Chapitre 6

Récurrences

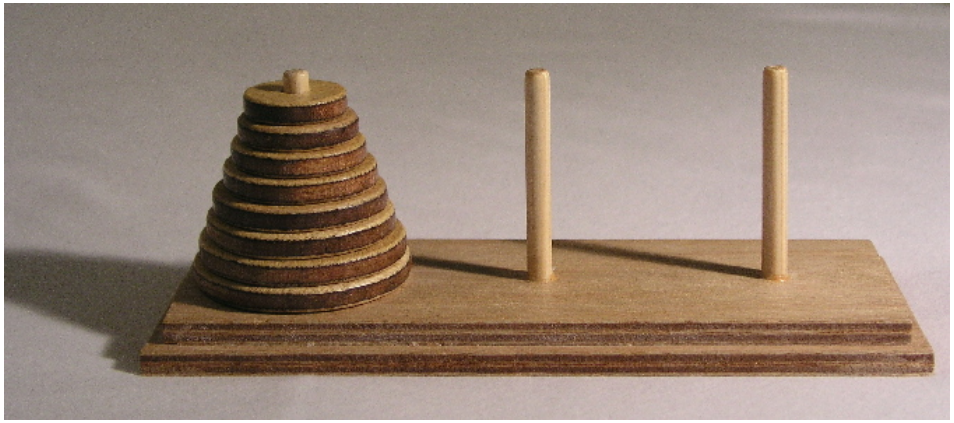
Introduction

Rappel : La notation asymptotique vue dans le chapitre 5 permet d'approximer la complexité des algorithmes.

But de ce chapitre : Étudier des méthodes permettant de résoudre des *équations récurrentes.*

Motivation : La complexité des *algorithmes récursifs* est souvent calculable à partir d'équations récurrentes.

Tours de Hanoï



Source : `http ://fr.wikipedia.org/wiki/Tours_de_Hanoï`

But : Déplacer la tour complète de la première tige vers une des deux autres tiges.

Contraintes :

- On ne peut déplacer qu'un seul disque à la fois.
- Un disque ne peut jamais être déposé sur un disque de diamètre inférieur.

Solution récursive :

- Cas de base : Déplacer une tour d'un seul disque est immédiat.
- Cas récursif : Pour déplacer une tour de $n+1$ disques de la première vers la troisième tige en connaissant une solution pour le déplacement d'une tour de n disques :
  1. Par récursion, déplacer n disques vers la deuxième tige ;
  2. Déplacer le disque restant vers la troisième tige ;
  3. Par récursion, déplacer les n disques de la deuxième tige vers la troisième tige.

Notation : Soit T_n le nombre minimum d'étapes nécessaires au déplacement d'une tour de n disques d'une tige vers une autre.

Propriété (borne supérieure) : On a $T_n \leq 2T_{n-1} + 1$.

Remarques :

- Pour déplacer une tour, il faut obligatoirement déplacer le disque du bas.
- Accéder au disque du bas nécessite de déplacer tous les autres disques vers une tige libre (au moins T_{n-1} étapes).
- Ensuite, il faut remettre en place le reste de la tour (au moins T_{n-1} étapes).

On a donc la propriété suivante :
Propriété (borne inférieure) : On a $T_n \geq 2T_{n-1} + 1$.

Récurrence : On sait à présent que $T_n = 2T_{n-1} + 1$.

Question : Comment calculer *efficacement* T_n, pour de grands n ?

Solution : Trouver une solution analytique pour T_n.

Méthode “Deviner-et-Vérifier”

Principes :

1. Calculer les quelques premières valeurs de T_n ;
2. Deviner une solution analytique ;
3. Démontrer qu'elle est correcte (par exemple par induction).

Application :

n	1	2	3	4	5	6	$\cdots$
T_n	1	3	7	15	31	63	$\cdots$
- On devine $T_n = 2^n - 1$
- On peut le démontrer par induction (exercice).

Méthode “Plug-and-Chug” (force brute)

1. “Plug” (appliquer l’équation récurrente) et “Chug” (simplifier)

$$\begin{aligned} T_n &= 1 + 2T_{n-1} \\ &= 1 + 2(1 + 2T_{n-2}) \\ &= 1 + 2 + 4T_{n-2} \\ &= 1 + 2 + 4(1 + 2T_{n-3}) \\ &= 1 + 2 + 4 + 8T_{n-3} \\ &= \dots \end{aligned}$$

Remarque : Il faut simplifier *avec modération*.

