Cours2
Complexité d’un algorithme

Année 2022-2023 J-F. Kamp - IUT de Vannes

PLAN

- Problème versus algorithme
- Comparaison d’algorithmes
- Complexité théorique des algorithmes
- Ordre de grandeur
- Principales classes de complexité
- Limites de la complexité

Notion de complexité

Problème et algorithmes

Un problème peut généralement être résolu de +sieurs façons.

Donc il existe +sieurs algorithmes différents pour résoudre un même problème.

Exemple : calcul du factoriel

n! = 1 x 2 x 3 x … x (n – 1) x n

En math : f(n) telle que

f(0) = 1

f(n) = 1 x 2 x 3 x … x (n – 1) x n

Problème et algorithmes

Premier algorithme possible : l'algorithme itératif.

```
int factorielle ( int n ) {

    int ret, i:

    ret = 1;

    for ( i = 1; i <= n; i++ ) {

        ret = ret ∗ i;
    }

    return ret;
}
```

Problème et algorithmes

Deuxième algorithme possible : la récursivité.

```
int factorielle ( int n ) {

    int ret;

    if ( n == 0 ) {
        ret = 1;
    }

    else {

        ret = n * factorielle ( n – 1 );
    }

    return ret;
}
```

Quel est l’algorithme le + performant : récursif ou itératif ?

Comparaison des algos

Sur quels critères comparer les algorithmes ?

- L’espace mémoire occupé par le fichier contenant le code ?
- L’espace mémoire occupé par le fichier compilé ?
- La quantité de mémoire vive occupée lors de l’exécution ?
- Le nombre d’octets transférés sur le réseau ?
- **Le temps de calcul.**

Critère de comparaison
Le temps de calcul

Première suggestion : chronométrer le temps de calcul.

Mais ça va dépendre :

- De la machine (brouette ou Ferrari)
- Du nombre de données traitées
- De la fréquence d’horloge
- Temps d’accès aux données
- …

Critère de comparaison
Le temps de calcul

Le temps de calcul va dépendre de la complexité (taille) du travail à accomplir.

Exemple : pour un algorithme de tri, la complexité va dépendre du nombre N de valeurs à trier.

Si N est la taille du problème, on voudrait exprimer le temps de calcul en fonction de N.

Le temps de calcul empirique

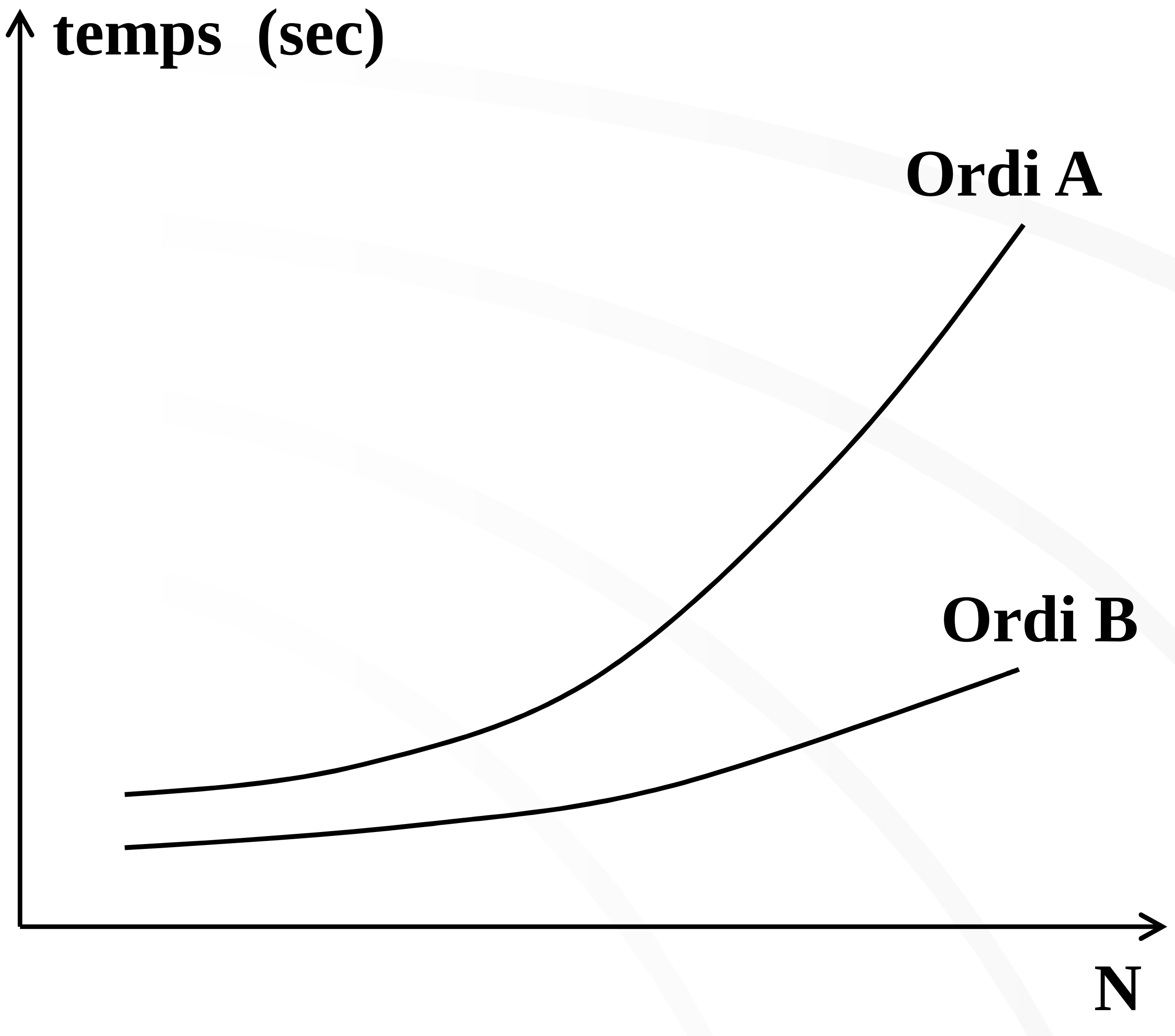
Exemple

Mesures du temps de calcul (en secondes) pour un même algo. de tri sur 2 machines A et B différentes.

