times \overline{y_{\mathrm{L}}} + \overline{y_{\mathrm{R}}}), \\
&= 2^{n/2} \times \overline{x_{\mathrm{L}}} \times 2^{n/2} \times \overline{y_{\mathrm{L}}} + 2^{n/2} \times \overline{x_{\mathrm{L}}} \times \overline{y_{\mathrm{R}}} + \overline{x_{\mathrm{R}}} \times 2^{n/2} \times \overline{y_{\mathrm{L}}} + \overline{x_{\mathrm{R}}} \times \overline{y_{\mathrm{R}}} \\
&= (2^{n} \times \overline{x_{\mathrm{L}}} \times \overline{y_{\mathrm{L}}} + \overline{x_{\mathrm{R}}} \times \overline{y_{\mathrm{R}}}) + (2^{n/2} \times (\overline{x_{\mathrm{L}}} \times \overline{y_{\mathrm{R}}} + \overline{x_{\mathrm{R}}} \times \overline{y_{\mathrm{L}}}) + \overline{0^{n/2}}) \\
&\overset{\text{T.48}}{=} \overline{\langle \overline{x_{\mathrm{L}}} \cdot \overline{y_{\mathrm{L}}} \rangle \, \langle \overline{x_{\mathrm{R}}} \cdot \overline{y_{\mathrm{R}}} \rangle} + \overline{\langle \overline{x_{\mathrm{L}}} \times \overline{y_{\mathrm{R}}} + \overline{x_{\mathrm{R}}} \times \overline{y_{\mathrm{L}}} \rangle \cdot 0^{n/2}}
\end{aligned}
$$

Algoritmo Recursivo

$$
\begin{aligned}
\overline{x} \times \overline{y} &= \overline{x_{\mathrm{L}} \cdot x_{\mathrm{R}}} \times \overline{y_{\mathrm{L}} \cdot y_{\mathrm{R}}} \\
&\overset{\text{T.48}}{=} (2^{n/2} \times \overline{x_{\mathrm{L}}} + \overline{x_{\mathrm{R}}})(2^{n/2} \times \overline{y_{\mathrm{L}}} + \overline{y_{\mathrm{R}}}), \\
&= 2^{n/2} \times \overline{x_{\mathrm{L}}} \times 2^{n/2} \times \overline{y_{\mathrm{L}}} + 2^{n/2} \times \overline{x_{\mathrm{L}}} \times \overline{y_{\mathrm{R}}} + \overline{x_{\mathrm{R}}} \times 2^{n/2} \times \overline{y_{\mathrm{L}}} + \overline{x_{\mathrm{R}}} \times \overline{y_{\mathrm{R}}} \\
&= (2^{n} \times \overline{x_{\mathrm{L}}} \times \overline{y_{\mathrm{L}}} + \overline{x_{\mathrm{R}}} \times \overline{y_{\mathrm{R}}}) + (2^{n/2} \times (\overline{x_{\mathrm{L}}} \times \overline{y_{\mathrm{R}}} + \overline{x_{\mathrm{R}}} \times \overline{y_{\mathrm{L}}}) + \overline{0^{n/2}}) \\
&\overset{\text{T.48}}{=} \overline{\langle \overline{x_{\mathrm{L}}} \cdot \overline{y_{\mathrm{L}}} \rangle \, \langle \overline{x_{\mathrm{R}}} \cdot \overline{y_{\mathrm{R}}} \rangle} + \overline{\langle \overline{x_{\mathrm{L}}} \times \overline{y_{\mathrm{R}}} + \overline{x_{\mathrm{R}}} \times \overline{y_{\mathrm{L}}} \rangle \cdot 0^{n/2}}
\end{aligned}
$$

