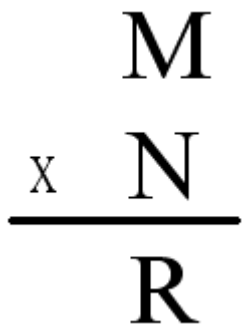
Retour sur la complexité algorithmique

Complexité algorithmique
Deuxième visite

Définition d'un algorithme

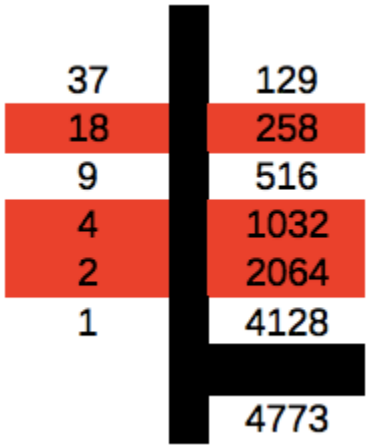
Un algorithme peut se comprendre comme un ensemble de règles permettant de résoudre un problème donné.

Exemple avec la multiplication R = M•N

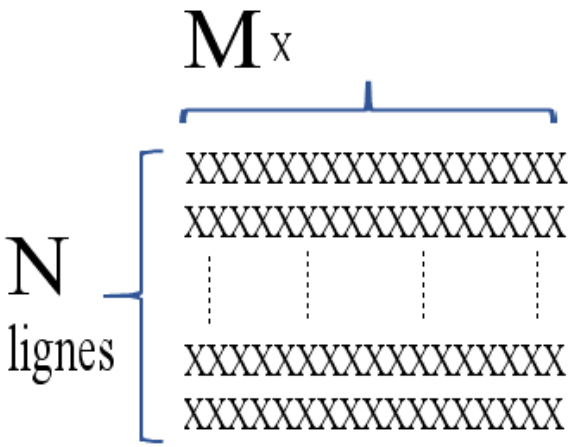


Comme à l'école

Boulier chinois



A la russe



Compter les croix

Complexité d'un algorithme

Pourquoi comparer les algorithmes ?

- ❑Pour trouver le plus rapide
- ❑Pour trouver le plus économe en mémoire
- ❑Pour trouver le meilleur compromis mémoire / vitesse

Les algorithmes doivent être justes

- ❑La terminaison du programme doit être vérifiée dans tous les cas
- ❑Ils doivent être sans 'bugs' ……. impossible ??
- ❑Ils doivent produire les résultats attendus

Complexité d'un algorithme

Comment comparer les algorithmes* ?

Un exemple : déterminer la liste des décompositions possibles d'un entier n comme somme de deux carrés

N = 1000 solutions possibles :

- $10, 30 => 10^2 + 30^2 = 1000$
- $18, 26 => 18^2 + 26^2 = 1000$

* D'après http://desaintar.free.fr/python/cours/complexite.pdf

Comparaison de quatre algorithmes

Durée de calcul en mS en fonction de n	Algorithme A1	Algorithme A2	Algorithme A3	Algorithme A4
1000	980	489	0,498	0,060
2000	3948	2017	0,953	0,080
4000	15835	7888	1,87	0,109

A vérifier avec le script : Comparaison_de_complexite.py

Comparaison de quatre algorithmes

Bilan pour la rapidité	Algorithme A1	Algorithme A2	Algorithme A3	Algorithme A4
Complexité théorique	$O(n^2)$	$O(n^2)$	$O(n)$	$O(\sqrt{n})$

Outil théorique de comparaison des complexités

La notation O (lire grand O)

