Master Theorem

(rédigé par Clarence Kineider)

Leçons : (903) ; 926 ; 931

Références : BEAUQUIER–BERSTEL–CHRÉTIENNE, *Éléments d'algorithmique* (p.30) [?]

Introduction :

On va présenter un théorème essentiel qui permet de donner la complexité d'algorithmes utilisant le paradigme "diviser pour régner". Cependant, on ne peut pas donner la complexité de tous les algorithmes de "diviser pour régner" avec ce théorème.

Théorème 1. Soit $T : \mathbb{N} \to \mathbb{R}_+$ croissante à partir d'un certain rang. On suppose qu'il existe un rang $n_0 \geqslant 1$, des entiers $b \geqslant 2$ et $k \geqslant 0$ ainsi que des réels $a, c, d > 0$ tels que la relation de récurrence suivante soit vérifiée :

$$\begin{cases} T(n_0) = d \\ T(n) = aT\left(\frac{n}{b}\right) + cn^k \text{ pour tout entier de la forme } n = b^p n_0 \text{ avec } p \in \mathbb{N} \end{cases}$$

Alors on a la disjonction de cas suivante :

(i) Si $a < b^k$ alors $T(n) = \Theta(n^k)$

(ii) Si $a = b^k$ alors $T(n) = \Theta(n^k \log_b(n))$

(iii) Si $a > b^k$ alors $T(n) = \Theta(n^{\log_b(a)})$

Démonstration. **Étape 1 :** *Cas $n = n_0 b^p$*

Soit $p \in \mathbb{N}$, $n = b^p n_0$. Par récurrence, on a :

$$T(n) = da^p + \sum_{i=0}^{p-1} ca^i \left(\frac{n}{b^i}\right)^k = \underbrace{da^p}_{\delta(n)} + cn^k \underbrace{\sum_{i=0}^{p-1} \left(\frac{a}{b^k}\right)^i}_{\gamma(n)}$$

On a $\delta(n) = da^p = da^{\log_b(n) - \log_b(n_0)} = \frac{d}{a^{\log_b(n_0)}} b^{log_b(a) log_b(n)} = \frac{d}{a^{\log_b(n_0)}} n^{\log_b(a)} = \Theta(n^{\log_b(a)})$.

De plus, $\gamma(n)$ est une somme partielle d'une série géométrique. On distingue trois cas :

— Si $a < b^k$, la série géométrique de raison $\frac{a}{b^k}$ converge, donc $\gamma(n)$ est bornée. On a donc

$$T(n) = \Theta(n^{\log_b(a)}) + \Theta(n^k) = \Theta(n^k)$$

car $\log_b(a) < k$.

— Si $a = b^k$, alors $\gamma(n) = p = \log_b(n) - \log_b(n_0) = \Theta(\log_b(n))$. Donc

$$T(n) = \Theta(n^{\log_b(a)}) + \Theta(n^k \log_b(n)) = \Theta(n^k \log_b(n))$$

car $\log_b(a) = k$.

— Si $a > b^k$, alors $\gamma(n) \sim \alpha \frac{a^p}{b^{kp}}$ avec $\alpha = \frac{1}{\frac{a}{b^k} - 1}$. Donc $cn^k \gamma(n) \sim c\alpha \left(\frac{n}{b^p}\right)^k a^p \sim c\alpha n_0^k a^p$. Or $a^p = \Theta(n^{\log_b(a)})$, donc on a

$$T(n) = \Theta(n^{\log_b(a)}) + \Theta(n^{\log_b(a)}) = \Theta(n^{\log_b(a)})$$

On a donc le résultat pour n de la forme $n = b^p n_0$ avec $p \in \mathbb{N}$.

Étape 2 : *Cas $n \in \mathbb{N}$*

Soit $n \geq n_0$ et $p \in \mathbb{N}$ tel que $b^p n_0 \leq n < b^{p+1} n_0$. Puisque T est croissante à partir d'un certain rang, pour n assez grand on a

$$T(b^p n_0) \leq T(n) < T(b^{p+1} n_0)$$

Or pour $f \in \left\{m \mapsto m^k, m \mapsto m^k \log_b(m), m \mapsto m^{\log_b(a)}\right\}$, on a $f(bm) = \Theta(f(m))$. Par encadrement, on a le résultat pour tout n à partir d'un certain rang. □

Astuces de l'agrégatif :

Suivant votre vitesse, vous pouvez ajouter une application à l'algorithme de votre choix (tri fusion pour la leçon 903 par exemple).

Si on a seulement une majoration dans l'hypothèse de récurrence (i.e. $T(n) \leq aT\left(\frac{n}{b}\right) + cn^k$), on a le même résultat mais avec des O à la place des Θ. En effet, il suffit de voir qu'on a $T(n) \leq \delta(n) + cn^k\gamma(n)$ pour les mêmes δ et γ que dans la démonstration. Cela permet d'appliquer le théorème à des relations de type $T(n) = aT\left(\frac{n}{b}\right) + O(n^k)$.

Remarques :

Voici une technique pour se passer de l'hypothèse de croissance de T :
Je la rédige pour le tri fusion, mais elle devrait s'adapter à n'importe quel algorithme.
On note $T(n)$ le temps d'exécution dans le cas le pire pour une entrée de taille n. On pose $T_m(n) = \max_{k \leq n} T(n)$ le temps d'exécution dans le cas le pire pour une entrée de taille inférieure ou égale à n. On a alors T_m croissante. On montre que T_m vérifie les hypothèses du Master Theorem avec une inégalité :

$$T_m(n) \leq T_m\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T_m\left(\left\lceil \frac{n}{2} \right\rceil\right) + cn \leq 2T_m\left(\left\lceil \frac{n}{2} \right\rceil\right) + cn$$

Donc pour $n = 2^k$ on a :

$$T_m(2^k) \leq 2T_m(2^{k-1}) + cn$$

Par le Master Theorem, on a donc $T_m(n) = O(n \log(n))$ car T_m est croissante.
Donc $T(n) \leq T_m(n) = O(n \log(n))$.

Remarques :

Voilà quelques problèmes qui utilisent ce théorème :

- Tri fusion $(a = 2, b = 2, k = 1)$ (cas (ii))
- Multiplication de Matrices par Strassen $(a = 7, b = 2, k = 2)$ (cas (iii))
- Dichotomie sur tableau trié $(a = 1, b = 2, k = 0)$ (cas (ii))
- Multiplication d'entiers par la relation de Karatsuba $(a = 3, b = 2, k = 1)$ (cas (iii))
- Dichotomie sur une liste triée $(a = 1, b = 2, k = 1)$ (cas (i))

Attention, on ne peut pas utiliser ce théorème pour le tri rapide (car on ne divise pas la taille de notre objet par une quantité constante (la constante b))