ou seja,

Algoritmo Recursivo

$$
\begin{aligned}
\overline{x} \times \overline{y} &= \overline{x_L \cdot x_R} \times \overline{y_L \cdot y_R} \\
&\overset{\text{T.48}}{=} (2^{n/2} \times \overline{x_L} + \overline{x_R})(2^{n/2} \times \overline{y_L} + \overline{y_R}), \\
&= 2^{n/2} \times \overline{x_L} \times 2^{n/2} \times \overline{y_L} + 2^{n/2} \times \overline{x_L} \times \overline{y_R} + \overline{x_R} \times 2^{n/2} \times \overline{y_L} + \overline{x_R} \times \overline{y_R} \\
&= (2^{n} \times \overline{x_L} \times \overline{y_L} + \overline{x_R} \times \overline{y_R}) + (2^{n/2} \times (\overline{x_L} \times \overline{y_R} + \overline{x_R} \times \overline{y_L}) + \overline{0^{n/2}}) \\
&\overset{\text{T.48}}{=} \overline{\langle \overline{x_L} \cdot \overline{y_L} \rangle \, \langle \overline{x_R} \cdot \overline{y_R} \rangle} + \overline{\langle \overline{x_L} \times \overline{y_R} + \overline{x_R} \times \overline{y_L} \rangle \cdot 0^{n/2}}
\end{aligned}
$$

ou seja,

$$
\langle \overline{x} \times \overline{y} \rangle = \left\langle \overline{\langle \overline{x_L} \times \overline{y_L} \rangle \, \langle \overline{x_R} \times \overline{y_R} \rangle} + \overline{\langle \overline{x_L} \times \overline{y_R} + \overline{x_R} \times \overline{y_L} \rangle \, 0^{n/2}} \right\rangle .
$$

Algoritmo Recursivo

$$\begin{aligned}
\overline{x} \times \overline{y} &= \overline{x_L \cdot x_R} \times \overline{y_L \cdot y_R} \\
&\overset{\text{T.48}}{=} (2^{n/2} \times \overline{x_L} + \overline{x_R})(2^{n/2} \times \overline{y_L} + \overline{y_R}), \\
&= 2^{n/2} \times \overline{x_L} \times 2^{n/2} \times \overline{y_L} + 2^{n/2} \times \overline{x_L} \times \overline{y_R} + \overline{x_R} \times 2^{n/2} \times \overline{y_L} + \overline{x_R} \times \overline{y_R} \\
&= (2^n \times \overline{x_L} \times \overline{y_L} + \overline{x_R} \times \overline{y_R}) + (2^{n/2} \times (\overline{x_L} \times \overline{y_R} + \overline{x_R} \times \overline{y_L}) + \overline{0^{n/2}}) \\
&\overset{\text{T.48}}{=} \overline{\langle \overline{x_L} \cdot \overline{y_L} \rangle \, \langle \overline{x_R} \cdot \overline{y_R} \rangle} + \overline{\langle \overline{x_L} \times \overline{y_R} + \overline{x_R} \times \overline{y_L} \rangle \cdot 0^{n/2}}
\end{aligned}$$

ou seja,

$$\langle \overline{x} \times \overline{y} \rangle = \left\langle \overline{\langle \overline{x_L} \times \overline{y_L} \rangle \, \langle \overline{x_R} \times \overline{y_R} \rangle} + \overline{\langle \overline{x_L} \times \overline{y_R} + \overline{x_R} \times \overline{y_L} \rangle \, 0^{n/2}} \right\rangle .$$

Teorema 51

Dados $n > 0$ e $x, y \in \{0,1\}^n$,

$$\overline{x} \times \overline{y} = \begin{cases} \overline{x} \times \overline{y}, & \text{se } n = 1, \\ \overline{\langle \overline{x_L} \times \overline{y_L} \rangle \, \langle \overline{x_R} \times \overline{y_R} \rangle} + \overline{\langle \overline{x_L} \times \overline{y_R} + \overline{x_R} \times \overline{y_L} \rangle \cdot 0^{n/2}}, & \text{se } n \geq 2 \text{ é par}, \\ \overline{(0) \cdot x} \times \overline{(0) \cdot y}, & \text{se } n \geq 3 \text{ é ímpar}. \end{cases}$$

