Algorithmique fonctionnelle - Complexité

Complexité

Présentation

Un bon algorithme

- **répond correctement au problème posé**
- rapide (complexité temporelle)
- peu gourmand en mémoire (complexité spatiale)

Les performances dépendent

- de la taille, structure de données en entrée
- du nombre d'opérations élémentaires exécutées
  - opérations arithmétiques
  - affectations
  - instructions de contrôles

Complexité

But

La complexité algorithmique permet de mesurer la performance d'un algorithme et de la comparer avec d'autres algorithmes réalisant les mêmes fonctionnalités.

Exemple

Pour un dictionnaire de 10^6 mots, et un test d'égalité qui prend une milliseconde de temps de calcule :

- recherche séquentielle

$$5.10^5 x 10^{-3} = 500s$$

- recherche dichotomique

$$log_2 10^6 x 10^{-3} = 20 millisecondes$$

Notations asymptotiques

On dit qu'une fonction g **est dominée asymptotiquement** par f, noté $g \in O(f)$ ssi :
$\exists C \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}^+ : g(n) \leq C * f(n) \ \forall n \geq n_0$

On dit qu'une fonction g **domine asymptotiquement** f, noté $g \in \Omega(f)$ ssi $f \in O(g)$, c'est-à-dire :
$\exists C \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}^+ : g(n) \geq C * f(n) \ \forall n \geq n_0$

On dit qu'une fonction g **est asymptotiquement équivalente** à f, noté $g \in \Theta(f)$ ssi :
$g \in O(f)$ et $g \in \Omega(f)$

Exemples

$$f(n) = \frac{n^3}{2},\ g_1(n) = n^2 + 12n + 8,\ g_2(n) = 5n^3$$

Exercice : Trouvez des constantes C et n_0 pour montrer que

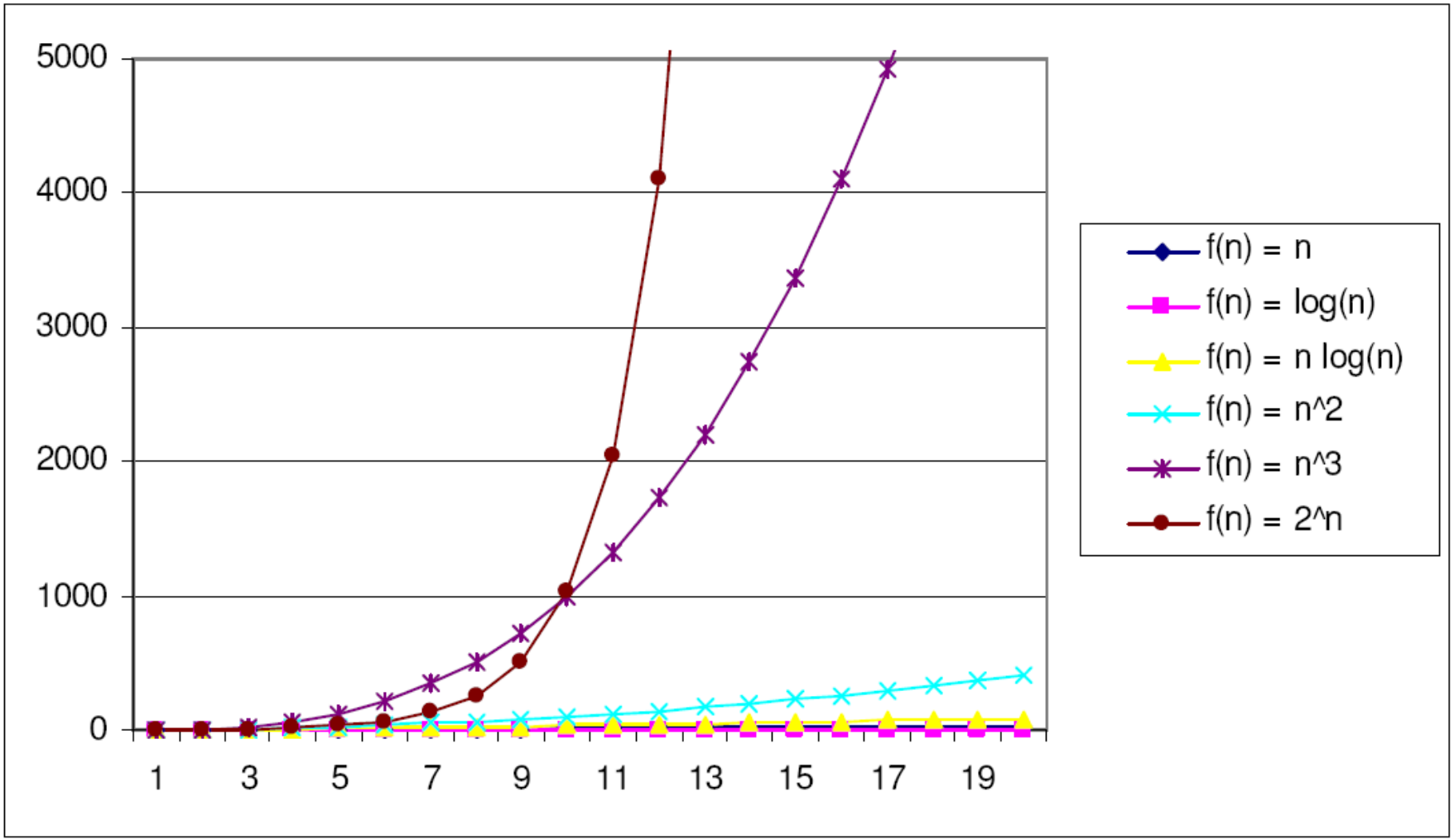
- $f \in O(g_2)$
- $g_2 \in O(f)$
- $g_1 \in O(n^2)$
- $g_1 \in O(f)$

Exercice : Prouver que $f \notin O(n^2)$

Complexité

Types de complexité

- $O(1)$: complexité constante
- $O(logn)$: complexité logarithmique
- $O(n)$: complexité linéaire
- $O(nlogn)$: complexité quasi-linéaire
- $O(n^k)$: complexité polynomiale
- $O(e^n)$: complexité exponentielle
- $O(n!)$: complexité factorielle
- ...


