Multiplication rapide de polynômes et de matrices

Tahina Ramananandro
ramanana@clipper.ens.fr

Lycée Louis-le-Grand – MPSI CAML
9 avril 2008

Ce TP propose deux algorithmes simples de multiplication rapide à base de diviser pour régner.

1 Polynômes

On représente un polynôme sous la forme de la liste de ses coefficients, la tête correspondant au degré zéro :

```
type polynome = float list;;
```

Ainsi, la liste `3::4::5::[]` correspond au polynôme $3 + 4X + 5X^2$.

1.1 Opérations préliminaires

Écrire des fonctions récursives :

- `ajout : polynome -> polynome -> polynome`
- `mulscal : float -> polynome -> polynome`
- `soustr : polynome -> polynome -> polynome`

qui calculent respectivement la somme de deux polynômes, la multiplication d'un polynôme par un scalaire, et la différence de deux polynômes.

1.2 Multiplication naïve

La multiplication de deux polynômes $P = \sum_k p_k X_k$ et $Q = \sum_k q_k X_k$ est donnée par la formule :

$$PQ = \sum_k \sum_{i+j=k} p_i q_j X^k$$

Mais la représentation des polynômes sous forme de liste rend cette formule rédhibitoire (car le simple accès à un coefficient quelconque du polynôme se fait en temps linéaire).

Cependant, on sait accéder en temps constant au coefficient de degré zéro. On peut donc écrire :

$P = P_1 X + p_0$

$Q = Q_1 X + p_0$

avec Q_1, P_1 deux polynômes.

À l'aide du développement de $(P_1 X + p_0)(Q_1 X + q_0)$, écrire une fonction récursive :

```
mult : polynome -> polynome -> polynome
```

qui renvoie le produit de deux polynômes.

1.3 Algorithme de Karatsuba (1960)

Le développement ci-dessus est trop naïf en termes de complexité. On s'intéresse donc à une autre façon de décomposer un polynôme.

Pour tout entier k, si on a $P = RX^k + S$ et $Q = TX^k + U$, alors le produit s'écrit :

$$PQ = X^{2k} RT + X^k (RU + ST) + SU$$

En négligeant la multiplication par un X^j, cette méthode naïve requiert quatre multiplications de polynômes plus petits (si on impose les degrés de S et U strictement inférieurs à k).

Or, Karatsuba (1960) a remarqué que, si on écrit ce produit sous la forme :

$$PQ = X^{2k}RT + X^k\left((R+S)(T+U) - (RT+SU)\right) + SU$$

alors le produit ne requiert que trois multiplications (au lieu de quatre naïvement) de polynômes plus petits : RT, SU et (R+S)(T+U).

Cette remarque ouvre la voie à une implémentation en *diviser pour régner*.

Écrire une fonction récursive :

```
karatsuba : int -> polynome -> polynome -> polynome
```

telle que `karatsuba k p q` renvoie le produit de deux polynômes suivant une méthode diviser pour régner avec le paramètre k pour la décomposition de deux polynômes. On pourra supposer que k est une puissance de 2.

2 Matrices

On représente les matrices sous la forme de vecteurs de vecteurs.

```
type matrice = float array array;;
```

Ainsi, la matrice $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ sera représentée sous la forme `[| [| 1; 2 |] ; [| 3; 4 |] |]`.

Une matrice nulle se crée par l'appel `make_matrix lignes colonnes 0.`.

2.1 Opérations préliminaires

Écrire des fonctions itératives :

- `ajout : matrice -> matrice -> matrice`
- `mulscal : float -> matrice -> matrice`
- `soustr : matrice -> matrice -> matrice`

qui calculent respectivement la somme de deux matrices, la multiplication d'une matrice par un scalaire, et la différence de deux matrices. On supposera que les matrices sont bien formées (toutes les lignes ont la même longueur), et que les dimensions sont compatibles.

2.2 Multiplication naïve

Si $A = (a_{i,j})$ et $B = (b_{i,j})$ sont deux matrices, alors leur produit $AB = (\sum_k a_{i,k}b_{k,j})$.

Écrire une fonction itérative qui calcule le produit de deux matrices (supposées bien formées et de dimensions compatibles) suivant cette formule.

2.3 Multiplication par blocs

Si $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ et $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$ sont des matrices écrites par blocs (supposons-les carrées de dimension une puissance de 2, tous les blocs ayant même dimension), alors le produit AB peut également se calculer par blocs « comme si les blocs étaient des scalaires » :

$$AB = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

Ceci ouvre la voie à un algorithme *a priori* de type diviser pour régner.

Écrire une fonction récursive `mulblocs` telle que :

```
mulblocs a b c dim la ca lb cb lc cc
```

écrive, dans le bloc de dimension dim dont le premier coefficient est en ligne lc, colonne cc, le produit des blocs de a et b de dimension dim, et dont les premiers coefficients.

L'idée est que initialement la matrice c est supposée être nulle, mais qu'on y ajoute progressivement les résultats des sous-produits par blocs.

Montrer que la multiplication par blocs est naïve (i.e. en $O(n^3)$ multiplications scalaires).

2.4 Algorithme de Strassen (1969)

Montrer que si on pose :

$$\begin{aligned} M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\ M_2 &= (A_{21} + A_{22}) B_{11} \\ M_3 &= A_{11}(B_{12} - B_{22}) \\ M_4 &= A_{22}(B_{21} - B_{11}) \\ M_5 &= (A_{11} + A_{12}) B_{22} \\ M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\ M_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \end{aligned}$$

alors, $\begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 - M_6 \end{bmatrix} = AB.$

Écrire alors une fonction récursive qui calcule le produit de deux matrices carrées de dimensions une puissance de 2. Contrairement à l'algorithme précédent, on ne s'attachera pas à ce que les opérations se fassent sur place (i.e. des matrices annexes pourront être créées).

On sera éventuellement amené à écrire une fonction de copie de blocs de matrices, dont on négligera le coût.

On montre que le coût :

$$T(n) = 7T(n/2) + O\left(n^2\right)$$

en nombre de multiplications scalaires.

En déduire que cet algorithme est en complexité $O\left(n^{\log_2 7}\right)$.