Algorithmique sur les nombres

Philippe Langevin

IMATH, université de Toulon

Mars 2021

Division euclidienne

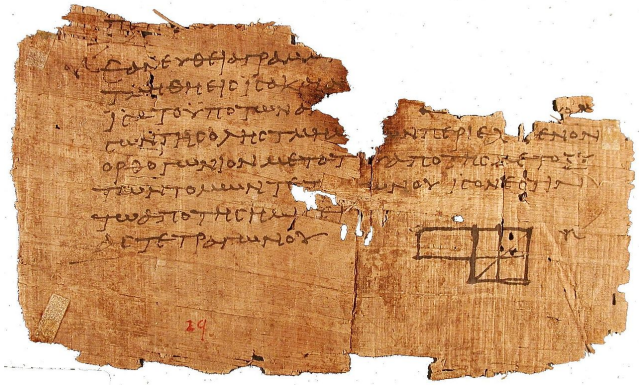


FIGURE – Le plus ancien fragment des éléments d'Euclide : an 100 ou 300.

Théorème (Les éléments, Livre VII)

Soient a et β deux entiers naturels. Si β n'est pas nul, il existe un unique couple (q, r) d'entiers tel que

$$a = \beta \times q + r, \quad 0 \leq r < \beta.$$

1. q est le quotient, notation algo `a div b`, `a / b` en C et `//` en python ;

Système de numération

Un entier $\beta > 2$ fixé, un entier $a > 0$ se décompose d'une et une seule manière sous la forme :

$$a = a_{n-1}\beta^{n-1} + \ldots + a_1\beta^1 + a_0 \quad \text{notation} \quad (a_{n-1} \ldots a_1 a_0)_\beta$$

avec $0 \leq a_i < \beta$ et $a_{n-1} > 0$.

- les a_i sont des chiffres ;
- $a_{n-1} > 0$ est le chiffre dominant ;
- n la taille a, caractérisée par $\beta^{n-1} \leq a < \beta^n$.

Proposition (taille)

La taille du nombre a en base β vaut $[\log_\beta(a)] + 1$.

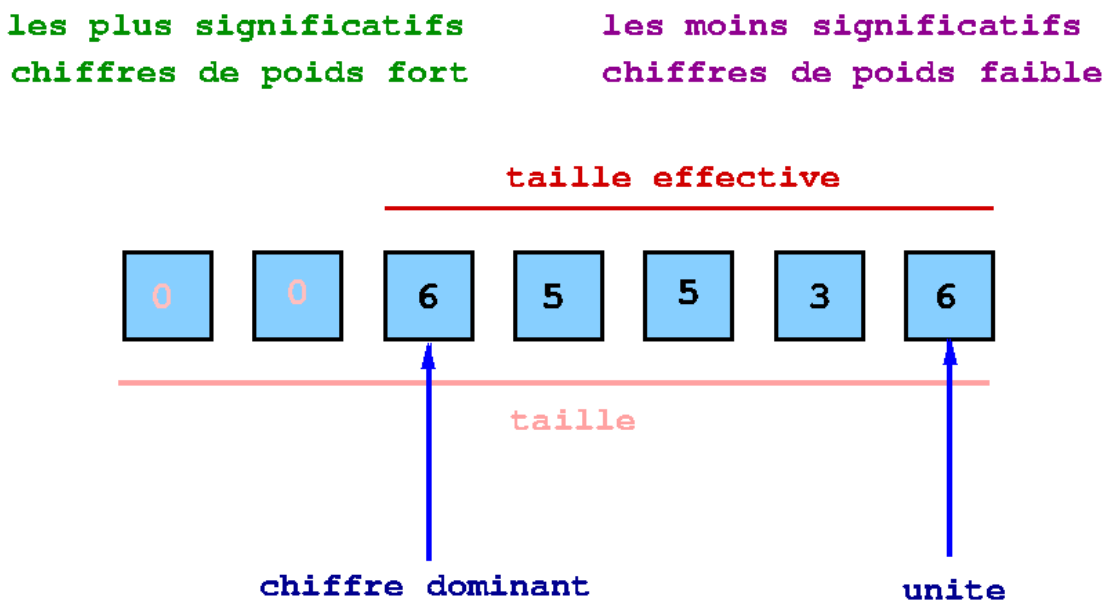
Les 100 bases usuelles

- décimale : humain !
- binaire : informaticien !
- hexadécimale : hacker !
- octale : système unix !

