Matrix Multiplication $n^2 \log n^2$

Mohamed Hamlil

January 2024

1 resume de la methode

La méthode suivie est de transformer nos matrices A et B en polynôme 2D, faire leur multiplication, puis leur somme, et enfin retransformer le polynôme C en matrice.

2 introduction

Soient A et B deux matrices carrées de taille n où $A \cdot B = C$,

$$C[i,j] = \sum_{t=0}^{n} A[i,t] \cdot B[t,j]$$

N'importe quelle multiplication de deux polynômes de degré n a une complexité temporelle de $n\log(n)$.
Les coefficients des matrices sont les images du polynôme

$$A(i,j) = \sum_{k=0}^{n} \left[\sum_{l=0}^{n} a_{kl} \cdot (i^k) \cdot (j^l) \right]$$

aux coordonnées de la forme
(w$^0, w^0)(w^0, w^1)$........(w^0, w^n)
$(w^1, w^0)(w^1, w^1)$........(w^1, w^n)
$(w^j, w^0)(w^j, w^1)(w^j, w^i)(w^j, w^n)$
$(w^n, w^0)(w^n, w^1)(w^n, w^i)(w^n, w^n)$
$pour w = e^{i\frac{2\pi}{n}}$.
de sorte que la matrice A=
A(w$^0, w^0)A(w^0, w^1) \dots A(w^0, w^n)$
$A(w^1, w^0)A(w^1, w^1) \dots A(w^1, w^n)$
$A(w^j, w^0)A(w^j, w^1)A(w^j, w^i)A(w^j, w^n)$
$A(w^n, w^0)A(w^n, w^1)A(w^n, w^i)A(w^n, w^n)$

Où chaque coefficient de la matrice A, $A[i,j] = A(w^j, w^i)$ (on notera $A[i,j]$ les coefficients de la matrice A et $A(i,j)$ le polynôme qui lui correspond).

3 Algorithm

On transforme les matrices en polynômes, ce qui prend $(n^2) \cdot \log(n)$ opérations (2D Inverse Fast Fourier Transform),

$$A(i,j) = \sum_{k=0}^{n} \left[\sum_{l=0}^{n} a_{kl} \cdot (i^k) \cdot (j^l) \right]$$

$$B(i,j) = \sum_{k=0}^{n} \left[\sum_{l=0}^{n} b_{kl} \cdot (i^k) \cdot (j^l) \right]$$

On regroupe les polynômes selon la variable i et j,

$$A(i,t) = \sum_{k=0}^{n} a'_k \cdot (i^k)$$

$$B(t,j) = \sum_{k=0}^{n} b'_k \cdot (j^k)$$

avec a'_k et b'_k des polynômes de sorte

$$a'_k = \sum_{l=0}^{n} a_l \cdot (t^l)$$

$$b'_k = \sum_{l=0}^{n} b_l \cdot (t^l)$$

Ce qui prendra n^2 opérations.
On sait que

$$C[i,j] = \sum_{t=0}^{n} A[i,t] \cdot B[t,j]$$

Et étant donné que $C[i,j] = C(w^j, w^i)$ Donc,

$$C(i,j) = \sum_{t=0}^{n} A(i,w^t) \cdot B(w^t,j)$$

$$C(i,j) = \sum_{t=0}^{n} \left(\sum_{k=0}^{n} a'_{kt} \cdot (i^k) \right) \cdot \left(\sum_{k=0}^{n} b'_{kt} \cdot (j^k) \right)$$

avec

$$a'_{kt} = \sum_{l=0}^{n} a_l \cdot (w^{t.l})$$

$$b'_{kt} = \sum_{l=0}^{n} b_l \cdot (w^{t.l})$$

donc

$$C(i,j) = \sum_{t=0}^{n} \left(\sum_{k=0}^{n} \left(\sum_{l=0}^{n} a_{kl} \cdot w^{t.l} \right) \cdot (i^k) \right) \cdot \left(\sum_{k=0}^{n} \left(\sum_{l=0}^{n} b_{kl} \cdot w^{t.l} \right) \cdot (j^k) \right)$$

pour

$$A_{kt} = \sum_{l=0}^{n} a_{kl} \cdot (w^{t.l})$$

$$B_{kt} = \sum_{l=0}^{n} b_{kl} \cdot (w^{t.l})$$

$$C(i,j) = \sum_{t=0}^{n} \left(\sum_{k=0}^{n} A_{kt} \cdot (i^k) \right) \cdot \left(\sum_{k=0}^{n} B_{kt} \cdot (j^k) \right)$$

