

Motivação, Multiplicação, Recorrências

Informalmente, um algoritmo é uma receita, i.e., uma sequência de regras bem definidas, para resolver um problema.

Mas, por que estudar algoritmos é tão importante? *IPA, CAP, AED1, AED2, PAA*

Algoritmos (e a área irmã estruturas de dados) são centrais na computação, sendo relevantes muitas áreas da mesma, como:

- roteamento de rede - usa algoritmos para caminhos mínimos e tabelas hash
- criptografia - usa algoritmos de teoria dos números
- computação gráfica - usa algoritmos geométricos
- biologia computacional - usa algoritmos de programação dinâmica
- bancos de dados - usa árvores de busca balanceadas e tabelas hash
- inteligência artificial - usa algoritmos em grafos
- otimização combinatória - usa algoritmos para resolver problemas nas mais diversas áreas

São importantes para a inovação tecnológica, pois algoritmos mais eficientes para um problema permitem aproveitar melhor o poder computacional disponível, possibilitando atacar instâncias maiores do problema.

São uma abordagem interessante para interpretar processos fora da computação, como o funcionamento de mercados em economia e da evolução na biologia.

A busca por bons algoritmos para os mais variados problemas é desafiadora, permitindo fortalecer raciocínio lógico e capacidade analítica.

É uma área cheia de resultados interessantes e surpreendentes, que apresenta soluções inesperadas mesmo para problemas clássicos.

Multiplicação

Definição do problema da multiplicação de inteiros.

Entrada:

- dois inteiros x e y com n dígitos cada

Solução:

- o produto x vezes y

Vamos avaliar a eficiência do algoritmo, em função de n , contando o número de operações básicas que ele realiza.

No contexto deste problema, uma operação básica é:

- soma ou multiplicação de dígitos
- multiplicação de inteiro por potência de 10

Exemplo: multiplicar $5678 \times 1234 = 7006652$.

$n = 4$

Handwritten multiplication of 5678×1234 showing the number of basic operations.

The multiplication is shown as follows:

$$\begin{array}{r} \overset{2\ 2\ 3\ 2\ 3\ 2}{5\ 6\ 7\ 8} \\ \times \quad \textcircled{1\ 2\ 3\ 4} \\ \hline 4 - \quad \textcircled{2\ 2\ 7\ 1\ 2} \\ 3 - \quad 1\ 7\ 0\ 3\ 4\ - \\ 2 - \quad 1\ 1\ 3\ 5\ 6\ -\ - \\ 1 - \quad 5\ 6\ 7\ 8\ -\ -\ - \\ \hline 7\ 0\ 0\ 6\ 6\ 5\ 2 \end{array}$$

Annotations:

- For the first row (4): $\# \text{ de operações básicas é da ordem de } n \leq 2n \rightarrow \# \text{ de operações é } \leq 2n^2$ (n linhas)
- For the second row (3): $\# \text{ de operações básicas é da ordem de } n \leq n \rightarrow \# \text{ de operações é } \leq 2n^2$ (2n colunas)

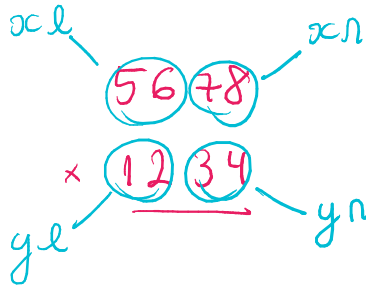
Concluimos que o número de operações básicas cresce de modo proporcional a uma função quadrática em n , digamos $4n^2$

"Perhaps the most important principle for the good algorithm designer is to refuse to be content" - Aho, Hopcroft and Ullman, The Design and Analysis of Computer Algorithms, 1974.

- Ou, simplesmente, conseguimos fazer melhor?

Exemplo do Algoritmo de Karatsuba: multiplicar $\underline{5678} \times \underline{1234} = 7006652$.

$n=4$



$$x = x_l \cdot 10^{n/2} + x_n = 56 \cdot 10^2 + 78$$

$$y = y_l \cdot 10^{n/2} + y_n = 12 \cdot 10^2 + 34$$

$$a = x_l \cdot y_l = 56 \cdot 12 = 672$$

$$b = x_n \cdot y_n = 78 \cdot 34 = 2652$$

$$c = (x_l + x_n)(y_l + y_n) = (56 + 78)(12 + 34) = 134 \cdot 46 = 6164$$

$$d = c - a - b = 6164 - 672 - 2652 = 2840$$

$$x * y = a \cdot 10^4 + d \cdot 10^2 + b = 672 \cdot 10^4 + 2840 \cdot 10^2 + 2652 = 7006652$$

- Por que isso funcionou? Isso sempre funciona? Por que isso é interessante?

Algoritmo recursivo simples

Dado um inteiro v com n dígitos, seja vl a metade mais significativa e vr a metade menos significativa. Note que,

$$v = vl \cdot 10^{n/2} + vr$$

Ideia do algoritmo:

1. Sendo x e y os inteiros na entrada do problema, considere a expressão

$$x * y = (xl \cdot 10^{n/2} + xr)(yl \cdot 10^{n/2} + yr) = \underbrace{xl \cdot yl}_{a} 10^n + \underbrace{(xl \cdot yr + xr \cdot yl)}_{b} \cdot 10^{n/2} + \underbrace{xr \cdot yr}_{c}$$

2. Como xl , xr , yl , yr tem metade dos dígitos dos números originais,
 - calcule recursivamente os produtos presentes na expressão anterior.
3. De posse do resultado de cada produto, para obter a solução, basta
 - multiplicá-lo pela potência de 10 adequada e
 - somar seguindo a expressão anterior.

