

Dossier algorithme : Algorithme de Karatsuba

Pascal Malingrey

20 août 2019

Table des matières

1	Présentation	2
1.1	Exemple introductif	2
1.1.1	Méthode classique	2
1.1.2	Réduction du nombre de multiplications	2
1.1.3	Passage à 4 chiffres : récurrence	3
2	Algorithme de KARATSUBA	3
2.1	Ecriture	3
2.2	Preuve de terminaison	4
2.3	Preuve partielle	5
2.4	Complexité en temps	6
2.5	Compléments : représentation de la complexité	7
3	Optimalité du problème	8
3.1	Aperçu de l'algorithme de Tom-Cook	8
3.2	Dernières avancées	9

1 Présentation



Le problème

Nous désirons effectuer la multiplication de deux grands entiers écrits dans une base donnée α : la base décimale et la base binaire seront privilégiées.

Dans la suite, la distinction sera faite entre la multiplication par α notée \cdot et la multiplication classique notée \times . La multiplication par α n'a pas le même coût que la multiplication classique.

Par exemple en binaire, il s'agit d'un décalage de bit et en décimal, l'ajout d'un 0 à la fin d'une liste si on représente le nombre par une liste.

1.1 Exemple introductif

Dans ce premier exemple nous allons traiter la multiplication de 57 par 34.

1.1.1 Méthode classique

$57 \times 34 = (5 \cdot 10 + 7) \times (3 \cdot 10 + 4) = 5 \times 3 \cdot 10^2 + (5 \times 4 + 7 \times 3) \cdot 10 + 7 \times 4 = 15 \cdot 10^2 + 41 \cdot 10 + 28 = 1938$
Le coût de cette opération est de 4 multiplications et de trois additions (les trois additions ne sont pas équivalentes).

1.1.2 Réduction du nombre de multiplications

Pour comprendre le fonctionnement, il est plus simple de travailler avec une écriture générique. On considère deux entiers m et p avec a_1, a_2, b_1 et b_2 entiers inférieurs strictement à x .

$$m \times p = \overline{a_1 a_2} \times \overline{b_1 b_2} = a_1 b_1 x^2 + (a_1 b_2 + a_2 b_1) x + a_2 b_2 \quad (1)$$

L'idée est de remplacer les deux produits croisés (du milieu), par une seule multiplication. Or $a_1 b_2 + a_2 b_1$ est une partie du développement de $(a_2 - a_1) \times (b_2 - b_1)$

$$a_1 b_2 + a_2 b_1 = a_1 b_1 + a_2 b_2 - (a_2 - a_1)(b_2 - b_1)$$

Remarque on aurait pu utiliser l'addition, mais pour éviter des dépassement lors d'une implémentation de l'algorithme la soustraction est privilégiée.

Le produit devient :

$$m \times p = \overline{a_1 a_2} \times \overline{b_1 b_2} = a_1 b_1 x^2 + [\underbrace{a_1 b_1 + a_2 b_2}_{\text{termes déjà apparents}} - (a_2 - a_1)(b_2 - b_1)] x + a_2 b_2 \quad (2)$$

Nous avons besoin de calculer que trois multiplications différentes.

