TD #10 — Trois récurrences à analyser (avec solutions)

Exercice 1. *Tours de Hanoï*

L'algorithme récursif usuel de résolution du problème des *tours de Hanoï* (si vous ne connaissez pas allez voir, c'est rigolo) a un coût $T(1) = 1$, et $T(n) = 2T(n-1) + 1$ pour une tour de n étages.

1. Dessinez l'arbre de la récurrence et donnez une estimation de $T(n)$ en fonction de n.

En dessinant l'arbre, on constate qu'il a n niveaux de $1, 2, \ldots, 2^{n-1}$ appels récursifs (ou cas de base), chacun de coût marginal 1. On peut donc estimer $T(n) = \sum_{i=0}^{n-1} 2^i = 2^n - 1$.

2. Prouvez que l'on a $T(n) = \textit{ici : votre estimation}$.

On prouve par récurrence que $T(n) = 2^n - 1$.

— Cas de base : $T(1) = 1 = 2^1 - 1$.
— Conservation : On a :

$$\begin{aligned} T(n) &= 2^n - 1 && \text{Hyp. de récurrence} \\ T(n+1) &= 2T(n) + 1 && \text{Hyp. initiale} \\ &= 2 \times (2^n - 1) + 1 && \text{Substitution} \\ &= 2^{n+1} - 1 \end{aligned}$$

Remarque : L'arbre de récurrence que l'on a considéré est un *arbre binaire complet*, qui pour un problème à n étages est de *profondeur* $n-1$. On vient donc en fait de montrer qu'un arbre binaire complet de profondeur n possède $2^{n+1} - 1$ *nœuds*.

Exercice 2. *Calcul naïf d'un nombre de lapins*

Un calcul récursif naïf des termes de la suite « de Fibonacci » a un coût (s'approchant par) $T(0) = T(1) = 1$, et $T(n) = T(n-1) + T(n-2)$.

1. Montrez par récurrence forte que l'on a $T(n) \leq \varphi^n$, avec $\varphi = \frac{1+\sqrt{5}}{2}$ satisfaisant $\varphi^2 = \varphi + 1$.

— Cas de base : $T(0) = T(1) = 1 < 1.5 < \varphi$
— Conservation (pour $n \geq 2$).

$$\begin{aligned} T(0 \leq x \leq n-1) &\leq \varphi^x && \text{Hyp. de récurrence} \\ T(n) &= T(n-1) + T(n-2) && \text{Hyp. initiale} \\ &\leq \varphi^{n-1} + \varphi^{n-2} && \text{Substitution} \\ &= \varphi^{n-2}(\varphi + 1) \\ &= \varphi^{n-2}(\varphi^2) && \text{Définition de } \varphi \\ &= \varphi^n \end{aligned}$$

Exercice 3. *Algorithme « de Strassen » de multiplication de matrices*

Le coût total de l'algorithme « de Strassen » de multiplication de matrices peut s'approcher par la récurrence $T(1) = 1$, $T(n) \leq 7T(n/2) + a(n/2)^2$.

1. Dessinez l'arbre de la récurrence pour n une puissance de deux, et donnez une estimation de $T(n)$ sous la forme d'un $\mathcal{O}$.
(Dans cette question on pourra prendre $a = 4$ pour simplifier le terme $a(n/2)^2$ en n^2.)

On obtient un arbre de $1 + \log n$ niveaux de coûts marginaux $7^i(n/2^i)^2$, pour $0 \leq i \leq \log n$. On estime donc :

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log n} 7^i \left(\frac{n}{2^i}\right)^2 \\
&= n^2 \sum_{i=0}^{\log n} \left(\frac{7}{4}\right)^i \\
&\leq C \times n^2 \times \left(\frac{7}{4}\right)^{1+\log n} && C \text{ constante} \\
&= C \times n^2 \times \left(\frac{7^{1+\log n}}{(2^{1+\log n})^2}\right) \\
&= \mathcal{O}(7^{\log n}) \\
&= \mathcal{O}(n^{\log 7})
\end{aligned}
$$

2. Prouvez que pour $N = 2^n$ une puissance de deux l'on a $T(N) \leq \lambda N^\omega + \mu N^2$, avec $\omega = \log 7$ et λ et μ deux constantes à déterminer. Procédez par récurrence en considérant d'abord le cas récursif pour déterminer μ (en fonction de a), puis seulement ensuite le cas de base pour déterminer λ.

Cas récursif. On a :

$$
\begin{aligned}
T(N) &\leq \lambda N^\omega + \mu N^2 && \text{Hyp. de récurrence} \\
T(2N) &\leq 7T(N) + aN^2 && \text{Hyp. initiale} \\
T(2N) &\leq 7(\lambda N^\omega + \mu N^2) + aN^2 && \text{Substitution} \\
&= \lambda(2N)^\omega + (7\mu + a)N^2 && 7 = 2^\omega
\end{aligned}
$$

On peut alors choisir μ satisfaisant $7\mu + a = 4\mu$ afin d'avoir $(7\mu + a)N^2 = \mu(2N)^2$, ce qui donne $\mu = -a/3$.

Cas de base. Il suffit de choisir λ de façon à avoir $T(1) = 1 \leq \lambda + \mu$, ce qui est satisfait pour $\lambda = 1 - \mu = 1 + a/3$.

Remarque : Dans l'algorithme original « de Strassen » la constante a vaut 18, ce qui donne $T(N) \leq 7N^\omega - 6N^2$. La variante « de Winograd » abaisse a à 15, ce qui donne $T(N) \leq 6N^\omega - 5N^2$.