Multiplication rapide

Vincent Pilaud

Mai 2004

Les algorithmes de multiplication rapide sont les piliers de nombreux programmes actuels : leur rôle central dans des problèmes divers et variés en fait un enjeu majeur de l'algorithmique. Nous étudions dans ce problème quelques exemples de méthodes algorithmiques plus ou moins efficaces pour la multiplication.

1 Description du problème

Problème

Soient n et m deux entiers (très grands). Le but est de calculer rapidement leur produit nm.

Bien que la question puisse paraître toute simple, nous allons voir qu'il existe de nombreuses façons de la résoudre, plus ou moins astucieuses et plus ou moins efficaces.

Notations

On notera :

$n = \sum_{k=0}^{p-1} 2^k a_k + 2^p = a_0 + 2a_1 + \ldots + 2^{p-1}a_{p-1} + 2^p = \overline{1a_{p-1}\ldots a_1 a_0}^2$ avec $p \in \mathbb{N}$ et $\forall i \in \{0;\ldots;p-1\}, a_i \in \{0;1\}$ la décomposition en base 2 de n.

On verra plus tard que $p = E[\frac{\ln n}{\ln 2}]$ (c'est ce qui nous servira a retrouver p en sachant n dans les programmes suivants).

$m = \sum_{k=0}^{q-1} 2^k b_k + 2^q = b_0 + 2b_1 + \ldots + 2^{q-1}b_{q-1} + 2^q = \overline{1b_{q-1}\ldots b_1 b_0}^2$ avec $q \in \mathbb{N}$ et $\forall i \in \{0;\ldots;q-1\}, b_i \in \{0;1\}$ la décomposition en base 2 de m.

Hypothèses

On supposera que les entiers sont donnés en base 2 (c'est le cas qui nous intéresse pour les machines).

2 Quelques notions utiles pour le problème

2.1 Logarithme Néperien

On admet l'existence d'une fonction continue strictement croissante $\ln : \begin{array}{c} \mathbb{R}^{+*} \longrightarrow \mathbb{R} \\ t \longmapsto \ln(t) \end{array}$ telle que :

$$\begin{cases} \ln(1) = 0 \\ \forall a, b \in \mathbb{R}^{+*}, \ln(ab) = \ln(a) + \ln(b) \end{cases}$$

Question 1. *Montrer que* $\forall a \in \mathbb{R}, \forall x \in \mathbb{K}, \quad \ln(a^x) = x\ln(a)$
pour $\mathbb{K} = \mathbb{N}$*, puis* $\mathbb{K} = \mathbb{Z}$*, puis* $\mathbb{K} = \mathbb{Q}$*, puis par densité pour* $\mathbb{K} = \mathbb{R}$*.*

Montrer que la fonction ln *est positive sur* $[1;+\infty[$ *et négative sur* $]0;1[$*.*

Montrer que si $n = \sum_{k=0}^{p-1} 2^k a_k + 2^p$*, alors* $p = E[\frac{\ln n}{\ln 2}]$*.*

Correction 1.

1. Montrons par récurrence que $\forall a \in \mathbb{R}, \forall x \in \mathbb{N}, \quad \ln(a^x) = x\ln(a)$:
   – initialisation : $\ln(a^0) = \ln(1) = 0$ par hypothèse.

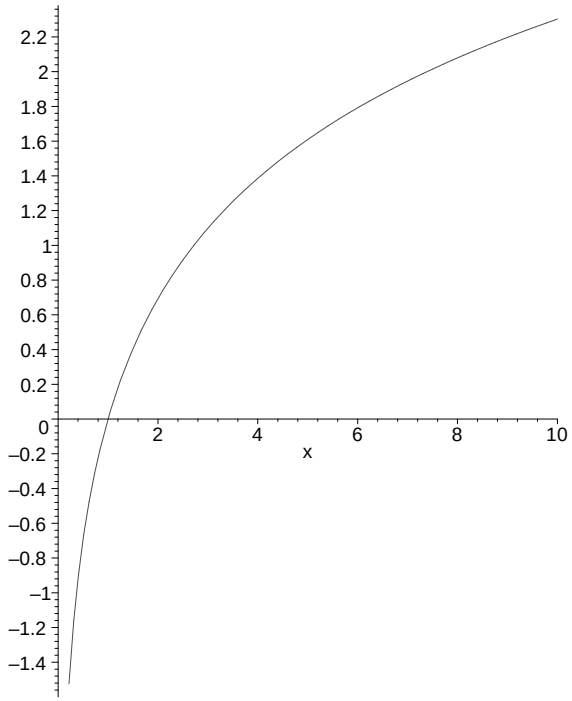
– transmission : soit $n \in \mathbb{N}$ et $a \in \mathbb{R}$. On suppose que $\ln(a^n) = n\ln(a)$. Alors $\ln(a^{n+1}) = \ln(a^n a) = \ln(a^n) + \ln(a) = n\ln(a) + \ln(a) = (n+1)\ln(a)$.

