1 Sorting Networks

We will assume that a comparator outputs the smaller value on the top and the larger value on the bottom wire.

a) Wrong. Consider the following input, from top to bottom: $I = (0, 0, 1, 0, 0, 0)$ which produces the output $O = (0, 0, 0, 1, 0, 0)$. This is obviously not sorted.

b) Wrong. A comparator leaves a sorted sequence intact.

c) Correct. A correct sorting network needs to be able to sort any input sequence, and a comparator added to the front merely changes the input for the sorting network.

d) Correct. Assume otherwise: There is a correct sorting network $S$ such that there is no comparator between wires $i$ and $i + 1$. Consider the input sequence $I$ consisting of 0’s, then a 1 at the $i$th spot, a 0 at the spot $i + 1$, and the rest 1s. Now every wire will leave $I$ intact, because $I$ is already sorted except for the wires $i$ and $i + 1$. But since there is no wire, the output to $S$ is the same as the input, $O = I$, which is not sorted. Thus, no such $S$ exists.

e) Wrong. Consider the following network with three wires depicted in Figure 1. It does not sort $I = (1, 1, 0)$ because the 0 does not have a chance to get to the top wire, so $O = (1, 0, 1)$.

![Figure 1: A network with a Comparator between each pair of wires.](image1)

f) Wrong. This can be seen in two ways. The first is by a simple counter example as in Figure 2. If we leave out the marked comparator, then we have a correct sorting network (easy to see or otherwise check all 0-1 sequences). Adding it gives us the same problem as the network in Figure 1 above.

![Figure 2: A network with an additional Comparator added inside.](image2)

Another indication for why this statement is wrong comes from considering Batcher’s Sorting Network from the lecture. It relies on creating and sorting bitonic sequences. A Comparator can easily destroy a bitonic sequence, such as $(0, 1, 1, 0)$ into $(0, 1, 0, 1)$. 