Introduction

- <u>But</u> : Trouver la complexité asymptotique temporelle à partir d'une équation de récurrence.

- <u>Exemples de récurrences</u> :
  - $T(n) = T(n-1) + 1$
  - $T(n) = T(n-1) + n^3$
  - $T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + cste$

- <u>Méthodes</u> :
  I. Méthode de substitution
  II. Méthode de l'arbre récursif
  III. Méthode générale

I. Méthode de substitution

<u>Rappel</u> : Démonstration par récurrence

<u>Exemple</u> : On définit la suite $\{a_n\}_{n\geq 0}$ par : $\begin{cases} a_0 = 1 \\ a_{n+1} = 3a_n + 5 \text{ pour tout } n \geq 0. \end{cases}$

Démontrez la formule $a_n = \frac{7}{2}3^n - \frac{5}{2}$ pour tout $n \geq 0$.

- **Initialisation** : Si $n = 0$, la proposition est vraie, car $\frac{7}{2}3^0 - \frac{5}{2} = 1 = a_0$.
- **Récurrence** : On suppose la proposition vraie au rang $k = n$: $a_n = \frac{7}{2}3^n - \frac{5}{2}$.

On doit démontrer qu'elle est vraie au rang $k = n + 1$. Or, on a :

$a_{n+1} = 3a_n + 5 = 3\left(\frac{7}{2}3^n - \frac{5}{2}\right) + 5 = \frac{7}{2}3^{n+1} - \frac{15}{2} + 5 = \frac{7}{2}3^{n+1} - \frac{5}{2}$.

La propriété est donc vraie au rang $k = n + 1$.

- **Conclusion** : $a_n = \frac{7}{2}3^n - \frac{5}{2}$ pour tout $n \geq 0$.

Exemple de résolution par substitution

La complexité d'un algorithme vérifie la relation de récurrence :
$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$, on veut vérifier que $T(n) = O(n\lg(n))$.
On doit donc trouver une constante $c > 0$ telle que $T(n) \leq cn\lg(n)$.

- Les valeurs initiales ne sont **en général** pas données, elles ne changent **en général** pas l'ordre de grandeur.
- **Récurrence** : on suppose $T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \leq c\left\lfloor \frac{n}{2} \right\rfloor \lg\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$. Si on remplace (substitution) dans la relation de récurrence, on obtient :

$$\begin{aligned} T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n &\leq 2c\left\lfloor \frac{n}{2} \right\rfloor \lg\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \\ &\leq 2c\frac{n}{2}\lg\left(\frac{n}{2}\right) + n \\ &\leq cn\lg(n) - cn\lg(2) + n \\ &\leq cn\lg(n), \end{aligned}$$

la dernière égalité étant vraie pour toute constance $c \geq \frac{1}{\lg(2)} = 1$.

Exemple de résolution par substitution

- **Initialisation** : Si $n = 1$, la proposition ne peut pas être vraie, car $\lg(1) = 0$.

Comme $\left\lfloor \frac{2}{2} \right\rfloor = \left\lfloor \frac{3}{2} \right\rfloor = 1$, et $\left\lfloor \frac{n}{2} \right\rfloor > 1$ pour tout $n > 3$, on peut initialiser la récurrence avec les **deux** valeurs $n = 2$ et $n = 3$.
Or, on a bien $T(2) \leq 2c\lg(2)$ et $T(3) \leq 3c\lg(3)$,
à condition d'avoir $c \geq \max\left(\frac{T(2)}{2}, \frac{T(3)}{3\lg(3)}\right)$.

- **Conclusion** : si on prend $c \geq \max\left(1, \frac{T(2)}{2}, \frac{T(3)}{3\lg(3)}\right)$, on obtient donc que

pour tout $n \geq 2, T(n) \leq cn\lg(n)$ $(n_0 = 2)$.

