Informatique

TD 1 : Complexité

Exercice 0 (Addition) Donnez des algorithmes pour additionner deux entiers.

Exercice 1 (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 2 (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 3 (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] 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.

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 4 (Quicksort) Cet algorithme est lui aussi une application du *diviser-pour-régner*. Son principe est le suivant : On sépare les éléments du tableau à trier T en deux : ceux qui sont plus petits que le premier élément de T & ceux qui sont plus grands. On obtient deux "petits tableaux" que l'on trie puis que l'on concatène.

Donnez une version de cet algorithme n'utilisant pas de tableaux auxiliaires.