2. On sait que : $\forall n \in \mathbb{N}, \quad 0 = \ln(1) = \ln(a^{-n}a^n) = \ln(a^{-n}) + \ln(a^n) = \ln(a^{-n}) + n\ln(a)$ d'où $\ln(a^{-n}) = -n\ln(a)$ et on a le résultat pour $\mathbb{K} = \mathbb{Z}$.
3. On sait que : $\forall p \in \mathbb{N}, q \in \mathbb{N}^*, \quad p\ln(a) = \ln(a^p) = \ln(a^{p\frac{q}{q}}) = \ln((a^{\frac{p}{q}})^q) = q\ln(a^{\frac{p}{q}})$ d'où $\ln(a^{\frac{p}{q}}) = \frac{p}{q}\ln(a)$.
4. On sait que $\mathbb{Q}$ est dense dans $\mathbb{R}$. Soit $x \in \mathbb{R}$ et $(y_n)_{n\in\mathbb{N}}$ une suite de rationnels qui convergent vers x. Alors par continuité des fonctions ln et $x \mapsto a^x$, on a $\ln(a^x) = \lim_{n\to\infty} \ln(a^{y_n}) = \lim_{n\to\infty} y_n \ln(a) = x\ln(a)$

$\ln(1) = 0$ et ln est strictement croissante, donc c'est évident.

Si $n = \sum_{k=0}^{p-1} 2^k a_k + 2^p$, alors $2^p \leq n < 2^{p+1}$, donc par croissance de ln, $p\ln 2 \leq \ln n < (p+1)\ln 2 \Rightarrow p \leq \frac{\ln n}{\ln 2} < p+1$, et comme p est entier, on a bien $p = E[\frac{\ln n}{\ln 2}]$.

On représente ici la fonction logarithme Néperien :



2.2 Nombres complexes

Histoire des nombres

1. Les nombres entiers ont été inventés pour faciliter les échanges [si tu me donnes 3 moutons, je te donne 5 chèvres].
2. Les mauvais payeurs ont motivé la découverte des nombres relatifs [tu me dois toujours 3 moutons : crédit montons $= -3$].
3. Pour trouver le prix du mouton pour une chèvre, on a du trouver les nombres rationnels [1 mouton$=\frac{5}{3}$ chèvre].
4. Le cours du mouton évoluant, il a fallut inventer les nombres algébriques (ie racines d'un polynôme à coefficients dans $\mathbb{Q}$) [si tu me donne n troupeaux de n chèvres, je te donne n moutons : on obtient la racine carré].
5. Aujourd'hui, nous avons besoin d'un nombre, noté i, tel que $i^2 = -1$. On parlera de nombre imaginaire et on construira ensuite les complexes $\mathbb{C}$ (cf. cours de terminale S).

Racines de l'unité

Soit $n \in \mathbb{N}$.
On admet l'existence d'un nombre complexe noté w_n et appelé racine primitive n-ième de l'unité qui vérifie :

$$\begin{cases} \forall i \in \{2; \ldots; n-1\}, (w_n)^i \neq w_n \\ (w_n)^n = 1 \end{cases}$$

3 Algorithme naïf

Question 2. *Rappeler l'algorithme naïf de multiplication de n par m (celui que l'on apprend dans les petites classes) et écrire un programme l'executant.*

Quelle est la complexité de cet algorithme en nombre d'opérations ?

Que penser du calcul en base b lorsque $b \neq 2$?

Correction 2.

