DM #3 — Deux exercices d'algorithmique d'automne (avec solutions)

Exercice 1. *Les œufs mystérieux*

On vous donne un panier d'œufs mystérieux et vous indique une tour de N étages. Votre tâche est de déterminer à partir de quel étage lâcher un œuf le cassera. Vous savez que cet étage existe et est le même pour tous les œufs, que si un œuf n'est pas cassé (respectivement, est cassé) à un étage il ne sera pas cassé (resp., sera cassé) à tout étage inférieur (resp., supérieur). De plus, un œuf lâché mais non cassé n'est pas endommagé : il ne cassera pas plus bas qu'initialement.

L'objectif de cet exercice est de concevoir trois algorithmes permettant de déterminer le premier étage à partir duquel les œufs cassent, avec trois limites différentes sur le nombre *maximum* (c'est à dire, dans le pire cas) d'œufs que vous pouvez être amené à casser afin de trouver le résultat.

On demande seulement une description informelle (mais néanmoins précise !) des algorithmes.

1. Donnez un algorithme cassant *un* œuf dans le pire cas. Quel est le nombre maximum de lâchés effectués dans le pire cas ?

> On commence par lâcher l'œuf à l'étage 1, et tant qu'il n'est pas cassé on le lâche à l'étage immédiatement supérieur. On renvoie le numéro de l'étage où l'œuf casse. Le nombre maximum de lâchés effectués dans le pire cas est N (ou $N-1$ si l'on exploite la garantie d'existence d'un étage où les œufs cassent).

2. Donnez un algorithme effectuant le plus petit nombre de lâchés dans le pire cas. Quel est le nombre maximum d'œufs cassés dans le pire cas (à une constante près) ?

> On procède par dichotomie : on définit un intervalle d'étages candidats `b, t` valant initialement `1, N` et on lâche un œuf au milieu de l'intervalle `m = b + (t-b)/2`. S'il casse on met à jour l'intervalle à `b, m - 1`, et sinon à `m + 1, t`, et on continue récursivement jusqu'à atteindre un intervalle de taille 1. On renvoie l'étage ainsi déterminé si l'œuf s'y casse, ou celui immédiatement supérieur sinon. Le nombre maximum d'œufs cassés est égal au nombre maximum de lâchés, qui est au plus $\lceil \log(N) \rceil >$ puisque la taille de l'intervalle candidat est (environ) divisée par deux à chaque étape.

3. Donnez un algorithme cassant *deux* œuf dans le pire cas, dont le nombre de lâchés effectués dans le pire cas est $\sqrt{N}$ (à d'éventuelles constantes multiplicatives et additives près).

> On définit $s = \lceil \sqrt{N} \rceil$, et on lâche un premier œuf aux étages de numéros $s, 2s, 3s, \ldots$ jusqu'à ce qu'il casse ou qu'on atteigne le sommet de la tour. Ceci nous prend au plus s lâchés puisque $s^2 \geq N$. On note t cet étage et a l'entier $t/s - 1$. On lâche maintenant un second œuf à partir de l'étage $as + 1$, et tant qu'il n'est pas cassé on le lâche à l'étage immédiatement supérieur. Ceci nous prend au plus s lâchés, et l'on renvoie le numéro de l'étage où l'œuf casse. Le nombre de lâchés dans le pire cas est majoré par $2s$, ce qui satisfait la contrainte donnée.

Exercice 2. *Le puzzle des tours de Hanoï*

On considère une tour de N étages (qui peut ou non servir à lancer des œufs) constitués de disques concentriques de rayon strictement décroissant (du bas en haut de la tour). Ces disques sont creux en leur centre et empilés dans un piquet, et l'on dispose de deux autres piquets vides dans lesquels aucun disque n'est initialement empilé. L'objectif du puzzle « des tours de Hanoï » consiste à transférer la tour du piquet sur lequel elle se trouve initialement sur un autre piquet (éventuellement désigné), en observant les contraintes suivantes :

- on ne peut retirer que les disques au sommet des piquets ;
- tout disque ne peut être empilé qu'au sommet d'un piquet vide ou sur un piquet dont le disque sommital est de rayon strictement plus grand (notamment, on ne peut pas stocker les disques ailleurs que sur un piquet).

1. Proposez un algorithme (décrit informellement mais précisément) permettant de résoudre ce problème.

> Notons p_1 le piquet initial, p_2 le piquet sur lequel on doit transférer la tour, p_3 le troisième piquet. Si la tour ne contient qu'un seul disque, on le transfère sur p_2 et l'on a gagné ; sinon on applique récursivement l'algorithme pour transférer la sous-tour constituée des $N-1$ disques sommitaux en le piquet p_3, on transfère le disque de base en p_2, puis l'on transfère à nouveau la sous-tour se trouvant en p_3 en p_2.

2. Analysez le coût de votre algorithme selon la métrique « nombre de déplacements effectués ».

Le déplacement d'une tour sans disque coûte zéro, et celui d'une tour d'un seul disque coûte un ; le déplacement d'une tour de N disques coûte deux fois le déplacement d'une tour de $N-1$ disques plus 1. La suite des coûts de l'algorithme pour des tours de hauteur 0, 1, 2,… est donc donnée par : $c_0 = 0$, $c_1 = 1$, $c_{n>1} = 2c_{n-1} + 1$. Les premiers termes de cette suite sont 0, 1, 3, 7, 15, 31, …, soit $2^0 - 1$, $2^1 - 1$, $2^2 - 1$, $2^3 - 1$, $2^4 - 1$, $2^5 - 1$, … On peut dès lors faire l'hypothèse que le coût c_n de l'algorithme pour une tour de n disques est donné par $2^n - 1$, ce que l'on va prouver par récurrence.
Pour $n \in \mathbb{N}$, on note $\mathcal{P}(n)$ la proposition « $c_n = 2^n - 1$ » :

— Initialisation : $c_0 = 0 = 2^0 - 1$: la proposition est vérifiée.
— Hérédité : on suppose $\mathcal{P}(n)$; dès lors :

$$\begin{aligned} c_{n+1} &= 2c_n + 1 && \text{par définition de la suite} \\ &= 2 \times (2^n - 1) + 1 && \text{par hypothèse de récurrence} \\ &= 2^{n+1} - 1 && \text{(par calcul)} \end{aligned}$$

On a a donc bien $\mathcal{P}(n) \Rightarrow \mathcal{P}(n+1)$.
— Conclusion : d'après le principe de récurrence, $\mathcal{P}(n)$ est vraie pour tout n.

3. Montrez que toute résolution du puzzle nécessite *au moins* $2^N - 1$ déplacements de disques.

La proposition à montrer est trivialement vérifiée pour $N = 0$ et $N = 1$. Il suffit ensuite de remarquer que les contraintes du puzzle imposent de déplacer les $N-1$ disques sommitaux avant de pouvoir déplacer le disque le plus grand, et qu'une fois ce déplacement effectué au moins un autre déplacement des $N-1$ disques sommitaux sera nécessaire. Le nombre de déplacements nécessaires pour des tours de hauteur 0, 1, … est donc minoré par $r_0 = 0$, $r_1 = 1$, $r_{n>1} = 2r_{n-1} + 1$. Cette suite est de définition identique à celle de la question précédente, et l'on a donc $r_n = 2^n - 1$, ce que l'on devait montrer.