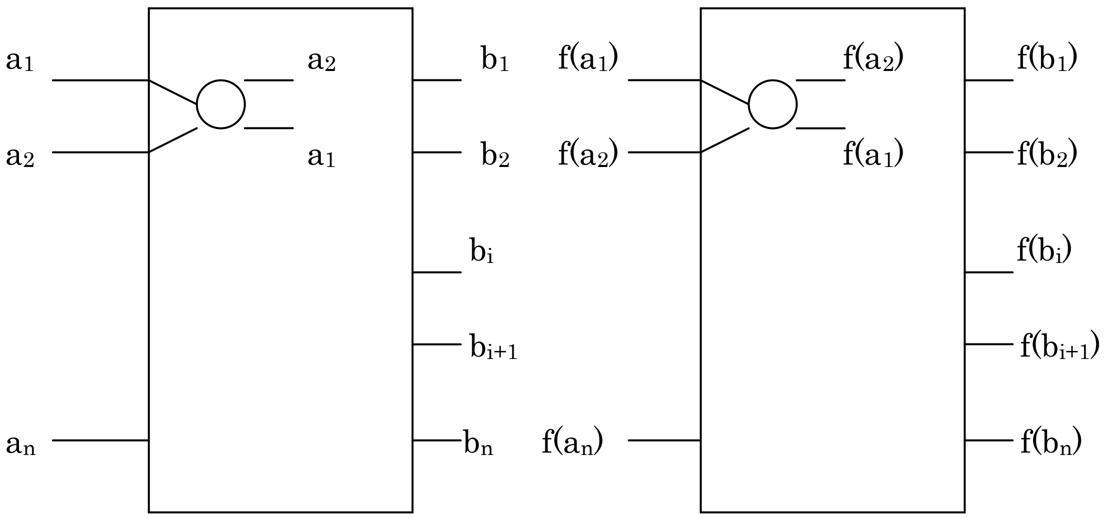
0-1 Principle and Correctness of Sorting Networks

Theorem (0-1 principle). If a sorting network sort any sequence of 0s and 1s, it sorts any sequence of real numbers.

Proof. Suppose there is a sorting network N with inputs a_1, …, a_n and out puts b_1, …, b_n. Suppose there is a monotone function f, that' is, for any x, y, $x \le y \Rightarrow f(x) \le f(y)$.. Then if the inputs to N are $f(a_1)$, …, $f(a_n)$, the outputs must be $f(b_1)$, …, $f(b_n)$. See the following figure. The comparator in N is hypothetical.



Suppose network ≥≥≥N does not sort the input sequence a_1, ,,,, a_n. Then there must be i such that $b_i > b_{i+1}$.

Now we define f to satisfy the following condition.

For all b_j such that $b_j \ge b_i$, $f(b_j)=1$.
For all b_k such that $b_k \le b_{i+1}$, $f(b_k)=0$.

That is, N does not sort the 0-1 sequence $f(a_1)$, …, $f(a_n)$. In a logical form where $\mathbf{a}=(a_1, …, a_n)$ and $f(\mathbf{a}) =(f(a_1), …, f(a_n))$, we have

For some input sequence of real values a, N does not sort a ➔
For some 0-1 input sequence $f(\mathbf{a})$, N does not sort $f(\mathbf{a})$

This is equivalent to

If N sorts any 0-1 input sequence ➔ N sotrs any input of real values,

Corollary. If a network merges two sorted 0-1 sequences, it merges two sorted sequences of real values.
In the following, we compare 0's and 1's in the two merge networks
By induction, we assume the correctness of networks of half size

Correctness of bitonic merge network.
0-1 bitonic sequence 0 0 0 1 1 …1 1 0 0 … 0
Cut the sequence into left half anf right half. We have the following typical cases.

		before	after
Case 1	left	0 0 0 0 0 1 1 1	0 0 0 0 0 0 0 0
	Right	1 1 0 0 0 0 0 0	1 1 0 0 0 1 1 1
Case 2		0 0 1 1 1 1 1 1	0 0 1 1 0 0 0 0
		1 1 1 1 0 0 0 0	1 1 1 1 1 1 1 1
Case 3		0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
		0 0 1 1 1 0 0 0	0 0 1 1 1 0 0 0
Case 4		1 1 1 0 0 1 1 1	1 1 1 0 0 1 1 1
		1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1

In the stage of "after", left and right are all bitonic sequences and left ≤ right

Correctness of odd-even merge network p 0's in left and q 0's in right

Case 1	left	0 0 0 1 1 1	0 0 0 0 1 1
(p: odd, q: odd)	right	0 0 0 1 1 1	0 0 1 1 1 1
Case 2.	left	0 0 0 1 1 1	0 0 0 1 1 1
(p: odd , q: even)	right	0 0 1 1 1 1	0 0 1 1 1 1
Case 3			
(p: even, q: even)	left	0 0 0 0 1 1	0 0 0 1 1 1
	right	0 0 1 1 1 1	0 0 0 1 1 1