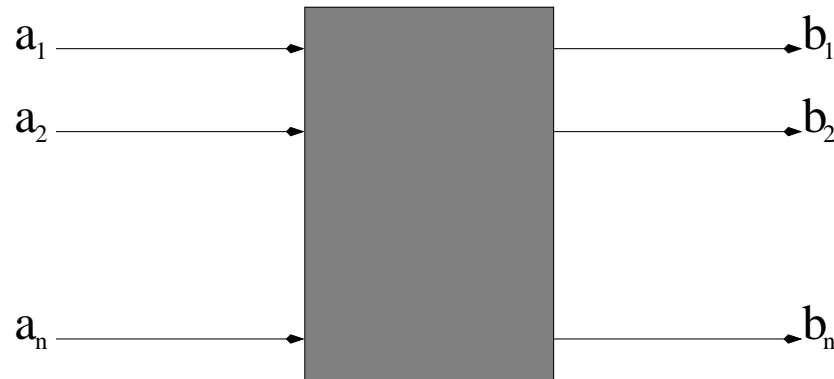


Analysis of Algorithms

SORTING NETWORKS

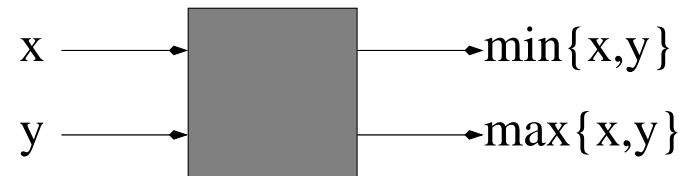
Sorting Networks



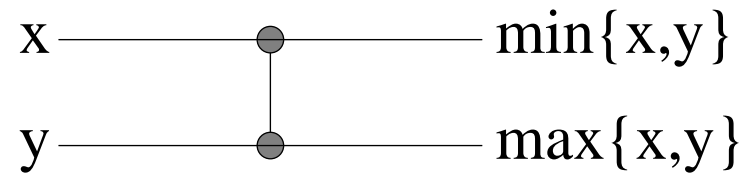
- ★ n unsorted inputs a_1, a_2, \dots, a_n .
- ★ n sorted outputs $b_1 \leq b_2 \leq \dots \leq b_n$.
- ★ The set $\{b_i\}_{i=1}^n$ is the same as the set $\{a_i\}_{i=1}^n$.
- ★ The networks can use only **comparators**.

A Comparator

Sorting Network for $n = 2$:

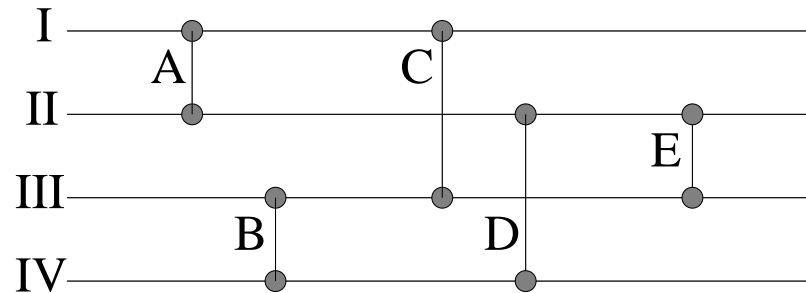


Notation:



- ★ 2 inputs in **any** order.
- ★ 2 outputs such that the top is " \leq " the bottom.

Example – Sorting 4 Inputs



Correctness:

- ★ After **A** and **B**: the minimum is either on **I** or on **III** and the maximum is either on **II** or on **IV**.
- ★ After **C** and **D**: **I** is the minimum and **IV** is the maximum.
- ★ After **E**: the outputs are sorted.

Complexity:

- **Size**: 5 comparators.
- **Depth**: 3 time steps.

Complexity Criteria

Size of a network: Number of comparators.

Depth of a network:

- ★ Depth of the network input lines is 0.
- ★ Depth of output lines of a comparator whose input lines have depth d_x and d_y is $1 + \max\{d_x, d_y\}$.
- ★ Depth of a comparator is the depth of its output lines.
- ★ Depth of a network is the maximum depth of its comparators.

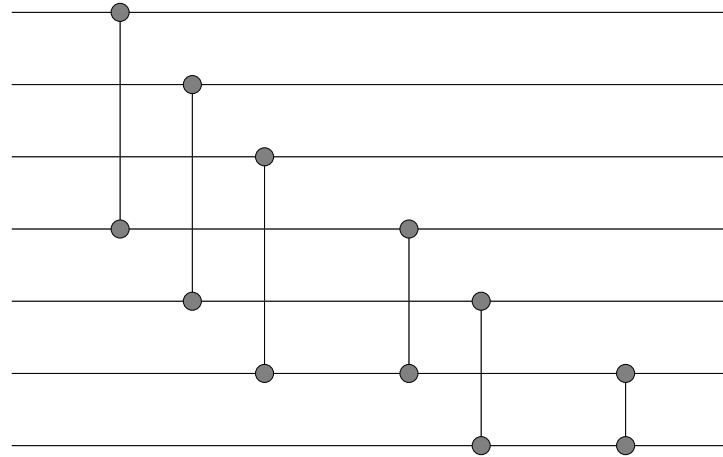
Time Complexity

Assumption: It takes one unit of time to **execute** a comparator.

- ★ A depth d comparator **produces** its outputs at time d .
- ★ A comparator network of depth d produces **all** of its outputs **no later** than time d .
- ★ **Some** of the outputs may be ready earlier.

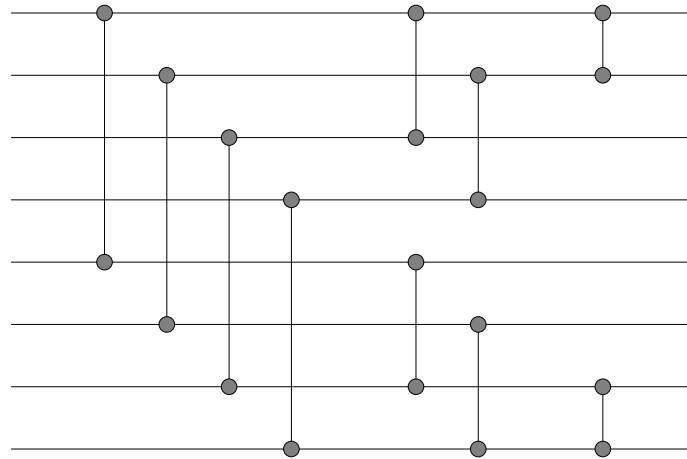
Proposition: The depth of the network could be **greater** than the number of comparators on any line.