Análise:

- Quantos são os produtos/chamadas recursivas no passo 2? 4
- Qual o tamanho dos inteiros nos subproblemas do passo 2? $n/2$
- Qual o trabalho para obter a solução no passo 3? $c \cdot n$
- Assim, o tempo gasto por esse algoritmo é dado pela recorrência:
$$T(n) = 4T(n/2) + c \cdot n$$
- Mas, qual é a função de tempo deste algoritmo?

Resolvendo a recorrência por substituição: $T(n) = 4 * T(n/2) + cn$

$$T(n) = 4[T(n/2)] + c.n = 4^1 T\left(\frac{n}{2^1}\right) + (2^1 - 1)c.n$$

$$T(n) = (4) \cdot (4T(n/4) + c.n/2) + c.n$$

$$= 4^2 [T(n/4)] + 3c.n = 4^2 T\left(\frac{n}{2^2}\right) + (2^2 - 1)c.n$$

$$T(n) = 4^2 (4T(n/8) + c.n/4) + 3c.n$$

$$= 4^3 [T(n/8)] + 7c.n = 4^3 T\left(\frac{n}{2^3}\right) + (2^3 - 1)c.n$$

$$T(n) = 4^3 (4T(n/16) + c.n/8) + 7c.n$$

$$= 4^4 T(n/16) + 15cn = 4^4 T\left(\frac{n}{2^4}\right) + (2^4 - 1)c.n$$

$\vdots \rightarrow$ prova por indução matemática

$$T(n) = 4^h T(n/2^h) + (2^h - 1)c.n$$

$$n/2^h = 1 \Rightarrow 2^h = n \Rightarrow h = \lg n$$

- Conseguimos fazer melhor?

$$T(n) = 4 T(n/2) + c.n$$

$$T(n/2) = 4 T(n/4) + c.n/2$$

$$T(n/4) = 4 T(n/8) + c.n/4$$

$$T(n/8) = 4 T(n/16) + c.n/8$$

$\vdots$

$$T(n) = 4^{\lg n} T\left(\frac{n}{2^{\lg n}}\right) + (2^{\lg n} - 1)c.n$$

$$= 2^{2\lg n} T\left(\frac{n}{n}\right) + (n - 1)c.n$$

$$= n^2 \cdot 1 + c \cdot n^2 - cn = O(n^2)$$

Ideia do Algoritmo de Karatsuba: revisitando multiplicar $5678 \times 1234 = 7006652$.

$n=4$

$x_l \leftarrow 5678 \leftarrow x_n$

$y_l \leftarrow 1234 \leftarrow y_n$

$$x = x_l \cdot 10^{n/2} + x_n \quad || \quad y = y_l \cdot 10^{n/2} + y_n$$

$$x \cdot y = (x_l \cdot 10^{n/2} + x_n)(y_l \cdot 10^{n/2} + y_n) = \underbrace{x_l \cdot y_l}_{a} 10^n + \underbrace{(x_l y_n + x_n y_l)}_{d} \cdot 10^{n/2} + \underbrace{x_n y_n}_{b}$$

- Não interessam os produtos $x_l \cdot y_n$ e $x_n \cdot y_l$ individualmente, só a soma deles.
- Conseguimos essa soma sem usar duas chamadas recursivas?

$$c = (x_l + x_n)(y_l + y_n) = \underbrace{x_l y_l}_{a} + \underbrace{(x_l y_n + x_n y_l)}_{d} + \underbrace{x_n y_n}_{b}$$

$$[d = c - a - b]$$

- Truque de Carl Friedrich Gauss para multiplicar números complexos.

$$a = x_l \cdot y_l = 56 \cdot 12 = 672 \quad || \quad b = x_n \cdot y_n = 78 \cdot 34 = 2652$$

$$c = (x_l + x_n)(y_l + y_n) = (56 + 78)(12 + 34) = 6164 \quad || \quad d = c - a - b = 6164 - 672 - 2652 = 2840$$

$$xy = a \cdot 10^n + d \cdot 10^{n/2} + b = 672 \cdot 10^4 + 2840 \cdot 10^2 + 2652 = 7006652$$

Algoritmo de Karatsuba

Pseudocódigo do algoritmo:

int multiKaratsubaRec (int x, int y):

(se $\text{tam}(x) == 1$ e $\text{tam}(y) == 1$:

 devolva $x * y$

$$n = \text{tam}(x) = \text{tam}(y)$$

(quebra x em x_l e x_r

 quebra y em y_l e y_r

$a = \text{multiKaratsubaRec}(x_l, y_l)$

$b = \text{multiKaratsubaRec}(x_r, y_r)$

$c = \text{multiKaratsubaRec}(x_l + x_r, y_l + y_r)$

($d = c - a - b$

 devolva $a \cdot 10^n + d \cdot 10^{n/2} + b$

Análise:

- Quantos são os produtos/chamadas recursivas? 3
- Qual o tamanho dos inteiros nos subproblemas? $n/2$
- Qual o trabalho local (não recursivo)? $C \cdot n$
- Assim, o tempo gasto por esse algoritmo é dado pela recorrência:

$$T(n) = 3T(n/2) + Cn$$

- Mas, qual é a função de tempo deste algoritmo?