Exercice

Convertir 666 en binaire, décimale, octale.

représentation



Pour une base donnée β, on utilise un tableau de taille n pour représenter tous les nombres de taille strictement inférieure à β^n, indexé par les entiers de 0 à $n-1$.

bc - An arbitrary precision calculator language

```
1 BC_BASE_MAX : The maximum output base is set
    at 999. The maximum input base is 16.
2 BC_DIM_MAX  : This is currently an arbitrary
    limit of 65535
```

```
1 pl@812/notes> bc <<< '2^1000000' | wc  -m
2 309883
```

```
1 pl@812/notes> bc <<< 'obase=16; 47710'
2 BA5E
```

Exercice

Quelle est la taille décimale d'un nombre binaire de 1783 chiffres ?

Notation

Pour un nombre ou un tableau A de taille n en base β :

$$(A_{n-1} \dots A_1 A_0)$$

pour tout indice $i < n$, on introduit la notation :

$$A[0..i[= (A_{i-1} \dots A_1 A_0)$$

C'est le nombre composé des i-chiffres de poids faible.

$$A[0..i[+A_i\beta^i = A[0..i+1[$$

Incrémenter un nombre

En base décimale :

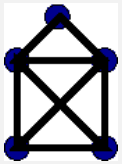
1	0	1	9	9	9

En binaire :

*	*	*	*	*	*	*	0	1	1	1	1

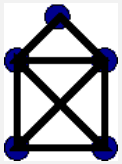
Incrémenter un nombre

```
1  INC( Z : nombre )
2  donnee   Z : n chiffres
3   variable      i : indice
4      aux :  chiffre
5  debut
6   aux ← base − 1
7   i   ← 0
8   tantque ( i < n et Z [ i ] == aux ) faire
9    Z[ i ] = 0
10    INC(i)
11   ftq
12   si ( i < n ) alors
13        Z[ i ] = Z [ i ] + 1
14             fsi
15   retourner i < n
16  fin
```



Temps de calcul

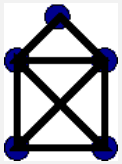
1. Analyser le pire des cas de l'algorithme `inc( Z : nombre )`.
2. Analyser le meilleur des cas.
3. Vérifier que le temps de calcul moyen est $\Theta(1)$.



puzzle

```
1 def wt( x ) :
2     r = 0
3     while x > 0 :
4         x = x & (x-1)
5         r = r + 1
6     return r
```

1. Que fait la fonction `wt( x )` ?
2. Trouver sur la toile l'histoire de ce puzzle !



Comparer deux nombres

Ecrire un algorithme `int cmp( x, y:nombre)` qui renvoie :

- zéro si les nombres sont égaux ;
- un nombre négatif si $x < y$;
- un nombre positif si $x > y$.

souvenir d'écolier !

Calculer la somme de 1855 et 1777 :

$$\begin{array}{cccc} & 1 & 7 & 7 & 7 \\ & 1 & 8 & 5 & 5 \\ \hline \end{array}$$

souvenir d'écolier !

Calculer la somme de 1855 et 1777 :

$$\begin{array}{cccc} & 1 & 7 & 7 & 7 \\ & 1 & 8 & 5 & 5 \\ \hline & & & & \\ & & & & 2 \end{array}$$

- Un procédé acquis. . .
- Pas forcément compris !
- Propagation de retenue.

souvenir d'écolier !

Calculer la somme de 1855 et 1777 :

$$\begin{array}{cccc} & 1 & 7 & 7 & 7 \\ & 1 & 8 & 5 & 5 \\ \hline & & & 1 & \\ & & & & 2 \end{array}$$

- Un procédé acquis. . .
- Pas forcément compris !
- Propagation de retenue.

souvenir d'écolier !

Calculer la somme de 1855 et 1777 :

$$\begin{array}{cccc} & 1 & 7 & 7 & 7 \\ & 1 & 8 & 5 & 5 \\ \hline & & & 1 & \\ & & & 3 & 2 \end{array}$$

- Un procédé acquis. . .
- Pas forcément compris !
- Propagation de retenue.

souvenir d'écolier !

Calculer la somme de 1855 et 1777 :

$$\begin{array}{cccc} & 1 & 7 & 7 & 7 \\ & 1 & 8 & 5 & 5 \\ \hline & & 1 & 1 & \\ & & & 3 & 2 \end{array}$$

- Un procédé acquis. . .
- Pas forcément compris !
- Propagation de retenue.

souvenir d'écolier !

