Algorithmique TD7
L3 Informatique – ENS Cachan

Guillaume Bury

24 octobre 2016

Exercice 1

On considère un tableau $A[1..n]$ qui contient toutes les valeurs de $\{0, 1, \ldots, n\}$ sauf une. On voudrait déterminer laquelle en temps linéaire.

Montrer comment résoudre ce problème si les éléments de A peuvent être lus et copiés en temps constant.

On suppose maintenant qu'un accès élémentaire dans A est une lecture du i-ème bit de $A[j]$. Montrer comment résoudre ce problème en temps linéaire et en utilisant une mémoire en $O(\log(n))$.

Exercice 2

On considère une pyramide de nombres, qu'on va ici représenter comme une matrice triangulaire : la première ligne (d'indice 0), contient un élément à la colonne 0, la deuxième ligne (d'indice 1), contient deux éléments numérotés 0 et 1, ... Un chemin est une suite d'indices $(u_i)_{i\in\mathbb{N}}$ tels que $u_i \leq u_{i+1} \leq u_i + 1$. Donner un algorithme qui calcule un chemin qui maximise la somme des nombres contenus dans les case parcourues.

Exercice 3

Soit Σ un alphabet fini et de cardinal supérieur ou égal à deux. On appelle codage binaire une application injective α de l'alphabet Σ dans $\{0,1\}^*$. En utilisant l'opération concaténation, α s'étend de manière naturelle en $\alpha : \Sigma^* \to \{0,1\}^*$.

Un codage est dit préfixe si aucune lettre n'est codée par un mot de code qui est préfixe du codage d'une autre lettre.

1. Montrer que pour un codage préfixe, α est injective sur Σ^*.
2. Montrer qu'on peut représenter un codage préfixe par un arbre binaire dont les feuilles sont les lettres de l'alphabet.

On dit qu'un codage est de longueur fixe quand toutes les lettres sont codées par un mot de code de même longueur. Mais on obtient des codes plus efficaces en associant des codes plus courts aux lettres qui apparaissent le plus fréquemment, quitte à devoir rallonger les codes des lettres qui apparaissent peu fréquemment.

On associe à chaque lettre a de Σ une fréquence d'apparition $f(a)$. Cette fréquence est généralement estimée à partir d'un ensemble de textes représentatif de la langue considérée.

On définit alors le coût d'un codage préfixe par la somme :

$$\sum_{a\in\Sigma} f(a) \cdot |\alpha(a)|$$

et on cherche un codage qui minimise ce coût.

3. Montrer qu'à un codage préfixe optimal correspond un arbre binaire où tout nœud interne a deux fils.

4. Montrer qu'il existe un codage préfixe optimal pour lequel les deux lettres dont le nombre d'occurrences est le plus faible sont sœurs dans l'arbre.

Étant données x et y les deux lettres dont le nombre d'occurrences est le plus faible dans w, on considère l'alphabet $\Sigma' = (\Sigma \setminus \{x, y\}) \cup z$ où z est une nouvelle lettre à laquelle on associe $f(z) = f(x) + f(y)$.

5. Soit T' l'arbre d'un codage optimal pour Σ', montrer que l'arbre T obtenu à partir de T' en remplaçant la feuille associée à z par un nœud interne ayant x et y comme feuilles représente un codage optimal pour Σ.
6. En déduire un algorithme recherchant un codage optimal et donner sa complexité.

Exercice 4

On s'intéresse ici à l'application d'une permutation miroir, que l'on définit comme suit. Prenons en entrée un tableau A de taille $n = 2^k$, avec $k \geq 0$. On peut voir chaque indice de A, comme un entier binaire de taille k que l'on écrira $< a_{k-1}, \ldots, a_1, a_0 >$, soit $a = \sum_{i=0}^{k-1} a_i 2^i$. On définit maintenant :

$$\begin{aligned} \mathrm{rev}_k(a) &= \mathrm{rev}_k(< a_{k-1}, \ldots, a_1, a_0 >) \\ &= < a_0, a_1, \ldots, a_{k-1} > \\ &= \sum_{i=0}^{k-1} a_{k-i-1} 2^i \end{aligned}$$

On s'intéresse donc à créer le tableau B défini par $B[i] = A[\mathrm{rev}_k(i)]$.

1. Donner un algorithme qui permet d'effectuer une permutation miroir en temps $O(nk)$.
2. On se propose maintenant d'implémenter la permutation miroir en se servant de la fonction incr définit comme : $\mathrm{incr}(a) = \mathrm{rev}_k(\mathrm{rev}_k(a)+1)$. On suppose que l'on a accès à des opérations de base sur les entiers binaires : décalage droite/gauche, ET logique bit à bit, OU logique bit à bit, ... Montrer comment implémenter la fonction incr de telle sorte que le calcul de la permutation miroir soit en temps $O(\log(n))$.