

Le théorème général

Jean-Stéphane Varré

Université Lille 1

jean-stephane.varre@univ-lille1.fr



Equations de partition

Le théorème général permet de connaître le comportement asymptotique des équations de récurrence qui s'écrivent sous la forme suivante :

$$c(n) = a \times c\left(\frac{n}{b}\right) + f(n)$$

Pour l'utiliser il faut définir le comportement asymptotique de la fonction $f(n)$ par rapport à $n^{\log_b a}$.

Equations de partition

Théorème général

Soient $a \geq 1$ et $b > 1$ deux constantes, soit $f(n)$ une fonction et soit $c(n)$ définie pour les entiers non négatifs par la récurrence

$$c(n) = a \times c\left(\frac{n}{b}\right) + f(n),$$

où l'on interprète n/b comme étant $\lfloor \frac{n}{b} \rfloor$ ou $\lceil \frac{n}{b} \rceil$. $c(n)$ peut alors être bornée asymptotiquement de la façon suivante.

Equations de partition

Théorème général

Soient $a \geq 1$ et $b > 1$ deux constantes, soit $f(n)$ une fonction et soit $c(n)$ définie pour les entiers non négatifs par la récurrence

$$c(n) = a \times c\left(\frac{n}{b}\right) + f(n),$$

où l'on interprète n/b comme étant $\lfloor \frac{n}{b} \rfloor$ ou $\lceil \frac{n}{b} \rceil$. $c(n)$ peut alors être bornée asymptotiquement de la façon suivante.

1 si $f(n) = \mathcal{O}(n^{\log_b a - \varepsilon})$ pour une certaine constante $\varepsilon > 0$, alors

$$c(n) = \Theta(n^{\log_b a}).$$

Equations de partition

Théorème général

Soient $a \geq 1$ et $b > 1$ deux constantes, soit $f(n)$ une fonction et soit $c(n)$ définie pour les entiers non négatifs par la récurrence

$$c(n) = a \times c\left(\frac{n}{b}\right) + f(n),$$

où l'on interprète n/b comme étant $\lfloor \frac{n}{b} \rfloor$ ou $\lceil \frac{n}{b} \rceil$. $c(n)$ peut alors être bornée asymptotiquement de la façon suivante.

1 si $f(n) = \mathcal{O}(n^{\log_b a - \varepsilon})$ pour une certaine constante $\varepsilon > 0$, alors

$$c(n) = \Theta(n^{\log_b a}).$$

2 si $f(n) = \Theta(n^{\log_b a})$, alors

$$c(n) = \Theta(n^{\log_b a} \log_2 n).$$

Equations de partition

Théorème général

Soient $a \geq 1$ et $b > 1$ deux constantes, soit $f(n)$ une fonction et soit $c(n)$ définie pour les entiers non négatifs par la récurrence

$$c(n) = a \times c\left(\frac{n}{b}\right) + f(n),$$

où l'on interprète n/b comme étant $\lfloor \frac{n}{b} \rfloor$ ou $\lceil \frac{n}{b} \rceil$. $c(n)$ peut alors être bornée asymptotiquement de la façon suivante.

1 si $f(n) = \mathcal{O}(n^{\log_b a - \varepsilon})$ pour une certaine constante $\varepsilon > 0$, alors

$$c(n) = \Theta(n^{\log_b a}).$$

2 si $f(n) = \Theta(n^{\log_b a})$, alors

$$c(n) = \Theta(n^{\log_b a} \log_2 n).$$

3 si $f(n) = \Omega(n^{\log_b a + \varepsilon})$ pour une certaine constante $\varepsilon > 0$, et si $a \times f(n/b) \leq k \times f(n)$ pour une certaine constante $k < 1$ et pour n suffisamment grand, alors

$$c(n) = \Theta(f(n)).$$

Exemple 1

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 4 c\left(\frac{n}{2}\right) + \sqrt{n}$$

Exemple 1

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 4 c\left(\frac{n}{2}\right) + \sqrt{n}$$

On commence par identifier

Exemple 1

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 4 c\left(\frac{n}{2}\right) + \sqrt{n}$$

On commence par identifier $a = 4$,

Exemple 1

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 4 c\left(\frac{n}{2}\right) + \sqrt{n}$$

On commence par identifier $a = 4$, $b = 2$,

Exemple 1

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 4c\left(\frac{n}{2}\right) + \sqrt{n}$$

On commence par identifier $a = 4$, $b = 2$, et $f(n) = \sqrt{n}$.

Exemple 1

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

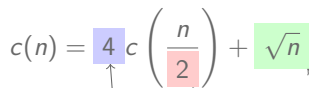
$$c(n) = 4c\left(\frac{n}{2}\right) + \sqrt{n}$$

On commence par identifier $a = 4$, $b = 2$, et $f(n) = \sqrt{n}$.

On calcule ensuite $\log_b a = \log_2 4 = 2$ puis $n^{\log_b a} = n^2$.

