IFT2125 - Introduction à l'algorithmique

L'analyse des algorithmes (BB, chapitres 3 et 4)

Pierre McKenzie

DIRO, Université de Montréal

Hiver 2018

Re-rappel : les ordres

Soit $f : \mathbb{N} \to \mathbb{R}^{\geq 0}$.

L'ordre de f est
$O(f) = \{t : \mathbb{N} \to \mathbb{R}^{\geq 0} \mid (\exists c \in \mathbb{R}^{\geq 0})(\underbrace{\overset{\infty}{\forall}}_{\substack{\text{pour tous les } n \\ \text{suffisamment grands}}} n \in \mathbb{N})[t(n) \leq cf(n)]\}$

L'oméga de f est
$\Omega(f) = \{t : \mathbb{N} \to \mathbb{R}^{\geq 0} \mid (\exists d \in \mathbb{R}^{+})(\overset{\infty}{\forall} n \in \mathbb{N})[t(n) \geq df(n)]\}$

L'ordre exact de f est
$\Theta(f) = O(f) \cap \Omega(f)$.

Outils disponibles [1] :

---

88 Asymptotic Notation Chapter 3

1. if $\lim_{n\to\infty} \frac{f(n)}{g(n)} \in \mathbb{R}^+$ then $f(n) \in \Theta(g(n))$,
2. if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ then $f(n) \in O(g(n))$ but $f(n) \notin \Theta(g(n))$, and
3. if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty$ then $f(n) \in \Omega(g(n))$ but $f(n) \notin \Theta(g(n))$.

---

1. Passage extrait de BB, de même que tout passage de mes transparents visiblement reproduit d'un livre et non explicitement attribué.

Notre approche pour l'analyse d'algos, en résumé

- Analytique
- En pire cas
- Souvent on comptera le nombre b d'exécutions d'une instruction baromètre
- On estimera $t(n)$ à l'ordre près, si possible à l'ordre exact-près
  - une f qui $\overset{\infty}{\forall}\, n$ et sur tout exemplaire de taille n majore b de cet exemplaire vérifie $t(n) \in O(f(n))$
  - une g qui $\overset{\infty}{\forall}\, n$ et sur au moins un exemplaire de taille n minore b de cet exemplaire vérifie $t(n) \in \Omega(g(n))$
  - pourvu ci-dessus que $g \in \Omega(f)$ on conclut $t(n) \in \Theta(f(n))$

Algorithmique versus théorie de la complexité du calcul

Un problème P est donné.

Ressort de l'algorithmique :

- développer un algorithme A efficace pour résoudre P
- déterminer l'ordre exact du temps d'exécution de l'algorithme A.

Ressort de la complexité du calcul :

- emprunter à l'algorithmique son meilleur algorithme, A, pour P
- démontrer qu'aucun algorithme ne fait mieux que A, pour P
- conclure que la complexité du problème P est donnée par le temps d'exécution de A.

Autres mesures possibles

- La mémoire utilisée.
  Peut-on toujours couper dans la mémoire ?
  Peut-on toujours le faire en échange d'un temps plus long ?
- Le temps "parallèle".
  L'accès à m processeurs permet-il d'accélérer ?
  Idéalement : m fois plus rapide, ou mieux encore.

Supplément sur les ordres (BB chapitre 3)

Ordre conditionnel

Les O, Ω et Θ n'ont plus de secrets (n'est-ce pas ?).

- Ordre conditionnel de $f(n)$ = notation qui permet d'énoncer une borne sur $f(n)$ qui ne vaut que pour certains n
- Couplé à une condition sur f,
  $t(n) \in$ ordre conditionnel de $f(n) \Rightarrow t(n) \in$ ordre inconditionnel

Exemple : $\Omega(f(n) \mid n$ est puissance de $2)$ versus $\Omega(f(n))$.

Ordre conditionnel

Définition formelle (cas de O, idem pour les autres)

Soient $f : \mathbb{N} \to \mathbb{R}^{\geq 0}$ et $P : \mathbb{N} \to \{\text{vrai,faux}\}$.

Alors

$$O(f(n) \mid P(n))$$

est

$$\{t : \mathbb{N} \to \mathbb{R}^{\geq 0} \mid \exists c \in \mathbb{R}^{\geq 0},\ \overset{\infty}{\forall} n \in \mathbb{N},\ [\ P(n) \Longrightarrow t(n) \leq cf(n)\ ]\ \}.$$

Règle de l'harmonie ("smoothness" rule)

Cas de Θ (idem pour les autres)

$b \in \mathbb{N}^{\geq 2}$
$t(n) \in \Theta(f(n) \mid n \text{ est puissance de } b)$.

