TD 2 : Complexité et algorithmes de tri

21 septembre 2017

1 Le tri par tas

Le tri par tas utilise une structure de données appelée *tas binaire*. C'est en fait un tableau qui peut être vu comme un arbre binaire presque parfait (voir Fig. 1), dans lequel chaque élément d'indice i a pour fils gauche (respectivement pour fils droit) l'élément d'indice $2 * i$ (respectivement $2 * i + 1$), s'il existe.

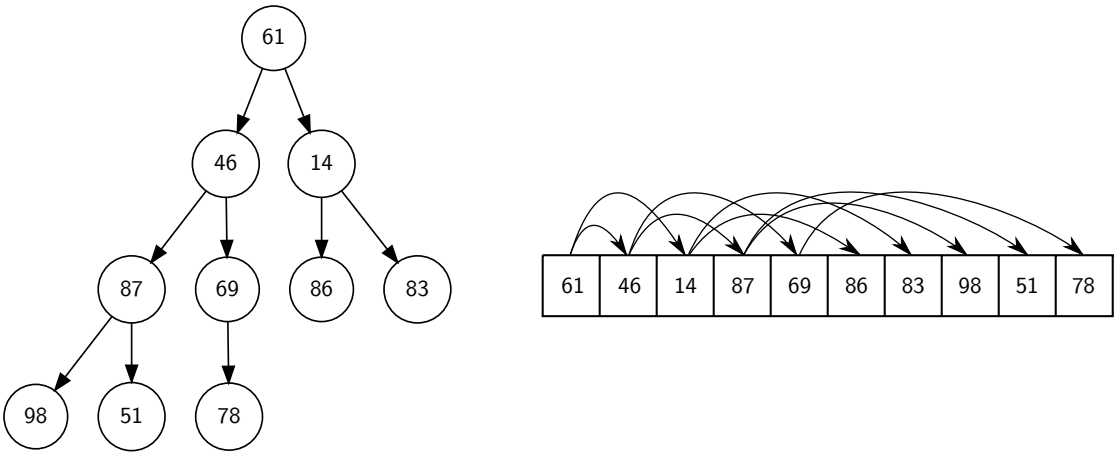

FIGURE 1 – Exemple de tas binaire de 10 éléments et sa représentation sous forme d'arbre.

Le principe du tri par tas est d'utiliser un *tas max*, c'est-à-dire un tas A qui vérifie la propriété suivante : la valeur de chaque élément de l'arbre est plus petite que celle de son parent. Dans la représentation sous forme de tableau, cela se traduit formellement par :

$$\forall i > 1, A[i] \leq A[i/2],$$

où $i/2$ permet d'obtenir par division entière l'indice du parent de l'élément d'indice i.

1.1 Construction d'un tas max

Pour construire un tas max à partir d'un tas binaire quelconque, on va procéder de façon ascendante. On a besoin d'une procédure auxiliaire $\text{ENTASSER-MAX}(A, i)$ qui suppose que les deux sous-arbres de l'élément d'indice i sont des tas max, mais que l'élément d'indice i peut être plus petit que ses deux enfants. Le rôle d'$\text{ENTASSER-MAX}(A, i)$ est de venir placer l'élément d'indice i dans le tas, de façon à ce que le sous-arbre enraciné en i devienne un tas max.

Question 1 – Écrire la procédure ENTASSER-MAX.

La procédure $\text{CONSTRUIRE-TAS-MAX}$ appelle itérativement $\text{ENTASSER-MAX}(A, i)$ pour i allant de $|A|$ à 1.

Question 2 – Donner la trace d'exécution de $\text{CONSTRUIRE-TAS-MAX}$ sur le tableau de la Fig. 1.

1.2 Tri par tas

Voici le code de l'algorithme du tri par tas :

Fonction $\text{TRI-PAR-TAS}(A)$
  $\text{CONSTRUIRE-TAS-MAX}(A)$
  pour $i \leftarrow |A|$ **à** 2 **faire**
    $temp \leftarrow A[1]$
    $A[1] \leftarrow A[i]$
    $A[i] \leftarrow temp$
    $\text{ENTASSER-MAX}(A[1, \ldots, |A| - 1], 1)$

Question 3 – À la suite de la question précédente, donner la trace d'exécution du TRI-PAR-TAS sur le tableau de la Fig. 1.

