Réseaux de tri

Résumé: Ces exercices sont en grande partie tirés du Cormen Leiserson Rivest que vous connaissez tous. On démontre d'abord le principe du 0-1 puis on l'applique sur différents réseaux de tris.
Olivier Beaumont et Fabrice Rastello sont les auteurs originaux de ce document. Mes apports ont été mineurs (coquilles, typographie, petits dessins,...).

1 Principe du 0-1

On donne ici une technique de preuve différente pour la preuve de correction des réseaux de fusion et de tri. On dit qu'une suite est de type 0-1 si elle n'est constituée que de 0 et de 1.

▷ **Question 1.** Montrer qu'un réseau de comparateurs implante correctement le tri si et seulement si il calcule bien cette fonction pour toutes suites de type 0-1 en entrée.

Réponse.

$\boxed{\Rightarrow}$ Ce sens est trivial puisque si un réseau implante correctement le tri, il la calcule bien en particulier pour toute suite de type 0-1.

$\boxed{\Leftarrow}$ Soit f une fonction monotone croissante. Un comparateur se comporte de la même manière sur (x_1, x_2) et sur $(f(x_1), f(x_2))$. On considère maintenant un réseau R fixé, appliqué à une suite donnée $(x_1, \ldots, x_n)$. La position finale de l'entrée x_i, i.e., le fil de sortie du réseau où cet x_i aboutit, ne dépend pas de la valeur de x_i mais de sa position relative par rapport aux autres x_j. Donc, si le réseau est appliqué à $(f(x_1), \ldots, f(x_n))$, $f(x_i)$ sort par le même fil de sortie que x_i plus haut.

Supposons maintenant que R ne calcule pas correctement la fonction de tri. Il existe donc une suite $(x_1, \ldots, x_n)$, et il existe i_1 et i_2 tels que x_{i_1} et x_{i_2} sortent de R sur deux sorties consécutives, mais $x_{i_1} > x_{i_2}$, Il suffit de définir une fonction monotone $f : \{x_1, \ldots, x_n\} \to \{0, 1\}$ telle que $f(x_{i_1}) = 1$ et $f(x_{i_2}) = 0$, par exemple :

$$f(x) = \begin{cases} 0 & \text{si } x \leq x_{i_2} \\ 1 & \text{si } x > x_{i_2} \end{cases}$$

Alors R ne calcule pas correctement le tri de $(f(x_1), \ldots, f(x_n))$, puisque on trouve 1 et 0 dans cet ordre sur deux sorties consécutives. Or $(f(x_1), \ldots, f(x_n))$ est une suite de type 0-1, d'où la conclusion.

□

▷ **Question 2.** Montrer qu'un réseau de comparateurs α trie correctement la séquence $< n, n-1, \ldots, 1 >$ ssi il trie correctement les séquences $< 1^i 0^{n-i} >$ pour tout $i \geqslant 1$.

Réponse.

$\boxed{\Rightarrow}$ Supposons tout d'abord que le réseau trie correctement la séquence $< n, n-1, \ldots, 1 >$. Alors (voir la démonstration du lemme 0-1) il trie également correctement les séquences $< f_i(n), f_i(n-1), \ldots, f_i(1) >$ pour tout i, où

$$f_i(x) = 0 \text{ si } x < i \text{ et } 1 \text{ sinon.}$$

$\boxed{\Leftarrow}$ Supposons maintenant que le réseau ne trie pas correctement la séquence $< n, n-1, \ldots, 1 >$ et montrons que nécessairement, il ne trie pas correctement une des séquences $< 1^i 0^{n-i} >$.

