A Toy Model: Sorting Networks.

Here we use *comparators*: a very small processor able to, given a and b, sort the two numbers. Can we organize and connect comparators in order to sort a sequence of numbers?

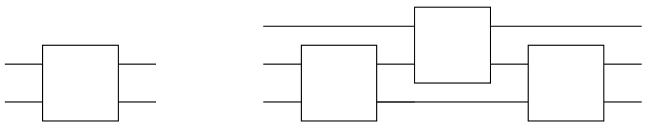

Figure 1 | *Sorting networks for two, three and four numbers*

Here we want to optimize in two ways: minimize the number of steps, and minimize the number of comparators.

1 Odd-Even merge sort.

The Odd-Even merge sort is a parallel algorithm very similar to merge sort: we start by sorting two halves and then merging the two. In this section, we will consider an input of size $n = 2^m$.

1.1 Odd-Even merging network.

Given a sequence $c = \langle c_1, c_2, \ldots, c_n \rangle$, we want to compute a sequence written $\mathsf{SORT}(\langle c_1, c_2, \ldots, c_n \rangle)$ which corresponds to the sorted sequence of the c_i's. If a sequence c is sorted, we write $\mathsf{SORTED}(\langle c_1, \ldots, c_n \rangle)$.

An important part of the merge sort is the following:

$$\text{if } \mathsf{SORTED}(\langle a_1, \dots, a_n \rangle) \text{ and } \mathsf{SORTED}(\langle b_1, \dots, b_n \rangle) \text{ then}$$

$$\begin{aligned} &\mathsf{MERGE}(\langle a_1, \dots, a_n \rangle, \langle b_1, \dots, b_n \rangle) \\ &\qquad = \mathsf{SORT}(\langle a_1, \dots, a_n, b_1, \dots, b_n \rangle). \end{aligned}$$

Let MERGE_m be a network merging two sequences of size $n = 2^m$.