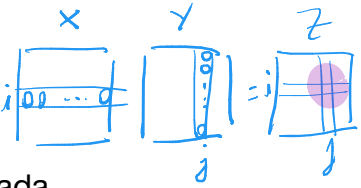
PAA - Aula ~~06~~ - Bônus

Multiplicação de Matrizes, Algoritmo de Strassen



Considere as matrizes quadradas X e Y, com n linhas e n colunas cada.

- No produto $X \cdot Y = Z$ temos que Z também é quadrada e tem dimensão n

Além disso, temos que a posição (i,j) de Z é dada por um somatório de produtos,

- i.e., $Z_{ij} = \sum_{k=1}^{n} X_{ik} \cdot Y_{kj}$

Exemplo:

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

Notem que a fórmula $Z_{ij} = \sum_{k=1}^{n} X_{ik} \cdot Y_{kj}$ sugere um algoritmo

- para multiplicar matrizes com três laços aninhados e eficiência $\Theta(n^3)$
- Isso porque calcular Z_{ij} leva tempo $\Theta(n)$ para qualquer par (i,j)
  - e existem n^2 pares na matriz Z

Obs.: vamos analisar a eficiência dos algoritmos usando n, mas vale lembrar que

- as matrizes tem tamanho n^2. Assim, ler a entrada já leva tempo $\Theta(n^2)$

Algoritmo recursivo simples

A ideia é aplicar o projeto de algoritmos por divisão e conquista

- Dividir: o problema original é dividido em subproblemas do mesmo tipo.
- Conquistar: os subproblemas são resolvidos recursivamente, sendo que os subproblemas pequenos são caso base.
- Combinar: as soluções dos subproblemas são combinadas numa solução do problema original.

Dividir X e Y em quatro matrizes

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

- sendo que $A, \ldots, H$ são matrizes quadradas com $\frac{n}{2}$ linhas e $\frac{n}{2}$ colunas.

Notem que

$$X \cdot Y = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE+BG & AF+CH \\ CE+DG & CF+DH \end{pmatrix}$$

$$X \cdot Y = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)\left(\begin{array}{c|c} E & F \\ \hline G & H \end{array}\right) = \left(\begin{array}{c|c} AE+BG & AF+CH \\ \hline CE+DG & CF+DH \end{array}\right)$$

O produto anterior sugere o seguinte algoritmo recursivo:

- Dados X e Y na entrada
- Se X e Y tem tamanho 1 então devolva o produto dos seus elementos // caso base
- Caso contrário, divida X e Y nas 8 matrizes com metade da dimensão
- Calcule recursivamente os 8 produtos envolvendo essas submatrizes
- Faça as somas necessárias com os resultados dos produtos — $\Theta(n^2)$
- Devolva a matriz resultante

Eficiência dada pela recorrência: $T(n) = 8T(n/2) + c \cdot n^2$

- pois são 8 produtos de submatrizes (chamadas recursivas),
  - cada uma com matrizes que tem metade das linhas (e colunas),
- e o trabalho para combinar as soluções
  - envolve da ordem de n^2 somas.
- Mas, qual é a função de tempo deste algoritmo?

Teorema Mestre

$T(n) = aT(n/b) + c \cdot n^d$

$a = 8 \quad b = 2 \quad d = 2 \;\|\; a = 8 > 2^2 = b^d \Rightarrow T(n) = O(n^{\log_b a}) = O(n^{\log_2 8}) = O(n^3)$

Resolução de recorrência por substituição

$$T(n) = 8\,T(n/2) + c\,n^2$$

$T(n/2) = 8\,T(n/4) + c\,n^2/4 = 2^2$

$T(n/4) = 8\,T(n/8) + c\,n^2/16 = 2^4$

$T(n/8) = 8\,T(n/16) + c\,n^2/64 = 2^6$

$$T(n) = 8\,T(n/2) + c\,n^2 \qquad 2^1 - 1$$

$$= 8(8(T(n/4) + c\,n^2/2^2) + c\,n^2$$

$$= 8^2 T(n/4) + \underbrace{3}_{2^2-1}\,c\,n^2$$

$$= 8^2(8\,T(n/8) + c\,n^2/2^4) + 3\,cn^2$$

$$= 8^3 T(n/8) + \underbrace{7}_{2^3-1}\,cn^2$$

$$= 8^3(8\,T(n/16) + cn^2/2^6) + 7\,cn^2$$

$$= 8^4 T(n/16) + \underbrace{15}_{2^4-1}\,cn^2$$

$$\vdots$$

$$T(n) = 8^j T(n/2^j) + (2^j - 1)\,cn^2$$

no maior j temos $\frac{n}{2^j} = 1 \Rightarrow 2^j = n \Rightarrow j = \lg n$

Assim,

$$T(n) = 2^{3\lg n} T(1) + (2^{\lg n} - 1)\, c.n^2$$

$$= n^3 . c + (n-1) . cn^2$$

$$= 2cn^3 - cn^2$$

Vamos verificar que a fórmula deduzida está correta usando indução matemática.

Caso base: $T(1) = 2c.1^3 - c.1^2 = c$ ✓

H.I.: $T(n') = 2cn'^3 - cn'^2$ p/ $n' < n$

Passo: $T(n) = 8\,T(n/2) + cn^2$

pela H.I. $= 8\,(2.c.(n/2)^3 - c.(n/2)^2) + c.n^2$

$= 8.2.c.n^3/8 - 8.c.n^2/4 + c.n^2$

$= 2cn^3 - cn^2$

Portanto, T(n) = 2cn^3 - cn^2 = O(n^3)

Algoritmo recursivo de Strassen

$$X \cdot Y = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)\left(\begin{array}{c|c} E & F \\ \hline G & H \end{array}\right) = \left(\begin{array}{c|c} AE+BG & AF+CH \\ \hline CE+DG & CF+DH \end{array}\right)$$

Usando uma observação perspicaz e nada óbvia,

- o algoritmo de Strassen precisa de apenas 7 chamadas recursivas.

Considere as seguintes matrizes:

$P_1 = A.(F-H)$ $\quad P_4 = D.(G-E)$

$P_2 = (A+B).H$ $\quad P_5 = (A+D).(E+H)$

$P_3 = (C+D).E$ $\quad P_6 = (B-D).(G+H)$ $\quad P_7 = (A-C).(E+F)$

Verifique que elas podem ser usadas para compor o produto das matrizes e

$$X.Y = \left(\begin{array}{c|c} P_5+P_4-P_2+P_6 & P_1+P_2 \\ \hline P_3+P_4 & P_1+P_5-P_3-P_7 \end{array}\right)$$

Assim, a eficiência do algoritmo é dada pela recorrência $T(n) = 7T(n/2) + c.n^2$

- que tem uma chamada recursiva a menos que o algoritmo anterior.

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

$a = 7 > 2^2 = b^d \Rightarrow T(n) = O\left(n^{\log_b a}\right) = O\left(n^{\log_2 7}\right) \simeq O\left(n^{2,81}\right)$