5000
4000
3000
2000
1000
0
1
3
5
7
9
11
13
15
17
19
f(n) = n
f(n) = log(n)
f(n) = n log(n)
f(n) = n^2
f(n) = n^3
f(n) = 2^n

Complexité

Calcul précis de la complexité ?

Hypothèses :

- Opérations élémentaires
  - affectation
  - comparaison de scalaires
- Instructions conditionnelles
  - meilleur cas
  - **pire cas**
  - cas moyen

Complexité

Pire, meilleur cas, cas moyen

Soit d une donnée de taille n, et $C(d)$ un nombre d'opérations élémentaires pour exécuter l'algorithme sur d. On distingue 3 cas de complexité :

- pire des cas : $Max_d C(d)$
- meilleur des cas : $Min_d C(d)$
- cas moyen : $\sum_d p(d)C(d)$
  où $p(d)$ est la probabilité d'avoir en entrée une instance d parmi toutes les données de taille n

Complexité

Evaluer la complexité des fonctions récursives

1. Choisir l'opération élémentaire
2. Déterminer une équation de récurrence en fonction de nombre d'opérations élémentaires

$$\begin{cases} \textit{cas particulières} \\ \textit{cas général} \end{cases}$$

3. Résoudre l'équation

Exemple 1

Factorielle

Fonction facto(n : Entier) : Entier
Début
  Si n $\leq$ 1 **Alors**
    Retourner 1
  Sinon
    Retournern*(facto(n-1))
  FinSi
Fin

Exemple 1

Solution

1. On choisit la comparaison comme opération élémentaire
2. On note $C(n)$ le nombre de comparaisons au rang n

$$\begin{cases} C(1) = 1 \\ C(n) = 1 + C(n-1) \end{cases}$$

3. $C(n) = 1 + C(n-1)$
   $C(n-1) = 1 + C(n-2)$
   ...

   $\Rightarrow C(n) = n$
   $\Rightarrow$ complexité : $O(n)$

Exemple 2

Recherche dichotomique

La recherche dichotomique fait partie des problématiques **"diviser pour mieux régner"**. Le problème est le suivant : *On désire deviner en un nombre nb, minimal, de coups, un nombre x compris entre 0 et 1000.*
La solution est d'annoncer le nombre médian des 2 bornes afin qu'à chaque tentative, on puisse éliminer la moitié des possibilités restantes.

Exemple 2

Recherche dichotomique

Fonction dicho(x : Entier, deb : Entier, fin : Entier, nb : Entier) : Entier
Variables
  mil : Entier
Début
  mil $\leftarrow$ (deb+fin)/2
  Si (x = mil) **Alors**
    retourner nb
  SinonSi (x > mil)
    Retourner dicho(x, $mil + 1$, fin, $nb + 1$)
  Sinon
    Retourner dicho(x, deb, $mil - 1$, $nb + 1$)
  FinSi
Fin

Cherchons un nombre entre 0 et 1000

Appel : dicho(12, 0, 1000, 1)

Exemple 2

Solution

1. On choisit la comparaison comme opération élémentaire
2. On note $C(n)$ le nombre de comparaisons au rang n, où n est le nombre de valeurs entre deb et fin

$$\begin{cases} C(1) = 1 \\ C(n) = 1 + C(n/2) \end{cases}$$

Exemple 2

Solution

On opère un changement de variable ;
posons $n = 2^k$, on obtient :

$$C(2^k) = 1 + C(2^{k-1})$$
$$C(2^{k-1}) = 1 + C(2^{k-2})$$
$$\ldots$$
$$C(2^1) = 1 + C(2^0) = 1 + 1$$
$$\Rightarrow C(2^k) = k + 1$$

$\Rightarrow C(n) = log_2(n) + 1$
$\Rightarrow$ complexité : $O(log_2 n)$

Exemple 3

Fibonacci

```
Fonction fibo (n : Entier) : Entier
Début
   Si n < 2 Alors
      Retourner n
   Sinon
      Retourner fibo(n-1) + fibo(n-2)
   FinSi
Fin
```

Exemple 3

Solution

1. On choisit la compairison comme opération élémentaire
2. On note $C(n)$ le nombre d'additions au rang n

$$\begin{cases} C(0) = 1 \\ C(1) = 1 \\ C(n) = 1 + C(n-1) + C(n-2) \end{cases}$$

3. $C(n) = 1 + C(n-1) + C(n-2) < 1 + 2C(n-1) < 1 + 2(1 + C(n-2)) < ...$
   $\Rightarrow C(n) < 1 + 2 + 4 + 8 + ... + 2^{n-1} = 2^n - 1$
   $\Rightarrow$ complexité : $O(2^n)$
   Plus compliqué à montrer : $O((\frac{1+\sqrt{5}}{2})^n)$

Exemple 3

Fibonacci récursif non terminal : Complexité exponentielle

Addition de 2 entiers : 100ns (10^{-7} s)

n	temps
20	1,3 ms
30	1,9s
50	40 min
100	2 millions d'années !