Exemple Calcul de 57×34

$$5 \times 3 = 15$$

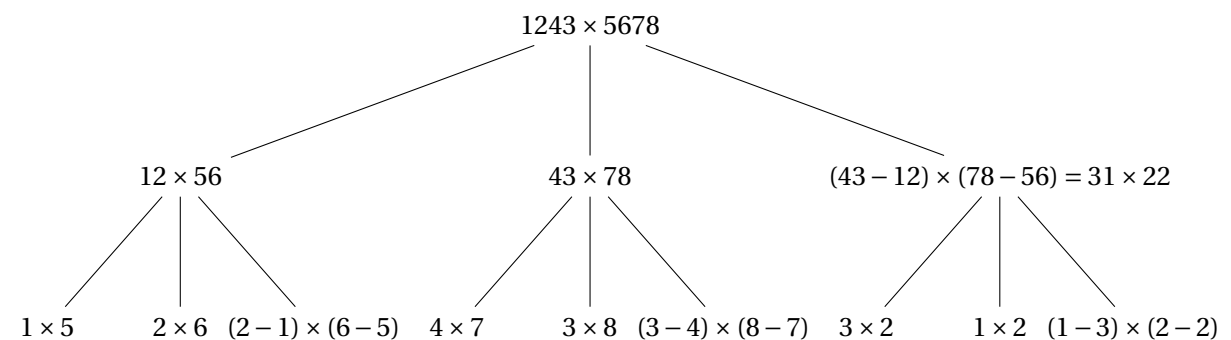
$$7 \times 4 = 28$$

$$(7 - 5) \times (4 - 3) = 2$$

Avec la formule (2), on obtient $57 \times 34 = 15 \cdot 10^2 + (15 + 28 - 2) \cdot 10 + 28 = 15 \cdot 10^2 + 41 \cdot 10 + 28 = 1938$

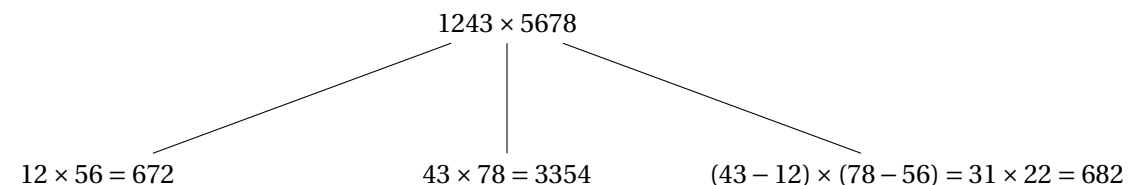
1.1.3 Passage à 4 chiffres : récurrence

Prenons maintenant deux nombres de 4 chiffres ou digits, $1243 \times 5678 = (12 \cdot 10^2 + 43)(56 \cdot 10^2 + 78)$. On va devoir calculer les trois produits à deux chiffres puis pour chacun de ces produits à nouveau trois produits à 1 chiffre.



On obtient :

- $12 \times 56 = 5 \cdot 10^2 + (5 + 12 - 1) \cdot 10 + 12 = 672$
- $43 \times 78 = 28 \cdot 10^2 + (28 + 24 - (-1)) \cdot 10 + 24 = 3354$
- $31 \times 22 = 6 \cdot 10^2 + (6 + 2 - 0) \cdot 10 + 2 = 682$



Au final :

$$1243 \times 5678 = 672 \cdot 10^4 + (672 + 3354 - 682) \cdot 10^2 + 3354 = 672 \cdot 10^4 + 3344 \cdot 10^2 + 3354 = 7057754$$

Ces exemples permettent d'envisager un algorithme récursif avec un nombre réduit de multiplications (algorithme du type diviser pour régner).

2 Algorithme de KARATSUBA

2.1 Ecriture

La version présentée l'est pour la base décimale et donc le log, ici est le logarithme décimal. Pour une base différente, les changements sont minims. On considère deux entiers naturels a et b .

Algorithme 1 : Fonction karatsuba(a, b)

```

si ( $a < 10$ ) ou ( $b < 10$ ) alors
|   retourner  $a*b$ 

 $m \leftarrow \left\lceil \frac{\min(\log(a+0.5), \log(b+0.5))}{2} \right\rceil$ ;
 $a_1, a_2 \leftarrow \text{decompose}(a, m)$ ;
 $b_1, b_2 \leftarrow \text{decompose}(b, m)$ ;
 $\epsilon = \text{signe}(a_2 - a_1) \times \text{signe}(b_2 - b_1)$ ;
 $R1 \leftarrow \text{karatsuba}(a_1, b_1)$ ;
 $R2 \leftarrow \text{karatsuba}(a_2, b_2)$ ;
 $R3 \leftarrow \text{karatsuba}(|a_2 - a_1|, |b_2 - b_1|)$ ;
retourner  $R1 \times 10^{2 \times m} + (R1 + R2 - \epsilon \times R3) \times 10^m + R2$ 

```

Remarque La fonction décompose dépend de la façon de représenter le nombre et donc en partie de l'implémentation de l'algorithme. Voici une possibilité de la fonction **decompose**.

Dans cette fonction x et m sont des entiers naturels.