- **Remarque** : Si on pose $T(1) = 1$ par exemple, on peut calculer explicitement à l'aide de la récurrence $T(2) = 4$ et $T(3) = 5$ et il suffit alors de prendre $c = 2$.

Exemple de résolution par substitution

La complexité d'un algorithme vérifie la relation de récurrence :
$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$, on veut vérifier maintenant que $T(n) = \Omega(n\lg(n))$.
On doit donc trouver une constante $c > 0$ telle que $T(n) \geq cn\lg(n)$.

- **Récurrence** : on suppose $T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \geq c\left\lfloor \frac{n}{2} \right\rfloor \lg\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$. Si on remplace (substitution) dans la relation de récurrence, on obtient :

$$\begin{aligned} T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n &\geq 2c\left\lfloor \frac{n}{2} \right\rfloor \lg\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \\ &\geq 2c\left(\frac{n}{2} - \frac{1}{2}\right)\lg\left(\frac{n}{2} - \frac{1}{2}\right) + n \\ &\geq c(n-1)\lg(n-1) - cn + n \end{aligned}$$

…??

« Remède » : Règle de l'harmonie

<u>Définitions</u> : Soit une fonction $T : \mathbb{N} \to [0; +\infty[$ et $b \in \mathbb{N}, b \geq 2$, alors :

- T est dite finalement croissante s'il existe $n_0 \in \mathbb{N}$ tel que $\forall n \geq n_0$, $T(n) \leq T(n+1)$.
- T est dite b-harmonieuse, si les deux conditions suivantes sont vérifiées :
  - T est finalement croissante,
  - $T(bn) = O(T(n))$.
- T est dite harmonieuse, si elle est b-harmonieuse pour tout $b \in \mathbb{N}, b \geq 2$.

- <u>Exemples</u> :
  - Les fonctions $n, n^2, \lg(n), n\lg(n)$, tous les polynômes de coefficient dominant positif sont harmonieux.
  - Les fonctions $n^{\lg(n)}, 3^n, n!$ ne sont pas harmonieuses.

Énoncé de la règle de l'harmonie

<u>Théorème</u> : On considère un entier $b \in \mathbb{N}, b \geq 2$ et deux fonctions $T, g : \mathbb{N} \to [0; +\infty[$ qui vérifient les deux propriétés suivantes :
- T est finalement croissante,
- g est b-harmonieuse.

Alors, $T(n) = \Theta(g(n) | n \text{ est une puissance de } b) \Rightarrow T(n) = \Theta(g(n))$.
Le théorème est également valable en remplaçant Θ par O ou Ω.

- <u>Conséquence</u> : Pour les récurrences $T(n) = aT\left(\left\lfloor \frac{n}{b} \right\rfloor\right) + f(n)$ ou $T(n) = aT\left(\left\lceil \frac{n}{b} \right\rceil\right) + f(n)$, où $a \geq 1, b > 1$ sont des entiers, on peut « oublier » les fonctions plancher ou plafond, si on peut prouver que T est finalement croissante et que la borne ou l'équivalent asymptotique $g(n)$ est b-harmonieux.

Application de la règle de l'harmonie

$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$ et on veut vérifier maintenant que $T(n) = \Omega(n\lg(n))$.
- T est finalement croissante (démonstration page suivante),
- $g(n) = n\lg(n)$ est 2-harmonieuse :
  - $g(n)$ est croissante comme produit de deux fonctions croissantes,
  - $g(2n) = 2n\lg(2n) = 2n(\lg(n) + 1) = \Theta(n\lg(n))$.

On peut donc démontrer $T(n) = \Omega(n\lg(n))$ en utilisant la récurrence $T(n) = 2T\left(\frac{n}{2}\right) + n$.

- **Récurrence** : on suppose $T\left(\frac{n}{2}\right) \geq c\frac{n}{2}\lg\left(\frac{n}{2}\right)$. Si on remplace (substitution) dans la relation de récurrence, on obtient :

$$\begin{aligned} T(n) = 2T\left(\frac{n}{2}\right) + n &\geq 2c\frac{n}{2}\lg\left(\frac{n}{2}\right) + n \\ &\geq cn\lg(n) - cn + n \\ &\geq cn\lg(n), \end{aligned}$$

la dernière égalité étant vraie pour toute constance $c \leq 1$.

