# Sorting networks

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

## 1 All sequences are 0-1

▷ **Question 1** *Show that a comparator network sorts every sequence of integers if and only if it sorts correctly every $\{0, 1\}$-valued sequence.* *Let us prove directly the more general statement (relevant for question 2).*

*Let $w$ be a sequence. A comparator network sorts $w$ if and only if it sorts $\tilde{f}(w)$ for every increasing $f : \mathbb{N} \to \{0, 1\}$ (where $\tilde{f}$ designates the induced map $\mathbb{N}^k \to \{0, 1\}^k$).*

*Let $s : \mathbb{N}^k \to \mathbb{N}^k$ be the function denoting the action of the sorting network. A straightforward induction shows that $s \circ \tilde{f} = \tilde{f} \circ s$ for every monotone $f$.*

*The direct implication is then obvious.*

*For the converse, suppose that all $\tilde{f}(w)$ are sorted by $s$. Assume $\pi_j(s(w)) < \pi_k(s(w))$ and take $f$ to be the characteristic function of $[\![0, \pi_j(s(w))]\!]$. The following concludes the proof.*

$$
\begin{array}{llcll}
 & \pi_j(s(w)) & < & \pi_k(s(w)) & \\
\Rightarrow & f(\pi_j(s(w))) & < & f(\pi_k(s(w))) & \textit{by definition of } f \\
\Rightarrow & \pi_j(\tilde{f}(s(w))) & < & \pi_k(\tilde{f}(s(w))) & \\
\Rightarrow & \pi_j(s(\tilde{f}(w))) & < & \pi_k(s(\tilde{f}(w))) & \textit{since } s \textit{ is a comparator network} \\
\Rightarrow & j & < & k & \textit{since } s \textit{ sorts } \tilde{f}(w)
\end{array}
$$

## 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 $0^i1^j0^k$ or $1^i0^j1^k$ 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 sequences?* *$w$ bitonic $\Rightarrow \tilde{f}(w)$ bitonic + previous question*

**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 ?* *Separator are sketched on figure* **??**. *The separator outputs two bitonic sequences of size $n/2$, one of which is constant.*

*This result follows from a case analysis. Suppose that there are more 1s than 0s in the input sequence.*

- *Suppose that the sequence is of shape $1^i0^j1^k$.*
  - *If $k \geqslant \frac{n}{2}$, then the output is $1^i0^j1^k$.*
  - *If $i \geqslant \frac{n}{2}$, then the output is $1^{i-\frac{n}{2}}0^j1^k1^{\frac{n}{2}}$.*

Figure 1: A seprator of size 8 applied to two bitonic sequences

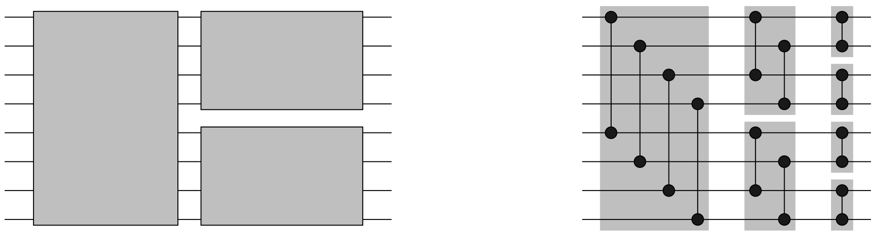


Figure 2: Sorting network from separators

- *If $i < \frac{n}{2}$ and $k < \frac{n}{2}$ (but $i + k \geqslant \frac{n}{2}$ since we have more* 1*s than* 0*s), then the output is* $0^{\frac{n}{2}-k}1^{\frac{n}{2}-j}0^{\frac{n}{2}-i}1^{\frac{n}{2}}$.

- *Otherwise, the sequence is of the shape $0^i1^j0^k$.*
  *Since $j \geqslant \frac{n}{2}$, the output is of shape $O^i1^{j-\frac{n}{2}}0^k1^{\frac{n}{2}}$.*

*The case where there are more* 0*s than* 1*s is completely symmetrical.*
*The layout to build a sorting network is indicated figure* **??**.
*One can easily check that the depth is $O(\log n)$ and the size is $O(n \log n)$.*

▷ **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). The fusion network is built recurively. Notice that if we have two sequences sorted by the subnetworks $0^i1^{n-i}$ and $0^k1^{n-k}$, it suffices to reverse the second one and to concatenate them to obtain the bitonic sequence $0^i1^{2n-i-k}0^k$, which we can sort using our network (see figure* **??**).