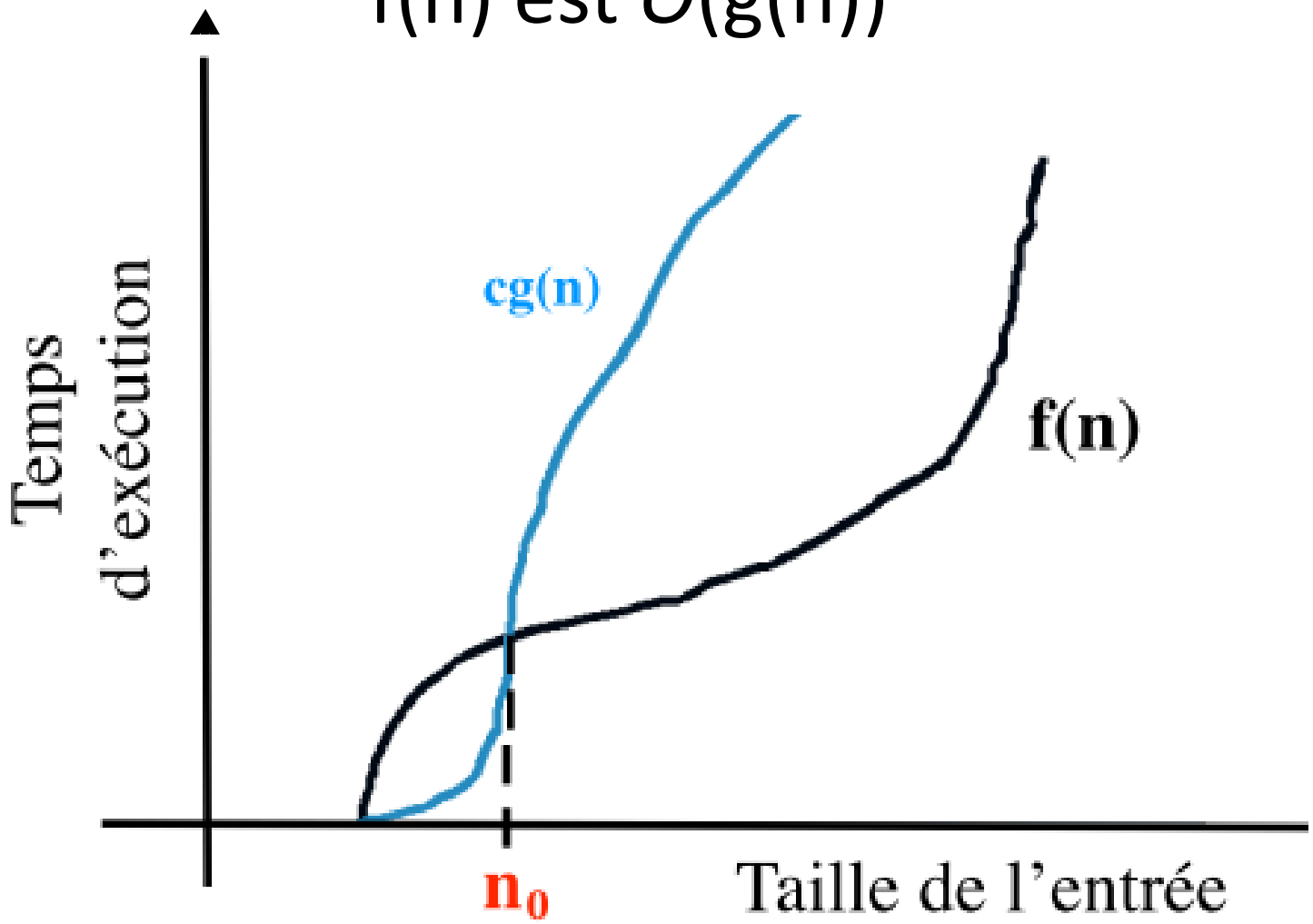
Complexité algorithmique

La notation grand *O*

$\exists n_0 \geqq 0, \exists C > 0$
tel que $\forall\ n \geqq n_0$
on a $f(n) \leqq C \cdot g(n)$

Le taux de croissance de f(n) est plus petit ou égal au taux de croissance de g(n)

f(n) est *O*(g(n))



Comparaison de taux de croissance de fonctions

$$1 \leqq \log_2(n) \leqq n \leqq n \cdot \log_2(n) \leqq n^2 \leqq 2^n \leqq n^n$$

Donc

$$O(1) \leqq O\ \log_2(n)$$

$$O\ (\log_2 n) \leqq O\ (n \cdot \log_2 n)$$

.....

