

Plan	Introduction ○○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
------	--------------------------------------	----------------------------------	--

Complexité des algorithmes (1)

Nour-Eddine Oussous, Éric Wegrzynowski

Licence ST-A, USTL - API2

28 septembre 2009

Plan	Introduction ●○ ○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
Motivation			

Coût d'un algorithme

L'exécution d'un programme a toujours un *coût*. On distingue habituellement deux coûts :

- ▶ le temps d'exécution : *la complexité temporelle* (point sensible)
- ▶ l'espace mémoire requis : *la complexité spatiale*

Historique	Vitesse μ processeur	Mémoire
Fin 70	10 MHz	16 ko
Fin 90	$\times 40$ 400 MHz	$\times 4000$ 64 Mo
Fin 00	$\times 2.5$ 1 GHz	$\times 16$ 1 Go
Fin 04	3 GHz	1 Go

Plan	Introduction ○○ ○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
------	---	----------------------------------	--

Introduction

Motivation

Objectifs

Exemple 1

Exemple 2

Le pire, le meilleur et le moyen

Classes de complexité

Les principaux ordres de grandeur

Taux de croissance

Plan	Introduction ●○ ○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
Motivation			

Influence de la taille des données

Ces coûts c dépendent de la taille n des données à traiter

$$c = f(n)$$

f étant une fonction de \mathbb{N} dans \mathbb{R}

Plan	Introduction ○○ ●○○○ ○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○○ ○○
Objectifs			

- ▶ Proposer des méthodes pour
 - ▶ *estimer le coût* d'un algorithme
 - ▶ *comparer* deux algorithmes sans avoir à les programmer
- ▶ Estimer l'influence de la *taille des données* n sur les ressources nécessaires c .

$$c = f(n)$$

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○●○○ ○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○○ ○○
Exemple 1			

Somme des éléments d'un tableau (analyse)

Le temps de calcul de la somme dépend évidemment

1. de la taille n du tableau
2. et du nombre d'opérations élémentaires effectuées
 - ▶ affectations
 - ▶ additions d'entiers
 - ▶ accès à un élément du tableau

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○●○○ ○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○○ ○○
Exemple 1			

Somme des éléments d'un tableau (algo)

```

1 // Donnee: un tableau d'entiers A indexe de 1 a n
2 // Sortie:  $\sum_{i=1}^n A[i]$ 
3 Somme(A):
4   s := 0
5   {s =  $\sum_{k=1}^0 A[k]$ }
6   pour i variant de 1 a n faire
7     s := s + A[i]
8   {s =  $\sum_{k=1}^i A[k]$ }
9   fin pour
10  {s =  $\sum_{k=1}^n A[k]$ }
```

En retirant les commentaires :

```

1 Somme(A):
2   s := 0
3   pour i variant de 1 a n faire
4     s := s + A[i]
5   fin pour
```

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○●○○ ○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○○ ○○
Exemple 1			

Somme des éléments d'un tableau (analyse)

En notant

- ▶ $t_{:=}$ le temps d'une affectation d'entiers
- ▶ t_{+} le temps d'une addition de deux entiers
- ▶ $t_{[]}$ le temps d'un accès à un élément du tableau
- ▶ $a(n)$ le nombre d'affectations pour un tableau de taille n
- ▶ $b(n)$ le nombre d'additions pour un tableau de taille n
- ▶ $c(n)$ le nombre d'accès pour un tableau de taille n

le temps $T(n)$ de calcul s'exprime par

$$T(n) = a(n) \cdot t_{:=} + b(n) \cdot t_{+} + c(n) \cdot t_{[]}$$

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○○○● ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
Exemple 1			

Somme des éléments d'un tableau (analyse)

```

1 Somme(A) :
2   s := 0
3   pour i variant de 1 à n faire
4     s := s + A[i]
5   fin pour

```

- ▶ La ligne 4 de l'algorithme montre que
 $b(n) = c(n)$
- ▶ et en tenant compte de la ligne 2
 $a(n) = 1 + b(n)$

donc

$$T(n) = b(n) \cdot (t_{:=} + t_{+} + t_{[]}) + t_{:=}$$

Il suffit donc de déterminer $b(n)$.

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○○○ ●○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
Exemple 2			

Insertion d'un élément dans un tableau trié (algo)

Données : un tableau $A[1..MAX]$, un indice $n > 1$

CU : la tranche $A[1..n-1]$ est triée

But : placer l'élément $A[n]$ à sa place dans la tranche $A[1..n]$

Var. locales : i

```

1 insérer(A, n) :
2   i := n
3   {A[1..i-1] trié, A[i] < A[i+1..n], A[i+1..n] trié}
4   tant que i > 1 et A[i] < A[i-1] faire
5     échanger(A[i], A[i-1])
6     dec(i)
7   {A[1..i-1] trié, A[i] < A[i+1..n], A[i+1..n] trié}
8   fin tant que
9   {A[1..i-1] trié, A[i] < A[i+1..n], A[i+1..n] trié
10  et (i = 1 ou A[i] ≥ A[i-1])}

```

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○○○● ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
Exemple 1			

Somme des éléments d'un tableau (analyse)

- ▶ À chaque étape i de la boucle **pour**, il y a une addition
- ▶ donc pour les n étapes il y a

$$b(n) = \sum_{i=1}^n 1 = n$$

additions

Le coût de l'algorithme Somme est donc

$$T(n) = n \cdot (t_{:=} + t_{+} + t_{[]}) + t_{:=}$$

soit de la forme

$$T(n) = \alpha \cdot n + \beta$$

α et β ne dépendant pas de n , mais uniquement du coût des opérations élémentaires.

