Seminário Algoritmos e Fibonacci

Escrita e números

O Kish tablet, datado de 3500 AC, é o mais antigo registro escrito conhecido.

- é uma escrita proto cuneiforme
- e seu significado é desconhecido

O Ebla tablet, datado de 2500 AC, é o mais antigo registro escrito já traduzido.

- Ele apresenta foco principal em registros econômicos,
    - como controle de inventário, importação e exportação.
    - Inclusive, revelam que Ebla produzia diversas cervejas,
        - incluindo uma que levava o nome da cidade.

Esse exemplo mostra que, desde de sua origem,

- a escrita estava intimamente ligada com a matemática.

Revolução da imprensa e sistema posicional

Avançando quase 4 mil anos na história, vamos para o final da idade média,

- quando Johannes Gutenberg introduziu a prensa de tipos móveis na Europa,
    - provocando assim a revolução da imprensa,
        - que barateou e democratizou o acesso à informação e conhecimento.

Neste mesmo período, ocorria outra revolução de imensa importância,

- mas esta é muito menos conhecida. Curiosamente, ela também
    - foi iniciada por um grande nome da história
    - e também se baseou em conhecimento vindo do oriente.

Tal conhecimento é o sistema posicional, inventado na Índia em 600 DC.

- Apesar de imensamente superior a outros sistemas numéricos,
    - levou muito tempo para se espalhar, provavelmente devido a barreiras
        - como linguagem, distância, tradição e ignorância.

No meio desse longo caminho, teve papel chave um influente livro-texto

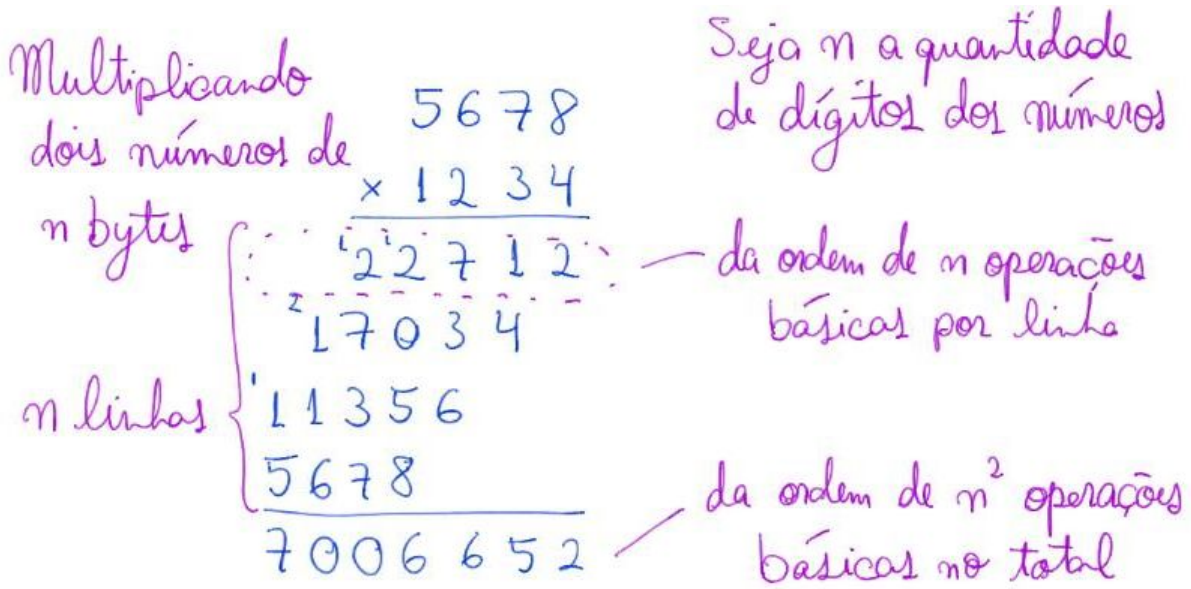
- escrito em Bagdá no século IX por **Al Khwarizmi**.
- Este livro estabeleceu os métodos básicos para
    - adicionar, multiplicar, dividir, obter a raiz quadrada
    - e até para calcular os dígitos de Pi.

