Ordre de grandeur du temps nécessaire à l'exécution d'un algorithme d'un type de complexité

Temps	Type de complexité	Temps pour n = 5	n = 10	n = 20	n = 50	n = 250	n = 1000	n = 10000	n = 1000000	Exemple de problème
$O(1)$	***constante***	10 ns	10 ns	10 ns	10 ns	10 ns	10 ns	10 ns	10 ns	**accès à une cellule de tableau**
$O(\log(n))$	***logarithmique***	10 ns	10 ns	10 ns	20 ns	30 ns	30 ns	40 ns	60 ns	**recherche dichotomique**
$O(\sqrt{n})$	racinaire	22 ns	32 ns	45 ns	71 ns	158 ns	316 ns	1 µs	10 µs	test de primalité naïf
$O(n)$	***linéaire***	50 ns	100 ns	200 ns	500 ns	2.5 µs	10 µs	100 µs	10 ms	**parcours de liste**
$O(n \log^*(n))$	quasi linéaire	50 ns	100 ns	200 ns	501 ns	2.5 µs	10 µs	100,5 µs	10,05 ms	triangulation de Delaunay
$O(n \log(n))$	***linéarithmique***	40 ns	100 ns	260 ns	850 ns	6 µs	30 µs	400 µs	60 ms	t**ris par comparaisons optimaux ex. tri fusion** ou tri par tas)
$O(n^2)$	***quadratique*** (polynomiale)	250 ns	1 µs	4 µs	25 µs	625 µs	10 ms	1 s	2.8 heures	**parcours de tableaux 2D**
$O(n^3)$	cubique (polynomiale)	1.25 µs	10 µs	80 µs	1.25 ms	156 ms	10 s	2.7 heures	316 ans	multiplication matricielle naïve
$2^{poly(\log(n))}$	sous-exponentielle	30 ns	100 ns	492 ns	7 µs	5 ms	10 s	3.2 ans	10^{20} ans	factorisation d'entiers avec GNFS (meilleur algorithme connu en 2018)
$2^{poly(n)}$	***exponentielle***	320 ns	10 µs	10 ms	130 jours	10^{59} ans	...	...	...	**problème du sac à dos** par force brute
$O(n!)$	factorielle	1.2 µs	36 ms	770 ans	10^{48} ans	...	...	...	...	problème du voyageur de commerce avec une approche naïve
$2^{2^{poly(n)}}$	doublement exponentielle	4.3 s	10^{278} ans	...	...	...	...	...	...	décision de l'arithmétique de Presburger

Classes de complexité

O	Type de complexité
$\mathcal{O}(1)$	constante
$\mathcal{O}(log(n))$	logarithmique
$\mathcal{O}(n)$	linéaire
$\mathcal{O}(n \times log(n))$	quasi-linéaire
$\mathcal{O}(n2)$	quadratique
$\mathcal{O}(n3)$	cubique
$\mathcal{O}(2n)$	exponentielle
$\mathcal{O}(n!)$	factorielle

CLASSES DE COMPLEXITÉ

Classe	Notation O	Exemple
Constante	O(1)	Accéder au premier élément d'un ensemble de données
Linéaire	O(n)	Parcourir un ensemble de données
Logarithmique	O(log(n))	Couper un ensemble de données en deux parties égales, puis couper ces moitiés en deux parties égales, etc.
Quasi-linéaire	O(n log(n))	Couper répétitivement un ensemble de données en deux et combiner les solutions partielles pour calculer la solution générale
Quadratique	$O(n^2)$	Parcourir un ensemble de données en utilisant deux boucles imbriquées
Polynomiale	$O(n^P)$	Parcourir un ensemble de données en utilisant P boucles imbriquées
Exponentielle	$O(a^n)$	Générer tous les sous-ensembles possibles d'un ensemble de données

14