Soit j le plus grand élément de $[1, n]$ qui n'est pas à sa place après l'application du réseau de comparateurs α sur la séquence $< n, n-1, \ldots, 1 >$. Alors, il existe une permutation σ de $[1, j]$ telle que

$$\alpha(< n, n-1, \ldots, 1 >) = < \sigma(1), \ldots, \underbrace{\sigma(k)}_{=j}, \ldots, \underbrace{\sigma(j)}_{<j}, j+1, \ldots, n > .$$

Considérons la fonction f_j précédemment définie. f_j étant une fonction monotone croissante, on a

$$\begin{aligned}\alpha(< 1^j, 0^{n-j} >) &= \alpha(f_j(< n, n-1, \ldots, 1 >)) \\ &= f_j(\alpha(< n, n-1, \ldots, 1 >)) \\ &=< 0, \ldots, 0, 1, 0, \ldots, 0, 1, \ldots, 1 >,\end{aligned}$$

ce qui est absurde.

□

2 Réseau de tri bitonique

Définition 1. On appelle **séquence bitonique** une séquence qui est soit croissante puis décroissante, soit décroissante puis croissante. Ainsi, les séquences $< 2, 3, 7, 7, 4, 1 >$ et $< 12, 5, 10, 11, 19 >$ sont bitoniques. Les séquences binaires bitoniques sont de la forme $0^i1^j0^k$ ou de la forme $1^i0^j1^k$.

Définition 2. Un **réseau de tri bitonique** est un réseau de comparaison triant toute séquence bitonique.

Définition 3. Un réseau de comparaison triant toute séquence binaire bitonique est un réseau de tri bitonique.

Définition 4. On appelle **séparateur** un réseau de comparaison de profondeur 1 dans lequel chaque ligne i est comparée à la ligne $i + \frac{n}{2}$ pour $i \in \{1, 2, ..., \frac{n}{2}\}$ (on considère que n est pair).

▷ **Question 3.** Imaginez comment construire un réseau de tri bitonique à partir de séparateurs. Quelle est sa profondeur et le nombre de comparateurs utilisés ?

Réponse. L'effet d'un séparateur sur une séquence bitonique est décrit dans la Figure 1. À la sortie du séparateur, on obtient deux séquences bitoniques dont une au moins est pure (c'est à dire composée soit uniquement de 0, soit uniquement de 1).

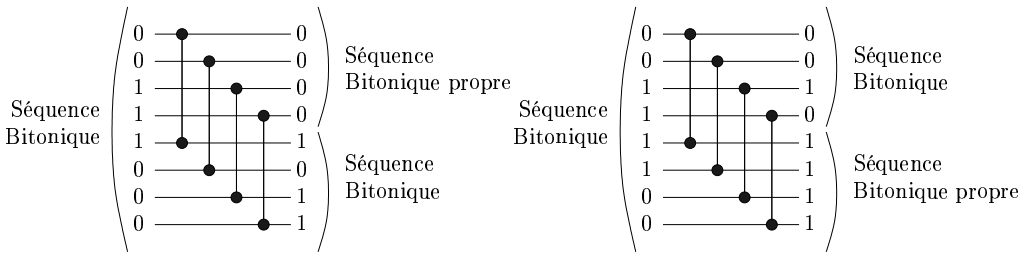

FIG. 1 – Un séparateur de taille 8 appliqué à deux séquences binaires bitoniques différentes.

Supposons qu'il y ait plus de 1 que de 0 dans la séquence bitonique initiale.

- Supposons que la séquence initiale est de la forme $1^i0^j1^k$.
  - Si $k > \frac{n}{2}$ alors la séquence de sortie est $1^i0^j1^k$.
  - Si $i > \frac{n}{2}$ alors la séquence de sortie est $1^{\frac{n}{2}-i}0^j1^k1^{\frac{n}{2}}$.
  - Si $i < \frac{n}{2}$ et $k < \frac{n}{2}$ (mais $i + k \geqslant \frac{n}{2}$ puisqu'il y à plus de 1), alors la séquence de sortie est de la forme $0^{\frac{n}{2}-k}1^{\frac{n}{2}-j}0^{\frac{n}{2}-i}$.
- Supposons que la séquence initiale est de la forme $0^i1^j0^k$.
  Comme $j \geqslant \frac{n}{2}$, la séquence de sortie est de la forme $O^i1^{j-\frac{n}{2}}0^k1^{\frac{n}{2}}$.