Métodos para resolver um problema, que sejam

- precisos, não ambíguos, mecânicos, eficientes e corretos
    - nada mais são que **algoritmos**.
- Não à toa, esta palavra foi cunhada em homenagem ao autor do livro.
    - De fato, a palavra **algarismo** também deriva de seu nome,
        - e **álgebra** deriva da palavra de **al-jabr**,
            - que aparece no título da sua obra.

Mas é possível (e até provável) que você nunca tenha pensado

- nas operações fundamentais que aprendeu na escola,
    - como sendo algoritmos.
- Para deixar esse fato mais evidente, tomemos como exemplo
    - o algoritmo para multiplicação de dois números inteiros.
- Através do seguinte exemplo, vamos
    - relembrar o método e analisá-lo.



Voltando à nossa história, considere o impacto que o sistema posicional
- e seus algoritmos numéricos tiveram e têm no nosso mundo.
- Pense se a ciência, tecnologia, indústria, comércio, comunicações
    - poderiam existir se tivéssemos que resolver problemas como
        - somar MCDXLVIII com LII,
        - ao invés de somar 1448 com 52.
- Observe, particularmente, que utilizamos o sistema posicional
    - na organização de bits dos computadores.

Dito isso, nossa história ainda está na Europa de Gutenberg
- alguns séculos distante do tempo atual.
- Nela ainda eram utilizados numerais romanos
    - e especialistas em ábaco eram necessários
        - para resolver operações aritméticas
            - que hoje crianças executam com facilidade.

É neste cenário que surge o matemático italiano Leonardo Fibonacci,
- que trabalhou muito para divulgar o sistema posicional na Europa,
    - efetivamente ajudam a moldar a história e o mundo atual.
- Mas, apesar da importância de tal trabalho de divulgação científica,
    - Fibonacci tornou-se mundialmente conhecido
        - por conta da sequência que leva seu nome.
    - Assim, falemos um pouco da famosa sequência
        - para descobrir conexões entre ela e os algoritmos.

Para quem não se lembra,
- seguem os primeiros termos da sequência de Fibonacci

    - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …
- sua regra de formação corresponde a somar os dois números anteriores
- mais formalmente, podemos utilizar a seguinte definição recursiva

Fn = { Fn-1 + Fn-2, se n > 1
1, se n = 1
0, se n = 0 }

Esta sequência largamente estudada pode ser aplicada a várias áreas, como

- arte e arquitetura
    - por conta de sua relação com a proporção áurea
- biologia e demografia
    - por produzir boas descrições do crescimento de populações.
        - Em particular, de populações que apresentam tempo até a maturação sexual,
            - como no caso de coelhos.
    - Isso deriva da sequência de Fibonacci se comportar como
        - uma exponencial com atraso (tempo de maturação)
    - É curioso ela também surgir no número de pétalas de flores,
        - sugerindo que a mitose das células tronco que dão origem às flores não simétrica.
    - Para comparar, as potências de 2 são boas para descrever
        - o crescimento de bactérias, que realizam bipartição simétrica.
- Falando na exponencial de base 2,
    - provavelmente a única sequência mais relevantes na computação
        - que a própria sequência de Fibonacci,
    - vale destacar sua semelhança com esta última.
        - Para tanto usamos a definição recursiva da exponencial

En = { En-1 + En-1, se n >= 1,
1, se n = 0 }

Essa semelhança implica que,

- Fn cresce aproximadamente como 2^0,694n,
    - o que nos permite estimar seu valor
        - e concluir que seu crescimento é muito rápido.
- Mas, quanto vale exatamente F100 ou F200?
    - Para responder a essas perguntas precisamos de algoritmos para calcular os números de Fibonacci.