Calculer la somme de 1855 et 1777 :

$$\begin{array}{cccc} & 1 & 7 & 7 & 7 \\ & 1 & 8 & 5 & 5 \\ \hline & & 1 & 1 & \\ & & 6 & 3 & 2 \end{array}$$

- Un procédé acquis. . .
- Pas forcément compris !
- Propagation de retenue.

souvenir d'écolier !

Calculer la somme de 1855 et 1777 :

$$\begin{array}{cccc} & 1 & 7 & 7 & 7 \\ & 1 & 8 & 5 & 5 \\ \hline & 1 & 1 & 1 & \\ & & 6 & 3 & 2 \end{array}$$

- Un procédé acquis. . .
- Pas forcément compris !
- Propagation de retenue.

souvenir d'écolier !

Calculer la somme de 1855 et 1777 :

$$\begin{array}{cccc} & 1 & 7 & 7 & 7 \\ & 1 & 8 & 5 & 5 \\ \hline & 1 & 1 & 1 & \\ 3 & 6 & 3 & 2 & \end{array}$$

- Un procédé acquis. . .
- Pas forcément compris !
- Propagation de retenue.

souvenir d'écolier !

Calculer la somme de 1855 et 1777 :

	1	7	7	7
	1	8	5	5
0	1	1	1	
	3	6	3	2

- Un procédé acquis. . .
- Pas forcément compris !
- Propagation de retenue.

algorithme d'addition

```
1  ADDITION( A, B : nombre  )
2  donnees   A, B : n  chiffres
3   variable      i  :  indice
4      tmp : double
5       R  : nombre de n  chiffres
6      ret  :  booléen
7  debut
8    ret  ← 0
9    i    ← 0
10   tant que ( i < n ) faire
11     tmp  ← A[i] + B[i] + ret
12     ret  ← tmp div  base
13     R[i] ← tmp mod  base
14     INC(i)
15   ftq
16
17   retourner  R, ret
18   fin
```

Analyse et Preuve de correction

- La retenue propagée est inférieure ou égale à 1.
- $\text{TAILLE}(A+B) \leq n+1$.
- L'arrêt est clair !
- La complexité est linéaire.
- La preuve n'a pu être faite en primaire ! !

Correction : $R[0..i[+r\beta^i = A[0..i[+B[0..i[$ invariant !

Quand on entre dans la boucle avec $r\beta^i + R_i = A_i + B_i$:

$$A[i]\beta^i + B[i]\beta^i + r\beta^i + R[0..i[= A[i]\beta^i + B[i]\beta^i + A[0..i[+B[0..i[$$

$$(A[i] + B[i] + r)\beta^i + R[0..i[= A[0..i+1[+B[0..i+1[$$

$$\text{TMP} := \text{A[i]+B[i]+RET}$$

$$\text{TMP}\beta^i + R[0..i[= A[0..i+1[+B[0..i+1[$$

$$((\text{TMP} \div \beta)\beta + \text{TMP}\%\beta)\beta^i + R[0..i[= A[0..i+1[+B[0..i+1[$$

$$\text{R[i]} = \text{TMP}\% \text{ BASE}$$

$$\text{TMP} \div \text{BASE}\beta^{i+1} + R[i]\beta^i + R_i = A[0..i+1[+B[0..i+1[$$

$$\text{RET} = \text{TMP} \div \text{BASE}$$

$$r\beta^{i+1} + R[0..i+1[= A[0..i+1[+B[0..i+1[$$

$$i \leftarrow i+1$$

$$r\beta^i + R[0..i[= A[0..i[+B[0..i[$$

multiplication par la base

La multiplication d'un nombre par la base est une simple affaire de décalage !

multiplication par la base

La multiplication d'un nombre par la base est une simple affaire de décalage !

```
1  BUGMULT( a : nombre )
2  donnee   a : nombre de n chiffres
3   variable      i : indice
4  debut
5    i ← 1
6    tant que ( i < n ) faire
7      a[ i ] = a[ i − 1 ]
8      inc( i )
9    ftq
10   a[ 0 ] ← 0
11 fin
```

multiplication par la base

La multiplication d'un nombre par la base est une simple affaire de décalage !

```
1  BUGMULT( a : nombre )
2  donnee   a : nombre de n chiffres
3   variable      i : indice
4  debut
5    i ← 1
6    tant que ( i < n ) faire
7      a[ i ] = a[ i − 1 ]
8      inc( i )
9    ftq
10   a[ 0 ] ← 0
11 fin
```

