ALGORITHME DE KARATSUBA

25 octobre 2020

Problème

Le problème ici est de pouvoir faire le produit de deux polynome en un temps plus court que l'argorithme naif $\Theta(n^2)$.

Principe

Soit deux polynome A(X) et B(X) de degré 2n, on pose

$$A(X) = A_1(X) \cdot X^n + A_0(X) \quad et \quad B(X) = B_1(X) \cdot X^n + B_0(X) \quad avec \quad degA_i, degB_i \leq n$$

On calcule :

$$A \cdot B = A_1B_1 \cdot X^{2n} + ((A_0 + A_1)(B_0 + B_1) - A_0B_0 - A_1B_1) \cdot X^n + A_0B_0$$

Ainsi nous avons 3 multiplications et 4 additions (au lieu de 4 multiplication et une addition dans le cas se l'algorithme naif), mais $\mathbf{A_i}$ et $\mathbf{B_i}$, sont eux aussi des polynomes sur lesquels on va refaire la meme procédure jusqu'a arriver au cas de bases, ou le degré du polynome est 1 ou 0.

Degré 1

Le calcul devient :

$$(aX + b)(cX + d) = acX^2 + (u - ac - bd)X + bd \quad ou \quad u = (a + b)(c + d)$$

Degré 0

On a une simple multiplication :

$$a \cdot b = ab$$

Algorithme

On suppose que les polynomes sont definit comme des tableau de coeficient stocker par ordre croissant de degré.

```
fonc transforme(réel P[1...n+1], entier k)
debut
    // transforme un polynome de degré n en n+k
    m ← n+k+1
    réel S[1..m]
    pour i allant de 1 à k faire
        S[i] ← 0
    fpour
    pour i allant de 1 à n+1 faire
        S[k+i] ← P[i]
    fpour
    retourner S
fin

fonc somme( réel P[1...n], réel Q[1...m])
    debut
        k ← min(m,n)
        t ← max(m,n)
        réel S[1...t], reste[];
        pour i allant de i à k faire
            S[i] ← P[i]+Q[i]
        fpour
        si m > n alors
            reste ← Q
        sinon
            reste ← P
        fsi
        pour i allant de k à t faire
            S[i] ← reste[i]
        fpour
        retourner S
    fin

fonc moins( réel P[1...n], réel Q[1...m])
    debut
        réel S[1...m];
        pour i allant de 1 à m faire
            S[i] ← -1*Q[i]
        fpour
        retourner somme(P,S)
    fin
```

```
algorithme karatsuba(réel P[1...n+1], réel Q[1...m+1])
    entier lenp, degp, lenq, degq, i;
    réel, a[], b[], c[], d[], ac, bd, u;
    debut
        degp ← n
        degq ← m
        // on les met au meme degré (ça peut arriver juste à la première exécution)
        si degp > degq alors
            tanque degp <> degq faire
                degq ← degq+1
                Q[degq] ← 0
            ftanque
        sinon si degp < degq
            tanque degp <> degq faire
                degp ← degp+1
                P[degp] ← 0
            ftanque
        fsi
        lenp ← degp+1
        lenq ← degq+1
        si degp = 0 alors
            réel S[1..1]
            S[1] ← P[0]*Q[0]
        sinon si degp = 1 alors
            réel S[1..3]
            ac ← P[1]*Q[1]
            bd ← P[0]*Q[0]
            u ← (P[0]+P[1])*(Q[0]+Q[1])
            S[1] ← bd
            S[2] ← u-ac-bd
            S[3] ← ac
        sinon
            i ← degp div 2
            a ← P[i+1...lenp]
            c ← Q[i+1...lenq]
            b ← P[0...i]
            d ← Q[0...i]
            ac ← karatsuba(a,c)
            bd ← karatsuba(b,d)
            u ← karatsuba(somme(a,b),somme(c,d))
            S ←somme(somme(transforme(ac,i+i),transforme(moins(moins(u,ac),bd),i)),bd)
        fsi
        retourner S
    fin
```

Complexité

Pour l'exemple vu plus au on à T(n) :

$$T(2n) = 3T(n) + 8n$$

Qui nous donne une compléxité de $\Theta(n^{\log_2 3})$