Travaux Dirigés d'Algorithmique n°3

Cours d'Informatique de Deuxième Année

—Deug SSM M3-MI—

L'objet de cette feuille d'exercices est de vous amener à utiliser des algorithmes récursifs pour résoudre un problème. Nous aborderons les différents points suivants :

- écriture de fonctions résursives ;
- tableaux en paramètres de fonctions récursives ;
- complexité en temps.

▸ **Exercice 1.** (**fonction factorielle**) Nous rappelons la définition récursive de la fonction `factorielle` :

$$\texttt{factorielle}(n) := \begin{cases} 1 & \text{si } n = 0, \\ n \times \texttt{factorielle}(n-1) & \text{si } n > 0. \end{cases}$$

1. Donner une fonction récursive de nom `fact` qui calcule la factorielle d'un entier `n` passé en argument ;
2. Donner la complexité en temps de votre fonction.

▸ **Exercice 2.** (**fonction puissance**) Nous nous proposons de calculer récursivement a^n, a et n entiers, de trois manières différentes. Donner une fonction pour chacune des formules récursives suivantes et estimer la complexité en temps des trois algorithmes de calculs : $a^0 = 1$ et si $n > 0$,

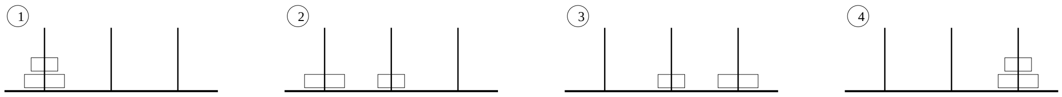
1. $a^n := a \times a^{n-1}$;
2. si n est pair alors $a^n := (a^2)^{\frac{n}{2}}$ sinon, $a^n := a \times (a^2)^{\frac{n-1}{2}}$;
3. si n est pair alors $a^n := (a^{\frac{n}{2}})^2$ sinon, $a^n := a \times (a^{\frac{n-1}{2}})^2$.

▸ **Exercice 3.** (**recherche simultanée du minimum et du maximum**) Nous souhaitons calculer simultanément le minimum et le maximum d'un tableau `t` de `n` entiers :

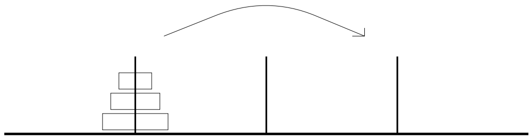
1. donner une fonction `MinMax` utilisant la méthode "naïve". Montrer que nous faisons au plus $2 \times (n-1)$ comparaisons entre les éléments du tableau ;

2. donner une fonction `MinMaxRec` utilisant la méthode "dichotomique"—récursive—. Montrer que nous faisons au plus $\lceil \frac{3\times n}{2} \rceil - 2$ comparaisons[1].

▸ **Exercice 4.** *Nous cherchons à résoudre le problème des "tours de Hanoï". Ce problème consiste à déplacer* `n` *disques d'un piquet de départ à un piquet d'arrivée en utilisant un troisième piquet. La seule contrainte tient au fait qu'il est interdit de poser un disque sur un disque plus petit que lui.*



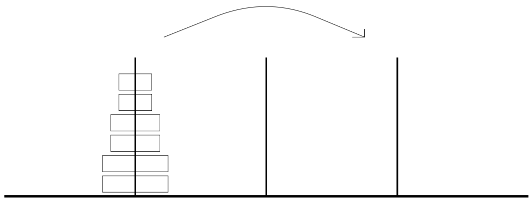
Résolution du problème pour $\mathtt{n} = 2$.



Exemple du problème pour $\mathtt{n} = 3$.

1. *résoudre "à la main" l'algorithme pour* $\mathtt{n} = 3$ *;*
2. *en déduire un algorithme récursif de résolution du problème ;*
3. *écrire la fonction correspondant à cet algorithme : la fonction devra afficher la suite des déplacements des disques ;*
4. *donner enfin, la complexité en temps de votre algorithme.*

▸ **Exercice 5.** (**exercice supplémentaire**) Nous considérons le problème des tours de Hanoï pour lequel chaque disque de taille `t` est présent `p` fois. Nous avons par ailleurs `n` tailles différentes de disque. Nous demandons dans un premier temps de résoudre le problème pour `p`= 2 puis d'en déduire l'algorithme général—et le programme *C*—correspondant à un nombre `p` quelconque d'occurences de chaque disque[2].



Exemple du problème pour $\mathtt{p} = 2$ *et* $\mathtt{n} = 3$.

[1] C'est-à-dire $\lfloor \frac{n}{2} \rfloor$ comparaisons en moins.
[2] Nous avons donc $\mathtt{n}\times\mathtt{p}$ disques.