2 Le tri comptage

On considère un algorithme de tri agissant sur un tableau A d'entiers bornés, appartenant à $\{1, 2, \ldots, k\}$, où $k \in \mathbb{N}$. Le principe est le suivant : on commence par compter le nombre d'occurences de chacun des entiers de $\{1, \ldots, k\}$ dans le tableau A. On parcourt ensuite ce tableau des occurences dans l'ordre croissant pour reconstruire les données de A, triées cette fois-ci.

Question 4 – Écrire la fonction TRI-COMPTAGE, qui prend en entrée un tableau A et implémente la stratégie décrite ci-dessus.

Question 5 – Quelle est la complexité de TRI-COMPTAGE, en fonction de k et de n, la taille du tableau d'entrée ?

Question 6 – Que dire de cette complexité, par rapport à la borne maximale vue en cours pour les algorithmes de tri par comparaison ?

3 Problème du drapeau tricolore

On aligne n boules, réparties en un nombre quelconque de boules de couleur bleue, blanche ou rouge. Ces boules sont diposées dans un ordre quelconque. La ligne de boules est représentée par un tableau B de taille n, dont chaque élément $B[i]$ appartient à l'ensemble {bleu, blanc, rouge}.

QUESTION 7 – Écrire un algorithme qui trie le tableau de telle façon que toutes les boules bleues apparaissent au début, suivies des boules blanches puis des boules rouges. La contrainte est que le tri du tableau doit être réalisé en seul parcours (on ne peut tester qu'une seule fois la couleur de chaque boule) et en place (sans utiliser de deuxième tableau).

4 Algorithme de tri parallèle

4.1 Définitions

On dispose de « composants » appelés *comparateurs* tous identiques qui permettent de comparer deux nombres x et y et les permuter le cas échéant.

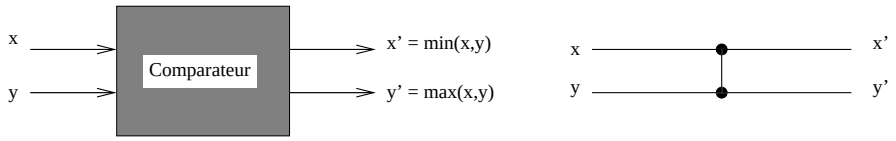

FIGURE 2 – Schématisation d'un comparateur

Un réseau de comparaisons est un ensemble de comparateurs reliés par des câbles (sans cycle). Il a donc autant d'entrées que de sorties, et est représenté par n lignes parallèles.

Un *réseau de tri* est un réseau de comparaisons tel que, pour toute séquence d'entrée (suite de n valeurs), la séquence de sortie est triée par ordre croissant.

La *profondeur* d'un câble est définie ainsi :

— si c'est une entrée du réseau, la profondeur vaut 0.

— si les entrées d'un comparateur sont des câbles de profondeur p_1 et p_2 alors la profondeur des deux câbles de sortie est $1 + \max(p_1, p_2)$.

QUESTION 8 – Quelle grandeur caractéristique du réseau permet de donner une mesure du temps nécessaire au tri ?

4.2 Un exemple

Voici un exemple de réseau de taille 4.

QUESTION 9 – Simuler l'exécution de ce réseau avec l'entrée $(9, 3, 1, 5)$ (on précise que les entrées sont lues de haut en bas).

QUESTION 10 – Montrer que ce réseau de comparaisons est un réseau de tri.

QUESTION 11 – En rajoutant un comparateur dans ce réseau de tri, peut-on perdre la propriété de tri de ce réseau ?

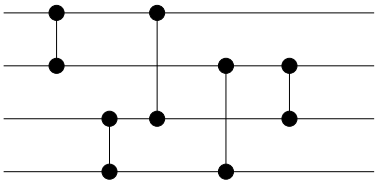

FIGURE 3 – Exemple de réseau de tri

QUESTION 12 – Construire un réseau de tri de taille 4 qui reprend le principe du tri par insertion.

QUESTION 13 – Montrer que tout réseau de tri de taille n a une profondeur p au moins égale à $2 \times \log_2 n$. NB : la profondeur d'un réseau est le maximum des profondeurs des câbles.