2. Identifier et vérifier un modèle
   - Identification :

$$T_n = 1 + 2 + 4 + \cdots + 2^{i-1} + 2^i T_{n-i}$$

   - Vérification en développant une étape supplémentaire :

$$\begin{aligned} T_n &= 1 + 2 + 4 + \cdots + 2^{i-1} + 2^i(1 + 2T_{n-(i+1)}) \\ &= 1 + 2 + 4 + \cdots + 2^{i-1} + 2^i + 2^{i+1} T_{n-(i+1)} \end{aligned}$$

3. Exprimer le $n^{\text{ème}}$ terme en fonction des termes précédents
   En posant $i = n - 1$, on obtient

$$\begin{aligned} T_n &= 1 + 2 + 4 + \cdots + 2^{n-2} + 2^{n-1} T_1 \\ &= 1 + 2 + 4 + \cdots + 2^{n-2} + 2^{n-1} \end{aligned}$$

4. Trouver une solution analytique pour le $n^{\text{ème}}$ terme

$$
\begin{aligned}
T_n &= 1 + 2 + 4 + \cdots + 2^{n-2} + 2^{n-1} \\
&= \sum_{i=0}^{n-1} 2^i \\
&= \frac{1 - 2^n}{1 - 2} \\
&= 2^n - 1
\end{aligned}
$$

Tri par fusion

Algorithme : Pour trier une liste de n éléments,

1. Diviser la liste en deux ;
2. Trier récursivement les deux sous-listes ;
3. Fusionner les deux listes triées.

Complexité : Soit T_n le nombre *maximum* de comparaisons à effectuer pour trier une liste de n éléments.

- On a $T_1 = 0$.
- Si $n > 1$, alors :
  - au pire $2T_{n/2}$ comparaisons pour trier les deux sous-listes ;
  - au pire $n - 1$ comparaisons pour fusionner deux sous-listes triées.

  Donc, $T_n = 2T_{n/2} + n - 1$.

Exercice : Trouver une solution analytique pour T_n.
(réponse : $T_n = n \log n - n + 1 \approx n \log n$).

Remarque

- On a supposé n puissance de 2 pour simplifier les développements
- La vraie récurrence est du type :
  - $T(1) = 1$
  - $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n - 1$ (Pour $n > 1$)
- Généralement, on peut ignorer les problèmes d'arrondis car ils n'affectent pas la complexité

Comparaison

- Tours de Hanoï :
  - $T_1 = 1$
  - $T_n = 2T_{n-1} + 1$
  - Solution analytique : $T_n = 2^n - 1$

- Tri par fusion :
  - $T_1 = 0$
  - $T_n = 2T_{n/2} + n - 1$
  - Solution analytique : $T_n \approx n \log n$

Générer des petits sous-problèmes mène en général à des solutions plus rapides que celles pour lesquelles on tente en priorité de réduire le travail additionnel à faire à chaque appel récursif.

Un algorithme rapide

Soit un algorithme pour lequel, à chaque étape, les données du problème sont divisées en deux, et une seule étape additionnelle est nécessaire pour regrouper les résultats.

La complexité de cet algorithme est décrite par la récurrence suivante :

- $S_1 = 0$
- $S_n = 2S_{n/2} + 1$ (avec $n \geq 2$).

A l'aide de la méthode "Deviner-et-Vérifier", on obtient

n	S_n
1	0
2	$2S(1)+1=1$
4	$2S(2)+1=3$
8	$2S(4)+1=7$
16	$2S(8)+1=15$

On devine donc la solution $S_n = n-1$.

La vérification est immédiate :

Théorème : Supposons

- $S_1 = 0$
- $S_n = 2S_{n/2} + 1$ (avec $n \geq 2$).

Si n est une puissance de 2, alors $S_n = n - 1$.

Démonstration : (par induction forte)

- Soit $P(n) =$ "Si n est une puissance de 2, alors $S_n = n - 1$".
- *Cas de base :* $P(1)$ est vrai car $S_1 = 1 - 1 = 0$.
- *Cas inductif :* Supposons $P(1), P(2), \ldots, P(n-1)$.
  - Si n n'est pas une puissance de 2, alors $P(n)$ est trivialement vrai.
  - Sinon, $S_n = 2S_{n/2} + 1 = 2(\frac{n}{2} - 1) + 1 = n - 1$. □

Attention aux pièges de l'induction

Théorème : Si n est une puissance de 2, alors $S_n \leq n$
(vrai puisque on vient de démontrer $S_n = n - 1$)