Finding the Maximum



n	Size	Depth
7	6	3
≥ 1	$n - 1$	$\lceil \log_2 n \rceil$

Finding the Minimum and the Maximum



n	Size	Depth
8	10	3
$2k$	$(3n/2) - 2$	$\lceil \log_2 n \rceil$
$2k + 1$	$\lceil 3n/2 \rceil - 2$	$\lceil \log_2 n \rceil + 1$

Complexity Objectives

- ★ A sorting network with a **simple** structure.
- ★ A sorting network with a **scalable** structure.
- ★ “**Good**” solutions for small values of n .
- ★ A **close** formula for the **size** and the **depth** of the network as a function of n .

Known Constructions

- ★ A “simple” sorting networks of size $O(n \log^2 n)$ and depth $O(\log^2 n)$.
- ★ A “very complicated” sorting network of size $O(n \log n)$ and depth $O(\log n)$.

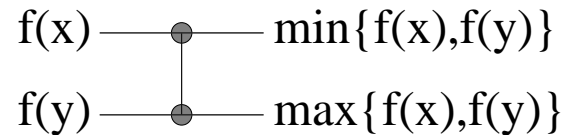
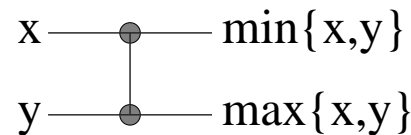
Lemma

Definition: f is a **monotone non-decreasing** function if
 $f(x) \leq f(y)$ for $x < y$.

Lemma: If a network produces the outputs b_1, \dots, b_n for the inputs a_1, \dots, a_n , then the network produces the outputs $f(b_1), \dots, f(b_n)$ for the inputs $f(a_1), \dots, f(a_n)$ for any **monotone non-decreasing** function f .

Proof

★ Assume first 2 inputs:



$$\min\{f(x),f(y)\}=f(\min\{x,y\})$$

$$\max\{f(x),f(y)\}=f(\max\{x,y\})$$

★ The rest of the proof is by **induction** on the comparators according to their **depth order**.

Corollary

Claim: An n -input network that sorts **all** the permutations on $1, \dots, n$ sorts **all** the sequences of **arbitrary** numbers.

- ★ There are $n!$ permutations on $1, \dots, n$.
- ★ There are ∞ sequences of arbitrary numbers.

Proof

- ★ Assume a network \mathcal{N} that sorts **all** the permutations.
- ★ Let a_1, \dots, a_n be an arbitrary sequence of numbers.
- ★ Let $b_1 \leq \dots \leq b_n$ be a **stable** sorted sequence of a_1, \dots, a_n .
- ★ For $1 \leq i \leq n$, define $f(a_i) = j$ if $a_i = b_j$.
 - f is a permutation.
 - f is a monotone non-decreasing function.
- ★ By the **monotone lemma**, \mathcal{N} **behaves** the same for the sequence a_1, \dots, a_n and the permutation $f(a_1), \dots, f(a_n)$.
- ★ Therefore, \mathcal{N} sorts **all** the sequences of arbitrary numbers.

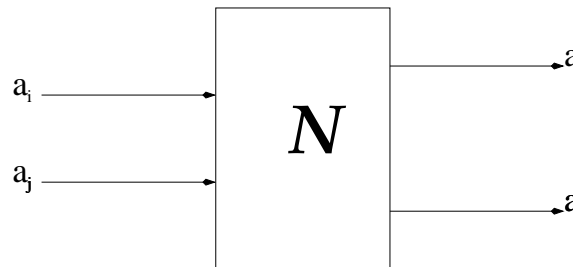
The Zero-One Principle

Claim: An n -input network that sorts **all** the sequences of 0's and 1's sorts **all** the sequences of **arbitrary** numbers.

- ★ There are 2^n sequences of 0's and 1's.
- ★ There are ∞ sequences of arbitrary numbers.

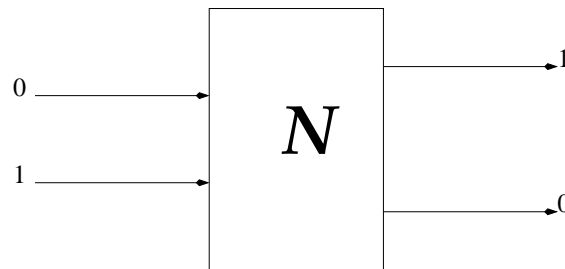
Proof

- ★ Assume a network \mathcal{N} that sorts **all** the 0-1 sequences.
- ★ Assume \mathcal{N} **does not** sort the arbitrary sequence of numbers a_1, \dots, a_n .
- ★ Let $a_i < a_j$ such that \mathcal{N} places a_j **above** a_i .



Proof

- ★ Let f be the **monotone non-decreasing** function:
 - $f(x \leq a_i) = 0$ and $f(x > a_i) = 1$.
 - $f(a_i) = 0$ and $f(a_j) = 1$.
- ★ By the **monotone lemma**, if the inputs for \mathcal{N} are $f(a_1), \dots, f(a_n)$, then $f(a_j)$ is **placed above** $f(a_i)$.



- ★ A **contradiction** to the assumption on \mathcal{N} .

Bitonic Sequences

Definition: A **bitonic** sequence is:

$$a_1 \leq a_2 \leq \cdots \leq a_h \geq a_{h+1} \geq \cdots \geq a_{n-1} \geq a_n$$

or

$$a_1 \geq a_2 \geq \cdots \geq a_h \leq a_{h+1} \leq \cdots \leq a_{n-1} \leq a_n$$

Goal: Construct a network that sorts **all** the bitonic sequences.

Modified goal: By the zero-one principle, it is enough to construct a network that sorts **all** the 0-1 bitonic sequences.

