Algoritmos

Pedro Hokama

Fontes

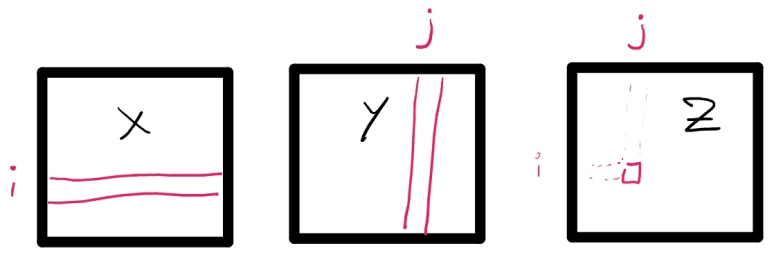
- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest, Ronald L. Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden

Apresentação Baseada:

- Stanford Algorithms
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V`
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6`
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende

O Problema da Multiplicação de matrizes é extremamente importante pois é a essencial para aplicações em diversas áreas:

- Computação Gráfica
- Machine Learning
- Biologia Computacional
- Algoritmos Matemáticos e Físicos.
- e muitas outras.



Problema de Multiplicação de Matrizes

Sejam X e Y duas matrizes quadrada $n \times n$, desejamos obter a matriz $Z = X.Y$ também $n \times n$, tal que

$$Z_{ij} = \sum_{k=1}^{n} X_{ik}.Y_{kj}$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{pmatrix}$$

Para obter cada número Z_{ij} precisamos multiplicar os n pares de elementos formados por X_{ik} e Y_{kj} para $k = 1 \ldots n$. Qual a complexidade para se obter cada Z_{ij}?

a. $\Theta(n)$
b. $\Theta(n \log n)$
c. $\Theta(n^2)$
d. $\Theta(n^3)$

E qual o tempo de execução total do algoritmo para obter todos os $O(n^2)$ valores de Z em função da largura das matrizes n?

a. $\Theta(n \log n)$
b. $\Theta(n^2)$
c. $\Theta(n^3)$
d. $\Theta(n^4)$

- Será possível fazer em tempo menor que cubico?
- Vamos tentar o paradigma de Divisão e Conquista!
- Identificar os passos de Divisão, Conquista e Combinação.
- Será que conseguimos dividir a matriz de alguma forma como no algoritmo de contagem de inversões.

Primeira Ideia

Dividir cada matriz em quadrantes:

$$X = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \text{ e } Y = \begin{pmatrix} E & F \\ G & H \end{pmatrix}$$

A multiplicação de matrizes nesse caso se comporta como se multiplicássemos elementos isolados.

$$Z = X.Y = \begin{pmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{pmatrix}$$

$$XY = \begin{pmatrix} a_{11} & a_{12} & b_{11} & b_{12} \\ a_{21} & a_{22} & b_{21} & b_{22} \\ c_{11} & c_{12} & d_{11} & d_{12} \\ c_{21} & c_{22} & d_{21} & d_{22} \end{pmatrix} \begin{pmatrix} e_{11} & e_{12} & f_{11} & f_{12} \\ e_{21} & e_{22} & f_{21} & f_{22} \\ g_{11} & g_{12} & h_{11} & h_{12} \\ g_{21} & g_{22} & h_{21} & h_{22} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}e_{11} + a_{12}e_{21} + b_{11}g_{11} + b_{12}g_{21} & a_{11}e_{12} + a_{12}e_{22} + b_{11}g_{12} + b_{12}g_{22} & z_{13} & z_{14} \\ a_{21}e_{11} + a_{22}e_{21} + b_{21}g_{11} + b_{22}g_{21} & a_{21}e_{12} + a_{22}e_{22} + b_{21}g_{12} + b_{22}g_{22} & z_{23} & z_{24} \\ z_{31} & z_{32} & z_{33} & z_{34} \\ z_{41} & z_{42} & z_{43} & z_{44} \end{pmatrix}$$

De fato o primeiro quadrante do resultado é:

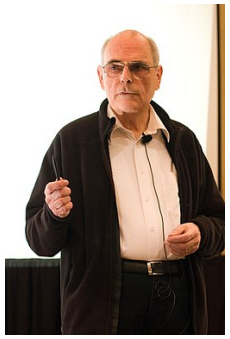
$$AE + BG$$

$$Z = XY = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{pmatrix}$$

- Podemos criar um algoritmo recursivo que calcula os 8 subproblemas de maneira recursiva, cada um com matrizes de dimensão $n/2$ e depois combina com as somas que podem ser feitas em tempo $O(n^2)$.
- A notícia ruim é que esse algoritmo também é $O(n^3)$ igual ao direto. Pfft! 😒
- Talvez aplicar algum *truque* como o de Gauss no algoritmo de Karatsuba?

Volker Strassen

- Volker Strassen, é um matemático alemão nascido em 1936.
- Professor emérito do Departamento de Matemática e Estatística da University of Konstanz.
- Recebeu diversos prêmios por suas contribuições em projeto e análise de algoritmos
- Em 1969 apresentou o primeiro algoritmo para fazer multiplicações de matrizes em tempo de execução inferior a $O(n^3)$.
- Causou um grande *frisson* na época, pois não acreditavam que tal multiplicação pudesse ter tempo subcúbico.



Algoritmo de Strassen - 1969

- Ideia: Reduzir o número de chamadas recursivas!
- Iremos computar apenas 7 chamadas recursivas.
- Faremos as adições e subtrações necessárias. O que ainda vai requisitar $O(n^2)$.

Os 7 produtos a serem computados são:

- $P_1 = A(F - H)$,
- $P_2 = (A + B)H$,
- $P_3 = (C + D)E$,
- $P_4 = D(G - E)$,
- $P_5 = (A + D)(E + H)$,
- $P_6 = (B - D)(G + H)$ e
- $P_7 = (A - C)(E + F)$.

$$X.Y = \begin{pmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{pmatrix} = \begin{pmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{pmatrix}$$

$$\begin{aligned} & P_5 + P_4 - P_2 + P_6 \\ & = (A + D)(E + H) + D(G - E) - (A + B)H + (B - D)(G + H) \\ & = AE + AH + DE + DH + DG - DE - AH - BH + BG + BH - DG - DH \\ & = AE + \cancel{AH} + \cancel{DE} + \cancel{DH} + \cancel{DG} - \cancel{DE} - \cancel{AH} - \cancel{BH} + BG + \cancel{BH} - \cancel{DG} - \cancel{DH} \\ & = AE + BG \end{aligned}$$

- Pelo teorema mestre a complexidade desse algoritmo é $O(n^{\log_2 7})$ (quando está no expoente a base do logaritmo importa sim) que é $\approx O(n^{2.8})$.
- Portando subcubico!
- As constantes no algoritmo de Strassen são bem maiores que as da multiplicação tradicional. Por isso só valem a pena para matrizes a partir de um certo tamanho. Para matrizes pequenas vale mais a pena a multiplicação tradicional.
- Minha sugestão para implementar o Algoritmo de Strassen é que na recursão, a partir de matrizes menores de 32 você use a multiplicação tradicional.
- Existem algoritmos teóricos mais eficientes como o de Coppersmith–Winogra, que é $O(n^{2.375})$ mas não é usado na prática porque só seria melhor que o Strassen para matrizes tão grandes que não são possíveis de serem processadas.