Essai de démonstration :

- Par induction forte avec $P(n) =$"Si n est une puissance de 2, alors $S_n \leq n$"
- *Cas de base :* $P(1)$ est vrai car $S_1 = 0 \leq 1$.
- *Cas inductif :* Supposons $P(1), P(2), \ldots, P(n-1)$.
  - Si n n'est pas une puissance de 2, alors $P(n)$ est trivialement vrai.
  - Sinon, $S_n = 2S_{n/2} + 1 \leq 2(n/2) + 1 = n + 1 \not\leq n$

□

Exercice : quid de $S_n \leq 2n$ ou $S_n \leq n - 2$

Variation du nombre de sous-problèmes

Supposons :

- $T_1 = 1$;
- $T_n = aT_{n/2} + n$.

Solution (qui peut être obtenue par la méthode "Plug-and-Chug") :

$$T_n \approx \begin{cases} \dfrac{2n}{2-a} & \text{pour } 0 \leq a < 2; \\ n \log n & \text{pour } a = 2; \\ \dfrac{an^{\log a}}{a-2} & \text{pour } a > 2. \end{cases}$$

Observation : La solution dépend fortement de la valeur de a.

Une première généralisation

Théorème (Master theorem) : Soient deux constantes $a \geq 1$ et $b > 1$ et une fonction $f(n) = O(n^d)$ avec $d \geq 0$. La complexité asymptotique de la récurrence suivante :

$$T(n) = aT(n/b) + f(n)$$

est :

$$T(n) = \begin{cases} O(n^d) & \text{pour } d > \log_b a \\ O(n^d \log n) & \text{pour } d = \log_b a; \\ O(n^{\log_b a}) & \text{pour } d < \log_b a. \end{cases}$$

(Introduction to algorithms, Cormen et al.)

Exemple d'application

- Soit la récurrence suivante :

$$T(n) = 7T(n/2) + O(n^2).$$

(Méthode de Strassen pour la multiplication de matrice)

- $T(n)$ satisfait aux conditions du théorème avec $a = 7$, $b = 2$, et $d = 2$.
- $\log_b a = \log_2 7 = 2.807... \Rightarrow d = 2 < \log_b a = 2.807...$
- Par le troisième cas du théorème, on a :

$$T(n) = O(n^{\log_b a}) = O(n^{2.807...}).$$

Une seconde généralisation

Forme générale d'une récurrence "Diviser pour régner" :

$$T(x) = \begin{cases} \text{est défini} & \text{pour } 0 \leq x \leq x_0 \\ \displaystyle\sum_{i=1}^{k} a_i T(b_i x) + g(x) & \text{pour } x > x_0 \end{cases}$$

avec

- $a_1, a_2, \ldots, a_k > 0$,
- $b_1, b_2, \ldots, b_k \in [0, 1[$,
- x_0 suffisamment grand,
- $|g'(x)| = O(x^c)$ pour un $c \in \mathbb{N}$.

Théorème (Akra-Bazzi) :

$$T(x) = \Theta\left(x^p\left(1 + \int_1^x \frac{g(u)}{u^{p+1}}\,\mathrm{d}u\right)\right)$$

où

- p satisfait l'équation $\sum_{i=1}^{k} a_i b_i^p = 1$

Exemple d'application

- Soit la récurrence "diviser pour régner" suivante :

$$T(x) = 2T(x/2) + 8/9\,T(3x/4) + x^2$$

- On a bien $|g'(x)| = |2x| = O(x)$
- Trouvons p satisfaisant :

$$2(\frac{1}{2})^p + \frac{8}{9}(\frac{3}{4})^p = 1$$

$\Rightarrow p = 2$

- Par application du théorème, on obtient :

$$\begin{aligned} T(x) &= \Theta\left(x^2\left(1 + \int_1^x \frac{u^2}{u^3}du\right)\right) \\ &= \Theta(x^2(1 + \log x)) \\ &= \Theta(x^2 \log x) \end{aligned}$$

Changement de variables

- Considérons la récurrence suivante :

$$T(n) = 2T(\sqrt{n}) + \log n$$

- Posons $m = \log n$. On a :

$$T(2^m) = 2T(2^{m/2}) + m.$$

- Soit $S(m) = T(2^m)$. On a :

$$S(m) = 2S(m/2) + m \Rightarrow S(m) = O(m \log m).$$

- Finalement :

$$T(n) = T(2^m) = S(m) = O(m \log m) = O(\log n \log \log n).$$