0-1 Bitonic Sequences

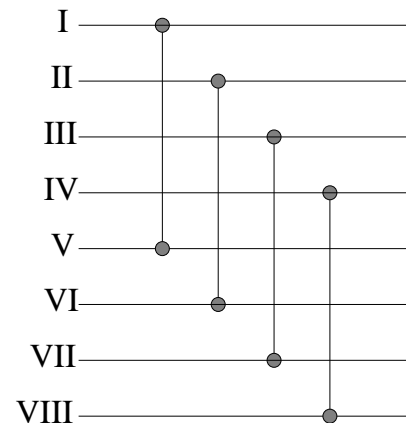
- ★ A **0-1 bitonic** sequence is: $0^i 1^j 0^k$ or $1^i 0^j 1^k$ for all possible $i, j, k \geq 0$ such that $i + j + k = n$.
- ★ **Special** cases of the $0^i 1^j 0^k$ bitonic sequences:
 - $i = 0 \Rightarrow 1, \dots, 1, 0, \dots, 0$.
 - $k = 0 \Rightarrow 0, \dots, 0, 1, \dots, 1$.
 - $i = 0$ and $k = 0 \Rightarrow 1, \dots, 1$.
 - $j = 0 \Rightarrow 0, \dots, 0$.
- ★ All of the above sequences are also **special** cases of the $1^i 0^j 1^k$ bitonic sequences.

Number of 0-1 Bitonic Sequences

- ★ $B(n)$: Number of 0-1 bitonic sequences of length n .
- ★ $0^i 1^j 0^k$ bitonic sequences for $i \geq 1$ and $j \geq 1$:
 - $n - 1$ choices for i .
 - $n - i$ choices for j for each i .
 - $k = n - i - j$.
 - All together: $\sum_{i=1}^{n-1} (n - i) = \frac{(n-1)n}{2}$.
- ★ $1^i 0^j 1^k$ bitonic sequences for $i \geq 1$ and $j \geq 1$: $\frac{(n-1)n}{2}$.
- ★ 0^n and 1^n are also bitonic sequences.
- ★ All together: $B(n) = \frac{(n-1)n}{2} + \frac{(n-1)n}{2} + 2 = n^2 - n + 2$.

Half Cleaner

Definition: For an even n , in a **half cleaner** network there is a comparator between line i and line $i + n/2$ for all $1 \leq i \leq n/2$.



Size: $n/2$

Depth: 1

Claim

- ★ If the input to a **half-cleaner** is a **0 – 1 bitonic sequence**, then the output satisfies the following properties:
 - **Both** the top half and the bottom half are bitonic.
 - **Either** the top half is all 0 **or** the bottom half is all 1.
 - **All** the outputs in the top half are less or equal to **all** the outputs in the bottom half.

Possible Outputs

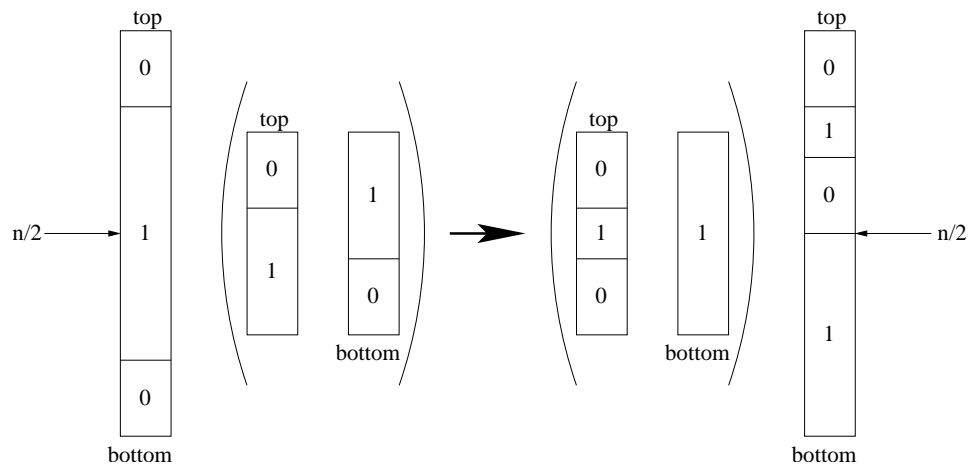
★ There are 11 **types** of outputs:

0	0	0	0	0	0	0	1	0	1	1
0	0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	0	1	0	1	0	1
0	0	0	0	0	0	1	0	0	1	1
0	0	1	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	1	1	1

Proof

- The proof is **only** for $0^i 1^j 0^k$ bitonic sequences.
- A **similar** proof works for $1^i 0^j 1^k$ bitonic sequences.
- A proof with **illustrations** handling four possible cases:
 - ★ $i \leq n/2; j \geq n/2; k \leq n/2.$
 - ★ $i \leq n/2; j \leq n/2; k \leq n/2.$
 - ★ $i \geq n/2; j \leq n/2; k \leq n/2.$
 - ★ $i \leq n/2; j \leq n/2; k \geq n/2.$

