TD d'algorithmique numéro 2
Natacha Portier et Daniel Hirschkoff
semaine du 4 au 8 octobre 1999

Question 1 ⋆ optimalité - oracle ⋆

On examine une autre méthode pour prouver l'optimalité de l'algorithme du tournoi vu la dernière fois, appelée *méthode de l'oracle*. L'oracle construit une donnée de taille n sur laquelle la complexité de l'algorithme est minorée par une valeur $c(n)$, qui est alors une minoration de la complexité au pire de l'algorithme.

On considère donc un algorithme quelconque utilisant des comparaisons et des affectations et qui trouve le maximum et le deuxième plus grand.

Ici l'oracle construit progressivement un tableau en respectant les régles suivantes, à chaque comparaison entre deux éléments x et y effectuée par l'algorithme (les règles sont symétriques en x et y) :

(R1) Si x n'a jamais perdu et y a perdu au moins une fois, on met à jour la valeur de x pour que x gagne

(R2) Si x et y n'ont jamais perdu, c'est celui qui a gagné le plus de matches qui gagne. S'ils ont gagné le même nombre de matches on détermine arbitrairement un vainqueur. Dans les deux cas on réajuste les valeurs en conséquence.

(R3) Si x et y ont perdu au moins une fois, le gagnant est celui qui a la plus grande valeur. S'ils ont la même on réajuste arbitrairement les valeurs.

Question 1.1 Se convaincre que les valeurs peuvent toujours être réajustées selon le schéma ci-dessus.

On cherche donc à minorer le nombre de pas que l'oracle force l'algorithme à faire. On raisonne pour cela sur un graphe orienté avec les conventions suivante :

- Il y a n nœuds, un par élément à comparer.
- On construit une relation d'ordre partiel : on rajoute un arc de x à y si x gagne une comparaison avec y ou si x gagne avec z et il y a un arc de z à y (on clôt transitivement le graphe).

Question 1.2 Montrer que si un joueur est vainqueur à la fin du jeu, alors du sommet correspondant partent $n-1$ arcs vers les autres sommets du graphe.

Question 1.3 Montrer que si x est invaincu et a gagné p matches contre des joueurs invaincus jusque là, de x partent au plus $2^p - 1$ arcs.

Question 1.4 Conclure.

Question 2 ⋆ résolution de récurrences ⋆

Dans l'algorithme de Strassen de multiplication de deux matrices de taille $n = 2^{s+1}$, soit $M(s)$ le nombre de multiplications et $A(s)$ le nombre d'additions à effectuer. Nous avons les relations de récurrence suivantes :

$M_{s+1} = 7M_s$ et $M_0 = 1$

$A_{s+1} = 7A_s + 18.4^s$ et $A_0 = 0$

Résoudre ces relations, ainsi que la suivante :

$B_{s+1} = 3B_s + 4.2^s - 4$, $B_0 = 0$, $B_1 = 4$, $B_2 = 24$

Question 3 ⋆ le *Master theorem* ⋆

On se propose de résoudre la relation de récurrence dans un cas assez général pour des algorithmes *diviser pour régner*. On s'intéresse à une propriété de récurrence de la forme :

$$T(n) = \underbrace{l.T(n/b)}_{\text{terme “diviser”}} + \underbrace{c.n^k,}_{\substack{\text{reconstruction}\\ \text{de la solution}}} \qquad \text{pour } n > n_0\,,$$

avec les hypothèses suivantes :

- $n_0 \geq 1$, $l \geq 1$, $k \geq 0$ et $b \geq 2$ sont des entiers;
- c est un réel strictement positif;
- n/n_0 est une puissance exacte de b.

Indiquer le comportement à l'infini de $T(n)$ en discutant suivant les valeurs de l, b et k.

Question 4.1 ⋆ matrices de Tœplitz ⋆

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

La somme de deux Tœplitz est-elle Tœplitz? Et le produit?

Question 4.2

Comment additionner deux matrices de Tœplitz en $O(n)$? On cherchera pour cela une représentation adéquate d'une matrice de Tœplitz.

Question 4.3

Comment calculer le produit d'une matrice $n \times n$ de Tœplitz par un vecteur de longueur n? Quelle est la complexité de l'algorithme?