Dans tous les cas, on vérifie donc que la séquence de sortie a bien la forme souhaitée (la démonstration est tout à fait identique s'il y a plus de 0 que de 1).

En utilisant la remarque précédente, il est facile de construire un réseau de tri bitonique en utilisant des séparateurs, comme indiqué Figure 2.

Soient t_m et p_m respectivement la profondeur et le nombre de comparateurs du réseau de tri bitonique pour une séquence bitonique d'entrée de taille $n = 2^m$. On vérifie que

$$\begin{array}{llll} t_1 = 1, & t_m = t_{m-1} + 1 & \text{ce qui implique que} & t_m = m, \\ p_1 = 1, & p_m = t_m 2^{m-1} & \text{ce qui implique que} & p_m = m2^{m-1}. \end{array}$$

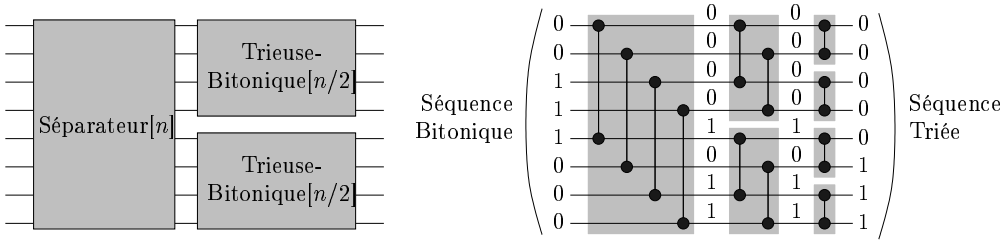

FIG. 2 – Construction d'un réseau d'une trieuse bitonique de séquence de taille n à partir de séparateurs.

Le réseau de tri bitonique ainsi construit a donc pour profondeur $O(\log n)$ et comporte $O(n \log n)$ comparateurs. □

▷ **Question 4.** En utilisant des trieuses bitoniques, construire un réseau fusionnant deux listes triées et déduisez-en la construction d'un réseau général de tri dont on déterminera la profondeur et le nombre de comparateurs.

Réponse. Il est facile de construire un réseau de fusion à partir de trieuses bitoniques. En effet, considérons la fusion de deux listes triées de taille n, $0^i 1^{n-i}$ et $0^k 1^{n-k}$ respectivement. Si on inverse la deuxième liste triée (ce qui revient à échanger des fils mais ne nécessite aucun comparateur supplémentaire), on obtient la séquence bitonique $0^i 1^{2n-i-k} 0^k$ sur laquelle on peut appliquer la trieuse bitonique pour obtenir la fusion des deux listes initiales. Pour inverser la seconde liste, il suffit simplement de modifier le premier étage de la trieuse bitonique comme montré en Figure 3.

Ensuite, on obtient le réseau général de tri en empilant les réseaux de fusion, comme vu en cours.

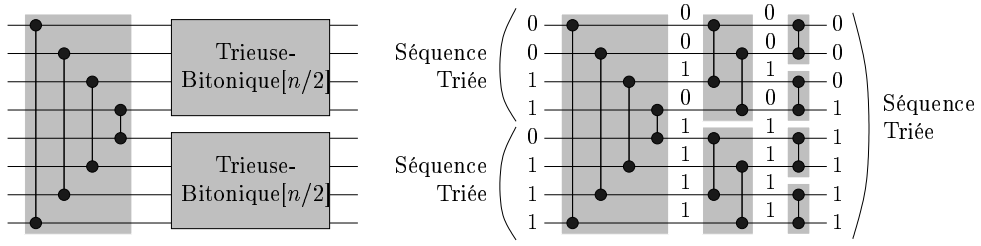

FIG. 3 – Construction d'un réseau de fusion à l'aide de trieuses bitoniques.