Multiplica(x, y)

Se $n = 1$
    Devolva $(x_1 \times y_1)$

$n \leftarrow \max\{|x|, |y|\}$
acrescente 0s ao início de x ou y até que ambas tenham tamanho n
acrescente um 0 ao início de x e de y /* para que nenhuma soma passe de tamanho n */
$n \leftarrow n + 1$
Se n *é ímpar*
    acrescente um 0 ao início de x e de y
    $n \leftarrow n + 1$

$x_L \leftarrow (x_{n-1}, \dots, x_{n/2})$
$x_R \leftarrow (x_{n/2-1}, \dots, x_0)$
$y_L \leftarrow (y_{n-1}, \dots, y_{n/2})$
$y_R \leftarrow (y_{n/2-1}, \dots, y_0)$

$A \leftarrow$ Multiplica(x_L, y_L)
$B \leftarrow$ Multiplica(x_R, y_R)
$C \leftarrow$ Multiplica(x_L, y_R)
$D \leftarrow$ Multiplica(x_R, y_L)

$E \leftarrow A \cdot B$
$F \leftarrow$ Soma$(C, D) \cdot$ Zero$(n/2)$
$r \leftarrow$ Soma(E, F)
Devolva r

Algoritmo Recursivo

- Que problema resolve? MI.
- Está correto? T.51
- Quanto custa? (próximo slide)

Quanto custa

Teorema 52

O tempo de execução de Multiplica(x, y) é $\Theta(n^2)$ onde

$$n = \max\{|x|, |y|\}.$$

Prova

Seja (x, y) uma instância de MI e seja $n = \max\{|x|, |y|\}$.
Excetuando-se o tempo consumido nas chamadas recursivas, o restante do tempo na execução de Multiplica(x, y) é $\Theta(n)$ pois cada linha toma tempo $\mathcal{O}(n)$ e a execução de Zero$(n/2)$ toma tempo $\Omega(n)$.

Na notação do T.43, temos então $T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n)$, onde

$$\begin{aligned} a &= 4, \\ b &= 2, \\ f(n) &= \Theta(n), \end{aligned}$$

Logo $n^{\log_b a} = n^{\log_2 4} = n^2$, e como $f(n) = \Omega(n^{2-1})$, então (T.43)

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2).$$

É possível fazer melhor? Sim.

Algoritmo de Karatsuba (1962)

Lema 53

Dados $a, b, c, d \in \mathbb{R}$, é possível computar os valores de $a \times c$, $a \times d + b \times c$ e $b \times d$ efetuando apenas três multiplicações.

Algoritmo de Karatsuba (1962)

Lema 53

Dados $a, b, c, d \in \mathbb{R}$, é possível computar os valores de $a \times c$, $a \times d + b \times c$ e $b \times d$ efetuando apenas três multiplicações.

Prova

Sejam $a, b, c, d \in \mathbb{R}$. Como

$$(a + b) \times (c + d) = a \times c + a \times d + b \times c + b \times d,$$

então

$$a \times d + b \times c = (a + b) \times (c + d) - a \times c - b \times d.$$

Algoritmo de Karatsuba (1962)

Lema 53

Dados $a, b, c, d \in \mathbb{R}$, é possível computar os valores de $a \times c$, $a \times d + b \times c$ e $b \times d$ efetuando apenas três multiplicações.