Case I

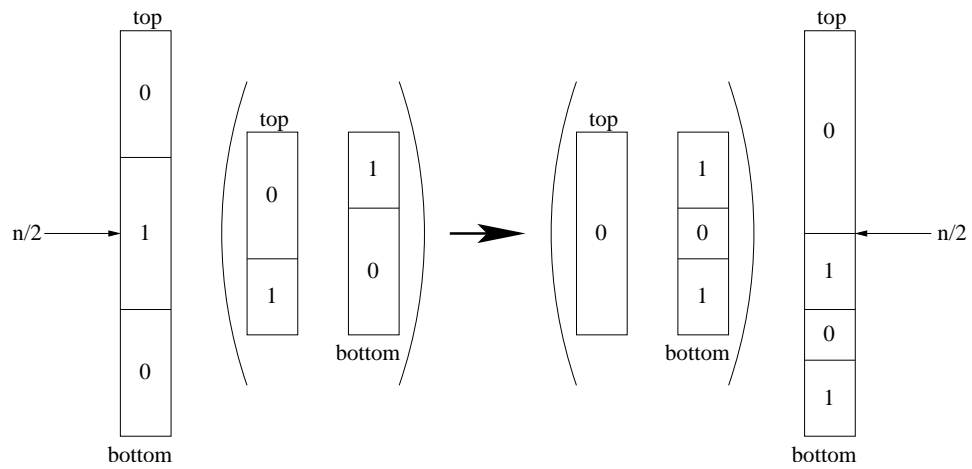


$$j \geq \frac{n}{2}$$

$$i \leq \frac{n}{2}$$

$$k \leq \frac{n}{2}$$

Case II

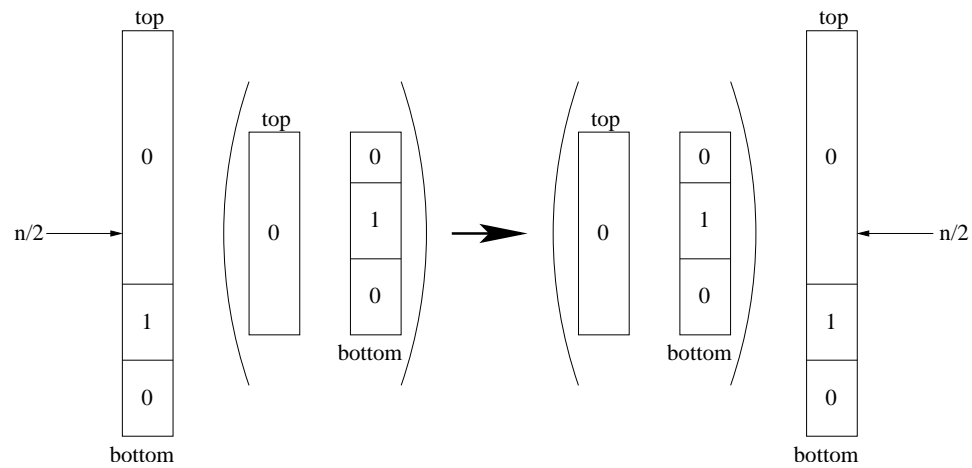


$$j \leq \frac{n}{2}$$

$$i \leq \frac{n}{2}$$

$$k \leq \frac{n}{2}$$

Case III

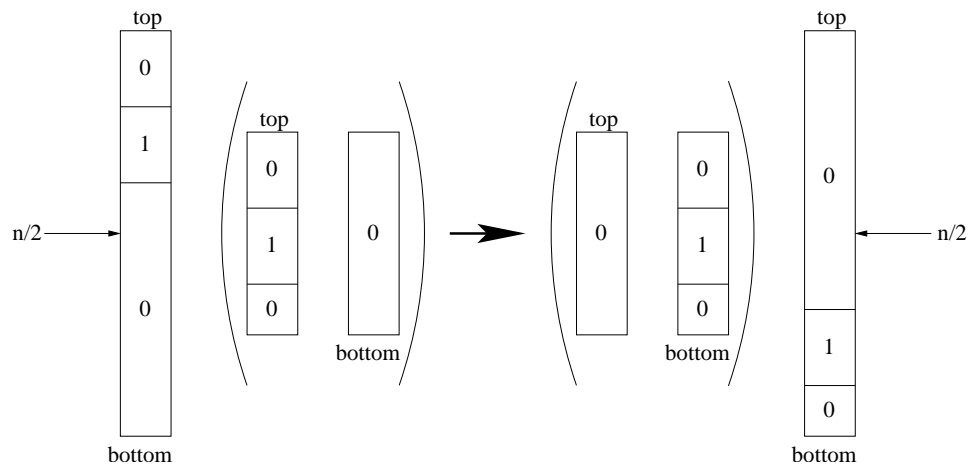


$$j \leq \frac{n}{2}$$

$$i \geq \frac{n}{2}$$

$$k \leq \frac{n}{2}$$

Case IV



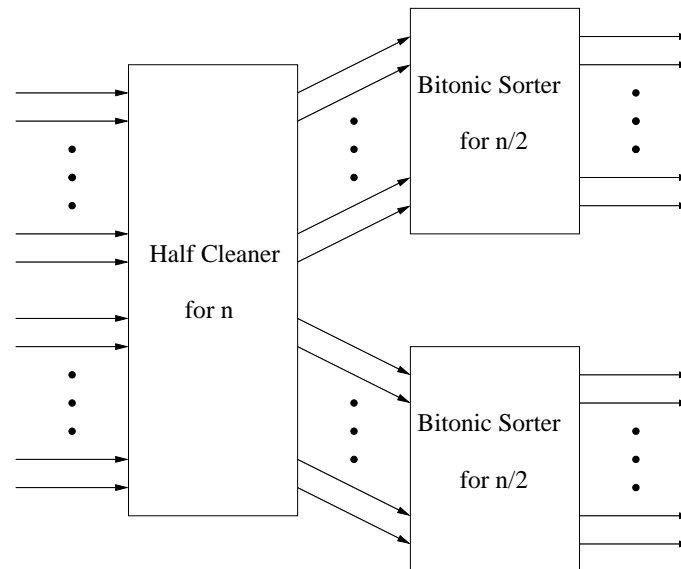
$$j \leq \frac{n}{2}$$

$$i \leq \frac{n}{2}$$

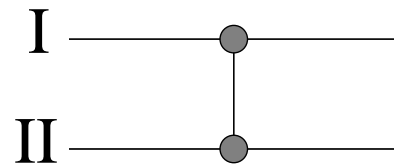
$$k \geq \frac{n}{2}$$

Bitonic Sorter

- ★ Assume $n = 2^k$ for an integer $k \geq 0$.
- ★ **First** apply a half cleaner on all n inputs.
- ★ **Recursively** apply bitonic sorters on the top half and the bottom half.