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○○○ ●○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
Exemple 2			

Insertion d'un élément dans un tableau trié (algo)

Sans les commentaires :

```

1 insérer(A, n) :
2   i := n
3   tant que i > 1 et A[i] < A[i-1] faire
4     échanger(A[i], A[i-1])
5     dec(i)
6   fin tant que

```

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○○○○ ○○●○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
Exemple 2			

Insertion d'un élément dans un tableau trié (analyse)

Le coût de l'insertion dépend

1. de la taille $n - 1$ de la tranche triée du tableau
2. de la valeur de l'élément $A[n]$
3. du coût des opérations de décrémentation et comparaison d'indices, d'accès à un élément du tableau, et enfin d'échange et de comparaison d'éléments du tableau.

Nous ne nous intéressons qu'à ces deux dernières opérations

- ▶ $e(n) = \text{nombre d'échanges}$
- ▶ $c(n) = \text{nombre de comparaisons d'éléments de } A$

dans l'insertion de $A[n]$ dans $A[1..n - 1]$

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○○○○ ○○○○●	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
Exemple 2			

Insertion d'un élément dans un tableau trié (coût)

Pour insérer l'élément d'indice n dans la tranche $A[1..n - 1]$:

```

1 insérer(A, n) :
2   i := n
3   tant que i > 1 et A[i] < A[i - 1] faire
4     échanger(A[i], A[i - 1])
5     dec(i)
6   fin tant que
```

- ▶ **Pire des cas** : $A[n] < A[1..n - 1]$
 - ▶ nombre échanges : $e(n) = n - 1$
 - ▶ nombre comparaisons : $c(n) = n - 1$

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○○○○ ○○●○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
Exemple 2			

Insertion d'un élément dans un tableau trié (coût)

Pour insérer l'élément d'indice n dans la tranche $A[1..n - 1]$:

```

1 insérer(A, n) :
2   i := n
3   tant que i > 1 et A[i] < A[i - 1] faire
4     échanger(A[i], A[i - 1])
5     dec(i)
6   fin tant que
```

- ▶ **Meilleur des cas** : $A[n] \geq A[1..n - 1]$ (le tableau est trié)
 - ▶ nombre échanges : $e(n) = 0$
 - ▶ nombre comparaisons : $c(n) = 1$

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
------	--------------------------------------	----------------------------------	--

Trois sortes de complexité

Pour une taille donnée n , le coût d'un algorithme peut dépendre de la donnée. On distingue alors trois cas :

- ▶ le **pire des cas** : c'est celui qui donne un coût maximal
- ▶ le **meilleur des cas** : c'est celui qui donne un coût minimal
- ▶ le **coût moyen** : c'est la moyenne des coûts sur l'ensemble de toutes les données de taille n

En analyse de complexité, on étudie souvent le pire cas ce qui donne une *majoration de la complexité* de l'algorithme.

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○○○○ ○○
------	--------------------------------------	----------------------------------	--

Remarques

- ▶ Lorsqu'on étudie la complexité d'un algorithme, on ne s'intéresse pas au temps de calcul exact mais à un *ordre de grandeur*.
- ▶ Pour une complexité polynomiale, par exemple, on ne s'intéresse qu'au terme de *plus haut degré*.

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ●○○○○○○○ ○○
Les principaux ordres de grandeur			

Logarithmique

$f(n)$ est logarithmique en n (en $\log n$)

- ▶ Exemple : recherche dichotomique dans un tableau trié $A[1..n]$

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ●○○○○○○○ ○○
Les principaux ordres de grandeur			

Les principaux ordres de grandeur

Lors de l'analyse de complexité, on essaie de situer un coût $c = f(n)$ par rapport aux ordres de grandeur suivants (donnés dans l'ordre croissant) :

1. logarithmique
2. linéaire
3. quasi-linéaire
4. polynomial (quadratique, cubique, ...)
5. exponentiel

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

Plan	Introduction ○○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ●●○○○○○ ○○
Les principaux ordres de grandeur			

Linéaire

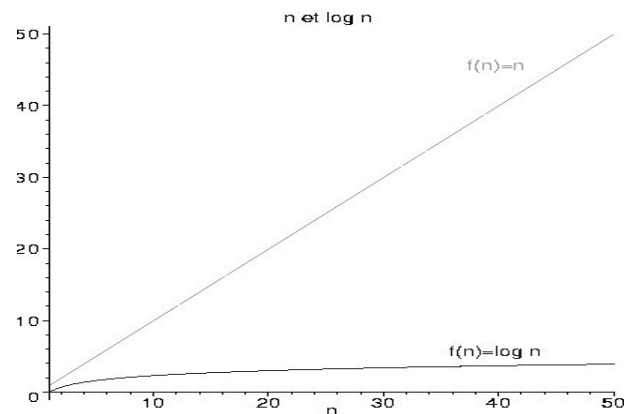
$f(n)$ est linéaire par rapport à n (de type αn)