Quel est le problème ?

multiplication par la base

La multiplication d'un nombre par la base est une affaire de décalage !

multiplication par la base

La multiplication d'un nombre par la base est une affaire de décalage !

```
1  DECALAGE( A : nombre )
2  donnee        A : nombre de n chiffres
3   variable     i : indice
4  debut
5    i ← n − 1
6    tant que ( i > 0 ) faire
7      A[ i ] = A[ i − 1 ]
8      dec( i )
9    ftq
10   A[0] = 0
11 fin
```

Multiplication par un chiffre

```
1  CMULT( a : nombre, c : chiffre )
2  donnee a : nombre de n chiffres
3   variable      i : indice
4                 tmp : double
5       ret   :  chiffre
6        r    : nombre de n chiffre
7  debut
8   ret ← 0
9   i   ← 0
10  tant que ( i < n ) faire
11   tmp ← c * a[ i ] + ret
12   r[i] ← tmp mod base
13   ret  ← tmp div base
14   inc( i )
15  ftq
16            retourner r, ret
17  fin
```

multiplication par un chiffre

$$\begin{array}{cccccccc} 6 & F & 6 & 0 & 6 & D & 3 & 7 \\ \times & & & & & & & 2 \\ \hline \end{array}$$

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
							E

multiplication par un chiffre

$$\begin{array}{cccccccc} 6 & F & 6 & 0 & 6 & D & 3 & 7 \\ \times & & & & & & & 2 \\ \hline & & & & & & 0 & \\ & & & & & & & E \end{array}$$

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
						0	
						6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
					0	0	
						6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
					0	0	
					A	6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
				1	0	0	
					A	6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
				1	0	0	
				D	A	6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
			0	1	0	0	
				D	A	6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
			0	1	0	0	
			0	D	A	6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
		0	0	1	0	0	
			0	D	A	6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
		0	0	1	0	0	
		C	0	D	A	6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
	0	0	0	1	0	0	
		C	0	D	A	6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
	0	0	0	1	0	0	
	E	C	0	D	A	6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
1	0	0	0	1	0	0	
	E	C	0	D	A	6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

multiplication par un chiffre

6	F	6	0	6	D	3	7
×							2
1	0	0	0	1	0	0	
D	E	C	0	D	A	6	E

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E

Multiplication par un chiffre

```
1  CMULT( a : nombre, c : chiffre )
2  donnee a : nombre de n chiffres
3   variable      i : indice
4                 tmp : double
5       ret   :  chiffre
6        r    : nombre de n chiffre
7  debut
8   ret ← 0
9   i   ← 0
10  tant que ( i < n ) faire
11   tmp ← c * a[ i ] + ret
12   r[i] ← tmp mod base
13   ret  ← tmp div base
14   inc( i )
15  ftq
16            retourner r, ret
17  fin
```

Analyse et Preuve de correction

- La retenue propagée est inférieure strictement à c.
- $\text{TAILLE}(c * A) \leq n + 1$.
- L'arrêt est clair !
- La complexité est linéaire.
- La preuve n'a pu être faite en primaire ! !

Démonstration.

...

Correction : $r\beta^i + R[0..i[= c * A[0..i[$ est invariant !

En effet, entrant dans la boucle avec $r\beta^i + R[0..i[= (cA[0..i[$:

$$cA[i]\beta^i + r\beta^i + R[0..i[= cA[i]\beta^i + c * A[0..i[$$

$$(cA[i] + r)\beta^i + R[0..i[= c * (A)[0..i+1[$$

$$\text{TMP} := \text{c*A[i]+RET}$$

$$\big((\text{TMP} \div \beta)\beta + \text{TMP}\%\beta\big)\beta^i + R[0..i[= \vdots$$

$$\text{R[i]} = \text{TMP}\% \text{ BASE}$$

$$(\text{TMP} \div \beta)\beta^{i+1} + R[i]\beta^i + R_i = \vdots$$

$$\text{RET} = \text{TMP} \div \text{BASE}$$

$$r\beta^{i+1} + R[0..i+1[= c * A[0..i+1[$$

$$i \leftarrow i + 1$$

$$r\beta^i + R[0..i[= c * A[0..i[$$

Produit de deux nombres

```
1  MULTIPLICATION( a, b : nombre de n chiffres )
2   variable              i  :  indice
3       prd , tmp : nombre de n   chiffres
4  debut
5    i ← 0
6    prd ← 0
7    tant que ( i < n ) faire
8      tmp ← CMULT( a, b[i] )
9      prd ← ADDITION( prd, tmp)
10     DECALAGE( a )
11     inc(i)
12   ftq
13  retourner   prd
14  fin
```

Complexite

L'algorithme de multiplication est quadratique en n.

