Sorting networks

Summary: The first exercise should be easy. The second one is a classic (see [2] or [1]). The third exercise will a cover a more sophisticated kind of sorting networks; the more eager will find numerous other examples in [3].

1 All sequences are 0-1

▷ **Question 1** *Let $w = \langle w_1, ..., w_n \rangle$ be a sequence. A comparator network sorts w correctly if and only if it sorts $\langle f(w_1), ..., f(w_n) \rangle$ correctly for every non-decreasing $f : \mathbb{N} \to \{0, 1\}$.*

2 Bitonic sorting networks

Definition 1. We call **bitonic** a sequence which is either increasing and then decreasing or decreasing and then increasing. Thus, sequences $\langle 2, 3, 7, 7, 4, 1 \rangle$ and $\langle 12, 5, 10, 11, 19 \rangle$ are bitonic. Binary bitonic sequence can all be written as $0^i 1^j 0^k$ or $1^i 0^j 1^k$ with $i, j, k \in \mathbb{N}$.

Definition 2. A **bitonic sorting network** is a comparator network sorting every bitonic binary sequence.

▷ **Question 2** *Does a bitonic sorting network sort* every *bitonic sequence?*

Definition 3. We call **separator** a network with n input, with n even, consisting of a column of $\frac{n}{2}$ comparators operating on inputs i and $i + \frac{n}{2}$ for $i \in [\![1, \frac{n}{2}]\!]$.

▷ **Question 3** *Build a bitonic sorting network using separators. How many comparators does it use ? How deep is it ?*

▷ **Question 4** *Using bitonic sorting networks, design a network merging two sorted lists. Use it as a stepping stone to build a general sorting network and estimate its complexity (depth, number of comparators).*

3 Sort a 2D grid

This exercise extends the odd-even mergesort over sequences seen during the lecture to 2D grids.

Definition 4. A square matrix $A = ((a_{i,j}))$ of size $n \times n$, $n = 2^m$ is in snakelike order if elements are placed as follows:

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

Notice that this snake induces a linear network within the grid (see figure 1).

Definition 5. A shuffle turns the $n = 2p$-long sequence of elements $\langle z_1, \ldots, z_n \rangle$ into the sequence $\langle z_1, z_{p+1}, z_2, z_{p+2}, \ldots, z_p, z_{2p} \rangle$. For instance, the "shuffle" of $(1, 2, 3, 4, 5, 6, 7, 8)$ is $(1, 5, 2, 6, 3, 7, 4, 8)$.

We propose to study the following algorithm, which merges four $2^{m-1} \times 2^{m-1}$ snalike-ordered matrices into a single $2^m \times 2^m$ snakelike-ordered matrix:

1. shuffle each row (using odd-even transpositions on the index of the elements), which is equivalent to shuffling columns

$$
\begin{array}{ccccccc}
a_{1,1} & \rightarrow & a_{1,2} & \rightarrow & a_{1,3} & \rightarrow & a_{1,4} \\
 & & & & & & \downarrow \\
a_{2,1} & \leftarrow & a_{2,2} & \leftarrow & a_{2,3} & \leftarrow & a_{2,4} \\
\downarrow & & & & & & \\
a_{3,1} & \rightarrow & a_{3,2} & \rightarrow & a_{3,3} & \rightarrow & a_{3,4} \\
 & & & & & & \downarrow \\
a_{4,1} & \leftarrow & a_{4,2} & \leftarrow & a_{4,3} & \leftarrow & a_{4,4}
\end{array}
$$

Figure 1: The snakelike order over a 4×4 grid.

2. sort every pair of columns (which are $n \times 2$ matrices) respecting the snakelike order, using $2n$ odd-even transpositions on the linear network induced over the relevant $2n$-long snakes
3. apply $2n$ odd-even transposition steps over the linear network induced by the snake of size n^2

▷ **Question 5** *Execute this merging algorithm with the following matrix (note that each 2×2 matrix is already snakelike sorted).*

$$
\begin{bmatrix}
1 & 3 & 5 & 6 \\
11 & 8 & 16 & 10 \\
4 & 7 & 2 & 9 \\
14 & 13 & 15 & 12
\end{bmatrix}.
$$

▷ **Question 6** *Show that the first step of the algorithm can be executed in time $2^{m-1} - 1$, a time unit spanning a swap between neighbours (you can parallelize!). Deduce that the merging algorithm is executed in time $\leqslant \frac{9}{2}n$.*

▷ **Question 7** *Admitting for now that the merging algorithm is correct, write an algorithm sorting sequences of length 2^{2m} over a $2^m \times 2^m$ grid. Estimate its complexity.*

▷ **Question 8** *Show that the odd-even transposition sorting step over a grid is correct (ie, $2n$ transposition steps in the third phase of the merging algorithm yield a correctly ordered snake).*

References

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, 2 edition, 1990. Traduction française publiée chez Dunod, Introduction à l'algorithmique, 2002.

[2] A. Gibbons and W. Rytter. *Efficient Parallel Algorithms.* Cambridge University Press, 1988.

[3] F.T. Leighton. *Introduction to parallel algorithms and architectures: arrays, trees, hypercubes.* Morgan Kaufmann, 1992.