Pour calculer les coefficients des i^k:

$$A_{k0} = \sum_{l=0}^{n} a_{kl} \cdot (w^{0.l})$$

$$A_{k1} = \sum_{l=0}^{n} a_{kl} \cdot (w^{1.l})$$

.
.

$$A_{kt} = \sum_{l=0}^{n} a_{kl} \cdot (w^{t.l})$$

.
.

$$A_{kn} = \sum_{l=0}^{n} a_{kl} \cdot (w^{n.l})$$

ce qui est équivalent à multiplier la matrice Vandermonde où les x_i sont les racines n-ièmes de l'unité. FFT calcule le produit de cette matrice avec le vecteur $[a_{k0}, a_{k1}, ..., a_{kl}, ..a_{kn}]$ en un temps de $O(n \log n^2)$ pour chaque k, donc $n^2 \log n^2$ pour calculer tous les A_k, et on fera de même pour B_k. Donc, on aura

$$C(i,j) = \sum_{t=0}^{n} A_t(i) \cdot B_t(j)$$

avec

$$A_t(i) = \sum_{k=0}^{n} a_{kt} \cdot (i^k)$$

$$B_t(j) = \sum_{k=0}^{n} b_{kt} \cdot (j^k)$$

Où a_{kt} et b_{kt} sont des constantes
Maintenant, on utilise l'algorithme HL. "'

4 Algorithme HL

soit

$$C(x,y) = \sum_{k=0}^{n} (A_k(x)) \cdot (B_k(y))$$

on sait que tout polynome est une combinaison de degree paire et impaire donc

$$(A_k x) = P(x) + I(x)$$

$$(B_k y) = P(y) + I(y)$$

donc

$$C(x,y) = \sum_{k=0}^{n} (P_k(x) + I_k(x)) \cdot (P_k(y) + I_k(y))$$

$$C(x,y) = \sum_{k=0}^{n} P_k(x) \cdot P_k(y) + P_k(x) \cdot I_k(y) + I_k(x) \cdot P_k(y) + I_k(x) \cdot I_k(y)$$

$$C(x,y) = \sum_{k=0}^{n} P_k(x) \cdot P_k(y) + x \cdot P_k(x) \cdot P_k(y) + y \cdot P_k(x) \cdot P_k(y) + xy \cdot P_k(x) \cdot P_k(y)$$

$$C(x,y) = \sum_{k=0}^{n} A'_k(x) \cdot B'_k(y) + x \cdot A'_k(x) \cdot B'_k(y) + y \cdot A'_k(x) \cdot B'_k(y) + xy \cdot A'_k(x) \cdot B'_k(y)$$

$$C(x,y) = \sum_{k=0}^{n} A'_k(x) \cdot B'_k(y) + x \cdot \sum_{k=0}^{n} A'_k(x) \cdot B'_k(y) + y \cdot \sum_{k=0}^{n} A'_k(x) \cdot B'_k(y) + xy \cdot \sum_{k=0}^{n} A'_k(x) \cdot B'_k(y)$$

$$C(x,y) = C'(x,y) + x \cdot C'(x,y) + y \cdot C'(x,y) + xy \cdot C'(x,y)$$

on appliquant la meme strategie que fft

$$C(x,y) = C'(x,y) + x \cdot C'(x,y) + y \cdot C'(x,y) + xy \cdot C'(x,y)$$

$$C(x,-y) = C'(x,y) + x \cdot C'(x,y) - y \cdot C'(x,y) - xy \cdot C'(x,y)$$

$$C(-x,y) = C'(x,y) - x \cdot C'(x,y) + y \cdot C'(x,y) - xy \cdot C'(x,y)$$

$$C(-x,-y) = C'(x,y) - x \cdot C'(x,y) - y \cdot C'(x,y) + xy \cdot C'(x,y)$$

pour generer n**2 points
selon the master theorem :l algorithm est de temps polynomial n$^2 logn$

5 contact

hamlilm@yahoo.fr
mohamed.hamlil@etu.univ-grenoble-alpes.fr