Algorithme 2 : Fonction decompose(x, m)

```

 $x_1 \leftarrow \left\lfloor \frac{x}{10^m} \right\rfloor$ ;
 $x_2 \leftarrow x - x_1 \times 10^m$ ;
retourner  $x_1, x_2$ 

```

2.2 Preuve de terminaison

Nous allons nous appuyer sur le théorème de terminaison

Théorème 1 Soient f un algorithme récursif défini sur un ensemble A et $<$ un ordre bien fondé sur A .

Soit $x \in A$, on note, A_x l'ensemble des y tels que $f(x)$ appelle $f(y)$.

- si pour x minimal, $f(x)$ se termine,
- et si, pour tout x de A , $\forall y \in A (y \in A_x \Rightarrow y < x)$.

alors $f(x)$ se termine.

Notre algorithme f est ici **karatsuba** qui est défini sur l'ensemble $A = \mathbb{N}^2$.

Nous définissons sur A , la relation binaire suivante :

Définition 1 Soit (a, b) et $(c, d) \in A$,

$(a, b) < (c, d)$ si et seulement si $\max(\text{long } a, \text{long } b) < \max(\text{long } c, \text{long } d)$.

où $\text{long } a$ est le nombre de chiffres de a en base 10.¹

On peut montrer que $<$, ainsi définie est un ordre bien fondé sur A , c'est à dire il n'existe pas de suite infinie (x_n) d'éléments de A telle qu'on ait $x_{n+1} < x_n$ pour tout n .

Soit $x = (a, b) \in A$. Déterminons maintenant $A_{(a, b)}$:

- si $a < 10$ ou $b < 10$, $A_{(a, b)} = \emptyset$ (a, b) est minimale et l'algorithme **karatsuba** renvoie bien le produit $a \times b$.
- sinon on définit trois couples de la manière suivante :

$$m = \left\lceil \frac{\min(\text{long } a, \text{long } b)}{2} \right\rceil$$

$$a = a_1 \cdot 10^m + a_2 \text{ avec } a_1 \text{ et } a_2 \text{ entiers naturels et } a_2 < 10^m$$

$$a_3 = |a_2 - a_1|$$

de même pour b .

Donc $A_{(a, b)} = \{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}$.

Montrons maintenant que $y \in A_{(a, b)} \Rightarrow y < (a, b)$

- si $a < 10$ ou $b < 10$, il n'a rien à prouver car $A_{(a, b)} = \emptyset$.
- sinon $\text{long}(a) \geq 2$ et $\text{long}(b) \geq 2$ d'où $m \geq 1$ et par définition des termes suivants :

$$\text{long}(a_2) < m \leq \left\lceil \frac{\text{long}(a)}{2} \right\rceil < \text{long}(a)^2,$$

$$\text{long}(a_1) + m = \text{long}(a) \text{ et comme } m \geq 1, \text{long}(a_1) < \text{long}(a)$$

$$\text{et donc } \text{long}|a_2 - a_1| \leq \max(\text{long}(a_1), \text{long}(a_2)) < \text{long}(a)$$

On obtient un résultat similaire pour b .

On vient de prouver que $\text{long } a_1 < \text{long } a$ et $\text{long } b_1 < \text{long } b$ donc :

$$\max(\text{long } a_1, \text{long } b_1) < \max(\text{long } a, \text{long } b) \text{ i.e. } (a_1, b_1) < (a, b)$$

De même $(a_2, b_2) < (a, b)$ et $(a_3, b_3) < (a, b)$.

L'algorithme de **karatsuba** vérifie toutes les hypothèses du théorème 1, et donc se termine.

2.3 Preuve partielle

Bien qu'il finisse, encore faut-il que l'algorithme donne le résultat attendu. Attachons-nous à prouver. Pour gagner en lisibilité renommons **karatsuba** par **kar**.

Définissons pour $n \in \mathbb{N}^*$, la propriété $\mathcal{P}(n)$ suivante : **kar(a,b)** renvoie $a \times b$ pour tout a, b de A avec $\text{long } a \leq n$ et $\text{long } b \leq n$. Démontrons par récurrence que $\mathcal{P}(n)$ est vraie pour tout entier $n \in \mathbb{N}^*$.