L'algorithme est simple : on écrit la multiplication comme on l'a appris :

$$\begin{array}{r}
1\ a_{p-1}\ a_{p-2}\ldots a_1\ a_0 \\
\times \quad 1\ b_{q-1}\ b_{q-2}\ \ldots b_1\ b_0 \\
\hline
b_0\ (b_0a_{p-1})\ldots(b_0a_1)\ (b_0a_0) \\
+ \quad b_1\ (b_1a_{p-1})\ldots(b_1a_1)\ (b_1a_0) \quad\quad \\
\ldots \quad\quad\quad \\
+ \quad b_q\ (b_qa_{p-1})\ldots(b_qa_1)\ (b_qa_0) \quad\quad\quad\quad\quad\quad \\
+ \quad 1 \quad a_{p-1} \quad \ldots \quad a_1 \quad a_0 \quad\quad\quad\quad\quad\quad\quad\quad\quad \\
\hline
\ldots \quad\quad\quad\quad\quad\quad\quad\quad \ldots(b_0a_0)
\end{array}$$

On suppose ici que n et m sont donnés en base 2, sous forme de tableaux, accompagnés de leur longueur ; on va renvoyer un tableau et sa longueur (il est très facile de passer du nombre au tableau et réciproquement). On peut alors écrire l'algorithme :

Ajoute(T, a, k)=
 SI ($a = 0$) ALORS () SINON
  ($bool$ ←VRAI ;
  TANTQUE ($bool$) FAIRE
   (SI ($T[k] = 1$) ALORS ($T[k] \leftarrow 0; k \leftarrow k + 1;$) SINON ($T[k] \leftarrow 1; bool$ ←FAUX ;) FSI ;)
  FTANTQUE ;)
 FSI ;

MultiplieNaif($(n, p), (m, q)$)=
 $res \leftarrow$ TABLEAU $(1, p + q)$ 0 ;
 POUR ($i = 1$) JUSQU'À (q) FAIRE
  (SI ($b[i] = 1$) ALORS (POUR ($j = 1$) JUSQU'À (p) FAIRE (Ajoute($res, a[j], i + j - 1$) ;) FPOUR ;)
  SINON ()
  FSI ;)
 FPOUR ;
 $(res, p + q - 1 + res[p + q]) \rightarrow;$;

La complexité dans le pire des cas est facile à calculer : dans le pire des cas, tous les a_i et tous les b_i valent 1. On fait alors p additions pour afficher la première ligne ; puis pour les $q - 1$ lignes suivantes, on fait $p - 1$ additions pour ajouter a_0 (car tous les chiffres sont des 1), 1 addition pour ajouter chaque a_i lorsque $i \in \{1; \ldots; p - 1\}$, puis encore 2 additions pour ajouter le 1 restant.

Donc en tout,

$$\boxed{C_{naif}(p, q) = p + (q - 1)(p - 1 + p - 1 + 2) = p(2q - 1)}$$

Le fait d'être en base 2 arrange bien les choses : tous les b_i valent 0 ou 1, donc les multiplications intermédiaires sont simples. En base b, il faudrait s'occuper de ces multiplications et faire attention aux retenues.

4 Algorithme de Karatsuba

Question 3. *Vérifier l'égalité :*

$$\forall a, b, c, d, k \in \mathbb{N}, \quad (a + b.2^k)(c + d.2^k) = ac - [(a - b)(c - d) - ac - bd].2^k + bd.2^{2k}$$

En déduire un algorithme basé sur l'idée de "diviser pour régner" qui calcule le produit de n et m.

Que dire de la complexité ?

Que se passe-t-il si on change de base de calcul ?

Correction 3.

L'égalité est claire. On en déduit que le calcul du produit de $(a+b.2^k)$ par $(c+d.2^k)$ ne requiert que le calcul des trois produits ac, bd et $(a-b)(c-d)$. Par conséquent, si k est bien choisit, le calcul du produit de deux grands nombres peut se faire en connaissant trois produit de nombres de taille moitié : c'est l'idée d'un algorithme du type "diviser pour régner".

```
REC Karatsuba(n, m)=
    p ← E[ln n / ln 2]; q ← E[ln m / ln 2];
    SI (p = 0) ALORS (SI (n = 1) ALORS (m →;) SINON (0 →;) FSI;)
    SINON (SI (q = 0) ALORS (SI (m = 1) ALORS (n →;) SINON (0 →;) FSI;)
        SINON (k ← max(E[p/2], E[q/2]);
            a ← E[n/2^k]; c ← E[m/2^k];
            b ← n − 2^k.a; d ← n − 2^k.c;
            u ← Karatsuba(a, c); v ← Karatsuba(b, d); w ← Karatsuba((a − b), (c − d));
            u + (w − u − v).2^k + v.2^{2k} →;)
        FSI;)
    FSI;;
```

