

PAA - Aula 06.2

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, sabemos 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:

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

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 .

Detalhe: 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

Ideia: aplicar o projeto de algoritmos por divisão e conquista

- Dividir: o problema original é dividido em subproblemas menores 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

$$X = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right) \quad Y = \left(\begin{array}{c|c} E & F \\ \hline G & H \end{array} \right)$$

sendo que A, ..., H são matrizes quadradas com $n/2$ linhas e $n/2$ colunas.

Além disso,

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

Isso 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 contrário, divida X e Y nas 8 matrizes com metade do tamanho
- Calcule recursivamente os 8 produtos envolvendo essas matrizes
- Faça as somas necessárias com os resultados dos produtos (em tempo $O(n^2)$)
- Devolva a matriz resultante

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

- pois são 8 produtos de matrizes/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?

Resolução de recorrência por substituição

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

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

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

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

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

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

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

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

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

$$= 8^3(8 T(n/16) + c n^2/2^6) + 7 c n^2$$

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

⋮

$$T(n) = 8^j T(n/2^j) + (2^j - 1) c n^2$$

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

Assim,

$$\begin{aligned} T(n) &= 2^{\lg n} T(1) + (2^{\lg n} - 1) c \cdot n^2 \\ &= n^3 \cdot c + (n - 1) \cdot c n^2 \\ &= \boxed{2cn^3 - cn^2} \end{aligned}$$

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

$$\text{Caso base: } T(1) = 2c \cdot 1^3 - c \cdot 1^2 = c \quad \checkmark$$

$$\text{H.I.: } T(n') = 2cn'^3 - cn'^2 \quad p/n' < n$$

$$\begin{aligned} \text{Passo: } T(n) &= 8T(n/2) + cn^2 \\ \text{pela H.I.} \quad &= 8(2c \cdot (n/2)^3 - c \cdot (n/2)^2) + cn^2 \\ &= \cancel{8} \cdot 2 \cdot c \cdot n^3 / \cancel{8} - \cancel{8}^2 \cdot c \cdot n^2 / 4 + cn^2 \\ &= \boxed{2cn^3 - cn^2} \end{aligned}$$

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

Algoritmo recursivo de Strassen

Usando uma observação perspicaz e nada óbvia,

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

Considere as seguintes matrizes:

$$\begin{array}{lll} P_1 = A(F-H) & P_4 = D(G-E) & P_7 = (A-C)(E+F) \\ P_2 = (A+B)H & P_5 = (A+D)(E+H) & \\ P_3 = (C+D)E & P_6 = (B-D)(G+H) & \end{array}$$

Verifique que elas podem ser usadas para compor a multiplicação das matrizes:

$$X.Y = \begin{pmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{pmatrix}$$

Assim, sua eficiência é dada pela seguinte recorrência

$$T(n) = 7 T(n/2) + c * n^2$$

- que tem uma chamada recursiva a menos que a do algoritmo anterior.
- Mas, qual é a função de tempo deste algoritmo?