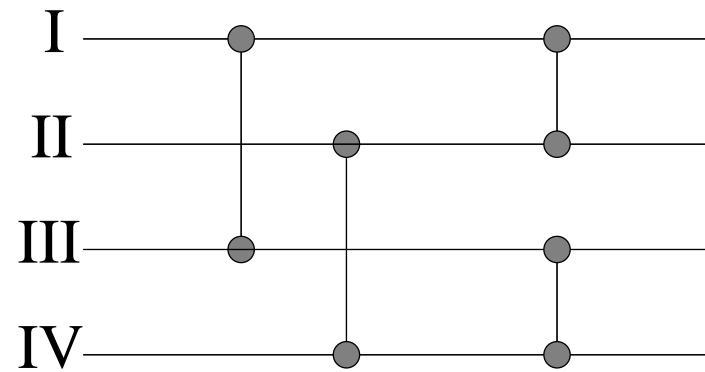
Bitonic Sorter for $n = 2$



Size: 1

Depth: 1

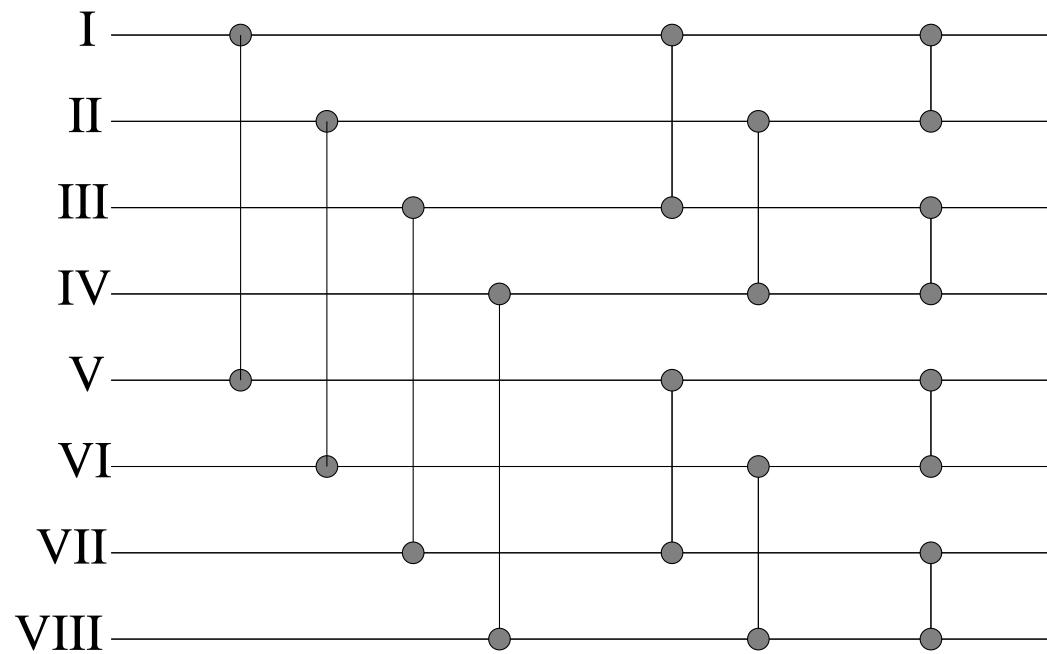
Bitonic Sorter for $n = 4$



Size: 4

Depth: 2

Bitonic Sorter for $n = 8$



Size: 12

Depth: 3

Correctness

- ★ **After** the first half cleaner:
 - Both halves are bitonic.
 - The top half is “ \leq ” the bottom half.
- ★ Therefore, the recursion **works**.

Depth Complexity – $D(n)$

$$D(1) = 0$$

$$D(2) = 1$$

$$D(4) = 2$$

$$D(8) = 3$$

$$\begin{aligned} D(n = 2^k) &= 1 + D(n/2) \\ &= 1 + 1 + \cdots + 1 \\ &= \log_2 n \end{aligned}$$

Size Complexity – $S(n)$

$$S(1) = 0$$

$$S(2) = 1$$

$$S(4) = 4$$

$$S(8) = 12$$

$$\begin{aligned} S(n = 2^k) &= (n/2) + 2S(n/2) \\ &= 1(n/2) + 2(n/4) + 4(n/8) + \dots + (n/2)1 \\ &= (n/2) \log_2 n \end{aligned}$$

Merger

Inputs for $n = 2^k$:

$$\star a_1 \leq a_2 \leq \cdots \leq a_{n/2}$$

$$\star a_{n/2+1} \leq a_{n/2+2} \leq \cdots \leq a_n.$$

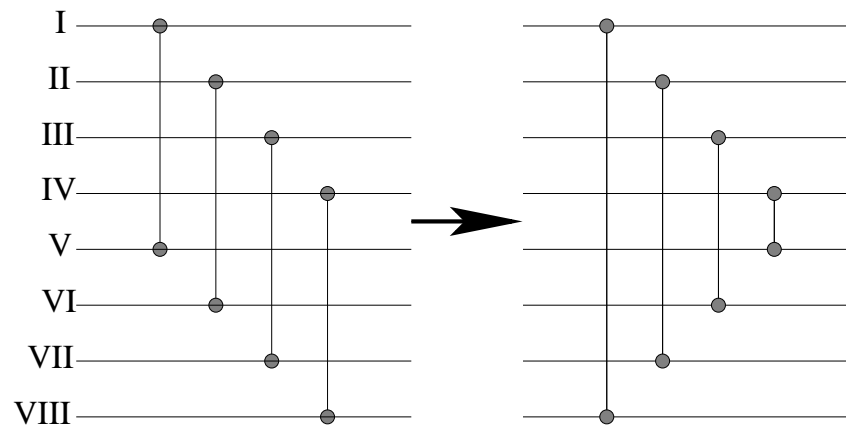
Outputs: $b_1 \leq b_2 \leq \cdots \leq b_n$.

\star The set $\{b_i\}_{i=1}^n$ is the same as the set $\{a_i\}_{i=1}^n$.

Idea: The following is a bitonic sequence:

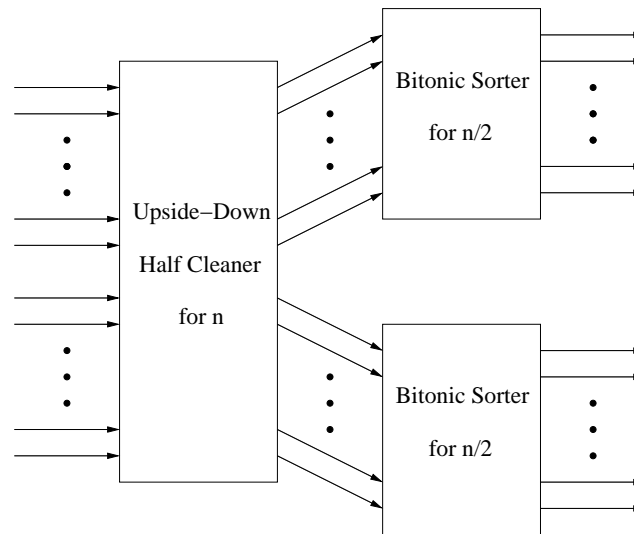
$$- a_1 \leq a_2 \leq \cdots \leq a_{n/2}, a_n \geq \cdots \geq a_{n/2+2} \geq a_{n/2+1}.$$

Merger – Construction



- ★ In a bitonic sorter, **replace** the first half cleaner for n with an **upside-down** half cleaner for n .
- ★ The outputs of **both** halves are bitonic sequences.
- ★ **Recursively**, continue with two **bitonic sorters** on $n/2$.