La complexité de l'algorithme est plus difficile a trouver. On suppose qu'on a deux entiers de même taille p. Alors si on appelle $C(p)$ la complexité pour multiplier ces deux nombres, on a :

$$C(p) = 3C(\frac{p}{2}) + \kappa p, \quad avec\ \kappa \in \mathbb{R}$$

On pose $X(q) = \frac{C(2^q)}{3^q}$. On a alors :

$$X(q+1) = \frac{C(2^{q+1})}{3^{q+1}} = \frac{3C(2^q) + \kappa 2^{q+1}}{3^{q+1}} = X(q) + \kappa(\frac{2}{3})^{q+1}$$

D'où :

$$X(q) = \kappa \sum_{j=0}^{q} (\frac{2}{3})^{j+1} \leq 2\kappa$$

On en déduit :

$$\boxed{C_{Karatsuba}(p,p) = C(p) \leq 2\kappa.p^{\frac{\ln 3}{\ln 2}}}$$

La complexité est donc bien meilleure (cf. "comparaison des résultats"). On retrouve ici un bel exemple de "diviser pour régner" très efficace.

Le changement de base de calcul ne change ici absolument rien.

5 Transformée de Fourier rapide

5.1 Rappels sur l'exponentiation rapide

On se donne deux entiers x et n. On veut calculer x^n.

Algorithme naif

La première idée qui vient à l'esprit est de multiplier x par lui même n fois à la suite :

```
Puissance(x, n)=
    res ← 1;
    TANTQUE (n > 0) FAIRE (res ← res × x; n ← n − 1;) FTANTQUE;
    res →;;
```

La complexité en temps est linéaire en n.

La méthode repose sur le résultat simple suivant :

$$\forall p \in \mathbb{N}, x^{2p} = (x^2)^p \text{ et } x^{2p+1} = x.(x^2)^p$$

On écrit l'algorithme :

```
rec PuissanceRapide(x, n)=
    si (n = 0) alors (1 →)
    sinon (si (n mod 2 = 0) alors (PuissanceRapide(x × x, n/2) →;)
            sinon (x×PuissanceRapide(x × x, (n − 1)/2) →;)
            fsi ;)
    fsi ; ;
```

L'algorithme est beaucoup plus rapide puisqu'il est maintenant de l'ordre de $\ln n$: si $n = 2^l + 2^{l-1}n_{l-1} + \ldots + 2n_1 + n_0$ (avec $l \in \mathbb{N}$ et $n_0, \ldots, n_{l-1} \in \{0:1\}^l$), on effectue $l + n_0 + \ldots + n_{l-1}$ multiplications pour le calcul de x^n, soit dans le pire des cas $2l = 2E[\frac{\ln n}{\ln 2}]$ multiplications.

5.2 Evaluation rapide de polynômes

On va utiliser la même méthode pour évaluer un polynôme en certains points bien choisis.

Soit $P(X) = \sum_{k=0}^{2^n-1} a_k X^k$. On pose : $Q(Y) = \sum_{k=0}^{2^{n-1}-1} a_{2k} Y^k$ et $R(Y) = \sum_{k=0}^{2^{n-1}-1} a_{2k+1} Y^k$, de sorte que :

$$P(X) = Q(X^2) + XR(X^2)$$

On veut évaluer le polynôme P aux points $(w_{2^n})^i$ pour $i \in \{1; \ldots; 2^n\}$.

Mais on sait que $(w_{2^n})^{-2^n} = 1$, donc

$$\forall i \in \{2^{n-1}+1; \ldots; 2^n\}, \quad ((w_{2^n})^i)^2 = ((w_{2^n})^{(i-2^{n-1})})^2$$

Par conséquent, on va juste évaluer Q et R aux points $((w_{2^n})^i)^2$ pour $i \in \{1; \ldots; 2^{n-1}\}$, et il suffira d'utiliser la première formule pour retrouver les valeurs de P recherchées.