Exemple 1

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 4c\left(\frac{n}{2}\right) + \sqrt{n}$$


On commence par identifier $a = 4$, $b = 2$, et $f(n) = \sqrt{n}$.

On calcule ensuite $\log_b a = \log_2 4 = 2$ puis $n^{\log_b a} = n^2$.

Il faut maintenant trouver le comportement asymptotique de $f(n)$ vis-à-vis de $n^{\log_b a - \epsilon}$, $n^{\log_b a}$ et $n^{\log_b a + \epsilon}$ pour savoir dans quel cas du théorème général on se place.

Exemple 1

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 4c\left(\frac{n}{2}\right) + \sqrt{n}$$

On commence par identifier $a = 4$, $b = 2$, et $f(n) = \sqrt{n}$.

On calcule ensuite $\log_b a = \log_2 4 = 2$ puis $n^{\log_b a} = n^2$.

Ici il paraît évident que $f(n) = \sqrt{n}$ ne majore pas $n^{2+\epsilon}$, on a donc $\sqrt{n} \notin \Omega(n^{2+\epsilon})$. On n'est donc pas dans le cas 3 du théorème général.

Exemple 1

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 4c\left(\frac{n}{2}\right) + \sqrt{n}$$

On commence par identifier $a = 4$, $b = 2$, et $f(n) = \sqrt{n}$.

On calcule ensuite $\log_b a = \log_2 4 = 2$ puis $n^{\log_b a} = n^2$.

De la même manière, il paraît évident que $f(n) = \sqrt{n}$ n'a pas le même comportement asymptotique que n^2 , on a donc $\sqrt{n} \notin \Theta(n^2)$. On n'est donc pas dans le cas 2 du théorème général.

Exemple 1

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 4c\left(\frac{n}{2}\right) + \sqrt{n}$$

On commence par identifier $a = 4$, $b = 2$, et $f(n) = \sqrt{n}$.

On calcule ensuite $\log_b a = \log_2 4 = 2$ puis $n^{\log_b a} = n^2$.

Comme $\sqrt{n} < n^{2-0.1}$ ($\varepsilon = 0.1$) pour tout n supérieur ou égal à 1, on en déduit que $\sqrt{n} = \mathcal{O}(n^{2-\varepsilon})$.

Exemple 1

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 4c\left(\frac{n}{2}\right) + \sqrt{n}$$

On commence par identifier $a = 4$, $b = 2$, et $f(n) = \sqrt{n}$.

On calcule ensuite $\log_b a = \log_2 4 = 2$ puis $n^{\log_b a} = n^2$.

Comme $\sqrt{n} < n^{2-0.1}$ ($\varepsilon = 0.1$) pour tout n supérieur ou égal à 1, on en déduit que $\sqrt{n} = \mathcal{O}(n^{2-\varepsilon})$.

On est donc dans le **cas 1** du théorème général, on conclut que :

$$c(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$$

Exemple 2

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8 c\left(\frac{n}{3}\right) + n^2$$

Exemple 2

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8 c\left(\frac{n}{3}\right) + n^2$$

On commence par identifier

Exemple 2

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{3}\right) + n^2$$

On commence par identifier $a = 8$,

Exemple 2

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{3}\right) + n^2$$

On commence par identifier $a = 8$, $b = 3$,

Exemple 2

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{3}\right) + n^2$$

On commence par identifier $a = 8$, $b = 3$, et $f(n) = n^2$.

Exemple 2

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{3}\right) + n^2$$

On commence par identifier $a = 8$, $b = 3$, et $f(n) = n^2$.

On calcule ensuite $\log_b a = \log_3 8 \leq 2$ (inutile d'avoir la valeur exacte qui est 1.89) puis $n^{\log_b a} = n^{1.89}$.

Exemple 2

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{3}\right) + n^2$$

On commence par identifier $a = 8$, $b = 3$, et $f(n) = n^2$.

On calcule ensuite $\log_b a = \log_3 8 \leq 2$ (inutile d'avoir la valeur exacte qui est 1.89) puis $n^{\log_b a} = n^{1.89}$.

Il faut maintenant trouver le comportement asymptotique de $f(n)$ vis-à-vis de $n^{\log_b a - \varepsilon}$, $n^{\log_b a}$ et $n^{\log_b a + \varepsilon}$ pour savoir dans quel cas du théorème général on se place.

Exemple 2

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{3}\right) + n^2$$

On commence par identifier $a = 8$, $b = 3$, et $f(n) = n^2$.

On calcule ensuite $\log_b a = \log_3 8 \leq 2$ (inutile d'avoir la valeur exacte qui est 1.89) puis $n^{\log_b a} = n^{1.89}$.

Ici il paraît évident que $f(n) = n^2$ majore $n^{1.89+\varepsilon}$ ($n^{1.89+0.01} < n^2$),

on a donc $n^2 \in \Omega(n^{1.89+\varepsilon})$. On est donc dans le **cas 3** du théorème général.

