Université Toulouse III – Paul sabatier
L2 Informatique

Complexité des algorithmes

Semestre 3

Toulouse, le 18 octobre 2012

Table des matières

Chapitre 1

Introduction

1.1 Complexité

On cherche à estimer le temps de calcul d'un algorithme A en fonction d'un paramètre n. Pour avoir une mesure indépendante de la machine, on identifie le temps de calcul avec le nombre d'instructions exécutées.

Ex Le paramètre n pourrait être la taille d'un tableau, par exemple.

Soit D_i l'ensemble des données possibles telle que $n = i$. Pour $d \in D_i$ on notera $T(A, d)$ le nombre d'instructions exécutée pendant l'exécution de $A(d)$.
On notera $\text{prob}(d|i)$ la probabilité que les données soit d étant donné qu'elles sont de taille i.

1.1.1 La complexité temporelle maximale

La complexité temporelle maximale [1] d'un algorithme A :

$$T_{\max}(i) = \max_{d \in D_i}\{T(A, d)\}$$

1.1.2 La complexité temporelle moyenne

La complexité temporelle moyenne [2] d'un algorithme A :

$$T_{\text{moy}} = \sum_{d \in D_i} \text{prob}(d|i) \times T(A, d)$$

R Pour pouvoir calculer T_{moy}, il faut connaître la distribution des données, ce qui n'est pas toujours évident (par exemple en traitement d'image)

1. Complexité dans le pire des cas
2. Complexité dans le cas moyen

1.1.3 La complexité temporelle minimale

La complexité temporelle minimale[3] d'un algorithme A :

$$T_{\min}(i) = \min_{d \in D_i}\{T(A, d)\}$$

R Peu utilisé, sauf pour prouver qu'un algorithme est mauvais. Si la complexité temporelle minimale est mauvaise même dans le meilleur des cas, alors l'algorithme n'est pas bon.

1.1.4 Compairaison de complexités en fonction de la machine

Complexité	**Nombre d'instructions pouvant executer la machine**	
	1 000 000	1 000 000 000 000
n	1 000 000	1 000 000 000 000
$n \log_2 n$	64 000	32 000 000 000
n^2	1 000	1 000 000
n^3	100	10 000
2^n	20	40

1.2 Complexité asymptotique

Pour comparer des algorithmes, on ne s'intéresse qu'à leur comportements pour n grand. On cherche une mesure de complexité qui soit indépendante du langage de programmation et de la vitesse de la machine.
⇒ On ne doit pas perdre en compte des facteurs constants.
⇒ Ordre de grandeur

1.2.1 La complexité asymptotique

La complexité asymptotique[4] est l'ordre de grandeur de sa limite lorsque $n \to \infty$

1.2.2 Notation

Soient T, f des fonctions positives ou nulles. Rotations de grandeur de fonction asymptotiques.

Grand O $T = O(f)$ si $\exists c \in \mathbb{R}^{>0}$ et $n_0 \in \mathbb{N}$ tels que $\forall n \geq n_0$, $T(n) \leq cf(n)$.

3. Complexité dans le meilleur des cas
4. Que ce soit maximale, moyenne ou minimale

Grand Oméga $T = \Omega(f)$ si $\exists c \in \mathbb{R}^{>0}$ et $n_0 \in \mathbb{N}$ tels que $i \forall n \geq n_0, T(n) \geq cf(n)$

Petit O $T = o(f)$ si $\frac{T(n)}{f(n)} \to O$ lorsque $n \to \infty$.

R T est négligeable devant f

Ex

1. $2n^2 + 5n + 10 = O(n^2)$
   Dans la définition $n_0 = 5, c = 4$:
   $\forall n \geq 5,\ 2n^2 + 5n + 10 \leq 4n^2$
2. $2n^2 + 5n + 10 = \Omega(n^2)$
   Dans la définition, $n_0 = 1,\ c = 2$
   $\forall n \geq 1,\ 2n^2 + 5n + 10 \geq 2n^2 \cdots$
   Donc $2n^2 + 5n + 10 = \Theta(n^2)$
3. $\frac{1}{5} + n = O(n \log_2 n)\ (n_0 = 2,\ c = 2)$
4. $\frac{1}{5} n \log_2 n + n = \Omega(n \log n)\ (n_0 = 1, c = \frac{1}{5})$
5. $\forall k \geq 0,\ n^k = O(n^{k+1})$ mais $n^k \neq \Omega(n^{k+1})$
6. $\forall a, b > 1, \log_a n = \Theta(\log_b n)$ car $\log_a n = \frac{\log_b n}{\log_b a}$ et $\log_b a$ est une constante.
   $\Rightarrow$ On a pas besoin de préciser la base de logarithme dnas une complexité asymptotique
7. $2n^2 + 5n + 10 = 2n^2 + 0(n^2)$
8. Pour toute constante $c > 0$, $C = \Theta(1)$
9. $2^n = o(3^n)$

R

1. O et Ω sont des pré-ordres [a] :
   $f = O(f)$ et $f = O(g)$ et $g = O(h)) \Rightarrow f = O(h)$
2. Θ est une relation d'équivalence [b] : $f = \Theta(g) \Leftrightarrow g = \Theta(f)$

a. Relations reflexives et transitives
b. relation reflexives, symétrique et transitive

Proposition

$$\text{Si } \lim_{n \to \infty} \frac{f(n)}{g(n)} = a > 0 \text{ Alors } f = \Theta(g)$$

R La réciproque est fausse

Notation

$$f \sim g \Rightarrow \lim_{n \to \infty} \frac{f(n)}{g(n)} = 1$$

Ex $(3n+1)^3 \sim 27n^3$

1.3 Exemple de complexités d'algorithmes

1.3.1 Le tri à bulles

$$\begin{aligned} T_{\min}(n) &= \Theta(n) \text{ Si le tableau est est déjà trié} \\ T_{\max}(n) &= \Theta(n^2) \text{ Si le tableau est trié en ordre décroissant} \\ T_{\text{moy}}(n) &= T_{\max}(n) = \Theta(n^2) \end{aligned}$$

1.3.2 Tri par fusion

$$T_{\min}(n) = T_{\max}(n) = T_{\text{moy}}(n) = \Theta(n \log n)$$

1.3.3 Tri rapide

$$\begin{aligned} T_{\min}(n) = T_{\text{moy}}(n) &= \Theta(n \log n) \\ T_{\max}(n) &= \Theta(n^2) \end{aligned}$$

Chapitre
2

Complexité des boucles

Chapitre 3

Complexité d'algorithmes définis par réccurence

Chapitre
4

Structure de données et complexité

Annexe A

Exercices

A.1 TD 1