Prova

Sejam $a, b, c, d \in \mathbb{R}$. Como

$$(a + b) \times (c + d) = a \times c + a \times d + b \times c + b \times d,$$

então

$$a \times d + b \times c = (a + b) \times (c + d) - a \times c - b \times d.$$

- Folclore (não está claro se é verdade): Gauss (1777 –1855) usava este fato para diminuir o número de multiplicações entre números reais no cômputo de produto de números complexos: $(a + bi)(c + di) = ac - bd + (bc + ad)i$

Algoritmo de Karatsuba (1962)

Aplicando o Lema 53 ao termo

$$\overline{x_L} \times \overline{y_R} + \overline{x_R} \times \overline{y_L}$$

Algoritmo de Karatsuba (1962)

Aplicando o Lema 53 ao termo

$$\overline{x_L} \times \overline{y_R} + \overline{x_R} \times \overline{y_L}$$

da recorrência do Teorema 51 substitui-mo-lo por

$$(\overline{x_L} + \overline{x_R}) \times (\overline{y_L} + \overline{y_R}) - \overline{x_L} \times \overline{y_L} - \overline{x_R} \times \overline{y_R}$$

Algoritmo de Karatsuba (1962)

Aplicando o Lema 53 ao termo

$$\overline{x_{\mathrm{L}}} \times \overline{y_{\mathrm{R}}} + \overline{x_{\mathrm{R}}} \times \overline{y_{\mathrm{L}}}$$

da recorrência do Teorema 51 substitui-mo-lo por

$$(\overline{x_{\mathrm{L}}} + \overline{x_{\mathrm{R}}}) \times (\overline{y_{\mathrm{L}}} + \overline{y_{\mathrm{R}}}) - \overline{x_{\mathrm{L}}} \times \overline{y_{\mathrm{L}}} - \overline{x_{\mathrm{R}}} \times \overline{y_{\mathrm{R}}}$$

Corolário 54

Dados $n > 0$ e $x, y \in \{0,1\}^n$,

$$\overline{x} \times \overline{y} = \begin{cases} \overline{x} \times \overline{y}, & \text{se } n = 1, \\ \overline{\langle \overline{x_{\mathrm{L}}} \times \overline{y_{\mathrm{L}}} \rangle \cdot \langle \overline{x_{\mathrm{R}}} \times \overline{y_{\mathrm{R}}} \rangle} + \overline{\langle (\overline{x_{\mathrm{L}}} + \overline{x_{\mathrm{R}}}) \times (\overline{y_{\mathrm{L}}} + \overline{y_{\mathrm{R}}}) - \overline{x_{\mathrm{L}}} \times \overline{y_{\mathrm{L}}} - \overline{x_{\mathrm{R}}} \times \overline{y_{\mathrm{R}}} \rangle \cdot 0^{n/2}}, & \text{se } n \geq 2 \text{ par} \\ \overline{(0) \cdot x} \times \overline{(0) \cdot y}, & \text{se } n \geq 3 \text{ ímpar} \end{cases}$$

Multiplica(x, y)

$n \leftarrow \max\{|x|, |y|\}$

acrescente 0s ao início de x ou y até que ambas tenham tamanho n
Se $n = 1$
    Devolva $(x_1 \times y_1)$

acrescente um 0 ao início de x e de y
$n \leftarrow n + 1$

Se *n é ímpar*
    acrescente outro 0 ao início de x e de y
    $n \leftarrow n + 1$

$x_L \leftarrow (x_{n-1}, \ldots, x_{n/2})$
$x_R \leftarrow (x_{n/2-1}, \ldots, x_0)$
$y_L \leftarrow (y_{n-1}, \ldots, y_{n/2})$
$y_R \leftarrow (y_{n/2-1}, \ldots, y_0)$

$A \leftarrow$ Multiplica(x_L, y_L)
$B \leftarrow$ Multiplica(x_R, y_R)
$C \leftarrow$ Soma(x_L, x_R)
$D \leftarrow$ Soma(y_L, y_R)
$E \leftarrow$ Multiplica(C, D)
$F \leftarrow$ Soma(A, B)
$G \leftarrow$ Subtrai(E, F)
remova os zeros do início de B

Devolva $Soma(A \cdot B, G \cdot Zero(n/2))$

onde Subtrai(x, y) devolve $\langle \overline{x} - \overline{y} \rangle$ (Exercício **??**).