La règle de l'harmonie sert à éliminer "n est puissance de b" :

Si

- $t(n)$ *est é.n.d. (éventuellement non décroissante), i.e.,*
  $\overset{\infty}{\forall} n \in \mathbb{N}, t(n) \leq t(n+1)$
- $f(n)$ *est harmonieuse, i.e.,*
  $f(n)$ *est é.n.d* et $f(bn) \in O(f(n))$

alors

- $t(n) \in \Theta(f(n))$.

Opérations sur les ordres

$O(n^2 + n^3)$ a été défini, mais est-ce que $O(n^2) + O(n^3)$ a un sens ? Non !

Il faut donc inventer une définition. La voici :

$$O(f) + O(g)$$

est

$$\{\, h : \mathbb{N} \to \mathbb{R}^{\geq 0} \mid \exists h_1 \in O(f), \exists h_2 \in O(g),\ \overset{\infty}{\forall} n \in \mathbb{N},\ [h(n) = h_1(n) + h_2(n)] \,\}$$

Parfois utile pour estimer le temps d'un bloc A suivi d'un bloc B.

S'étend à d'autres opérateurs, par exemple $\times$ au lieu de $+$, et à n'importe quelle paire d'ensembles de fonctions, comme $O(f) + \Theta(g)$.

Supplément sur les récurrences (BB Section 4.7)

Méthode du pifomètre ("intelligent guesswork")

Méthode : estimer à l'oeil la forme de la solution, puis prouver.

Ex : récurrence importante (ici n puissance de b)

$$T(b^k) = \begin{cases} c \neq 0 & \text{si } k = k_0 \\ aT(b^{k-1}) + f(b^k) & \text{si } k > k_0, \end{cases}$$

où $k_0 \in \mathbb{N}$, $a \in \mathbb{R}^{\geq 1}$, $b \in \mathbb{N}^{\geq 2}$, $f : \mathbb{N} \longrightarrow \mathbb{R}^{>0}$.

$$T(b^k) = \begin{cases} c \neq 0 & \text{si } k = k_0 \\ aT(b^{k-1}) + f(b^k) & \text{si } k > k_0, \end{cases}$$

Posons $g(b^{k_0}) = c/(a^{k_0})$ et $g(b^k) = f(b^k)/(a^k)$ pour $k > k_0$.
Par induction sur $k \geq k_0$:

$$T(b^k) = a^k \times [g(b^{k_0}) + g(b^{k_0+1}) + \cdots + g(b^k)].$$

Déjà $T(n) \in \Omega(\underbrace{n^{\log_b a}}_{(b^k)^{\log_b a} = a^k} \mid n \text{ est puissance de } b)$.

Pour obtenir mieux, il faut analyser $[+ \cdots +]$:

$$T(b^k) = \begin{cases} c \neq 0 & \text{si } k = k_0 \\ aT(b^{k-1}) + f(b^k) & \text{si } k > k_0. \end{cases}$$

Lemme ("des puissances de b")

- *Si* $\varepsilon > 0$ *et* $f(n) \in O(n^{\log_b a - \varepsilon})$ *alors* $T(n) \in \Theta(n^{\log_b a} \mid n \text{ est puissance de } b)$.
- *Si* $\varepsilon > 0$ *et* $f(n) \in O(n^{\log_b a + \varepsilon})$ *alors* $T(n) \in O(n^{\log_b a + \varepsilon} \mid n \text{ est puissance de } b)$.
- *Si* $\varepsilon \geq 0$ *et* $f(n) \in O(n^{\log_b a}(\log n)^{\varepsilon})$ *alors* $T(n) \in O(n^{\log_b a}(\log n)^{\varepsilon+1} \mid n \text{ est puissance de } b)$.

Application à une récurrence asymptotique

Soit $t : \mathbb{N} \longrightarrow \mathbb{R}^{>0}$ dont on ne connaît que

$$t(n) \in a_1 t(\lceil n/b \rceil) + a_2 t(\lfloor n/b \rfloor) + O(f(n)),$$

où

- $f : \mathbb{N} \longrightarrow \mathbb{R}^{>0}$
- $a_1, a_2 \in \mathbb{R}^{\geq 0}$, $a_1 + a_2 \geq 1$
- $b \in \mathbb{N}^{\geq 2}$.

Le théorème qui suit résout cette récurrence :

Théorème (Solution de la récurrence asymptotique)

Posons $a = a_1 + a_2$.