*The depth of the network is $\sum_{i=0}^{\log(n)} \log(i) \leq \log(n)^2$, the number of comparators $\sum_{i=0}^{\log(n)} i\log(i) \leq n\log(n)^2$.*

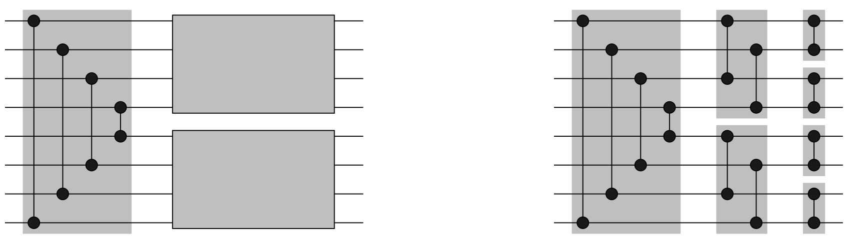


Figure 3: Fusion network from a bitonic sorting network.

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

Figure 4: The snakelike order over a $4 \times 4$ grid.

# 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:

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

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

**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 $(1, 5, 2, 6, 3, 7, 4, 8)$.

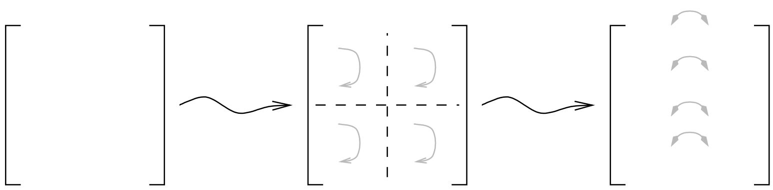
We propose to study the following algorithm, which merges four $2^{m-1} \times 2^{m-1}$ snalike-ordered matrices into a single $2^m \times 2^m$ snakelike-ordered matrix:

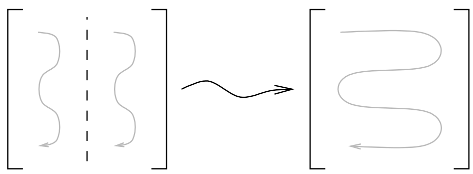
1. shuffle each row (using odd-even transpositions on the index of the elements), which is equivalent to shuffling columns

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 the induced sorting algorithm with $n = 4$ and $a_{i,j} = 21 - 4i - j$ for $1 \leqslant i, j \leqslant 4$. Le dessin ci-dessous représente l'évolution de la grille au cours des différentes étapes du tri serpent.*





▷ **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 $\leqslant \frac{9}{2}n$.*

*For $i$ ranging from 1 to $n = 2p$, we we define the index $c_i$ of the column $i$ as the index of its image via the "shuffle":*

$$c_i = 2i - 1 \text{ if } 1 \leqslant i \leqslant p \text{ and } 2(i - 2^{m-1}) \text{ otherwise}$$

*Then, we can check than we can build a network for the shuffle if we know a comparator network sorting this particular sequence (hardcode the potential crossings of wires instead of the comparators).*

*Let us consider the primitive network $\alpha$ of depth $p - 1$ whose $i$th step handles $i$ comparisons*

$$\langle p-i+1, p-i+2\rangle, \langle p-i+3, p-i+4\rangle, \ldots, \langle p+i-1, p+i\rangle \ .$$

*For instance, for $n = 8$, $p = 4$, we sort the sequence $(1, 5, 2, 6, 3, 7, 4, 8)$. The first three stage include comparators $\langle 4, 5\rangle$ (first stage), $\langle 3, 4\rangle, \langle 5, 6\rangle$ (second stage), and $\langle 2, 3\rangle, \langle 4, 5\rangle, \langle 6, 7\rangle$ (third stage).*

*A straightforward induction show that the sorting network that we define sorts correctly the sequence $c_i$, so we are done.*

*The network needed for this step is of detph $\frac{n}{2} - 1$ (counting crossing of wires as an elementary step) and the cost of the other two steps is $2n$, hence the fusion steps costs $\leqslant \frac{9}{2}n$.*

▷ **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. $\sum_{i=0}^{\log(n)} \frac{9}{2} 2^i \sim 9 \times 2^{\log(n)} \sim O(n)$; one can easily show that this is optimal (up to the constant in the $O$), since an element in a corner might forced to travel a path of length $2n - 2$.*

▷ **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). Of course, we show this result for $0-1$ grids using the first question.*

*Assume that the four submatrices $M_1, M_2, M_3, M_4$ are recursively sorted in a snakelike fashion. Since we restrict our attention to $0-1$ sequences, the shape of the $M_i$ is entirely determined by the number of 0 they contain. Furthermore, there are $x_i$ such that there are exactly $x_i$ or $x_i + 1$ 0s in a given column of $M_i$.*

*After the shuffle, every supercolumn of width 2 will contain $x_1 + x_2 + x_3 + x_4 + j$ 0s for some $j \leq 4$. Thus, after the snakelike sort of the column, there will be only 0s on the $\lfloor \frac{x_1+x_2+x_3+x_4}{2} \rfloor$th* *row, a mixture of* $0$ *and* $1$ *on the next two rows (accounting for this variation of* $2$ *elements on each column) and then only* $1$*. It is then known that a* $2n$*-deep sorting network, which happen to span the entire matrix, can handle the faulty rows.*

# 4 Answers