Algoritmo de Karatsuba (1962)

Teorema 55

O tempo de execução de Multiplica(x, y) é $\Theta(n^{\lg 3})$. onde

$$n = \max\{|x|, |y|\}.$$

Teorema 55

O tempo de execução de Multiplica(x, y) é $\Theta(n^{\lg 3})$. onde

$$n = \max\{|x|, |y|\}.$$

Prova

Seja (x, y) uma instância do MI e seja $n = \max\{|x|, |y|\}$.
Excetuando-se o tempo consumido nas chamadas recursivas, o restante do tempo na execução de Multiplica(x, y) é $\Theta(n)$.

Algoritmo de Karatsuba (1962)

Teorema 55

O tempo de execução de Multiplica(x, y) é $\Theta(n^{\lg 3})$. onde

$$n = \max\{|x|, |y|\}.$$

Prova

Seja (x, y) uma instância do MI e seja $n = \max\{|x|, |y|\}$.
Excetuando-se o tempo consumido nas chamadas recursivas, o restante do tempo na execução de Multiplica(x, y) é $\Theta(n)$. Na notação do T.43, temos

Algoritmo de Karatsuba (1962)

Teorema 55

O tempo de execução de Multiplica(x, y) é $\Theta(n^{\lg 3})$. onde

$$n = \max\{|x|, |y|\}.$$

Prova

Seja (x, y) uma instância do MI e seja $n = \max\{|x|, |y|\}$.
Excetuando-se o tempo consumido nas chamadas recursivas, o restante do tempo na execução de Multiplica(x, y) é $\Theta(n)$. Na notação do T.43, temos

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n),$$

Algoritmo de Karatsuba (1962)

Teorema 55

O tempo de execução de Multiplica(x, y) é $\Theta(n^{\lg 3})$. onde

$$n = \max\{|x|, |y|\}.$$

Prova

Seja (x, y) uma instância do MI e seja $n = \max\{|x|, |y|\}$.
Excetuando-se o tempo consumido nas chamadas recursivas, o restante do tempo na execução de Multiplica(x, y) é $\Theta(n)$. Na notação do T.43, temos

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n),$$

com

Algoritmo de Karatsuba (1962)

Teorema 55

O tempo de execução de Multiplica(x, y) é $\Theta(n^{\lg 3})$. onde

$$n = \max\{|x|, |y|\}.$$

Prova

Seja (x, y) uma instância do MI e seja $n = \max\{|x|, |y|\}$.
Excetuando-se o tempo consumido nas chamadas recursivas, o restante do tempo na execução de Multiplica(x, y) é $\Theta(n)$. Na notação do T.43, temos

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n),$$

com

$$\begin{aligned} a &= 3, \\ b &= 2, \\ f(n) &= \Theta(n), \end{aligned}$$

Algoritmo de Karatsuba (1962)

Teorema 55

O tempo de execução de Multiplica(x, y) é $\Theta(n^{\lg 3})$. onde

$$n = \max\{|x|, |y|\}.$$

Prova

Seja (x, y) uma instância do MI e seja $n = \max\{|x|, |y|\}$.
Excetuando-se o tempo consumido nas chamadas recursivas, o restante do tempo na execução de Multiplica(x, y) é $\Theta(n)$. Na notação do T.43, temos

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n),$$

com

$$\begin{aligned} a &= 3, \\ b &= 2, \\ f(n) &= \Theta(n), \end{aligned}$$

Daí $n^{\log_b a}$

Algoritmo de Karatsuba (1962)

Teorema 55

O tempo de execução de Multiplica(x, y) é $\Theta(n^{\lg 3})$. onde

$$n = \max\{|x|, |y|\}.$$

Prova

Seja (x, y) uma instância do MI e seja $n = \max\{|x|, |y|\}$.
Excetuando-se o tempo consumido nas chamadas recursivas, o restante do tempo na execução de Multiplica(x, y) é $\Theta(n)$. Na notação do T.43, temos

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n),$$

com

$$\begin{aligned} a &= 3, \\ b &= 2, \\ f(n) &= \Theta(n), \end{aligned}$$

Daí $n^{\log_b a} = n^{\log_2 3}$

Algoritmo de Karatsuba (1962)