Propriétés des logarithmes

$\log_a(x) = \log(x)/\log(a) = \ln(x)/\ln(a)$ logarithme en base a de x

$\log_e(x) = \ln(x)$ logarithme népérien

$\log_{10}(x) = \log(x)$ logarithme décimal

$$\log_b(x^a) = a \log_b(x)$$

$$\log_b(xy) = \log_b(x) + \log_b(y)$$

$$\log_b(x/y) = \log_b(x) - \log_b(y)$$

Opération fondamentale

Pour connaître le temps de calcul, nous choisissons une opération fondamentale et nous calculons le nombre d'opérations fondamentales exécutées par l'algorithme.

Problème	Opération fondamentale
Recherche d'un élément dans une liste	Comparaison
Tri d'une liste, d'un fichier, …	Comparaisons, déplacements
Multiplication des matrices réelles	Multiplications et additions
Addition des entiers binaires	Opération binaire

Complexité de certains problèmes courants

Complexité	Exemple
$O(1)$	Accès à un élément de tableau
$O(\log(n))$	Recherche dichotomique
$O(n)$	Recherche dans un tableau non trié
$O(n\log(n))$	Tri rapide
$O(n^2)$	Tri à bulles
$O(n^3)$	Multiplication de matrices
$O(2^n)$	Algorithme du "voyageur de commerce"

Complexité algorithmique

Que peut-on faire en fonction de la complexité

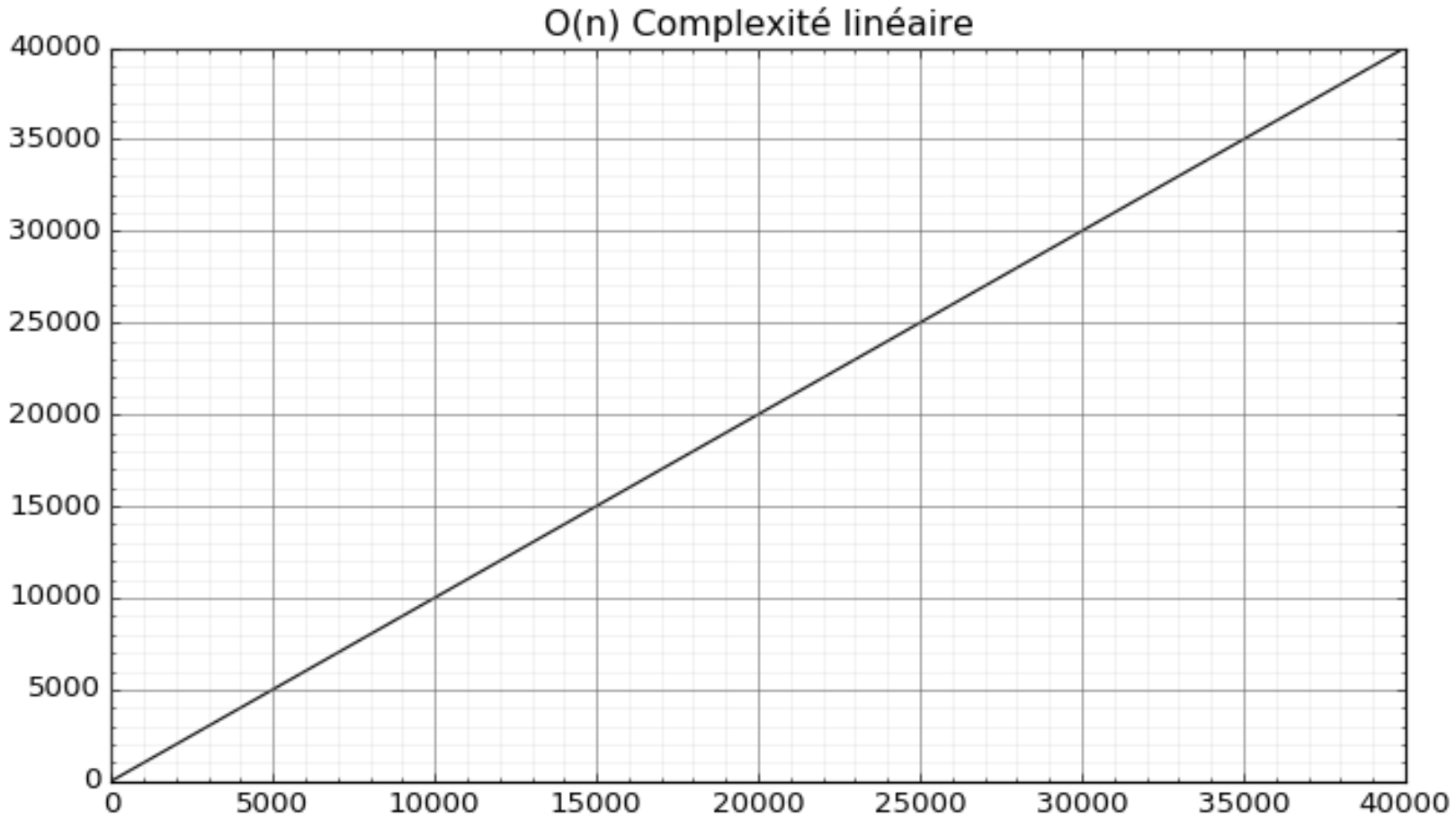
Quelle valeur de n est réaliste pour réaliser un calcul ?