Preuve Initialisation : pour tout $a < 10$ et $b < 10$ c'est à dire $\text{long } a = \text{long } b = 1$, le programme ne fait pas d'appel récursif et renvoie bien $a \times b$, donc $\mathcal{P}(1)$ est vraie.

Hérédité : Supposons que pour un entier n non nul, $\mathcal{P}(n)$ soit vraie. Étudions $\mathcal{P}(n+1)$.

Prenons deux entiers a et b de longueur $n+1$.

En reprenant les notations précédentes, on a $A_{(a, b)} = \{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}$ avec $(a_1, b_1) <$

1. une définition possible pour des entiers naturels est : $\text{long}(x) = \lceil \log(x+0.5) \rceil$

2. $\left\lceil \frac{x}{2} \right\rceil < \frac{x}{2} + 1 \leq \frac{x+2}{2} \leq x$ car $2 \leq x$.

$(a, b) \leq n+1$, $(a_2, b_2) < (a, b) \leq n+1$ et $(a_3, b_3) < (a, b) \leq n+1$ et donc $\text{long } a_1 \dots \text{long } b_3$ sont au plus de longueur n et nous pouvons appliquer l'hypothèse de récurrence.

$$\begin{aligned}
\mathbf{kar}(a, b) &= \mathbf{kar}(a_1, b_1) \cdot 10^{2m} + (\mathbf{kar}(a_1, b_1) + \mathbf{kar}(a_2, b_2) - \mathbf{kar}(a_3, b_3)) \cdot 10^m + \mathbf{kar}(a_2, b_2) \\
&= a_1 \times b_1 \cdot 10^{2m} + (a_1 \times b_1 + a_2 \times b_2 - \epsilon a_3 \times b_3) \cdot 10^m + a_2 \times b_2 \quad \text{hyp. rec} \\
&= a_1 \times b_1 \cdot 10^{2m} + (a_1 \times b_1 + a_2 \times b_2 - (a_2 - a_1) \times (b_2 - b_1)) \cdot 10^m + a_2 \times b_2 \\
&= a_1 \times b_1 \cdot 10^{2m} + (a_1 \times b_2 + a_1 \times b_2) \cdot 10^m + a_2 \times b_2 \\
&= (a_1 \cdot 10^m + a_2) (b_1 \cdot 10^m + b_2) \\
&= a \times b
\end{aligned}$$

remarque : $\epsilon = \text{signe}(a_2 - a_1) \times \text{signe}(b_2 - b_1)$ et donc $\epsilon |a_2 - a_1| |b_2 - b_1| = (a_2 - a_1)(b_2 - b_1)$

Donc d'après le principe de récurrence, on a pour tout entier naturel non nul $\mathcal{P}(n)$ qui est vraie, validant la preuve partielle de notre algorithme **karatsuba**.

2.4 Complexité en temps

La fonction log dans cette algorithme a certainement un coup en temps important, on va donc remplacer l'instruction : $(\log(a+0.5))$ par $\text{long}(\text{str}(a))$ qui aura un coût en temps plus faible au plus en $O(n)$

Algorithme 3 : Fonction karatsuba(a, b)	coût
si $(a < 10)$ ou $(b < 10)$ alors	
retourner $a*b$	$O(1)$
$m \leftarrow \left\lceil \frac{\min(\text{long}(\text{str}(a)), \text{long}(\text{str}(b)))}{2} \right\rceil$	$O(n)$
$a_1, a_2 \leftarrow \text{decompose}(a, m)$	$O(n)$
$b_1, b_2 \leftarrow \text{decompose}(b, m)$	$O(n)$
$\epsilon = \text{signe}(a_2 - a_1) \times \text{signe}(b_2 - b_1)$	$O(n)$
$R1 \leftarrow \text{karatsuba}(a_1, b_1)$	$T\left(\frac{n}{2}\right)$
$R2 \leftarrow \text{karatsuba}(a_2, b_2)$	$T\left(\frac{n}{2}\right)$
$R3 \leftarrow \text{karatsuba}(a_2 - a_1 , b_2 - b_1)$	$T\left(\frac{n}{2}\right) + O(n)$
retourner $R1 \times 10^{2 \times m} + (R1 + R2 - \epsilon \times R3) \times 10^m + R2$	$\Theta(n)$

Remarque La somme ou la soustraction d'entiers naturels est en $O(n)$ c'est pourquoi $\text{signe}(a-b)$ l'est également.