Démonstration de la croissance de $T(n)$

Démonstration par récurrence :

- **Initialisation** : $T(2) = 2T(1) + 2 \geq T(1)$
- **Récurrence** : on suppose $T(k) \leq T(k+1)$ **pour tout** $k \leq n$. On a alors :

$$T(n+1) = 2T\left(\left\lfloor \frac{n+1}{2} \right\rfloor\right) + n + 1$$

$$T(n+2) = 2T\left(\left\lfloor \frac{n+2}{2} \right\rfloor\right) + n + 2$$

Or, $T\left(\left\lfloor \frac{n+1}{2} \right\rfloor\right) \leq T\left(\left\lfloor \frac{n+2}{2} \right\rfloor\right)$ par hypothèse de récurrence, car $\left\lfloor \frac{n+2}{2} \right\rfloor = \left\lfloor \frac{n+1}{2} \right\rfloor$ ou $\left\lfloor \frac{n+2}{2} \right\rfloor = \left\lfloor \frac{n+1}{2} \right\rfloor + 1$.
On a donc bien $T(n+1) \leq T(n+2)$.

Deuxième exemple

La complexité d'un algorithme vérifie la relation de récurrence :
$T(n) = T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 1$, on veut vérifier que $T(n) = O(n)$.
On doit donc trouver une constante $c > 0$ telle que $T(n) \leq cn$.
- **Récurrence** :

$T(n) \leq c\left\lceil \frac{n}{2} \right\rceil + c\left\lfloor \frac{n}{2} \right\rfloor + 1 = cn + 1$ …??
Quelle que soit la valeur de c, on n'aura jamais $cn + 1 \leq cn$!!
Il faut alors « renforcer » l'hypothèse de récurrence en soustrayant un terme d'ordre inférieur : $T(n) \leq cn - b$.
- **Récurrence** :

$T(n) \leq c\left\lceil \frac{n}{2} \right\rceil - b + c\left\lfloor \frac{n}{2} \right\rfloor - b + 1 = cn - 2b + 1$
Or, $-2b + 1 \leq -b$, pour toute constante $b \geq 1$.

II. Méthode de l'arbre récursif

- On représente l'équation de récurrence par une arborescence pour les relations de la forme :

$$T(n) = aT\left(\frac{n}{b}\right) + f(n).$$

- On va étudier :

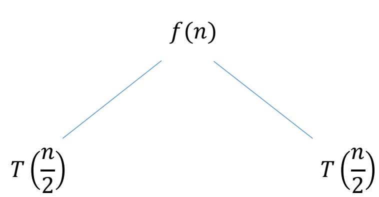
$$T(n) = 2T\left(\frac{n}{2}\right) + f(n), T(1) = 1.$$

- On suppose que $n = 2^p$.

Arbre de hauteur 1

$$T(n) = 2T\left(\frac{n}{2}\right) + f(n), T(1) = 1$$

<u>Travail effectué</u> :



Arbre de hauteur 2

$$T(n) = 2T\left(\frac{n}{2}\right) + f\ (n), T(1) = 1$$

$$T(n) = 2\left(2T\left(\frac{n}{4}\right) + f\left(\frac{n}{2}\right)\right) + f\ (n)$$

<u>Travail effectué</u> :

$$f(n)$$

$$f\left(\frac{n}{2}\right) \qquad f\left(\frac{n}{2}\right)$$

$$T\left(\frac{n}{4}\right) \quad T\left(\frac{n}{4}\right) \qquad T\left(\frac{n}{4}\right) \quad T\left(\frac{n}{4}\right)$$

