ALGORITHME DE STRASSEN

25 octobre 2020

.

1 Principe

L'algorithme de Strassen peut être vu comme une application de la bonne vieille technique diviser-pour-régner. Supposons qu'on veuille calculer le produit **C = AB**, où A, B et C sont des matrices **n** × **n**. En supposant que n soit une puissance exacte de 2, on divise A, B et C chacune en quatre matrices n/2 × n/2, ce qui fait réécrire l'équation C = AB de la manière suivante :

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

Avec :
r = ae + bg ,
s = af + bh ,
t = ce + dg ,
u = cf + dh .

Chacune de ces quatre équations spécifie deux multiplications de matrices n/2×n/2 et l'addition de leurs produits n/2×n/2. Ces équations permettent de définir une stratégie diviser-pour-régner immédiate, qui donne la récurrence suivante pour le temps T(n) requis pour multiplier deux matrices n × n :

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

Malheureusement, la récurrence (28.13) a pour solution $T(n) = \Theta(n^3)$, et cette méthode n'est donc pas plus rapide que la méthode ordinaire

Strassen a découvert une approche récursive différente, qui ne demande que 7 multiplications récursives de matrices n/2×n/2, et $\Theta(n^2)$ additions et soustractions scalaires, ce qui aboutit à la récurrence :

$$T(n) = 7T(n/2) + \ominus(n^2)$$

$$T(n) = \ominus(n^{lg7})$$

La méthode de Strassen est composée de quatre étapes :
1) Diviser les matrices d'entrée **A** et **B** en sous-matrices n/2 × n/2, comme dans la methode DPR
2) A l'aide de $\ominus(n^2)$ additions et soustractions scalaires, calculer 14 matrices de dimension n/2 × n/2.
3) Calculer récursivement les sept produits de matrices $P_i = A_i B_i$ pour $i = 1, 2, \ldots, 7$.
4) Calculer les sous-matrices désirées r, s, t, u de la matrice résultat C en additionnant et/ou soustrayant diverses combinaisons des matrices Pi, à l'aide de $\ominus(n^2)$ additions et soustractions scalaires seulement

2 Algorithme

```
Function Strassen(A, B, C, n){

        if n == 2 do
                /* On a deux matrice de taille 2*2 (le cas de base) */
                p1 = A[1][1] * ( B[1][2] - B[2][2] )
                p2 = (A[1][1] + A[1][2]) * B[2][2]
                p3 = (A[2][1] + A[2][2]) * B[1][1]
                p4 = A[2][2]*(B[2][1] - B[1][1])
                p5 = (B[1][1] + B[2][2]) * (A[1][1] + A[2][2])
                p6 = (B[2][1] + B[2][2]) * (B[1][2] - B[2][2])
                p7 = (A[1][1] - A[2][1]) * (B[1][1] - B[1][2])
```

```
        C[1][1] = p5 + p4 - p3 - p7
        C[1][2] = p1 + p2
        C[2][1] = p3 + p4
        C[2][2] = p5 + p1 -p3 - p7

        return C

else
        L = C = 0
        ml = mc =  1
        nl = nc = 1

        for i = 1 to n do

                for j=ml to rl do
                        L = L + 1
                        C = 0
                        for j = mc to rc
                                C = C + 1
                                ar[L][C] = A[i][j]
                                br[L][C] = B[i][j]
                        endfor
                enfor

                if( mc == ml == 1 && rc == rl == n/2 ) do
                for k=1 to n/2 do
                    for m=1 to n/2 do
                        a[k][m] = ar[k][m]
                        e[k][m] = br[k][m]
                endfor
            endfor
                 mc = n/2 + 1
                 rc = n
              endif

                if(ml == 1 && rl == n/2 && mc == n/2 + 1 && rc = n) do
                        for k=1 to n/2 do
                                for m=1 to n/2 do
                                        b[k][m] = ar[k][m]
                                        f[k][m] = br[k][m]
                                endfor
                        endfor
                        ml = n/2 + 1
                        rl = n
                endif

                if( mc == ml == n/2 && rc == rl == n ) do
                        for k=1 to n/2 do
                                for m=1 to n/2 do
                                        d[k][m] = ar[k][m]
                                        h[k][m] = br[k][m]
                                endfor
                        endfor
                        mc = 1
                        rc = n/2
                endif

                if(mc = 1 && rc == n/2 && ml == n/2 && rl == n) do
                        for k=1 to n/2 do
                                for m=1 to n/2 do
```

```
                                    c[k][m] = ar[k][m]
                                    g[k][m] = br[k][m]
                            endfor
                    endfor
            endif

        endfor

    /*  On vas copier les elements d'une matrice dans une autre:
        l'appel de strassen renvoi une matrice dont les elements
        dans seront copiés les matrices Mi */

        M1 = strassen(a, f-h, c, n/2)
        M2 = strassen(a+b, h, n/2)
        M5 = strassen(a+b, e+h, c, n/2)
        M4 = strassen(d, g-e, c, n/2)
        M3 = strassen(c+d, e, c, n/2)
        M6 = strassen(b-d, g+h, c, n/2)
        M7 = strassen(a-c, e-f)

        /* Addition et Soustraction de matrice */

        S = M1 + M2
        T = M3 + M4
        R = M5 + M4 - M2 + M6
        U = P5 + P1 - P3 - P7

        // On construit la matrice de retour
        for i = 1 to n/2 do
                for j = 1 to n/2 do
                        C[i][j] = R[i][j]
                enfor
        enfor

        for i = 1  to n/2 do
                for j = n/2 + 1 to n do
                        C[i][j] = S[i][j]
                enfor
        enfor

        for i = n/2 + 1 to n do
                for j = 1 to n/2 do
                        C[i][j] = T[i][j]
                enfor
        enfor

        for i = n/2 + 1 to n do
                for j = n/2 + 1 to n do
                        C[i][j] = U[i][j]
                enfor
        enfor

        return C

 endif

}
```

3 Complexité

Comme l'on a vue dans la section precedante, on :

$$T(n) = 7T(n/2) + \ominus(n^2)$$

$$T(n) = \ominus(n^{lg7})$$

En utilisant des techniques avancées qui sortent du cour, on peut en fait multiplier des matrices n $\times$ n plus rapidement qu'en temps $\ominus(n^{lg7})$. Le meilleur majorant connu à ce jour vaut environ $\bigcirc(n^{2,376})$.