Uma primeira tentativa é um algoritmo recursivo

- baseado na própria definição da sequência

```
Fib1(n)
    se n = 0 então devolva 0
    se n = 0 então devolva 1
    devolva Fib1(n - 1) + Fib1(n - 2)
```

Será que esse algoritmo é eficiente? Seu tempo é descrito pela seguinte recorrência

$T(n) = T(n - 1) + T(n - 2) + 3$

Isso implica que, infelizmente,

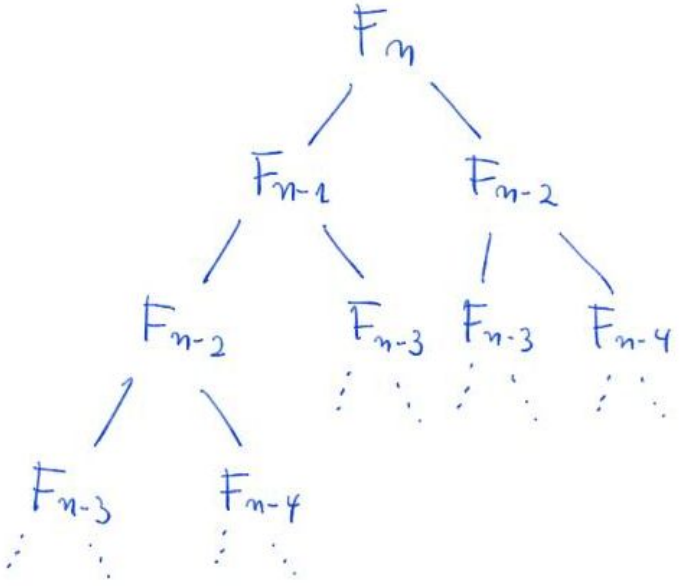
- o crescimento de T(n) é pelo menos tão rápido
    - quanto o crescimento da própria sequência de Fibonacci.
- Por isso, esse algoritmo é impraticável,
    - exceto para n muito pequeno.

Note que, para calcular F100

- o número de passos seria aproximadamente 2^69.
- Mesmo em um computador moderno que execute 10GHz instruções
    - esse cálculo levaria 1871 anos.
- Esse é um bom exemplo de que,
    - mesmo a imensa capacidade de processamento dos computadores modernos
        - não pode vencer o crescimento exponencial.

Será que podemos fazer melhor que isso?

- Observe que o algoritmo anterior é muito ineficiente porque
    - recalcula muitas vezes os mesmos subproblemas.



Usando a técnica de programação dinâmica,

- que salva resultados dos subproblemas menores numa tabela
    - para evitar recomputação,
- obtemos um algoritmo muito mais eficiente

```
Fib2(n)
    aloque um vetor f[0 .. n]
    f[0] = 0
    f[1] = 1
    para i = 2 até n
        f[i] = f[i - 1] + f[i - 2]
    devolva f[n]
```

À primeira vista poderíamos dizer que esse algoritmo leva tempo linear em n,

- já que ele tem apenas um laço que realiza n - 1 iterações.
- No entanto, olhando com atenção,
    - observamos que não é exatamente o caso.

Isso porque, para números de tamanho médio

- é razoável supor que uma operação aritmética, como soma, leva tempo constante.
- No entanto, números de Fibonacci crescem muito rapidamente,
    - de modo que o custo para somar f[i - 1] + f[i - 2] é proporcional ao número de bits usados para representar estes números.

No caso, o valor v de Fn cresce exponencialmente (v ~= 2^0,697n),

- mas em binário (base 2) usamos lg v bits para representar v.
- Portanto, o número de bits bn para representar Fn é aproximadamente
    - lg (2^0,697n) = 0,697n.
        - Ou seja, da ordem de n.

Como, seguindo os algoritmos de Al Khwarizmi,

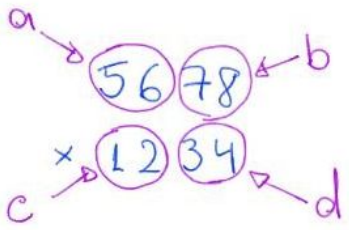
- somar dois números de tamanho n leva tempo linear,
    - temos que o algoritmo anterior utiliza n iterações
    - e em cada iteração é realizada uma soma que custa da ordem de n operações.
        - Portanto, ele leva tempo quadrático,
            - i.e., da ordem de n^2.

Será que podemos fazer melhor?

Vamos considerar o algoritmo de Karatsuba,

- que é um método bastante eficiente para multiplicar números grandes.
- Ele é baseado
    - na técnica de divisão e conquista
    - e no truque de Gauss,
        - originalmente desenvolvido pelo famoso matemático para facilitar operações com números complexos.