Exemple 2

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{3}\right) + n^2$$

On commence par identifier $a = 8$, $b = 3$, et $f(n) = n^2$.

On calcule ensuite $\log_b a = \log_3 8 \leq 2$ (inutile d'avoir la valeur exacte qui est 1.89) puis $n^{\log_b a} = n^{1.89}$.

Dans le cas 3 il faut vérifier en plus la condition

$a \times f(n/b) \leq k \times f(n)$ pour une certaine constante $k < 1$.

Exemple 2

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{3}\right) + n^2$$

On commence par identifier $a = 8$, $b = 3$, et $f(n) = n^2$.

On calcule ensuite $\log_b a = \log_3 8 \leq 2$ (inutile d'avoir la valeur exacte qui est 1.89) puis $n^{\log_b a} = n^{1.89}$.

$$a \times f\left(\frac{n}{b}\right) = 8 \left(\frac{n}{3}\right)^2 = \frac{8}{9}n^2$$

Existe-t-il $k < 1$ tel que cette quantité est inférieure ou égale à $k n^2$?

Oui il suffit de prendre $\frac{8}{9} \leq k < 1$, donc par exemple $\frac{8}{9}$ convient.

Exemple 2

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{3}\right) + n^2$$

On commence par identifier $a = 8$, $b = 3$, et $f(n) = n^2$.

On calcule ensuite $\log_b a = \log_3 8 \leq 2$ (inutile d'avoir la valeur exacte qui est 1.89) puis $n^{\log_b a} = n^{1.89}$.

$$a \times f\left(\frac{n}{b}\right) = 8 \left(\frac{n}{3}\right)^2 = \frac{8}{9}n^2$$

Existe-t-il $k < 1$ tel que cette quantité est inférieure ou égale à $k n^2$?

Oui il suffit de prendre $\frac{8}{9} \leq k < 1$, donc par exemple $\frac{8}{9}$ convient.

On est dans le cas 3 du théorème général, on conclut donc que

$$c(n) = \Theta(f(n)) = \Theta(n^2)$$

Exemple 3

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8 c\left(\frac{n}{2}\right) + 4n^3$$

Exemple 3

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8 c\left(\frac{n}{2}\right) + 4n^3$$

On commence par identifier

Exemple 3

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{2}\right) + 4n^3$$

On commence par identifier $a = 8$,

Exemple 3

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{2}\right) + 4n^3$$

On commence par identifier $a = 8$, $b = 2$,

Exemple 3

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{2}\right) + 4n^3$$


On commence par identifier $a = 8$, $b = 2$, et $f(n) = 4n^3$.

Exemple 3

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{2}\right) + 4n^3$$

On commence par identifier $a = 8$, $b = 2$, et $f(n) = 4n^3$.

On calcule ensuite $\log_b a = \log_2 8 = 3$ puis $n^{\log_b a} = n^3$.

Exemple 3

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{2}\right) + 4n^3$$

On commence par identifier $a = 8$, $b = 2$, et $f(n) = 4n^3$.

On calcule ensuite $\log_b a = \log_2 8 = 3$ puis $n^{\log_b a} = n^3$.

Il faut maintenant trouver le comportement asymptotique de $f(n)$ vis-à-vis de $n^{\log_b a - \epsilon}$, $n^{\log_b a}$ et $n^{\log_b a + \epsilon}$ pour savoir dans quel cas du théorème général on se place.

Exemple 3

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{2}\right) + 4n^3$$

On commence par identifier $a = 8$, $b = 2$, et $f(n) = 4n^3$.

On calcule ensuite $\log_b a = \log_2 8 = 3$ puis $n^{\log_b a} = n^3$.

Ici il paraît évident que $f(n) = 4n^3$ a le même comportement que n^3 (il existe les constantes 3 et 5 telles que $3n^3 < 4n^3 < 5n^3$ pour tout n), on a donc $4n^3 = \Theta(n^3)$. On est donc dans le **cas 2** du théorème général.

Exemple 3

Supposons avoir un algorithme dont la complexité est donnée par l'équation de récurrence suivante :

$$c(n) = 8c\left(\frac{n}{2}\right) + 4n^3$$

On commence par identifier $a = 8$, $b = 2$, et $f(n) = 4n^3$.

On calcule ensuite $\log_b a = \log_2 8 = 3$ puis $n^{\log_b a} = n^3$.

Ici il paraît évident que $f(n) = 4n^3$ a le même comportement que n^3 (il existe les constantes 3 et 5 telles que $3n^3 < 4n^3 < 5n^3$ pour tout n), on a donc $4n^3 = \Theta(n^3)$. On est donc dans le **cas 2** du théorème général.

On conclut donc que :

$$c(n) = \Theta(n^{\log_b a} \log n) = \Theta(n^3 \log n)$$