Multiplicación de matrices

- Sean Y y Z dos matrices de $n \times n$.
- El algoritmo común y corriente para calcular el producto $X = YZ$ hace $O(n^3)$ operaciones (multiplicaciones y sumas).
- Si cada operación se tarda $O(1)$ entonces este algoritmo se tarda $O(n^3)$.
- ¿Podemos hacerlo más rápido?

División de las matrices

- Dividamos cada una de X, Y y Z en cuatro matrices:

$$X = \begin{pmatrix} I & J \\ K & L \end{pmatrix} Y = \begin{pmatrix} A & B \\ C & D \end{pmatrix} Z = \begin{pmatrix} E & F \\ G & H \end{pmatrix}.$$

- Entonces tenemos que

$$\begin{aligned} I &= AE + BG \\ J &= AF + BH \\ K &= CE + DG \\ L &= CF + DH. \end{aligned}$$

Análisis

- Sea $T(n)$ el tiempo que nos toma multiplicar dos matrices de $n \times n$ con esta idea.
- Podemos suponer que $n = 2^k$.
- Entonces

$$T(n) = \begin{cases} c & \text{si } n = 1 \\ 8T(n/2) + dn^2 & \text{si } n > 1 \end{cases}$$

Sustitución

- Sustituyendo obtenemos

$$\begin{aligned} T(n) &= 8T(n/2) + dn^2 \\ &= 8(8T(n/4) + d(n/2)^2) + dn^2 \\ &= 8^2 T(n/2^2) + 2^1 dn^2 + 2^0 dn^2 \\ &= 8^i T(n/2^i) + dn^2 \sum_{j=0}^{i-1} 2^j \\ &= 8^k T(1) + dn^2(2^k - 1) \\ &= cn^3 + dn^2(n-1). \end{aligned}$$

- Por lo tanto $T(n) \in O(n^3)$, lo cual no es una mejora.

Algoritmo de Strassen

- $M_1 = (A + C)(E + F)$.
- $M_2 = (B + D)(G + H)$.
- $M_3 = (A - D)(E + H)$.
- $M_4 = A(F - H)$.
- $M_5 = (C + D)E$.
- $M_6 = (A + B)H$.
- $M_7 = D(G - E)$.
- $I = M_2 + M_3 - M_6 - M_7$.
- $J = M_4 + M_6$.
- $K = M_5 + M_7$.
- $L = M_1 - M_3 - M_4 - M_5$.

Análisis de correctitud

- Se puede verificar que los cálculos anteriores funcionan.
- Por ejemplo:

$$\begin{aligned} J &= M_4 + M_6 \\ &= A(F - H) + (A + B)H \\ &= AF - AH + AH + BH \\ &= AF + BH. \end{aligned}$$

Análisis

- Sea $T(n)$ el tiempo que nos toma multiplicar dos matrices de $n \times n$ con el algoritmo de Strassen.
- Podemos suponer que $n = 2^k$.
- Entonces

$$T(n) = \begin{cases} c & \text{si } n = 1 \\ 7T(n/2) + dn^2 & \text{si } n > 1 \end{cases}$$

Sustitución

- Sustituyendo obtenemos

$$\begin{aligned} T(n) &= 7T(n/2) + dn^2 \\ &= 7(7T(n/4) + d(n/2)^2) + dn^2 \\ &= 7^2 T(n/2^2) + 7^1 dn^2/4^1 + 7^0 dn^2/4^0 \\ &= 7^i T(n/2^i) + dn^2 \sum_{j=0}^{i-1} (7/4)^j \\ &= 7^k T(1) + dn^2 \frac{(7/4)^k - 1}{7/4 - 1} \\ &= cn^{\log_2 7} + \frac{4}{3} d(n^{\log_2 7} - n^2). \end{aligned}$$

- Por lo tanto $T(n) \in O(n^{\log_2 7}) \approx O(n^{2.807})$, lo cual sí es una mejora.

Algoritmos de multiplicación de matrices

- ¿Qué tan rápido se puede multiplicar dos matrices de $n \times n$?
- El algoritmo obvio tarda $O(n^3)$.
- El algoritmo de Strassen tarda $O(n^{\log_2 7})$ (descubierto en 1971).
- En 1990 Coppersmith y Winograd descubrieron un algoritmo que tarda $O(n^{2.376})$.
- Ningún algoritmo puede tardar menos de $O(n^2)$.
- Muchos investigadores creen que sí existen algoritmos que tardan $O(n^2)$.