Produit de deux nombres

```
1  MULTIPLICATION( a, b : nombre de n chiffres )
2   variable  i , j :  indice ; tmp: double; ret :  chiffre
3                  prd :  nombre de 2*n  chiffres
4  debut
5    prd ← 0
6    i ← 0
7    tant que ( i < n )
8      j ← 0 ; ret ← 0
9      tant que ( j < n )
10       tmp ← prd[ i + j ]  +  a[ j ] * b[ i ] + ret
11       ret ← tmp div base
12       prd[ i + j ] ←  tmp  mod base
13       inc( j )
14     ftq
15     prd[ i + n ] = ret
16     inc( i )
17   ftq
18  retourner  prd
```

Analyse et Preuve de correction

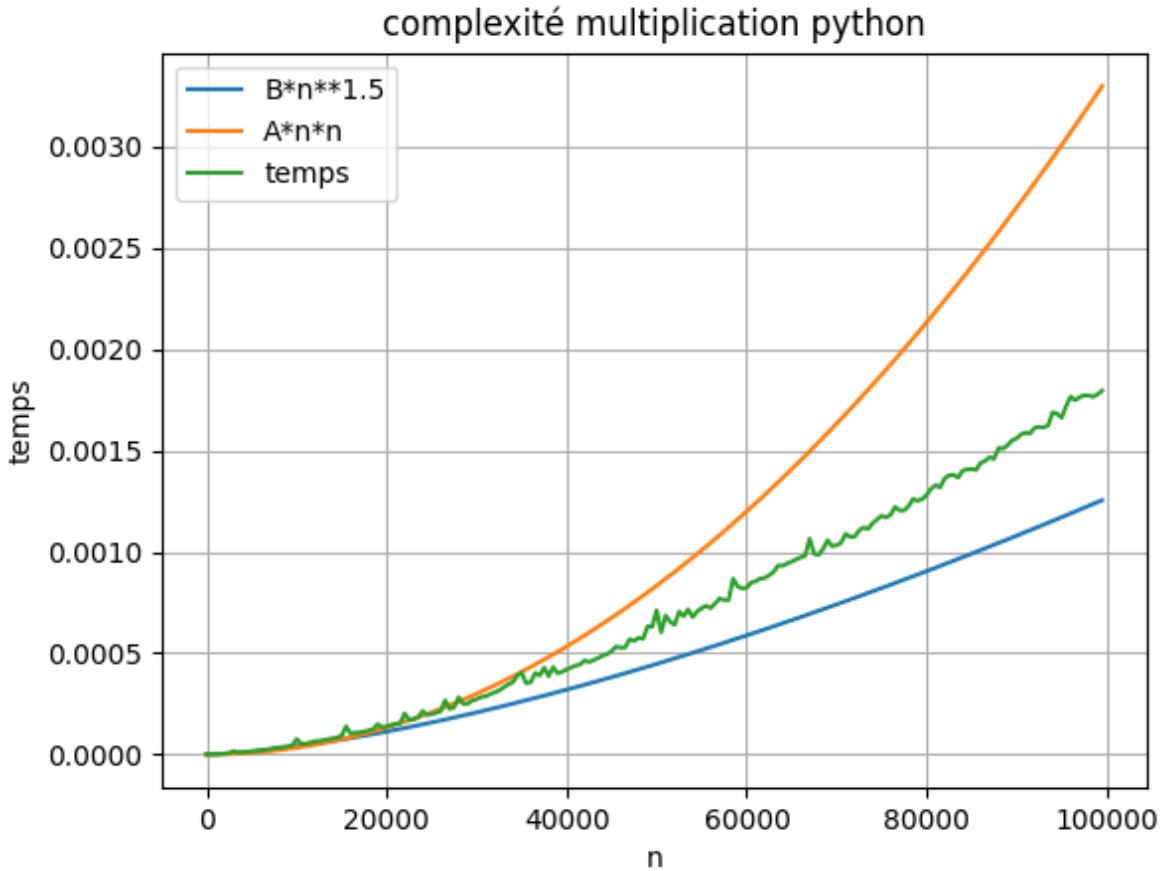
- La retenue propagée est inférieure strictement à $\textsc{base}$.
- $\textsc{taille}(a * b) \leq 2n$.
- L'arrêt est clair !
- La complexité est quadratique.
- La preuve n'a pu être faite en primaire ! !