Soient t'_m et p'_m respectivement la profondeur et le nombre de comparateurs du réseau de tri ainsi construit pour une séquence d'entrée de taille $n = 2^m$. On vérifie que

$$\begin{array}{llll} t'_1 = 1, & t'_m = t'_{m-1} + t_m & \text{ce qui implique que} & t'_m = O(m^2), \\ p'_1 = 1, & p'_m = 2p'_{m-1} + p_m & \text{ce qui implique que} & p'_m = O(m^2 2^m). \end{array}$$

Le réseau de tri ainsi construit a donc pour profondeur $O((\log n)^2)$ et comporte $O(n^2 (\log n)^2)$ comparateurs. □

3 Tri sur une grille 2D

Cette partie utilise le tri par transposition pair-impair sur un réseau linéaire vu en cours. On s'intéresse ici au tri par transposition pair impair sur une grille.

Définition 5. Un tableau carré $A = ((a_{i,j}))$ de taille $n \times n$, $n = 2^m$ est ordonné en serpent si les éléments

du tableau sont ordonnés comme suit :

$$\begin{aligned} a_{2i,j} &\leqslant a_{2i,j+1}, \quad \text{si} \quad 1 \leqslant j \leqslant n-1 \\ a_{2i+1,j+1} &\leqslant a_{2i+1,j}, \quad \text{si} \quad 1 \leqslant j \leqslant n-1 \\ a_{2i-1,n} &\leqslant a_{2i,n}, \quad \text{si} \quad 1 \leqslant i \leqslant 2^{m-1} \\ a_{2i,1} &\leqslant a_{2i+1,1}, \quad \text{si} \quad 1 \leqslant i \leqslant 2^{m-1}-1. \end{aligned}$$

On peut noter que ce serpent induit un réseau linéaire à l'intérieur de la grille (voir Figure 4).

$$\begin{array}{l} a_{1,1} \leftarrow a_{1,2} \leftarrow a_{1,3} \leftarrow a_{1,4} \\ \quad \downarrow \\ a_{2,1} \rightarrow a_{2,2} \rightarrow a_{2,3} \rightarrow a_{2,4} \\ \qquad\qquad\qquad\quad \downarrow \\ a_{3,1} \leftarrow a_{3,2} \leftarrow a_{3,3} \leftarrow a_{3,4} \\ \quad \downarrow \\ a_{4,1} \rightarrow a_{4,2} \rightarrow a_{4,3} \rightarrow a_{4,4} \end{array}$$

FIG. 4 – L'ordre serpent sur une grille 4×4

Définition 6. Un «shuffle» transforme la séquence de $n = 2p$ éléments $< z_1, \ldots, z_n >$ en la séquence $< z_1, z_{p+1}, \ldots, z_p, z_{2p} >$.

On se propose d'étudier l'algorithme suivant, qui réalise la fusion de 4 tableaux de taille $2^{m-1} \times 2^{m-1}$ ordonnés en serpent en un tableau de taille $2^m \times 2^m$ ordonné en serpent.

1. «shuffle» de chaque ligne du tableau (en utilisant des transpositions pair-impair sur les indices des éléments), ce qui revient à appliquer la transformation «shuffle» sur les colonnes.
2. Trier les paires de colonnes, c'est à dire les tableaux de taille $n \times 2$ en ordre serpent, en utilisant $2n$ étapes de transposition pair-impair sur le réseau linéaire induit sur chaque serpent de longueur $2n$.
3. Appliquer $2n$ étapes de transposition pair-impair sur le réseau linéaire induit par le serpent de taille n^2.

▷ **Question 5.** Faire tourner/l'algorithme de tri induit avec $n = 4$ et $a_{i,j} = 21 - 4i - j$ pour $1 \leqslant i, j \leqslant 4$.

