$1^{ère}$ année Magistère Informatique 2007 - 2008
Algorithmique

TD 2

Exercice 1 Démontrer le théorème suivant vu en cours.

Théorème : Récurrences de partitions

Soit $t : \mathbb{N} \to \mathbb{R}_+$ *une fonction croissante à partir d'un certain rang et telle que* $\exists n_0 > 0,\ \exists b > 1,\ \exists k \geq 0,\ \exists a, c, d > 0$ *vérifiant*

$$\begin{cases} t(n_0) = d \\ t(n) = at(n/b) + cn^k & n > n_0 \text{ et } n/n_0 \text{ puissance de } b \end{cases}$$

alors,

$$t_n = \begin{cases} \theta(n^k) & a < b^k \\ \theta(n^k log_b n) & a = b^k \\ \theta(n^{log_b a}) & a > b^k \end{cases}$$

Exercice 2

1. Proposer un algorithme simple pour le calcul de a^n ; analyser sa complexité.
2. Proposer un algorithme "diviser pour régner" pour le calcul de a^n ; analyser sa complexité.

Exercice 3 On représente un polynôme $p(x) = \sum_{i=0}^{n-1} a_i x^i$ par la liste de ses coefficients $[a_0, \ldots, a_{n-1}]$. Nous nous intéressons au problème de la multiplication de deux polynômes : étant donnés deux polynômes $p(x)$ et $q(x)$ de degré au plus $n-1$, calculer $p(x)q(x) = \sum_{i=0}^{i=2n-2} c_i x^i$.

1. Donner un algorithme simple qui calcule directement chaque coefficient c_i. Quelle est sa complexité ?
2. On découpe chaque polynôme en deux parties de tailles égales :

$$p(x) = p_1(x) + x^{n/2} p_2(x)$$
$$q(x) = q_1(x) + x^{n/2} q_2(x)$$

   Quelle est la complexité d'un algorithme de calcul de pq utilisant l'identité :

$$pq = p_1 q_1 + x^{n/2} p_1 q_2 + x^{n/2} p_2 q_1 + x^n p_2 q_2$$

3. On utilise maintenant la relation

$$p_1 q_2 + p_2 q1 = (p_1 + p_2)(q_1 + q_2) - p1q1 - p2q2$$

   Calculer la complexité du calcul de pq utilisant cette relation.

Exercice 4 Soit $T[1..n]$ un tableau représentant une liste d'éléments de taille n. Un élément e de T est majoritaire si l'ensemble $\{i \mid T[i] = e\}$ est de cardinalité strictement supérieure à $\frac{n}{2}$.

1. Proposer un algorithme simple qui recherche un élément majoritaire dans un tableau ; analyser sa complexité.
2. Proposer un algorithme "diviser pour régner" qui recherche un élément majoritaire dans un tableau ; analyser sa complexité.

Exercice 5 On dispose d'un tableau d'entiers relatifs de taille n. On cherche à déterminer la suite d'entrées consécutives du tableau dont la somme est maximale. Par exemple, pour le tableau $T = [-1, 9, -3, 12, -5, 4]$, la solution est 18 (somme des éléments de $T[2..4] = [9, -3, 12]$).

1. Proposer un algorithme simple qui recherche une telle séquence dans un tableau ; analyser sa complexité.
2. Proposer un algorithme "diviser pour régner" qui recherche une telle séquence dans un tableau ; analyser sa complexité.

Exercice 6 Une *matrice de Tœplitz* est une matrice $(a_{i,j})$ $n \times n$ telle que $a_{i,j} = a_{i-1,j-1}$ pour $2 \leq i, j \leq n$

1. La somme de deux matrices de Tœplitz est-elle une matrice de Tœplitz ? Et le produit ?
2. Trouver un moyen d'additionner deux matrices de Tœplitz en $\mathcal{O}(n)$.
3. Comment calculer le produit d'une matrice de Tœplitz $n \times n$ par un vecteur de longueur n ? Quelle est la complexité de l'algorithme ?

Exercice 7 Nous voulons insérer n éléments dans une table de manière dynamique. On insère à chaque fois un nouvel élément, directement si la table n'est pas pleine. Sinon, on crée une nouvelle table de taille double, où l'on recopie l'ancienne puis on insère. Le coût correspond alors à la taille de l'ancienne table, augmentée de 1. (La table est toujours à moitié pleine, donc il arrive que le coût soit élevé, mais on a dans ce cas libéré de la place pour les insertions suivantes.) Quel est le coût de n insertions successives ? Que se passe-t-il pour des tableaux dont les tailles suivent une progression arithmétique ?