Projeto e Análise de Algoritmos

PAA - Aula 01
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:

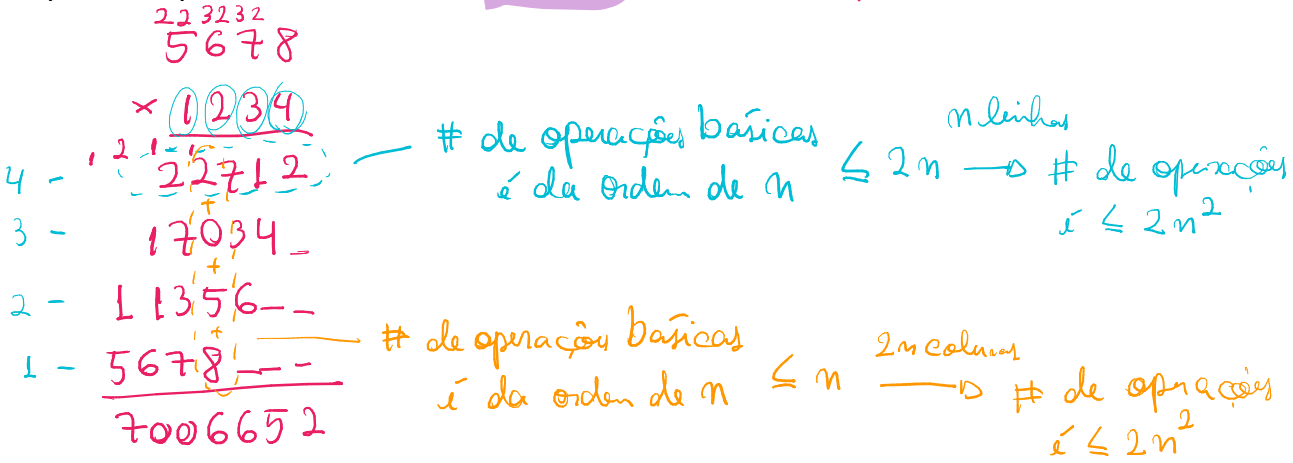
- o produto x vezes y

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 x 1234 = 7006652.



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

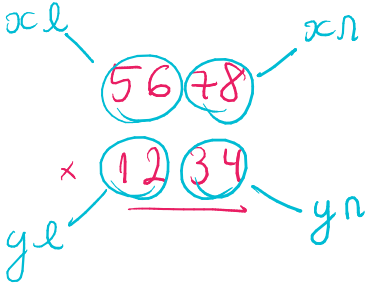
"Perhaps the most important principle for the good algorithm designer is to refuse to be content" - Aho, Hopcroft and Ullman, The Design and Analysis of Computer Algorithms, 1974.

- Ou, simplesmente, conseguimos fazer melhor?

Exemplo do Algoritmo de Karatsuba: multiplicar 5678 x 1234 = 7006652.

x y

$n = 4$



$$x = xl \cdot 10^{n/2} + xr = 56 \cdot 10^2 + 78$$

$$y = yl \cdot 10^{n/2} + yr = 12 \cdot 10^2 + 34$$

$a = xl \cdot yl = 56 \cdot 12 = 672$

$b = xr \cdot yr = 78 \cdot 34 = 2652$

$c = (xl + xr)(yl + yr) = (56 + 78)(12 + 34) = 134 \cdot 46 = 6164$

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

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

- 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

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

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:

$$T(n) = 4T(n/2) + cn$$

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

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

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

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

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

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

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

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

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

$\vdots$ → prova por indução matemática

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

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

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

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

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

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

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

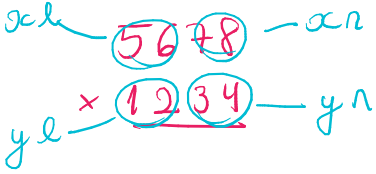
$T(n/4) = 4T(n/8) + c.n/4$

$T(n/8) = 4T(n/16) + cn/8$

$\vdots$

- Conseguimos fazer melhor?

Ideia do Algoritmo de Karatsuba: revisitando multiplicar 5678 x 1234 = 7006652.



$n = 4$

$x = xl \cdot 10^{n/2} + xr \quad \| \quad y = yl \cdot 10^{n/2} + yr$

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

- Não interessam os produtos xl * yr e xr * yl individualmente, só a soma deles.
- Conseguimos essa soma sem usar duas chamadas recursivas?

$$c = (xl + xr)(yl + yr) = \underbrace{xl \cdot yl}_{a} + \underbrace{(xl\,yr + xr\,yl)}_{d} + \underbrace{xr\,yr}_{b}$$

$$d = c - a - b$$

Truque de Gauss

- Truque de Carl Friedrich Gauss para multiplicar números complexos.

$a = xl \cdot yl = 56 \cdot 12 = 672 \quad \| \quad b = xr \cdot yr = 78 \cdot 34 = 2652$

$c = (xl + xr)(yl + yr) = (56 + 78)(12 + 34) = 6164 \quad \| \quad d = c - a - b = 6164 - 672 - 2652 = 2840$

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

Algoritmo de Karatsuba

Pseudocódigo do algoritmo:

```
int multiKaratsubaRec(int x, int y):
    Se tam(x) == 1 e tam(y) == 1:
        devolva x*y
    quebra x em xl e xr
    quebra y em yl e yr
    a = multiKaratsubaRec(xl, yl)
    b = multiKaratsubaRec(xr, yr)
    c = multiKaratsubaRec(xl+xr, yl+yr)
    d = c - a - b
    devolva a.10^n + d.10^(n/2) + b
```

$n = tam(x) = tam(y)$

Análise:

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

$$T(n) = 3\,T(n/2) + cn$$

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