Démonstration.

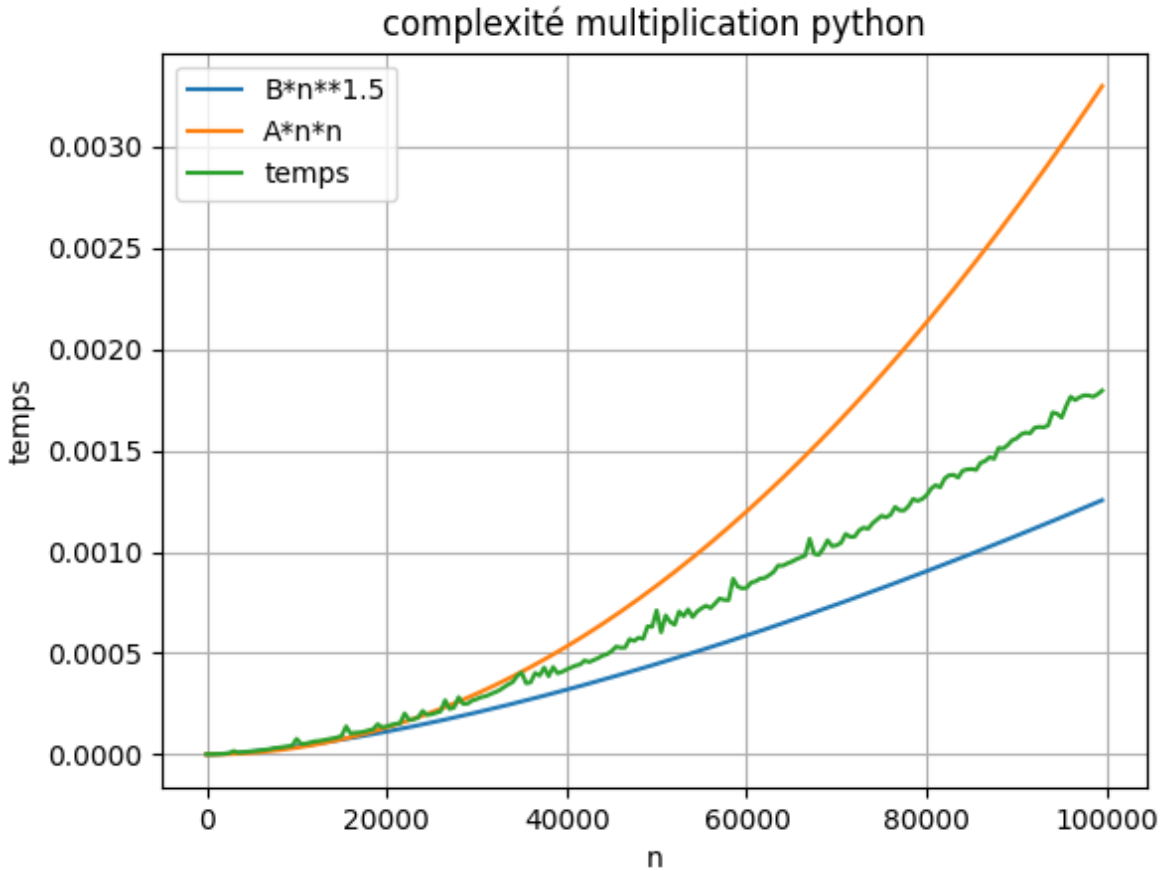
...

Peut-on faire mieux ?

Peut-on faire mieux ?