Réponse.

$$\begin{pmatrix} 16 & 15 & 14 & 13 \\ 12 & 11 & 10 & 9 \\ 8 & 7 & 6 & 5 \\ 4 & 3 & 2 & 1 \end{pmatrix} \xrightarrow[\substack{\text{tri fusion} \\ \text{grilles } 2\times 2}]{} \begin{pmatrix} 11 & 12 & 9 & 10 \\ 16 & 15 & 14 & 13 \\ 3 & 4 & 1 & 2 \\ 8 & 7 & 6 & 5 \end{pmatrix} \xrightarrow[\substack{\text{shuffle colonnes} \\ \text{grille } 4\times 4}]{} \begin{pmatrix} 11 & 9 & 10 & 12 \\ 16 & 14 & 15 & 13 \\ 3 & 1 & 4 & 2 \\ 8 & 6 & 7 & 5 \end{pmatrix}$$

$$\xrightarrow[\substack{\text{tri colonnes} \\ \text{grille } 4\times 4}]{} \begin{pmatrix} 1 & 3 & 2 & 4 \\ 8 & 6 & 7 & 5 \\ 9 & 11 & 10 & 12 \\ 16 & 14 & 15 & 13 \end{pmatrix} \xrightarrow[\substack{2n \text{ pair-impair} \\ \text{grille } 4\times 4}]{} \begin{pmatrix} 1 & 3 & 2 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}.$$

□

▷ **Question 6.** Montrer que la première étape de l'algorithme peut s'effectuer en temps $2^{m-1}-1$, l'unité étant un échange entre voisins (on pourra effectuer sur les transpositions pair-impair sur un ensemble d'indices astucieusement choisis). En déduire que l'algorithme global de fusion s'effectue en temps $\leqslant 4.5n$.

Réponse. On définit l'indice c_i de la colonne i par

$$c_i = 2i + 1 \text{ si } 1 \leqslant i \leqslant 2^{m-1} \text{ et } 2(i - 2^{m-1}) \text{ sinon.}$$

Ensuite, il suffit de trier cet ensemble d'indices en utilisant des échanges entre voisins. Considérons le réseau primitif α à 2^{m-1} étages dont le $i^{\text{ème}}$ étage effectue les comparaisons

$$< 2^{m-1} - i + 1, 2^{m-1} - i + 2 >, < 2^{m-1} - i + 3, 2^{m-1} - i + 4 >, \ldots, < 2^{m-1} + i - 1, 2^{m-1} + i > .$$

Une récurrence immédiate montre que le réseau de tri ainsi défini trie correctement la séquence des c_i, et peut donc être utilisé pour réaliser l'opération de «shuffle» sur les colonnes.
Comme le coût de cet étape est $\frac{n}{2} - 1$ et que le coût des autres étapes est $2n$, on vérifie donc que le coût global de l'algorithme de fusion est $\leqslant 4.5n$. □

▷ **Question 7.** En supposant l'algorithme de fusion correct, construire un algorithme qui trie une séquence de longueur 2^{2m} sur une grille $2^m \times 2^m$. Estimer sa complexité.

Réponse. Soit t_m le temps nécessaire pour trier sur une grille $2^m \times 2^m$. On vérifie que

$$t_m \leqslant t_{m-1} + 4.5 \;\; 2^m \leqslant 4.5 \;\; (2^{m+1} - 1) \leqslant 9 \cdot 2^m$$

puisqu'un tableau 1×1 est toujours ordonné en serpent. Le temps nécessaire pour trier une séquence de longueur $N = 2^{2m}$ sur une grille est donc de l'ordre de $\sqrt{N}$. □

▷ **Question 8.** Montrer que le tri par transposition pair-impair sur une grille est correct (il s'agit de montrer que $2n$ étapes de transposition pair-impair dans la troisième phase de l'algorithme de fusion suffisent à obtenir un serpent correctement ordonné).