Teorema 55

O tempo de execução de Multiplica(x, y) é $\Theta(n^{\lg 3})$. onde

$$n = \max\{|x|, |y|\}.$$

Prova

Seja (x, y) uma instância do MI e seja $n = \max\{|x|, |y|\}$.
Excetuando-se o tempo consumido nas chamadas recursivas, o restante do tempo na execução de Multiplica(x, y) é $\Theta(n)$. Na notação do T.43, temos

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n),$$

com

$$\begin{aligned} a &= 3, \\ b &= 2, \\ f(n) &= \Theta(n), \end{aligned}$$

Daí $n^{\log_b a} = n^{\log_2 3} = n^{\lg 3}$ e como $f(n) = \Omega(n^{\lg 3-(\lg 3-1)})$

Algoritmo de Karatsuba (1962)

Teorema 55

O tempo de execução de Multiplica(x, y) é $\Theta(n^{\lg 3})$. onde

$$n = \max\{|x|, |y|\}.$$

Prova

Seja (x, y) uma instância do MI e seja $n = \max\{|x|, |y|\}$.
Excetuando-se o tempo consumido nas chamadas recursivas, o restante do tempo na execução de Multiplica(x, y) é $\Theta(n)$. Na notação do T.43, temos

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n),$$

com

$$\begin{aligned} a &= 3, \\ b &= 2, \\ f(n) &= \Theta(n), \end{aligned}$$

Daí $n^{\log_b a} = n^{\log_2 3} = n^{\lg 3}$ e como $f(n) = \Omega(n^{\lg 3 - (\lg 3 - 1)})$, então (T.43):

Algoritmo de Karatsuba (1962)

Teorema 55

O tempo de execução de Multiplica(x, y) é $\Theta(n^{\lg 3})$. onde

$$n = \max\{|x|, |y|\}.$$

Prova

Seja (x, y) uma instância do MI e seja $n = \max\{|x|, |y|\}$.
Excetuando-se o tempo consumido nas chamadas recursivas, o restante do tempo na execução de Multiplica(x, y) é $\Theta(n)$. Na notação do T.43, temos

$$T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n),$$

com

$$\begin{aligned} a &= 3, \\ b &= 2, \\ f(n) &= \Theta(n), \end{aligned}$$

Daí $n^{\log_b a} = n^{\log_2 3} = n^{\lg 3}$ e como $f(n) = \Omega(n^{\lg 3-(\lg 3-1)})$, então (T.43):

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\lg 3}).$$

Algoritmo de Karatsuba (1962)

Corolário 56

O tempo de execução de Multiplica(x, y) é $\Omega(n^{1.58})$ e $\mathcal{O}(n^{1.59})$, onde, $n = \max\{|x|, |y|\}$.

Algoritmo de Karatsuba (1962)

Corolário 56

O tempo de execução de Multiplica(x, y) é $\Omega(n^{1.58})$ e $\mathcal{O}(n^{1.59})$, onde, $n = \max\{|x|, |y|\}$.

Prova

Basta observar que
$$1.58 < \lg 3 < 1.59.$$

Algoritmo de Karatsuba (1962)

Corolário 56

O tempo de execução de Multiplica(x, y) é $\Omega(n^{1.58})$ e $\mathcal{O}(n^{1.59})$, onde, $n = \max\{|x|, |y|\}$.

Prova

Basta observar que

$$1.58 < \lg 3 < 1.59.$$

- É possível fazer melhor?

Algoritmo de Karatsuba (1962)

Corolário 56

O tempo de execução de Multiplica(x, y) é $\Omega(n^{1.58})$ e $\mathcal{O}(n^{1.59})$, onde, $n = \max\{|x|, |y|\}$.

Prova

Basta observar que

$$1.58 < \lg 3 < 1.59.$$

- É possível fazer melhor?Sim, mas é complicado e ineficiente na prática