Complexité
TD 2 - Correction

Master 1 Informatique

Serge Miguet, Guillaume Metzler, Tess Masclef
Institut de Communication (ICOM)
Université de Lyon, Université Lumière Lyon 2
serge.miguet@univ-lyon2.fr guillaume.metzler@univ-lyon2.fr
tess.masclef@univ-lyon2.fr

Tour de Hanoï

Etude des cas simples Comme dans le TD précédent, nous allons commencer par étudier le nombre de mouvements nécessaires pour déplacer une tour de hauteur de taille $n = 0, 1, 2$ ou 3. On notera x_n le nombre d'opérations à effectuer et G-M-D les cases de *gauche*, du *milieu*, et de *droite* respectivement.

- $n = 0$: il n'y rien à faire, on a donc $x_0 = 0$.
- $n = 1$: on doit faire la manipulation (1) déplacer l'anneau sur M, on a donc $x_1 = 1$
- $n = 2$: on doit effectuer les manipulations suivantes : (1) petit anneau sur D (2) grand anneau sur M (3) petit anneau sur M On a donc $x_2 = 3$
- $n = 3$: les manipulations à faire sont les suivantes (1) petit anneau sur M (2) anneau moyen sur D (3) petit anneau sur D (4) grand anneau sur M (5) petit anneau sur G (6) anneau moyen sur M et (7) petit anneau sur M. On a donc $x_3 = 7$

Un algorithme récursif L'étude précédente semble suggérer que la complexité du problème n'est pas linéaire mais plutôt exponentielle. C'est ce que nous allons montrer par la suite.

Considérons une tour composée de n anneaux que l'on souhaite déplacer, notons alors x_n le nombre d'opérations nécessaires pour déplacer ces n anneaux. Nous allons essayer de décomposer les opérations afin d'en déduire un processus itératif.

Pour déplacer notre tour de taille n, nous allons :

1. déplacer une tour de taille $n-1$, ce qui nécessitera x_{n-1} opérations. De sorte à ce qu'il ne reste que l'anneau le plus grand sur notre pile initiale.
2. on déplacement ensuite ce plus grand anneau sur la pile qui est libre. On effectue donc un mouvement (on retrouve notre $x_1 = 1$)
3. on déplace à nouveau notre pile de n_1 anneaux sur le plus grand anneau, ce qui nécessite à nouveau x_{n-1} opérations.

On en déduit la relation de récurrence suivante :

$$x_n = 2x_{n-1} + 1. \tag{1}$$

Ce que l'on peut traduire par : le nombre d'opéprations (x_n) nécessaires pour déplacer une tour de taille n est égal à deux fois le nombre d'opérations pour déplacer une tour de taille $n-1$, auquel j'ajoute 1.

A partir de la relation de récurrence (1) obtenue, nous pouvons en déduire une expression de x_n en fonction de n et donc la complexité de notre algorithme itératif. La relation (1) porte également le nom d'*équation séquentielles*, on peut aussi y reconnaître une *suite arithmético-géométrique*. On se propose de trouver une expression de x_n par trois approche différentes :

- **Approche itérative.**

  On peut simplement utiliser la relation de récurrence pour en déduire une forme générale de $(x_n)_{n\in\mathbb{N}}$:

$$\begin{aligned} x_n &= 2x_{n-1} + 1, \\ &= 2^2 x_{n-2} + 2 + 1, \\ &= 2^3 x_{n-3} + 4 + 2 + 1, \\ &= 2^4 x_{n-2} + 8 + 6 + 2 + 1, \\ &= \cdots, \\ x_n &= 2^n x_0 + 2^{n-1} + 2^{n-2} + \cdots + 2 + 1, \end{aligned}$$

or $x_0 = 0$, on a donc

$$x_n = 2^{n-1} + 2^{n-2} + \cdots + 2 + 1,$$

On y reconnaît donc la somme des termes d'une suite géométrique de raison 2^n, ainsi

$$x_n = \sum_{k=1}^{n-1} 2^k = \frac{2^n - 1}{2 - 1} = 2^n - 1.$$

Nous aurions également pu rédiger notre récurrence proprement sur le même principe.

- **Approche équation séquentielle.**

  Pour trouver une expression de x_n on va additioner une solution de l'équation dite *homogène* avec une solution dite *particulière*.