En appliquant récursivement cette méthode, on diminue le temps d'évaluation : si on note $C_{Eval}(d)$ le nombre d'opérations nécessaires pour évaluer un polynôme de degré d en $d+1$ points, on a :

$$C_{Eval}(2^n - 1) = 2C_{Eval}(2^{n-1} - 1) + 2^{n+1}$$

On pose $X(n) = \frac{C_{Eval}(2^n-1)}{2^n}$ et on obtient comme d'habitude $X(p+1) = X(p) + 2$ d'où $X(p) = 2p$.

Par conséquent, $\boxed{C_{Eval}(d) = d \ln d}$.

5.3 Transformation de Fourier rapide

Question 4. *En remarquant que si $n = \sum_{k=0}^{p-1} 2^k a_k + 2^p$, alors $n = P_n(2)$ avec $P_n = \sum_{k=0}^{p-1} X^k a_k + X^p$, montrer que la multiplication de deux entiers peut être vu comme la multiplication de deux polynômes suivie d'un report de retenues.*

En se rappelant de la méthode d'interpolation de Lagrange, montrer que le produit PQ de deux polynômes P et Q de degré p et q peut être vu comme le produit de $p+q+1$ valeurs prises par ces polynômes.

En utilisant l'évaluation rapide de polynômes étudiée précédement, trouver une méthode calculant rapidement le produit de deux entiers.

Que dire de la complexité de ce nouvel algorithme ?

Correction 4.

Prenons un exemple simple : on veut calculer le produit de $13 = \overline{1101}^2$ par $9 = \overline{101}^2$. On remarque que $12 = P(2)$ où $P(X) = X^3 + X^2 + 1$ et que $9 = Q(2)$ où $Q(X) = X^2 + 1$. Par conséquent, $13 \times 9 = P(2)Q(2) = PQ(2)$. Si on calcule $PQ(X) = X^6 + X^5 + 2X^3 + X^2 + 1$, et que l'on reporte la retenue $PQ(X) = X^6 + X^5 + X^4 + X^2 + 1$, on obtient directement $13 \times 9 = \overline{1110101}^2 = 117$.

Pour calculer rapidement le produit des deux polynômes, il suffit d'utiliser les polynômes interpolateurs de Lagrange et donc de connaître l'évaluation de PQ en $3 + 2 + 1 = 7$ points. Si on utilise la méthode précédente, on aura

rapidement l'évaluation pour les puissances d'une racine primitive 7-ième de l'unité des polynômes P et Q, et donc en faisant $3+2+1$ produits simples, on obtiendra l'évaluation de PQ en ces points.

La méthode générale peut donc s'écrire ainsi : pour multiplier n par m,

1. poser P et Q les polynômes représentant n et m.
2. pour p et q les degrés de P et Q, poser $k = E[\frac{\ln(p+q+1)}{\ln 2}] + 1$ (ainsi, $p+q \leq 2^k - 1$ et le degré de PQ est plus petit que $2^k - 1$),
3. évaluer P et Q sur les puissances d'une racine primitive 2^k-ième de l'unité grâce à la méthode précédente,
4. effectuer les multiplications simples $P((w_{2^k})^i)Q((w_{2^k})^i)$ pour $i \in \{1; \ldots; 2^k\}$,
5. poser L le polynôme interpolateur de Lagrange tel que $L((w_{2^k})^i) = P((w_{2^k})^i)Q((w_{2^k})^i)$,
6. Effectuer les reports de retenues sur le polynôme L trouvé, et retrouver le produit nm.

La complexité de l'algorithme se situe essentiellement aux points 3, 4 et 5 de la méthode décrite :
- l'annexe montre que le point 3 se fait en $C_{Eval}(p+q+1) = (p+q+1)\ln(p+q+1)$ opérations,
- le point 4 consiste à faire $p+q+1$ multiplications simples,
- on peut montrer que l'interpolation du point 5 se fait aussi en $(p+q+1)\ln(p+q+1)$ opérations.

Au total et en supposant que les deux nombres ont la même taille (donc que $p = q$), on a à peu près :

$$\boxed{C_{TFR}(p,p) = 4p\ln 2p + 2p \simeq 4p\ln p}$$

On a encore gagner beaucoup grâce à cette méthode (cf. "comparaison des résultats").

6 Comparaison des résultats

On a représenté sur ce graphe les fonctions : $x \mapsto x^2$ (en haut), $x \mapsto x^{\frac{\ln 3}{\ln 2}}$ (milieu) et $x \mapsto x\ln(x)$ (en bas). La progression est nette.