Propriété 1 Notre algorithme **karatsuba** a une complexité en temps qui vérifie

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n)$$

Nous allons faire appel au Master Theorem pour déterminer cette complexité.

Théorème 2 Soient $a \geq 1$ et $b > 1$, et T une fonction vérifiant :

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^d)$$

- $d < \log_b a$ alors $T(n) = O(n^{\log_b a})$.
- $d = \log_b a$ alors $T(n) = O(n^d \log n)$.
- $d > \log_b a$ alors $T(n) = O(n^d)$.

Dans le cas qui nous intéresse, nous avons $a = 3$ et $b = 2$ et $d = 1$. Il se trouve que $1 < \log_2 3$ donc nous pouvons conclure d'après le théorème précédent que :

Propriété 2 Notre algorithme **karatsuba** a une complexité en temps qui vérifie

$$T(n) = O(n^{\log_2 3}) \text{ avec } \log_2 3 \simeq 1.585$$

L'algorithme classique est en $O(n^2)$ donc celui-ci est plus efficace. Le gain ne semble pas important et pourtant si on compare les évolutions des deux coûts voici le résultat :

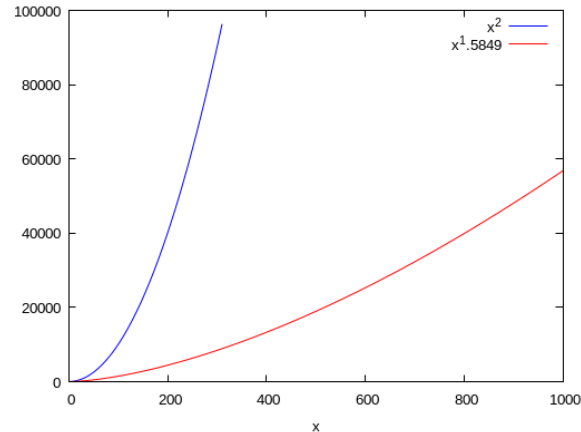
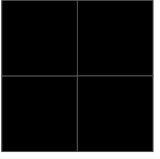
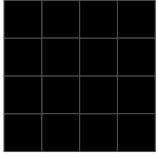
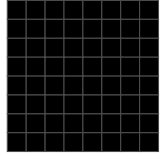
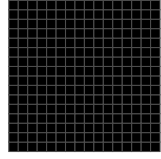
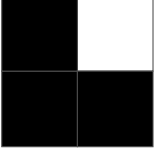
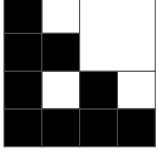
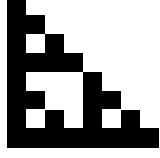
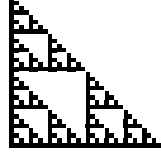


FIGURE 1 – Comparaison des croissances quadratique et en $n^{1.58}$

2.5 Compléments : représentation de la complexité

Nous allons illustrer la complexité en temps de cet algorithme d'une façon moins classique. Dans le cas d'un calcul normal mais récursif nous devons effectuer 4 multiplications et avec l'algorithme de **karatsuba** on n'en fait que 3.

Profondeur de récursivité	1	2	3	...	n
Algorithme classique					
Algorithme karatsuba					

Idée prise sur http://math.univ-lyon1.fr/~roblot/resources/ens_partie_2.pdf

La courbe limite est une fractale, qui ressemble au triangle de Sierpinski.

Pour les courbes fractales, on peut définir une dimension celle de Hausdorff. Et le calcul de cette dimension pour notre fractale est étonnamment $\log_2 3 = \frac{\ln 3}{\ln 2}$ à rapprocher de la complexité en temps de notre algorithme $O(n^{\log_2 3})$.

3 Optimalité du problème

La solution proposée par Karatsuba en 1962 a été la première avec une complexité sous quadratique. Une généralisation de cet algorithme a été proposée par Toom-Cook en 1963.

