TD3 – Complexité Algorithmique ; Diviser pour régner

Responsable : M. Bombar, L. Poyeton

Exercice 1 – Square And Multiply

On considère l'algorithme suivant.

Algorithme 1 :

Entrées : $x \in M$ et $\ell = \sum_{j=0}^{r-1} \ell_j 2^j \in \mathbb{N}$

Sortie : x^ℓ

```
y ← 1
z ← x
pour i = 0, ··· , r − 1 faire
    si ℓ_i = 1 alors
        y ← y · z
    si i < r − 1 alors
        z ← z^2
Renvoyer y
```

(Q1) On note y_i, z_i les valeurs de y et z après le i-ème tour de boucle. Montrer que

$$z_i = x^{2^{i+1}}; \qquad y_i = x^{s_i}, \quad \text{où } s_i = \sum_{j=0}^{i} l_j 2^j.$$

(Q2) En déduire que l'algorithme est correct.

(Q3) Implémenter l'algorithme en Sage. Attention, la fonction ne doit pas chercher à décomposer l'entrée ℓ en binaire !

(Q4) Combien réalise-t-on de multiplications ?

Exercice 2 – Karatsuba pour les entiers

On veut multiplier deux entiers positifs a et b. On suppose d'abord que leur taille est au plus $n = 2^k$ bits et on pose $m = n/2$ et $B = 2^m$. On écrit

$$a = a_0 + Ba_1, \qquad b = b_0 + Bb_1,$$

avec $0 \leq a_0, a_1, b_0, b_1 < B$.

(Q1) Exprimer le produit ab en fonction des quatre produits $a_i b_j$. Si l'on applique récursivement ce découpage, écrire une relation de récurrence pour le coût et retrouver la complexité quadratique de la multiplication naïve.

(Q2) De façon analogue à la version polynomiale vue en cours, on proposer de poser

$$Z_0 = a_0 b_0, \qquad Z_2 = a_1 b_1, \qquad D = (a_1 - a_0)(b_1 - b_0).$$

Montrer que

$$a_0 b_1 + a_1 b_0 = Z_0 + Z_2 - D,$$

puis en déduire une formule pour ab ne faisant intervenir que les trois produits Z_0, Z_2 et D.

(Q3) Pourquoi les deux facteurs intervenant dans D ont-ils, en valeur absolue, au plus m bits ?

(Q4) Dérouler un niveau de l'algorithme pour

$$a = 123, \qquad b = 91,$$

en choisissant $n = 8$. Vérifier le résultat obtenu.

(Q5) Implémenter cet algorithme en SageMath. La fonction devra fonctionner pour des entiers relatifs de tailles quelconques ; on pourra compléter implicitement la taille au prochain entier pair lors du découpage.

(Q6) Montrer que le coût satisfait une récurrence de la forme

$$T(n) = 3T(n/2) + \mathcal{O}(n),$$

et en déduire une complexité en $\Theta\big(n^{\log_2 3}\big)$ opérations élémentaires sur les bits.

Exercice 3 – Transformée de Walsh–Hadamard

Dans cet exercice, on suppose également que $n = 2^k$ est une puissance de 2. Pour une liste $x = (x_0, \dots, x_{n-1}) \in \mathbb{K}^n$, on définit récursivement $H_n(x)$ de la façon suivante. Pour $n = 1$, on pose $H_1(x) = x$. Si $n > 1$, on coupe $x = (u, v)$ en deux moitiés de longueur $n/2$, et l'on pose

$$\boxed{H_n(u, v) = \big(H_{n/2}(u + v),\, H_{n/2}(u - v)\big),}$$

où $u + v$ et $u - v$ sont calculés terme à terme.

(Q1) Écrire explicitement $H_2(a, b)$ puis $H_4(a, b, c, d)$. Combien d'additions ou soustractions sont effectuées dans chacun de ces deux cas ?

(Q2) En développant chaque niveau de récursion, calculer à la main

$$H_8(1, 0, 1, 1, 0, 0, 1, 0).$$

(Q3) Programmer une fonction récursive `walsh(x)` sur des listes de longueur une puissance de 2. La fonction ne doit utiliser que des additions, des soustractions et des opérations sur les listes.

(Q4) Soit $T(n)$ le nombre d'additions ou soustractions effectuées par l'algorithme. Montrer que

$$T(n) = 2T(n/2) + n, \qquad T(1) = 0,$$

puis en déduire que

$$\boxed{T(n) = n \log_2 n.}$$

(Q5) Montrer que

$$\boxed{H_n(H_n(x)) = nx},$$

et en déduire que

$$H_n^{-1} = \frac{1}{n} H_n.$$