Merger – Complexity

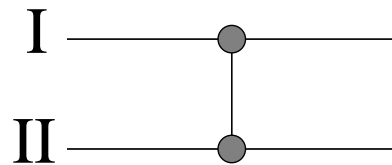


Complexity for $n = 2^k$: The **same** as bitonic sorter.

Depth: $D(n) = \log_2 n$.

Size: $S(n) = (n/2) \log_2 n$.

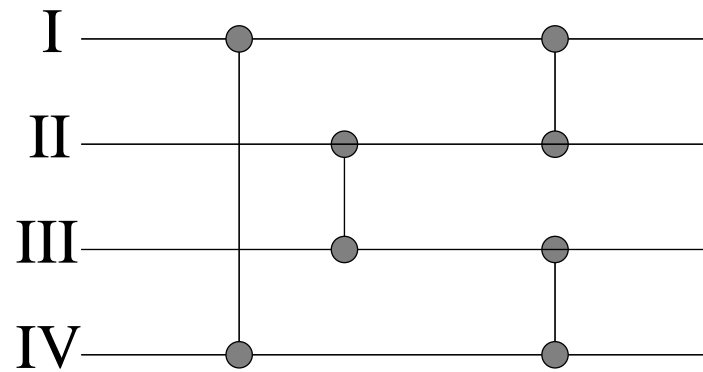
Merger for $n = 2$



Size: 1

Depth: 1

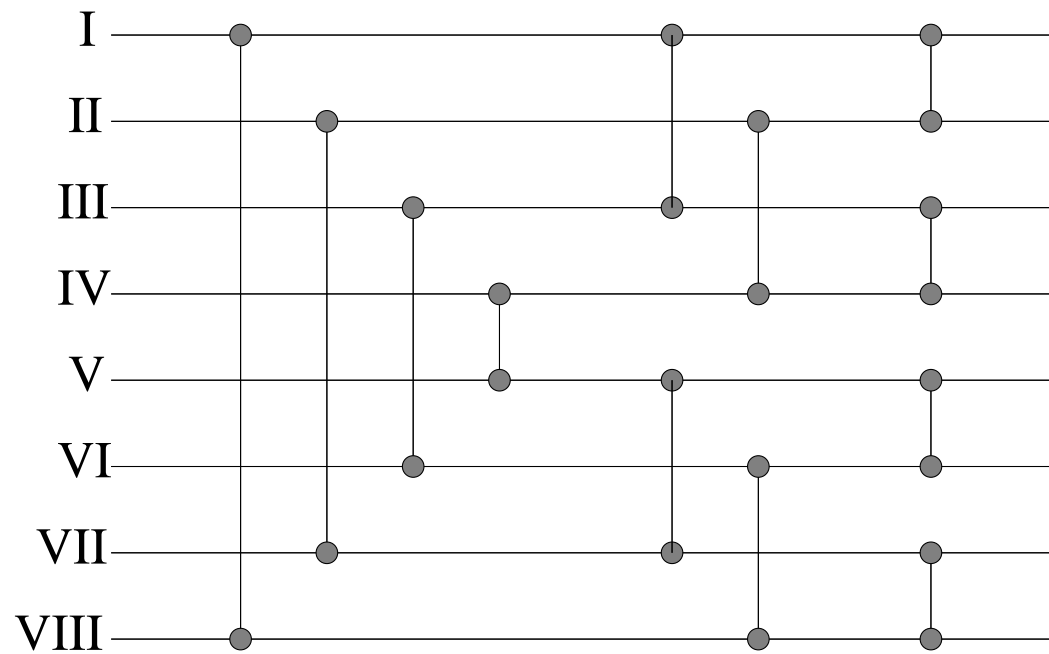
Merger for $n = 4$



Size: 4

Depth: 2

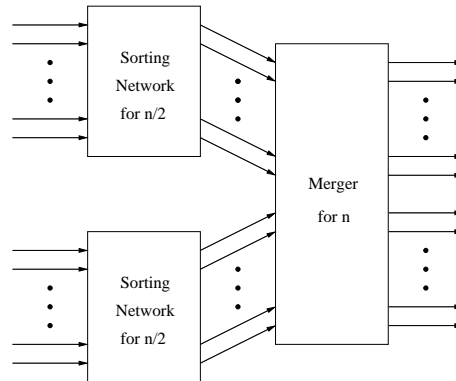
Merger for $n = 8$



Size: 12

Depth: 3

Sorting Network



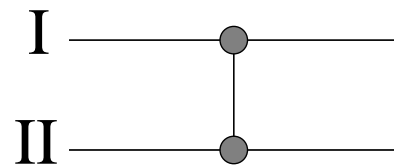
Construction: Merge-Sort.

- ★ **Recursively** sort the top half and the bottom half.
- ★ **Merge** both halves using a **Merger** for n inputs.

Correctness: By **induction** on n .

Remark: Construction is equivalent to the non-recursive implementation of **Merge-Sort**.