Complexité	Nom courant	Temps quand on double la taille de l'entrée	Max n
$O(n)$	linéaire	prend 2 fois plus de temps	10^{12}
$O(1)$	constant	prend le même temps	pas de limite
$O(n^2)$	quadratique	prend 4 fois plus de temps	10^6
$O(n^3)$	cubique	prend 8 fois plus de temps	10 000
$O(\log n)$	logarithmique	prend seulement une étape de plus	$10^{10^{12}}$
$O(n \log n)$	linearithmique	prend deux fois plus de temps $+ \log n$	10^{11}
$O(2^n)$	exponentiel	prend tellement de temps que c'est inconcevable	30

Dans ce tableau log n représente $\log_2$ n pour une analyse de complexité ils sont équivalents

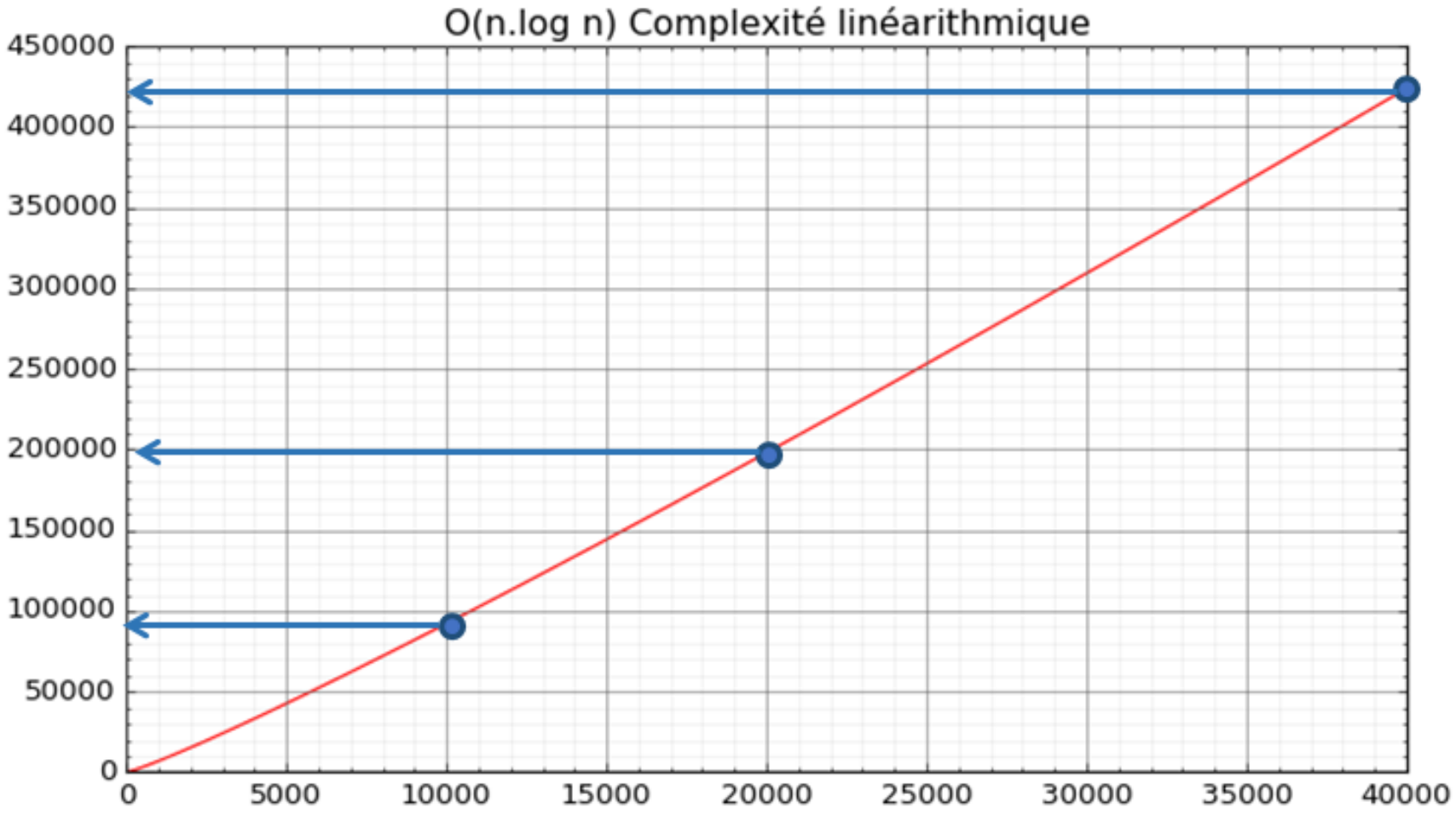
Complexité algorithmique