Arbre de hauteur p si $n = 2^p$

$$f(n) \qquad 0 \to 2^0$$

$$f\left(\frac{n}{2}\right) + f\left(\frac{n}{2}\right) \qquad 1 \to 2^1$$

$$f\left(\frac{n}{4}\right) + f\left(\frac{n}{4}\right) + f\left(\frac{n}{4}\right) + f\left(\frac{n}{4}\right) \qquad 2 \to 2^2$$

$$\vdots$$

$$T\left(\frac{n}{2^p}\right) + T\left(\frac{n}{2^p}\right) + \ldots + T\left(\frac{n}{2^p}\right) + T\left(\frac{n}{2^p}\right) \qquad p \to 2^p$$

Formule pour $T(n)$

- Travail non récursif effectué lors des appels récursifs (le total des niveaux de 0 à $p-1$) :

$$\sum_{i=0}^{p-1} 2^i f\left(\frac{n}{2^i}\right)$$

- Travail effectué par le cas de base (le dernier niveau, p) :

$$2^p T\left(\frac{n}{2^p}\right) = 2^p T(1) = 2^p = n$$

- Au total :

$$T(n) = \sum_{i=0}^{p-1} 2^i f\left(\frac{n}{2^i}\right) + n$$

La fonction $f(n)$

$$T(n) = 2T\left(\frac{n}{2}\right) + f\ (n), T(1) = 1$$

$$T(n) = \sum_{i=0}^{p-1} 2^i f\left(\frac{n}{2^i}\right) + n$$

Nous allons considérer 3 cas en fonction du taux de croissance de $f(n)$ par rapport à n :

1. $f(n)$ croît plus lentement que n.
   - Exemple : $f(n) = \sqrt{n}$
2. $f(n)$ croît au même rythme que n.
   - Exemple : $f(n) = 2n$
3. $f(n)$ croît plus vite que n.
   - Exemple : $f(n) = 3n^2$

$f(n) = \sqrt{n}$

$$T(n) = \sum_{i=0}^{p-1} 2^i f\left(\frac{n}{2^i}\right) + n = \sum_{i=0}^{p-1} 2^i \sqrt{\frac{n}{2^i}} + n$$

$$= n + \sqrt{n} \sum_{i=0}^{p-1} \sqrt{2}^i = n + \sqrt{n} \frac{\sqrt{2}^p - 1}{\sqrt{2} - 1}$$

$$= n + \sqrt{n} \frac{\sqrt{n} - 1}{\sqrt{2} - 1} = \left(1 + \frac{1}{\sqrt{2} - 1}\right) n - \frac{1}{\sqrt{2} - 1} \sqrt{n}$$

$$\in \Theta(n)$$

$f(n) = 2n$

$$T(n) = \sum_{i=0}^{p-1} 2^i f\left(\frac{n}{2^i}\right) + n = \sum_{i=0}^{p-1} 2^i \frac{2n}{2^i} + n$$

$$= n + n \sum_{i=0}^{p-1} 2 = n + 2np$$

$$= n + 2n\lg(n)$$

$$\in \Theta(n\lg(n))$$

$f(n) = 3n^2$

$$T(n) = \sum_{i=0}^{p-1} 2^i f\left(\frac{n}{2^i}\right) + n = \sum_{i=0}^{p-1} 2^i 3\left(\frac{n}{2^i}\right)^2 + n$$

$$= n + 3n^2 \sum_{i=0}^{p-1} \frac{1}{2^i} = n + 3n^2 \frac{\frac{1}{2^p} - 1}{\frac{1}{2} - 1} = n - 6n^2\left(\frac{1}{n} - 1\right)$$

$$= n - 6\frac{n^2}{n} + 6n^2 = 6n^2 - 5n$$

$$\in \Theta(n^2)$$

Résumé

$$T(n) = 2T\left(\frac{n}{2}\right) + f\ (n), T(1) = 1$$

$f\ (n)$	$T(n)$	**Complexité**
$\sqrt{n}$	$\left(1 + \frac{1}{\sqrt{2}-1}\right)n - \frac{1}{\sqrt{2}-1}\sqrt{n}$	$\Theta(n)$
$2n$	$n + 2n\lg(n)$	$\Theta(n\lg(n))$
$3n^2$	$6n^2 - 5n$	$\Theta(n^2)$