Exemplificando as ideias do algoritmo de Karatsuba



$$x = a.10^{\frac{n}{2}} + b$$
$$y = c.10^{\frac{n}{2}} + d$$

$$x * y = (a.10^{\frac{n}{2}} + b) + (c.10^{\frac{n}{2}} + d)$$
$$= a.c.10^{n} + (a.d + bc)10^{\frac{n}{2}} + bd$$

Note que $(a+b)(c+d) = a.c + a.d + bc + bd$ } Truque de Gauss
Logo $ad + bc = (a+b)(c+d) - ac - bd$ }

- multiplique $a.c$ — $*_1$ $\quad 56.12 = 672$
- multiplique $b.d$ — $*_2$ $\quad 78.34 = 2652$
- multiplique $(a+b)(c+d)$ $\quad (56+78)(12+34) = 6164$
- calcule $(a+b)(c+d) - ac - bd$ — $*_3$

$$6164 - 672 - 2652 = 2840$$

$$x * y = *_1 10^{n} + *_3 10^{\frac{n}{2}} + *_2$$

$$6720000 + 284000 + 2652 = 7006652$$

Observe que realizamos três multiplicações para calcular x * y.

- Estas correspondem a chamadas recursivas,
    - portanto, o tempo do algoritmo de Karatsuba segue a recorrência

      T(n) = 3 T(n/2) + cn

    - Pelo Teorema Mestre temos

      T(n) = O(n^log2 3) ~= O(n^1.584)

Note que o algoritmo de Karatsuba, ao menos para números grandes,

- é melhor que o algoritmo tradicional,
    - que é quadrático, i.e., leva tempo proporcional a O(n^2)

Voltando o foco para Fibonacci, vamos tentar uma abordagem matricial

$$\begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}, \text{ ou seja, } F_1 = F_1 \text{ e } F_2 = F_1 + F_0$$

$$\text{Note que } \begin{pmatrix} F_2 \\ F_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^2 \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

$$\text{Generalizando } \begin{pmatrix} F_{n-1} \\ F_n \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

$$\text{Calculando eficientemente } \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{n/2} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{n/2}$$

ou seja, para calcular $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n$ precisamos de $\lg n$ produtos entre matrizes $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

Cada produto matricial pode ser feito usando 4 adições e 8 multiplicações.

- Assim, o total de multiplicações é 8lg n.
- Como sabemos realizar uma multiplicação em tempo O(n^1,584),
    - o tempo total será O(n^1,584 lg n),
        - que é melhor que O(n^2).

Para finalizar, observe quão surpreende descobrir que,

- mesmo um problema tão conhecido quanto a sequência de Fibonacci,
    - ainda guarda segredos quanto à eficiência dos algoritmos para obtê-la.
- Essa é a natureza do projeto de algoritmos,
    - para problemas nas mais diversas áreas.
    - E isto é particularmente válido para problemas NP-Difíceis.

Bibliografia

- Pág. 141 de M.R.T. Dumper, B. Stanley. Cities of the Middle East and North Africa: A Historical Encyclopedia. ABC-CLIO, 2006.
- Prólogo de S. Dasgupta, C.H. Papadimitriou, U. Vazirani. Algorithms. McGraw-Hill Higher Education, 2006.

Título: Algoritmos e Fibonacci

Resumo: Nos dias atuais, algoritmos eficientes são peça chave para o desenvolvimento da humanidade, ainda que normalmente não recebam o crédito devido, já que é comum se atribuir os avanços tecnológicos advindos da computação exclusivamente aos ganhos de performance dos computadores. De fato, não apenas algoritmos ganharam relevância na era da informação, mas eles foram fundamentais para o desenvolvimento da ciência e tecnologia desde muito antes do surgimento dos primeiros computadores. Nesta apresentação veremos uma perspectiva histórica sobre o desenvolvimento de algoritmos, descobriremos o que Leonardo Fibonacci tem a ver com isso, e emprestaremos sua famosa sequência como estudo de caso para vislumbrar a riqueza da área de projeto e análise de algoritmos.