Réponse. Pour simplifier la démonstration, nous utilisons évidement le principe du 0-1, qui nous permettra d'affirmer que le tri sur la grille est correct ssi on peut réaliser la troisième phase de l'algorithme avec seulement $2n$ transpositions pair-impair sur des séquences de 0-1.

Les différentes possibilités pour des grilles ordonnées en serpent sont illustrées dans la figure 5. En fait,

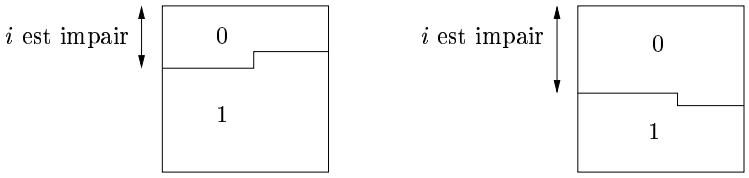

FIG. 5 – Les deux grilles en serpent possibles pour des séquences 0-1.

il suffit d'observer qu'à la fin de la deuxième étape de l'algorithme (les tris sur les réseaux linéaires de tailles $2n$), au plus deux lignes de la matrice ne sont pas exclusivement constituées de 0 et de 1, comme indiqué Figure 6. En effet, regardons le nombre de 0 et de 1 dans deux couples de colonnes après l'opération de

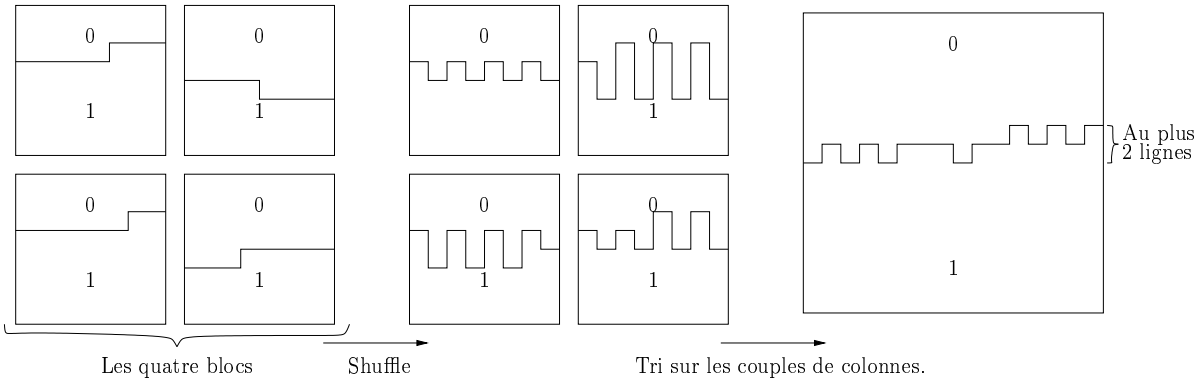

FIG. 6 – Fusion de deux serpents 0-1.

«shuffle». Dans chacune des quatre colonnes concernées, le nombre de 0 et de 1 varie au plus d'une unité.

La différence du nombre global de 0 et de 1 dans deux couples de colonnes après l'opération de «shuffle» est donc d'au plus 4.
Si le nombre de 0 diffère de moins de 3 unités avec le nombre de 1 entre deux couples de colonnes, la propriété est évidemment vérifiée.
Supposons maintenant que le nombre de 0 varie de 4 unités. Les couples de colonnes sont donc respectivement identiques aux deux premières et aux deux dernières colonnes.
Dans tous les cas, on vérifie donc qu'au plus deux lignes (de longueur n) ne sont pas exclusivement constituées de 0 ou de 1, et donc que le réseau linéaire induit par le serpent de taille n^2 trie la séquence globale en seulement $2n$ étapes, ce qui achève la démonstration. □