III. Méthode générale

Théorème : Soit $T(n) = aT\left(\frac{n}{b}\right) + f(n)$, où $a \geq 1, b > 1$ sont des entiers, $f(n) > 0$. On interprète ici $\frac{n}{b}$ comme signifiant $\left\lfloor\frac{n}{b}\right\rfloor$ ou $\left\lceil\frac{n}{b}\right\rceil$.

- Cas 1 : f croît « **moins vite** » que $n^{\log_b(a)}$.
  S'il existe $\epsilon > 0$ tel que $f(n) = O\left(n^{\log_b(a)-\epsilon}\right)$, alors $T(n) = \Theta\left(n^{\log_b(a)}\right)$.
- Cas 2 : f croît **au même rythme** que $n^{\log_b(a)}$.
  Si $f(n) = \Theta\left(n^{\log_b(a)}\right)$, alors $T(n) = \Theta\left(n^{\log_b(a)}\lg(n)\right)$.
- Cas 3 : f croît « **plus vite** » que $n^{\log_b(a)}$.
  S'il existe $\epsilon > 0$ tel que $f(n) = \Omega\left(n^{\log_b(a)+\epsilon}\right)$, et si $af\left(\frac{n}{b}\right) \leq cf(n)$ pour une certaine constante $c < 1$ et pour n suffisamment grand, alors $T(n) = \Theta(f(n))$.

Cas où la méthode générale ne s'applique pas

- Cas 1 : f croît **moins vite** que $n^{\log_b(a)}$ mais pas « polynomialement » moins vite.
  Exemple : $f(n) = \frac{n^{\log_b(a)}}{\lg(n)}$
- Cas 3a : f croît **plus vite** que $n^{\log_b(a)}$ mais pas « polynomialement » plus vite.
  Exemple : $f(n) = n^{\log_b(a)}\lg(n)$
- Cas 3b : f croît « polynomialement » **plus vite** que $n^{\log_b(a)}$ mais ne respecte pas la deuxième condition dite de « régularité ».

Exemples

$T(n) = 2T\left(\frac{n}{2}\right) + f\ (n)$

$a = b = 2$, donc $\log_b(a) = 1$.

1. $f(n) = \sqrt{n} = O\left(n^{1-\frac{1}{2}}\right)$, donc le cas 1 s'applique avec $\varepsilon = \frac{1}{2}$, et on obtient ainsi $T(n) = \Theta\left(n^{\log_b(a)}\right) = \Theta(n)$.

2. $f(n) = 2n = \Theta(n)$, donc le cas 2 s'applique, et on obtient ainsi $T(n) = \Theta\left(n^{\log_b(a)}\lg(n)\right) = \Theta(n\lg(n))$.

3. $f(n) = 3n^2 = \Omega(n^{1+1})$, donc le cas 3 s'applique avec $\varepsilon = 1$ (car $af\left(\frac{n}{b}\right) = 6\left(\frac{n}{2}\right)^2 = \frac{3}{2}n^2 \leq cf(n)$ avec $c = \frac{1}{2}$), et on obtient ainsi $T(n) = \Theta(f(n)) = \Theta(n^2)$.