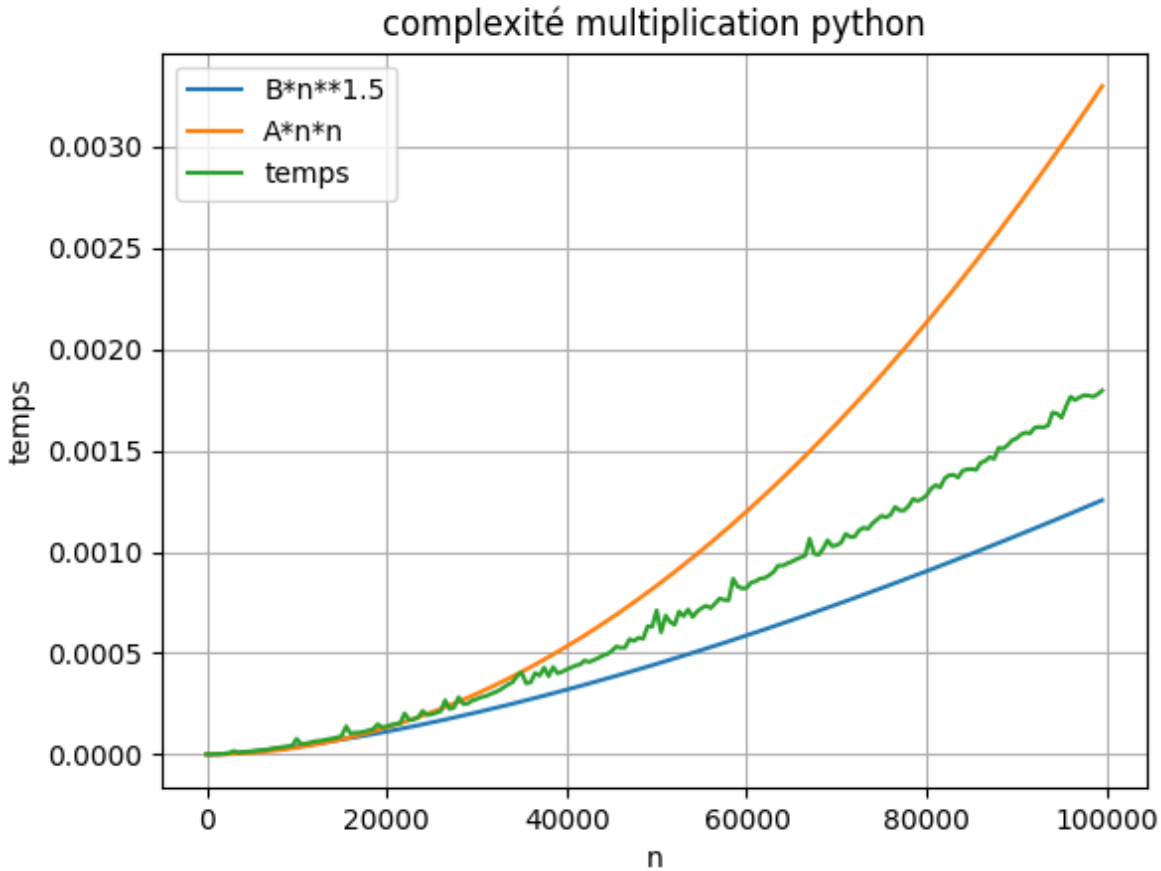


Peut-on faire mieux ?



- Karatsuba $\mathrm{O}(n\sqrt{n})$

Peut-on faire mieux ?



- Karatsuba $\mathrm{O}(n\sqrt{n})$
- Fourier $\mathrm{O}(n \log n)$

exponentiation

```
1  EXPONENTIATION( x, r : nombre )
2   variable   y : nombre
3  debut
4        y ← 1
5        tant que ( r > 0) faire
6              si impair( r ) alors
7                   y = y * x
8              fsi
9      x ← x * x
10             r ← r div 2
11       ftq
12       retourner y
13  fin
```

Complexite

Le nombre de multiplications est logarithmique en n.

Traçage des variables

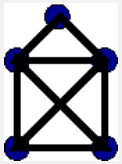
ligne	x	n	y
3	α		
⋮			
⋮			
⋮			
⋮			
⋮			
⋮			
⋮			
⋮			

Invariant ?

Invariant ?

Lemme

La valeur de $x^n y$ est invariante !