N	Ordi. A	Ordi. B
125	12,5	2,8
250	49,3	11
500	195,8	43,4
1000	780,3	172,9
2000	3114,9	690,5

Le temps de calcul empirique

Ajustement de courbes sur les données.



Le temps de calcul empirique

- Mise en équation

Ordi A

$t(N) = 0{,}00078\ N^2 + 0{,}003\ N + 0{,}001$

Ordi B

$t(N) = 0{,}00017\ N^2 + 0{,}0004\ N + 0{,}01$

- Observations

Les courbes sont en N^2 pour les 2 ordis.

Si $N > 500$, $t(N)$ ne dépend que du terme en N^2.

Le temps de calcul empirique

La variation du temps de calcul en fonction de la taille (ici N = nbre de valeurs à trier) est appelée complexité de l’algorithme.

Dans cet exemple, il s’agit d’une complexité empirique car fondée sur des mesures (ici du temps).

Complexité théorique

Il est possible de déterminer les équations précédentes de manière théorique pour une bonne majorité des problèmes.

Idée : compter les opérations élémentaires exécutées par l'algorithme en fonction de N (taille du problème).

Hypothèse (forte) : **une opération élémentaire s'exécute en un temps élémentaire** qui est 1 coup d'horloge (T = 1/F, F = fréquence en Hz).

Complexité théorique

Exemple de calcul théorique pour la factorielle.

Hypothèses (fixons les règles) :

- Une déclaration vaut 1 opération élémentaire
- Une affectation vaut 1 opération élém.
- Une addition ou multiplication vaut 1 opération élém.
- Une incrémentation (i++) ou décrémentation (i--) vaut 2 opérations élém.
- Un test de comparaison (<=, <, >=, >, !=, ==) vaut 1 opération élém.

Complexité théorique

```
int factorielle ( int n ) {

    int ret, i:                       // 2 op

    ret = 1;                          // 1 op
    i = 1;                            // 1 op

    while ( i <= n ) {                // 1 op

        ret = ret * i;                // 2 op
        i++;                          // 2 op
    }

    return ret;
}
```

- Avant la boucle : 2 + 1 + 1 = 4 op
- Boucle effectuée (n – 1 + 1) = n fois
- Corps de la boucle : 2 + 2 = 4 op
- Comparaison : 1 op
- Comparaison effectuée (n + 1) fois

Complexité théorique

- Avant la boucle : 2 + 1 + 1 = 4 op
- Boucle effectuée (n – 1 + 1) = n fois
- Corps de la boucle : 2 + 2 = 4 op
- Comparaison : 1 op
- Comparaison effectuée (n + 1) fois

Nbre opérations élémentaires = f (n)

= 4 + (n x 4) + (n + 1) x 1

= 5 + 5n

Temps de calcul : t (n) = f (n) x T

où T = 1/F (F = fréquence de l’horloge Hz)

Conclusion : le temps de calcul théorique de la factorielle est linéaire en « n ».

Une meilleure intuition

Supposons une horloge à 1MHz

=> $T = 10^{-6}$ secondes

- Premier algo. de tri (tri1)

  $t(n) = n^2 \times T$

- Second algo. de tri (tri2)

  $t(n) = n \log_2 (n) \times T$

Soit $n = 10^5$ (100.000)

- tri1 : $t(n) = 10^{10} \times 10^{-6} = 10^4$ secondes = 2,7 heures
- tri2 : $t(n) = 16,6 \times 10^5 \times 10^{-6} = 1,66$ secondes

Ordre de grandeur

Nbre opérations élémentaires

$f(n) = 5 + 5n$

L'expression **exacte** n'a pas beaucoup d'intérêt car :

- La constante « 5 » n'est pas significative elle concerne des détails d'implémentation.
- Le coût en temps sera d'autant plus important que **n** est grand et la constante n'a plus aucun poids dans le calcul.

On fera l'approximation suivante : pour **n** grand $\mathbf{f(n) \approx 5n}$, donc la complexité de l'algorithme « factorielle » est linéaire en **n**, ce que l'on notera $\mathbf{\Theta(n)}$.

Classes de complexité

Θ	Noms	Exemples
$\Theta(1)$	constante	tab[i]
$\Theta(\log n)$	logarithmique	rech. dichotom.
$\Theta(n)$	linéaire	factorielle
$\Theta(n \log n)$	nlogn	tris rapide
$\Theta(n^2)$	quadratique	tris simple
$\Theta(n^3)$	cubique	3 bcles imb.
$\Theta(2^n)$	exponentielle	IA (cavalier)

Limites de la complexité

Pour **n** petit, les constantes cachées peuvent faire la différence.

Exemple :

Soit algo1 : $f(n) = 20 + 100\, n \log_2(n)$

Soit algo2 : $f(n) = 40 + 2\, n^2$

algo1 est en $\Theta(n\log n)$

algo2 est en $\Theta(n^2)$

Pour $n = 8$ (donc petit) on a :

algo1 : $f(8) = 100 \times 8 \times 3 + 20 = 2420$ op

algo2 : $f(8) = 40 + 2 \times (8)^2 = 168$ op

Qui est le meilleur ?

Limites de la complexité