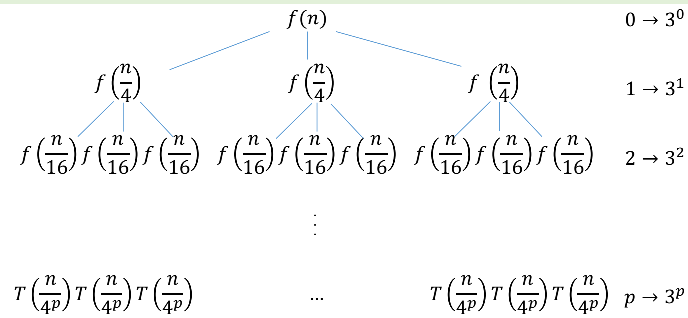
Exemple avec les trois méthodes

$$T(n) = 3T\left(\left\lceil\frac{n}{4}\right\rceil\right) + n^2, T(1) = 1.$$

Résoudre cette équation de récurrence :

- Avec la méthode de l'arbre récursif
- Avec la méthode de substitution
- Avec la méthode générale

Arbre de hauteur p si $n = 4^p$



Calcul de $T(n)$

$$T(n) = 3T\left(\frac{n}{4}\right) + n^2, T(1) = 1, f(n) = n^2, n = 4^p$$

$$T(n) = \sum_{i=0}^{p-1} 3^i f\left(\frac{n}{4^i}\right) + 3^p$$

Or, $3^p = 3^{\log_4(n)} = n^{\log_4(3)}$ et donc :

$$T(n) = \sum_{i=0}^{p-1} 3^i f\left(\frac{n}{4^i}\right) + n^{\log_4(3)}$$

$$T(n) = \sum_{i=0}^{p-1} 3^i \left(\frac{n}{4^i}\right)^2 + n^{\log_4(3)}$$

$$T(n) = n^2 \sum_{i=0}^{p-1} \left(\frac{3}{16}\right)^i + n^{\log_4(3)}$$

Calcul de $T(n)$

$$T(n) = n^2 \frac{\left(\frac{3}{16}\right)^p - 1}{\frac{3}{16} - 1} + n^{\log_4(3)}$$

$$T(n) = n^2 \frac{\left(\frac{3}{16}\right)^{\log_4(n)} - 1}{\frac{3}{16} - 1} + n^{\log_4(3)}$$

$$T(n) = \frac{16}{13} n^2 - \frac{16}{13} n^{\log_4(3)} + n^{\log_4(3)} \in \Theta(n^2)$$

Résolution par substitution

On veut vérifier que $T(n) = \Omega(n^2)$.
On doit donc trouver une constante $c > 0$ telle que $T(n) \geq cn^2$

- **Récurrence** : on suppose $T\left(\left\lceil\frac{n}{4}\right\rceil\right) \geq c\left\lceil\frac{n}{4}\right\rceil^2$. Si on remplace (substitution) dans la relation de récurrence, on obtient :

$$\begin{aligned} T(n) = 3T\left(\left\lceil\frac{n}{4}\right\rceil\right) + n^2 &\geq 3c\left\lceil\frac{n}{4}\right\rceil^2 + n^2 \\ &\geq \frac{3}{16}cn^2 + n^2 \\ &\geq \left(\frac{3}{16}c + 1\right)n^2 \\ &\geq cn^2, \end{aligned}$$

la dernière égalité étant vraie pour toute constance $c \leq \frac{16}{13}$.

- **Initialisation** : Si on prend $c = 1$, la propriété est vérifiée pour $n = 1$.
- Remarque : On montre que $T(n) = O(n^2)$ de la même manière, en utilisant la règle de la l'harmonie.

Méthode générale

- $T(n) = 3T\left(\left\lceil\frac{n}{4}\right\rceil\right) + n^2, T(1) = 1.$
- $a = 3, b = 4.$
- $f(n) = n^2 = \Omega\left(n^{\log_4(3)+2-\log_4(3)}\right)$, donc le cas 3 s'applique avec $\varepsilon = 2 - \log_4(3) > 0$.
- On doit vérifier la deuxième hypothèse :

$af\left(\frac{n}{b}\right) = 3\left(\frac{n}{4}\right)^2 = \frac{3}{16}n^2 \leq cf(n)$ avec $c = \frac{3}{16}$.

- Conclusion : $T(n) = \Theta(f(n)) = \Theta(n^2)$.