ALGORITHMIQUE

# TD 1 : Compter & Prouver

**Exercice 0 (Addition)** Donnez des algorithmes pour additionner deux nombres. Comparez leurs complexités.

**Exercice 1 (Puissance)** Donnez deux algorithmes (un itératif & un récursif) rendant, à partir de deux entiers positifs $x$ & $y$, le nombre $x^y$. Donnez leurs complexités.

**Exercice 1' (Puissance) **** Prouvez les algorithmes précédents.

**Exercice 2 (Maximum d'un tableau)** Donnez deux algorithmes (un itératif & un récursif) qui calculent la plus grande valeur d'un tableau de nombres. Comparez leurs fonctionnements.

**Exercice 3 (Algorithme mystère) **** Que fait l'algorithme 1 ? Justifiez votre réponse. Donnez sa complexité dans le cas le meilleur. En déduire la complexité dans le cas le pire & en moyenne. Donnez une version récursive de cet algorithme.

**Algorithme 1**

**Données :** $A$ & $B$ deux entiers positifs
**Début**
  $x \leftarrow A$ ; $y \leftarrow B$ ; $R \leftarrow 1$
  **tant que** $y \neq 0$ **:**
    **si** $y$ est impair **:**
      $R \leftarrow R \times x$
      $y \leftarrow$ y - 1
    **sinon :**
      $x \leftarrow x \times x$
      $y \leftarrow y/2$
  **rendre** $R$
**Fin**

**Exercice 4 (Palindrome) *** Implémentez un algorithme récursif permettant de savoir si un tableau de caractères est un palindrome (un palindrome se lit "à l'endroit" & "à l'envers" de la même façon, comme par exemple "À l'étape, épate-la !" (on considère ni la ponctuation, ni les espaces, ni les accents)). Quelle est sa complexité ?

**Exercice 5 (Tours de Hanoï)** Les *tours de Hanoï* sont un célèbre casse tête qui consiste à déplacer $n$ disques de diamètres différents d'une tour de "départ" à une tour d' "arrivée" en passant par une tour "intermédiaire", tout en respectant les règles suivantes :

- on ne peut déplacer qu'un disque à la fois,
- on ne peut placer un disque sur un disque plus petit que lui.

On suppose que cette dernière règle est également respectée dans la configuration de départ.

Donnez un algorithme récursif permettant de résoudre le problème. Quelle est sa complexité ? Peut-on faire mieux ?

**Exercice 6 (Quick Sort) *** Donnez une version de QuickSort qui n'utilise pas de tableaux auxiliaires.



**Exercice 7 (Tri par Base) **** Ce tri s'applique *uniquement aux entiers positifs*, que l'on considère écrits en base 2[1]. Le principe de ce tri est très simple :

1. On considère d'abord le bit de poids le plus faible (i.e. le plus à droite) & on répartit l'ensemble à trier en deux sous ensembles :
   - les entiers dont le bit de poids le plus faible est 0
   - les entiers dont le bit de poids le plus faible est 1
2. On concaténe les deux sous-ensembles, en commençant par celui des bits à 0.
3. On recommence sur le bit à gauche de celui qu'on vient de traiter.
   ...

Les parcours d'ensembles se font, toujours, de la gauche vers la droite.

Donnez le pseudo-code de cet algorithme (on supposera que l'on dispose d'une fonction qui, étant donnés deux entiers $n$ & $i$, donne le $i^{me}$ bit de $n$).

Prouvez cet algorithme.

Donnez la complexité de cet algorithme. Rappelez la complexité minimale du tri (dans le cas le pire). Commentaires.

**Exercice 8 (Ackermann) **** La fonction d'Ackermann se définit de la manière suivante, pour tous entiers $m$ & $n$ positifs :

$$A(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ A(m-1,1) & \text{si } n = 0 \\ A(m-1, A(m,n-1)) & \text{sinon} \end{cases}$$

Cette fonction est d'une importance théorique certaine (c'est un exemple de fonction récursive, & non récursive primitive) & est célèbre du fait qu'elle grossit très vite alors qu'elle s'écrit simplement.

Vérifiez que cette définition est valide (*i.e.* s'arrête quelque soient $m$ & $n$).

**Exercice 0' (Division) **** Donnez des algorithmes pour diviser deux nombres. Comparez leurs complexités.

[1] Il est adaptable aux autres types de nombres (& en particulier, on peut très bien compter en base 10), mais ce n'est pas ici le sujet.