1. *Si* $\varepsilon > 0$ *et* $(\log_b a - \varepsilon) \geq 0$ *et* $f(n) \in O(n^{\log_b a - \varepsilon})$ *alors* $t(n) \in O(n^{\log_b a})$.
2. *Si* $\varepsilon > 0$ *et* $f(n) \in O(n^{\log_b a + \varepsilon})$ *alors* $t(n) \in O(n^{\log_b a + \varepsilon})$.
3. *Si* $\varepsilon \geq 0$ *et* $f(n) \in O(n^{\log_b a}(\log n)^{\varepsilon})$ *alors* $t(n) \in O(n^{\log_b a}(\log n)^{\varepsilon+1})$.

Vaut également lorsque

$$t(n) \in a_1 t(\lceil n/b \rceil) + a_2 t(\lfloor n/b \rfloor) + \Omega(f(n)),$$

et tous les "O" remplacés par des "Ω".

Preuve (esquisse) du cas $f(n) \in O(n^{\log_b a - \varepsilon})$

$$t(n) \in a_1 t(\lceil n/b \rceil) + a_2 t(\lfloor n/b \rfloor) + O(n^{\log_b a - \varepsilon})$$

1. Choisir c et k_0 tels que pour tout $n \geq b^{k_0}$,
   $t(n) \leq a_1 t(\lceil n/b \rceil) + a_2 t(\lfloor n/b \rfloor) + \underbrace{cn^{\log_b a - \varepsilon}}_{\text{note : non décroissante}}$
2. Montrer que pour tout n, $t(n) \leq T(n)$ où
   $$T(n) = \begin{cases} \max\{t(0), t(1), \ldots, t(b^{k_0})\} & \text{si } n \leq b^{k_0} \\ a_1 T(\lceil n/b \rceil) + a_2 T(\lfloor n/b \rfloor) + cn^{\log_b a - \varepsilon} & \text{sinon.} \end{cases}$$
3. Lemme des puissances $\Rightarrow$ $T(n) \in O(n^{\log_b a} \mid n \text{ est puissance de } b)$
4. Démo 3 $\Rightarrow$ $T(n)$ est é.n.d.
5. $n^{\log_b a}$ est harmonieuse
6. Smoothness rule $\Rightarrow$ $T(n) \in O(n^{\log_b a})$
7. Points (2) et (6) $\Rightarrow$ $t(n) \in O(n^{\log_b a})$.

Méthode de l'équation caractéristique

Déjà étudiée dans les cours préalables pour la résolution de récurrences linéaires homogènes à coefficients constants.

Plusieurs exemples dans BB.

Résolution de récurrences

Récurrences linéaires homogènes à coefficients constants:

Soit la récurrence R

$$a_0 t_n + a_1 t_{n-1} + \ldots + a_k t_{n-k} = 0$$

Voici les étapes de la résolution:

1) Trouver le polynôme caractéristique $P(x)$ de la récurrence R

2) Trouver les racines de $P(x)$

Si ces racines sont distinctes

3) La solution générale est de la forme $t_n = \sum_{i=1}^{k} c_i r_i^n$

4) Résoudre le système d'équations linéaires donné par les conditions initiales pour trouver la valeur des constantes $c_1, c_2, \ldots, c_k$

5) Écrire la solution t_n en fonction de ces constantes c_i

Résolution de récurrences

Récurrences linéaires homogènes à coefficients constants:

Soit la récurrence R

$$a_0 t_n + a_1 t_{n-1} + \ldots + a_k t_{n-k} = 0$$

Voici les étapes de la résolution:

1) Trouver le polynôme caractéristique $P(x)$ de la récurrence R

2) Trouver les racines de $P(x)$

Si ces racines ne sont pas toutes distinctes

3) La solution générale est de la forme $t_n = \sum_{i=1}^{\ell} \sum_{j=0}^{m_i - 1} c_{ij} n^j r_i^n$

où on a ℓ racines r_i de multiplicité m_i

4) Résoudre le système d'équations linéaires donné par les conditions initiales pour trouver la valeur des constantes $c_1, c_2, \ldots, c_k$

5) Écrire la solution t_n en fonction de ces constantes c_i

Cas non homogène

Type de récurrence :

$$a_0 t_n + a_1 t_{n-1} + \cdots + a_k t_{n-k} = b_1^n p_1(n) + b_2^n p_2(n) + \cdots$$

où a_i, b_j sont des constantes,
$p_i(n)$ sont des polynômes.

Prendre comme équation caractéristique :

$$(a_0 x^k + a_1 x^{k-1} + \cdots + a_k)(x - b_1)^{1+\text{degré}(p_1)}(x - b_2)^{1+\text{degré}(p_2)} \cdots = 0.$$