- ▶ Exemple : calcul du produit scalaire de deux vecteurs de \mathbb{R}^n

Complexité des algorithmes (1)	Licence ST-A, USTL - API2
--------------------------------	---------------------------

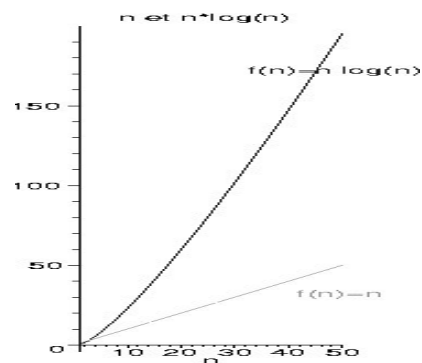
Plan	Introduction ○○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○●○○○○○ ○○
Les principaux ordres de grandeur			

Comparaison linéaire/logarithmique



Plan	Introduction ○○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○●○○○○ ○○
Les principaux ordres de grandeur			

Comparaison quasi-linéaire/linéaire



Plan	Introduction ○○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○●○○○○ ○○
Les principaux ordres de grandeur			

Complexité quasi-linéaire

$f(n)$ est de la forme $\alpha(n \log n)$

► Exemple : *Tri par fusion*

Plan	Introduction ○○ ○○○○○ ○○○○○	Le pire, le meilleur et le moyen	Classes de complexité ○○○○○○●○○○ ○○
Les principaux ordres de grandeur			

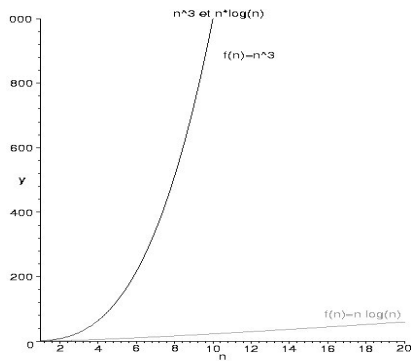
Complexité polynomiale

$f(n)$ est un polynôme dont le monôme dominant est de la forme αn^k

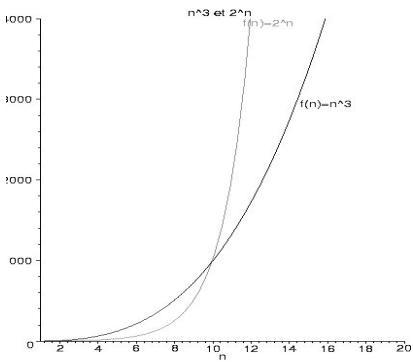
► Exemple : *Multiplication de deux matrices carrées d'ordre n par la méthode usuelle : αn^3*

Lorsque $k = 2$ ou 3 , on parle d'ordre de grandeur *quadratique* ou *cubique*

Comparaison polynomial/quasi-linéaire



Comparaison exponentiel/polynomial



Complexité exponentielle

$f(n)$ est de la forme αa^n avec $a > 1$

► Exemple : *Tours de Hanoi* : $\Theta(2^n)$

Comparaison

$T(n)$	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 60$
$\log n$	$1 \mu s$	$1,3 \mu s$	$1,5 \mu s$	$1,6 \mu s$	$1,7 \mu s$	$1,8 \mu s$
n	$10 \mu s$	$20 \mu s$	$30 \mu s$	$40 \mu s$	$50 \mu s$	$60 \mu s$
$n \log n$	$10 \mu s$	$26 \mu s$	$44 \mu s$	$64 \mu s$	$85 \mu s$	$107 \mu s$
n^2	$100 \mu s$	$400 \mu s$	$900 \mu s$	$1,6 ms$	$2,5 ms$	$3,6 ms$
n^3	$1 ms$	$8 ms$	$27 ms$	$64 ms$	$125 ms$	$216 ms$
n^5	$0,1 s$	$3 s$	$24 s$	$1,7 mn$	$5 mn$	$13 mn$
2^n	$1 ms$	$1 s$	$18 mn$	13 jours	36 ans	366 siècles
3^n	$60 ms$	1 heure	6 ans	3900 siècles	$2 \times 10^8 \text{ siècles}$	$1,3 \times 10^{13} \text{ siècles}$

Effet d’une amélioration technologique

Taille des données traitées à *temps constant*

$T(n)$	Aujourd’hui	100 fois + rapide	1000 fois + rapide
$\log n$	N	N^{100}	N^{1000}
n	N	$100\ N$	$1000\ N$
n^2	N	$10\ N$	$32\ N$
n^3	N	$4,6\ N$	$10\ N$
n^5	N	$2,5\ N$	$4\ N$
2^n	N	$N + 7$	$N + 10$
3^n	N	$N + 4$	$N + 6$