Analyser une complexité par le graphique



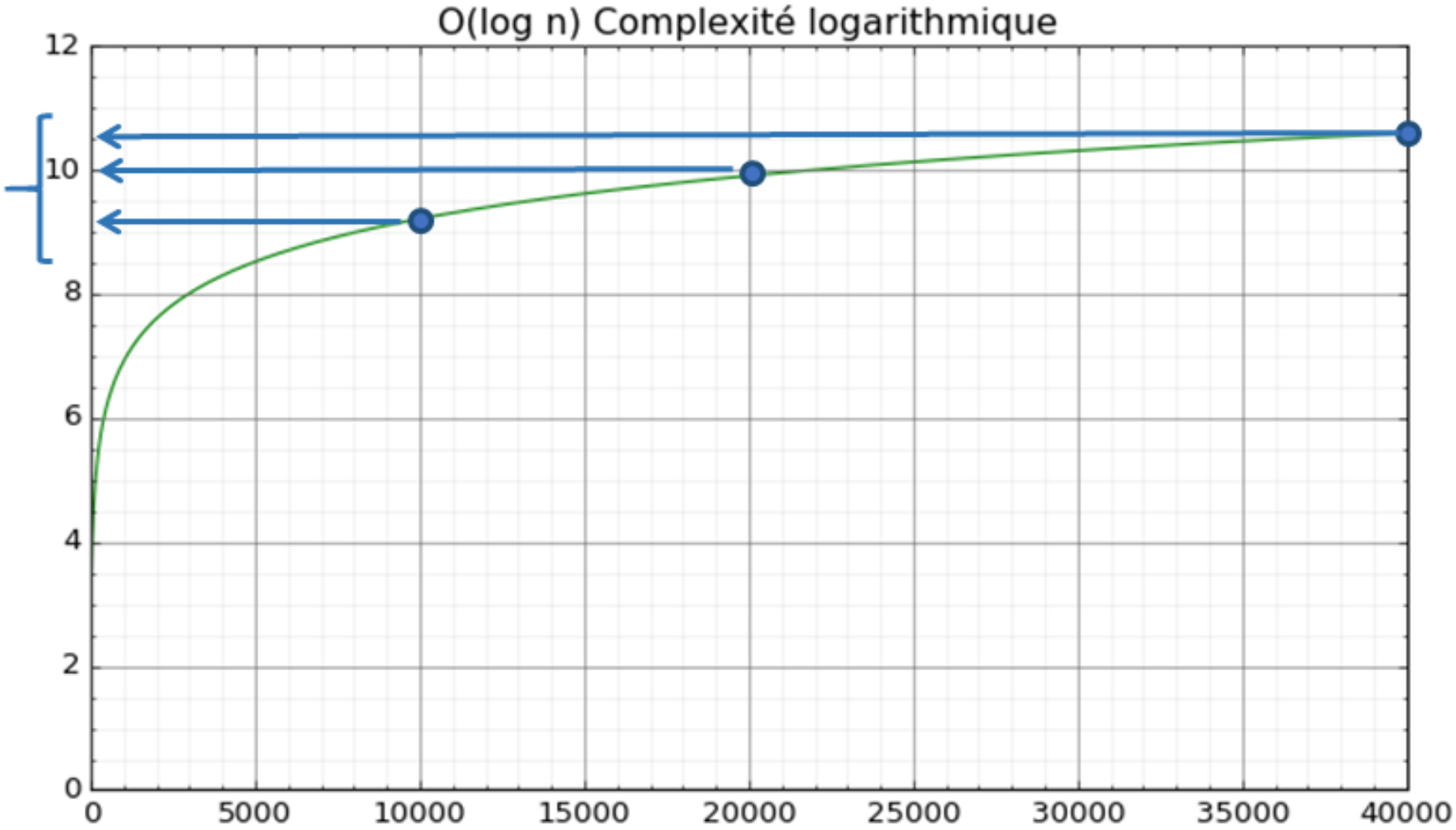
Complexité algorithmique

Analyser une complexité par le graphique



Complexité algorithmique

Analyser une complexité par le graphique



Complexité d'un algorithme

Revenir sur la comparaison de quatre algorithmes

Durée de calcul en mS en fonction de n	Algorithme A1	Algorithme A2	Algorithme A3	Algorithme A4
1000	980	489	0,498	0,060
2000	3948	2017	0,953	0,080
4000	15835	7888	1,87	0,109
Complexité théorique	$O(n^2)$	$O(n^2)$	$O(n)$	$O(\sqrt{n})$

Quelques ressources utilisées

1. **HAMEL, Sylvie.** Notation asymptotique.
http://www.iro.umontreal.ca/~hamelsyl/grandO.pdf.
[En ligne]

2. **BOUCHEZ TICHADOU, Florent.** *Complexité algorithmique*. 13 septembre 2018.

3. **TRICHET, Eric.** Introduction_complexite_algorithmique.
http://www.irem.unilim.fr/fileadmin/documents/2015_01_04-Introduction_complexite_algorithmique.pdf.
[En ligne]

4. **De SAINT JULIEN, Arnaud.** *Complexité et preuves d'algorithmes*. 9 octobre 2018.
http://desaintar.free.fr/python/cours/complexite.pdf
http://desaintar.free.fr/index.php
[En ligne]

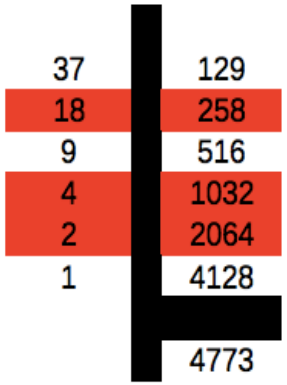
merci à leurs auteurs ….

Et pour multiplier avec ….

Un boulier chinois

https://plusbelleslesmaths.com/utiliser-boulier/

http://ecoleduboulier.com/



A la russe

http://math.lemur.pagesperso-orange.fr/math/fract/russe.htm