Temps de calcul

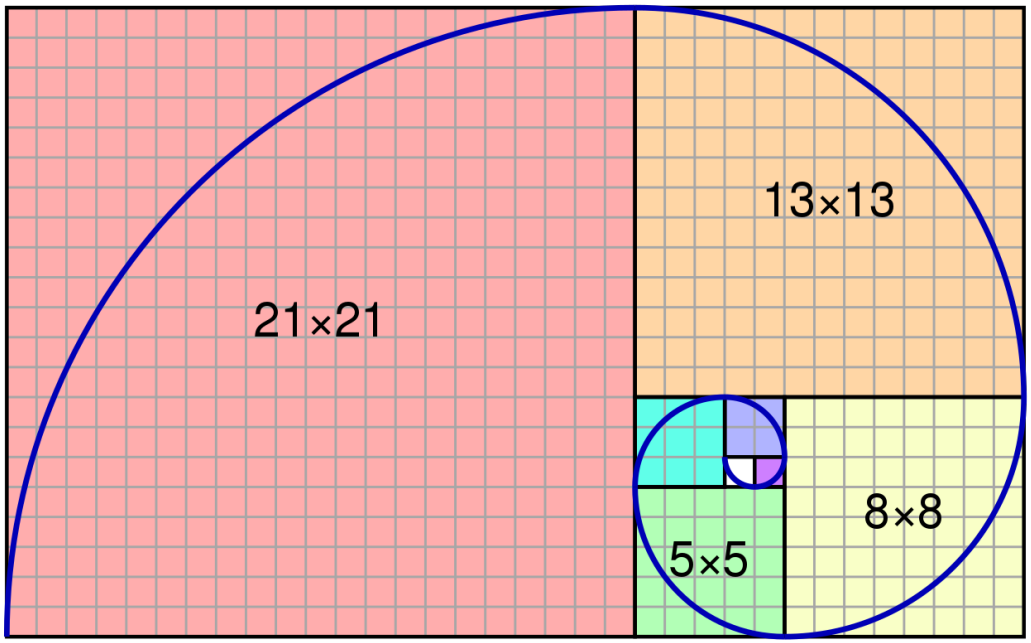
Estimer le temps de calcul de x^r pour des entiers de n chiffres en fonction de la complexité de la multiplication.

constant	Écolier	Karatsuba	Fourier
n			

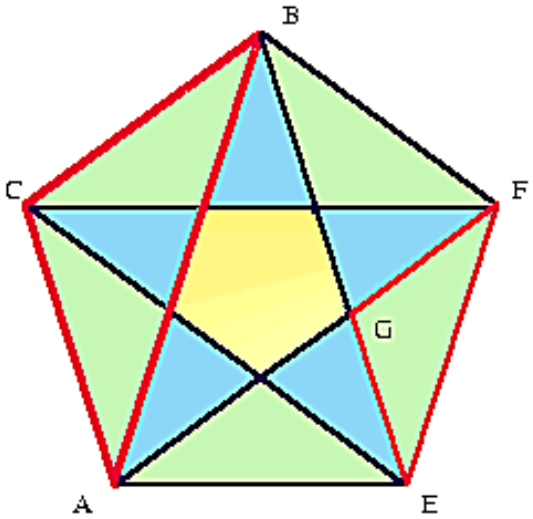
Application à la suite de Fibonacci

La suite de Fibonacci est définie par une récurrence d'ordre 2 :

$$F_0 := 0, \quad F_1 := 1, \qquad F_n := F_{n-1} + F_{n-2}$$



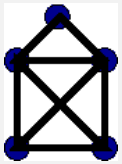
Comportement asymptotique de la suite de Fibonacci



On note ϕ la plus grande des racines du du trinôme $T^2 - T - 1$, c'est le nombre de Phidias ou nombre d'or. On note $\hat{\phi}$ la racine conjuguée à ϕ.

$$F_n = \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n)$$

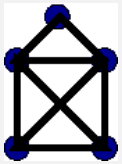
- Établir la formule reliant F_n à ϕ et $\hat{\phi}$.
- Montrer que $F_n \sim \frac{1}{\sqrt{5}}\phi^n$.
- Estimer la taille de F_n en binaire, en base 10.



Algorithme récursif

```
1 FIBONACCI( n : indice )
2 debut
3   si  ( n < 2 ) alors
4     retourner  n
5   fsi
6   retourner  FIBONACCI( n − 1) + FIBONACCI( n − 2)
7 fin
```

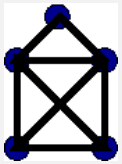
- Le temps de calcul est au moins exponentiel !
- Préciser le temps de calcul en multiprécision.



Algorithme itératif

```
1 def fibit( n ) :
2     x = 0
3     y = 1
4     i = 2
5     while  i <= n :
6         y, x = y+x, y
7         i = i + 1
8     return y
```

- En multiprécision, le temps de calcul de la version itérative est au plus quadratique. Pourquoi ?



Approche matricielle

$$F = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad F^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix}$$

- Établir la relation matricielle.
- Déduire une méthode de calcul de F_n basée sur l'exponentiation matricielle.
- Préciser le nombre de produits matriciels.
- Estimer la complexité en multiprécision.

Méthode itérative vs matricielle ?

Méthode itérative vs matricielle ?