Algorithmes et Complexité II : TD1

Exercice 1 [Suite de Fibonacci]

- Écrire une fonction récursive qui calcul la n^{ieme} puissance d'un nombre donné x en temps $O(log(n))$.
  Indication $x^n = x^{\lfloor \frac{n}{2} \rfloor} \times x^{\lfloor \frac{n}{2} \rfloor} \times x^{n \ modulo \ 2}$.

- Rappeler la définition de la suite de Fibonacci. Donald Knuth a proposé la formule suivante pour le calcul du n^{ieme} terme de la suite de Fibonacci:

$$\begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$$

  Écrire une fonction qui calcule le n^{ieme} terme de la suite de Fibonacci en temps $O(log(n))$?

Exercice 2 [Recherche du $k$ème élément]

Vous avez un fichier contenant une liste de prix de voitures (les prix sont des entiers compris entre 5000 et 10000 euros, un prix par ligne). Vous devez écrire une solution qui prend en argument un entier k et qui affiche le prix de la $k$ème voiture la moins chère.

Exercice 3 [3-SAT]

Soit une collection de n variables booléennes $v_1, \ldots, v_n$ et soit une formule logique $\mathcal{F}$ s'écrivant en `3-CNF` *i.e.* $\mathcal{F}$ est la conjonction de m clauses, $\mathcal{F} = c_1 \wedge \cdots \wedge c_m$. Chaque clause est la disjonction de trois littéraux, $c_i = l_{i1} \vee l_{i2} \vee l_{i3}$. Chaque littéral peut être une variable ou sa négation. Note : lire `ET` pour l'opérateur $\wedge$ et lire `OU` pour $\vee$.

1. Supposons que les variables v_i sont stockées dans un tableau de n cases. Proposer une représentation mémoire de la formule $\mathcal{F}$.
2. Nous cherchons à vérifier s'il existe une affectation de valeurs de vérité aux variables v_i qui satisfait la formule $\mathcal{F}$ *i.e.* une assignation de valeurs pour laquelle $\mathcal{F}$ devient *vraie*. Proposer un algorithme brute-force qui vérifie cela.
3. Quelle est la complexité de votre algorithme ?