  On commence par résoudre l'équation dite *homogène*

  $$x_n = 2x_{n-1}.$$

  On reconnait une suite géométrique de raison 2.

  Donc les solutions sont de la forme $x_n = \lambda \times 2^n$, où $\lambda \in \mathbb{R}$. Enfin on cherche une solution particulière à notre équation séquentielle. La suite constante égale à -1 vérifie notre équation.

  On en déduit :

  $$x_n = \lambda \times 2^n - 1.$$

  Or $x_0 = 0$, donc $\lambda = 1$, d'où :

  $$x_n = \times 2^n - 1.$$

- **Approche arithmético-géométrique.** Pour déterminer une expression de la suite $(x_n)_{n\in\mathbb{N}}$, nous allons considérer la suite auxiliaire $(y_n)_{n\in\mathbb{N}}$ définie par $y_n = x_n + 1$. On a alors

  $$y_{n+1} = \underbrace{x_{n+1}}_{\downarrow \text{ définition de } (x_n)_{n\in\mathbb{N}}} + 1,$$

$$
\begin{aligned}
&= 2x_n + 1 + 1, \\
&= 2x_n + 2, \\
&= 2y_n.
\end{aligned}
$$

La suite $(y_n)_{n\in\mathbb{N}}$ est donc suite géométrique de raison 2 et premier terme $y_0 = x_0 + 1 = 1$. On a donc

$$y_n = 2^n,$$

donc

$$x_n = y_n - 1 = 2^n - 1.$$

On ainsi montré que notre algorithme a une complexité exponentielle

Premier mouvement et parité Dans ce qui précède nous n'avons pas tenu compte du fait que l'on souhaite obtenir une nouvelle tour la case centrale (en réalité cela ne change pas la complexité) mais cela va avoir une influence sur la premier mouvement à effectuer comme nous avons pu le voir dans la première partie. Ainsi :

- Si $n \in 2\mathbb{N}$, alors on doit toujours déplacer l'anneau le plus petit, lors du premier mouvement, sur la case de droite.
- Si $n \in 2\mathbb{N} + 1$, on doit toujours déplacer le plus petit anneau, lors du premier mouvement, sur la case du milieu .

Quelques compléments Cet exercice nous a montré que si nous avons une relation de récurrence de la forme

$$x_n = ax_{n-1} + b,$$

où a et b sont des constantes réelles, alors la complexité est **exponentielle**. Plus précisément $x_n \underset{n\to\infty}{\sim} \mathcal{O}(a^n)$.

Si la relation de récurrence est de la forme $x_n = x_{n-1} + n$ où $b \in \mathbb{R}$, alors la complexité est **quadratique**. On a donc $x_n \underset{n\to\infty}{\sim} \mathcal{O}(n^2)$.
En effet, nous avons

$$
\begin{aligned}
x_n &= x_{n-1} + n, \\
&= x_{n-2} + n - 1 + n,
\end{aligned}
$$

$$
\begin{aligned}
&= x_{n-3} + n - 2 + n - 1 + n, \\
&= \cdots, \\
x_n &= 1 + 2 + \cdots + n, \\
x_n &= \frac{n(n+1)}{2}.
\end{aligned}
$$

Si la relation de récurrence est de la forme $x_n = x_{n-1} + b$ où $b \in \mathbb{R}$, alors la complexité est **linéaire**. On a donc $x_n \underset{n\to\infty}{\sim} \mathcal{O}(bn)$.

On rappelle aussi (vu en cours) qu'une relation de récurrence de la forme $x_n = x_{n/2} + 1$ implique que notre algorithme a une complexité **logarithmique**. On a donc $x_n \underset{n\to\infty}{\sim} \mathcal{O}(log_2(n))$, ici le log en base 2.

Enfin, la relation de récurrence est de la forme $x_n = x_{n-1}$, alors la complexité est **constante**, *i.e.* elle ne dépend pas de le taille des objets considérés. On a donc $x_n \underset{n\to\infty}{\sim} \mathcal{O}(1)$.

Implémentation Un code permettant de mettre en évidence la complexité mais aussi la résolution du problème est disponible au lien ci-dessous :

Code Commenté