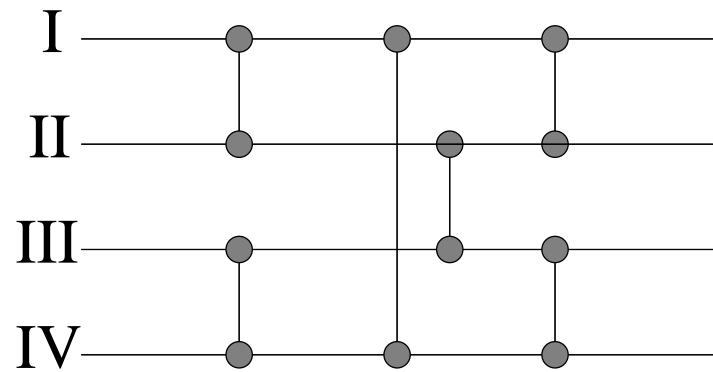
Sorting Networks for $n = 2$



Size: 1

Depth: 1

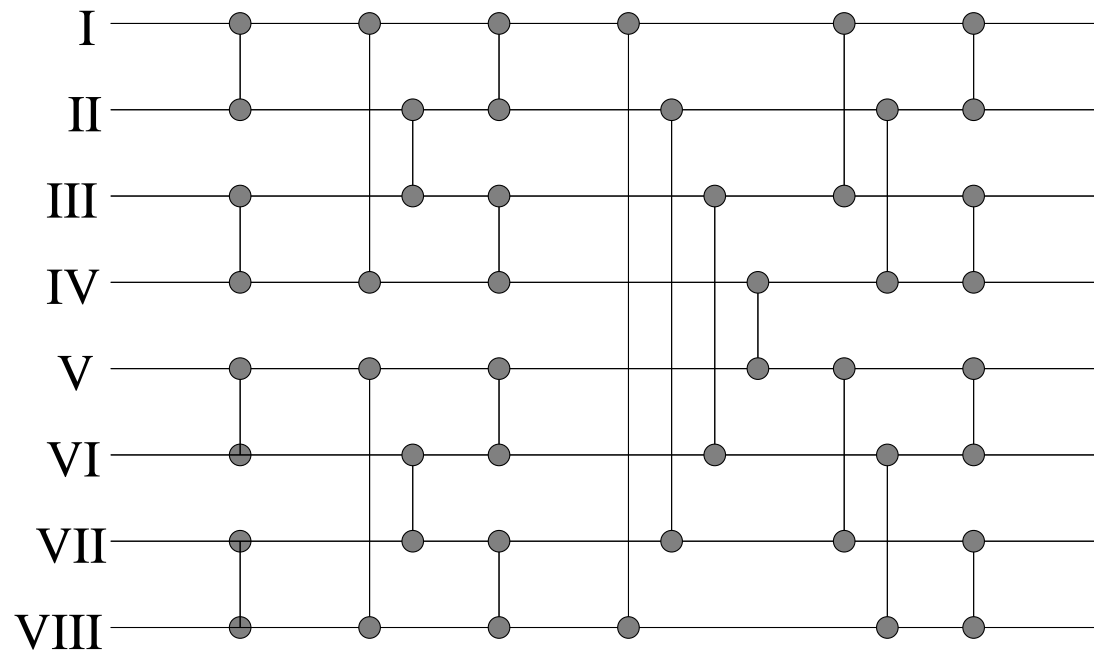
Sorting Networks for $n = 4$



Size: 6

Depth: 3

Sorting Networks for $n = 8$



Size: 24

Depth: 6

Depth Complexity – $D(n)$

$$D(1) = 0$$

$$D(2) = 1$$

$$D(4) = 3$$

$$D(8) = 6$$

$$\begin{aligned} D(n = 2^k) &= \log_2 n + D(n/2) \\ &= \log_2 n + \log_2(n/2) + \log_2(n/4) + \cdots + 1 \\ &= (\log_2 n (\log_2 n + 1)) / 2 \\ &= \Theta(\log^2 n) \end{aligned}$$

Size Complexity – $S(n)$

$$S(1) = 0$$

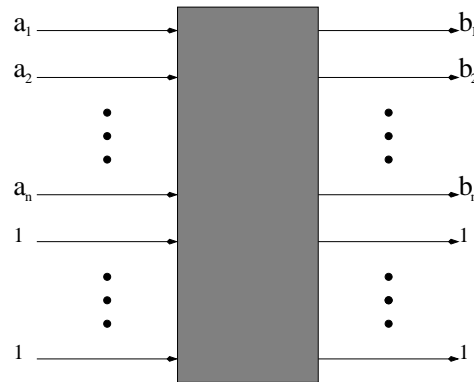
$$S(2) = 1$$

$$S(4) = 6$$

$$S(8) = 24$$

$$\begin{aligned} S(n = 2^k) &= (n/2) \log_2 n + 2S(n/2) \\ &= (n/2)(\log_2 n + \log_2(n/2) + \log_2(n/4) + \cdots + 1) \\ &= (n/2)((\log_2 n(\log_2 n + 1))/2) \\ &= \Theta(n \log^2 n) \end{aligned}$$

n is not power of 2



- ★ $2^{k-1} < n < 2^k$ unsorted inputs a_1, a_2, \dots, a_n .
- ★ n sorted outputs $b_1 \leq b_2 \leq \dots \leq b_n$.
- ★ The set $\{b_i\}_{i=1}^n$ is the same as the set $\{a_i\}_{i=1}^n$.
- ★ **Bottom** $(2^k - n)$ inputs and outputs are 1.

Correctness

- ★ A 0-1 sorting network is **stable**.
- ★ Bottom $(2^k - n)$ outputs are bottom $(2^k - n)$ inputs.
- ★ \Rightarrow The **top** n lines sort the **top** n inputs.

Remark: Does not work for bitonic sorters. Because, bottom $(2^k - n)$ inputs may create a non-bitonic input sequence.

Complexity

Depth:

$$\begin{aligned} D(n) &= (\log_2(2^k)(\log_2(2^k) + 1))/2 \\ &= \lceil \log_2 n \rceil (\lceil \log_2 n \rceil + 1)/2 \\ &= \Theta(\log^2 n) . \end{aligned}$$

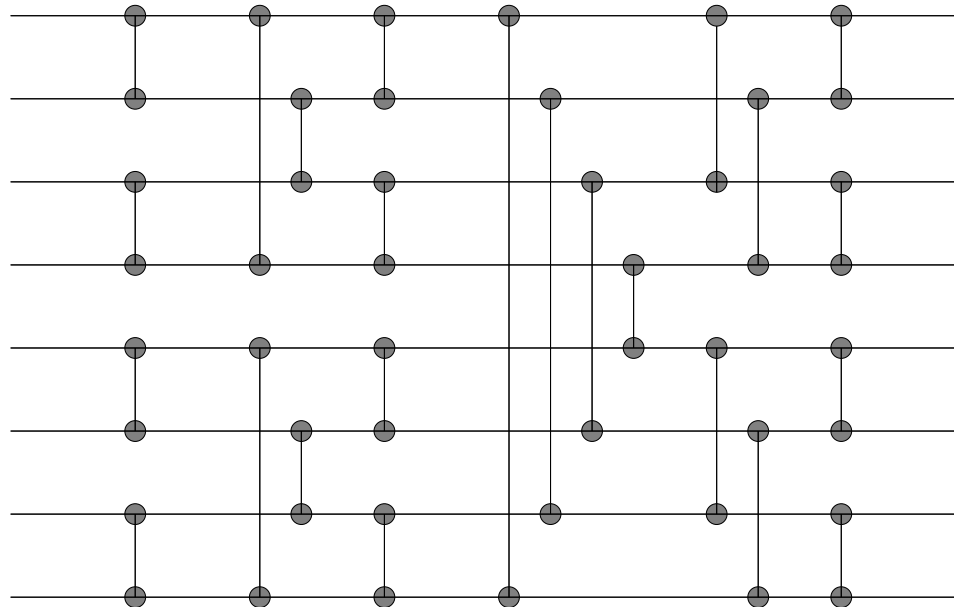
Size:

$$\begin{aligned} S(n) &= (2^k/2)((\log_2(2^k)(\log_2(2^k) + 1))/2) \\ &< n(\lceil \log_2 n \rceil (\lceil \log_2 n \rceil + 1)/2) \\ &= \Theta(n \log^2 n) . \end{aligned}$$