Pour comprendre cette généralisation revenons sur l'algorithme **Karatsuba**, et en répondant à la question comment trouver le terme central en économisant le nombre de multiplication. Nous utilisons les notations de la première partie :

$$m(x) \times p(x) = (a_1x + a_0) \times (b_1x + b_0) = p_2x^2 + p_1x + p_0 \quad (3)$$

Le résultat de la multiplication est un polynôme en x , et nous allons prendre des valeurs particulières pour trouver le résultat.

- Avec le coefficient dominant, on obtient $a_1 \times b_1 = p_2$
- Avec $x = 0$, on obtient $a_0 \times b_0 = p_0$
- Avec $x = -1$, on obtient $(a_1 - a_0) \times (b_1 - b_0) = p_2 - p_1 + p_0$ et donc

$$p_1 = p_2 + p_0 - (a_1 - a_0) \times (b_1 - b_0)$$

On retrouve le résultat de la première partie.

3.1 Aperçu de l'algorithme de Tom-Cook

Tom-Cook utilise cette idée en décomposant chacun des nombres en trois parties $m(x) = (a_2x^2 + a_1x + a_0)$ et $p(x) = (b_2x^2 + b_1x + b_0)$. Le résultat est $p(x) = \sum_{i=0}^4 p_i x^i$. En évaluant le système avec le terme dominant puis la valeur en 2, 1, -1 et 0, on obtient un système d'équa-

tions qui est

$$\begin{pmatrix} p(\infty) \\ p(2) \\ p(1) \\ p(-1) \\ p(0) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 16 & 8 & 4 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_4 \\ p_3 \\ p_2 \\ p_1 \\ p_0 \end{pmatrix}$$

Maintenant on utilise le fait que $p(x) = a(x)b(x)$ et que la matrice carrée A qui est présente est inversible³, pour obtenir :

$$\begin{pmatrix} p_4 \\ p_3 \\ p_2 \\ p_1 \\ p_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 16 & 8 & 4 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} a(\infty)b(\infty) \\ a(2)b(2) \\ a(1)b(1) \\ a(-1)b(-1) \\ a(0)b(0) \end{pmatrix}$$

Nous avons donc 5 multiplications⁴ pour trouver la valeur du produit de $m \times p$.

Remarque D'un point de vue algorithmique on aura un algorithme récursif sous forme d'arbre avec pour chaque nœud, 5 branches et comme on divise la taille du nombre de départ par 3, une profondeur de l'ordre de $\log_3(n)$.

Propriété 3 L'algorithme de Tom-Cook a une complexité qui vérifie $T(n) = 5T\left(\frac{n}{3}\right) + \Theta(n)$ et donc est en $O(n^{\log_3 5})$.

Comme $\log_2 3 > \log_3 5$, l'algorithme de **Karatsuba** n'est pas optimal.

3.2 Dernières avancées

Début de cette année 2019, des chercheurs

$$3. \det(A) = -12 \text{ et } A^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -2 & \frac{1}{6} & -\frac{1}{2} & -\frac{1}{6} & \frac{1}{2} \\ -1 & 0 & \frac{1}{2} & \frac{1}{2} & -1 \\ 2 & -\frac{1}{6} & 1 & -\frac{1}{3} & -\frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

4. on ne compte que les multiplications entre grand nombre, la multiplication par 2 ou 4 est un décalage de bit lors de l'implémentation de l'algorithme.

Annexe

Algorithme pour la base binaire

Algorithme 4 : Fonction karatsuba(a, b)

si ($a < 2$ ou ($b < 2$) **alors**

 | **retourner** $a*b$

$m \leftarrow \left\lceil \frac{\min(\log(a+0.5), \log(b+0.5))}{2} \right\rceil$;

$a_1, a_2 \leftarrow \text{decompose}(a, m)$;

$b_1, b_2 \leftarrow \text{decompose}(b, m)$;

$\epsilon = \text{signe}(a_2 - a_1) \times \text{signe}(b_2 - b_1)$;

$R1 \leftarrow \text{karatsuba}(a_1, b_1)$;

$R2 \leftarrow \text{karatsuba}(a_2, b_2)$;

$R3 \leftarrow \text{karatsuba}(|a_2 - a_1|, |b_2 - b_1|)$;

retourner $R1 \times << 2 \times m + (R1 + R2 - \epsilon \times R3) << m + R2$