Improvements

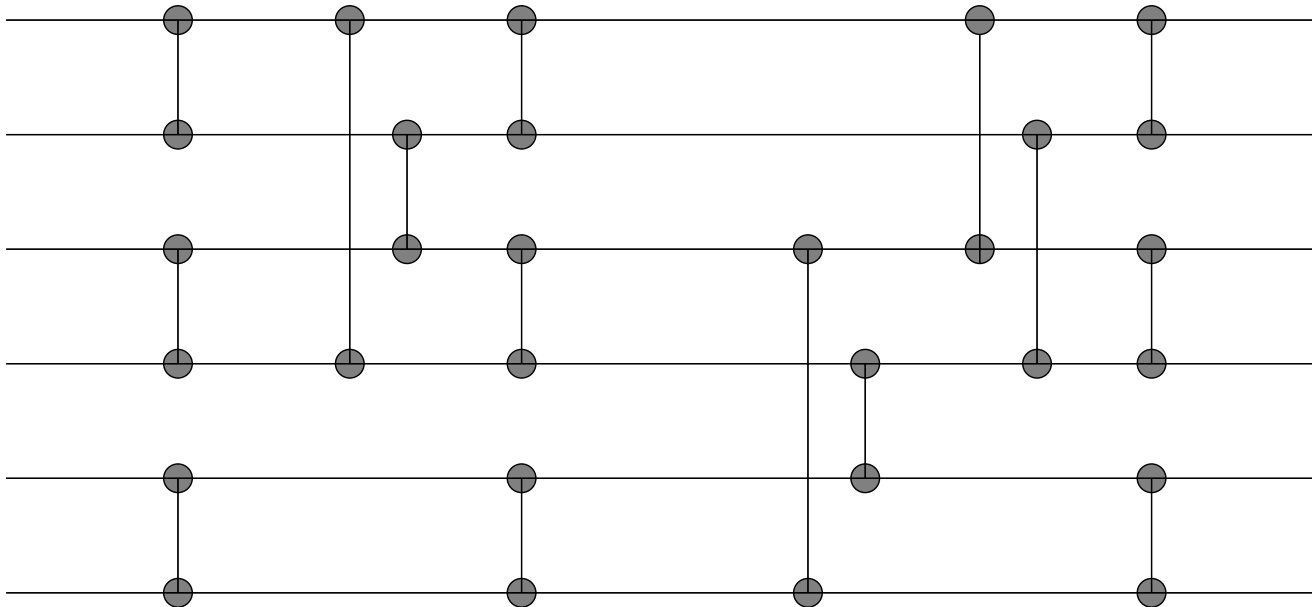
- ★ Omit **bottom** $(2^k - n)$ lines and **all** comparators connected to these lines.
- ★ Omit comparators between already **sorted** lines.

Example – $n = 6$



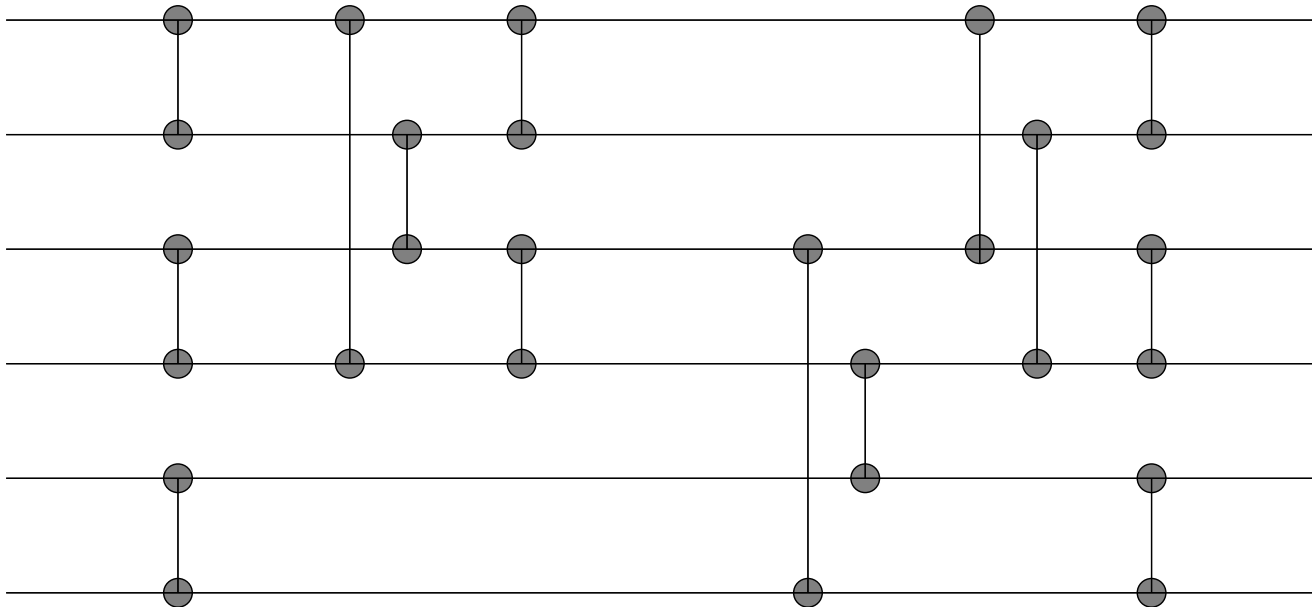
Sorting network for $n = 8$: size 24 and depth 6.

Example – $n = 6$



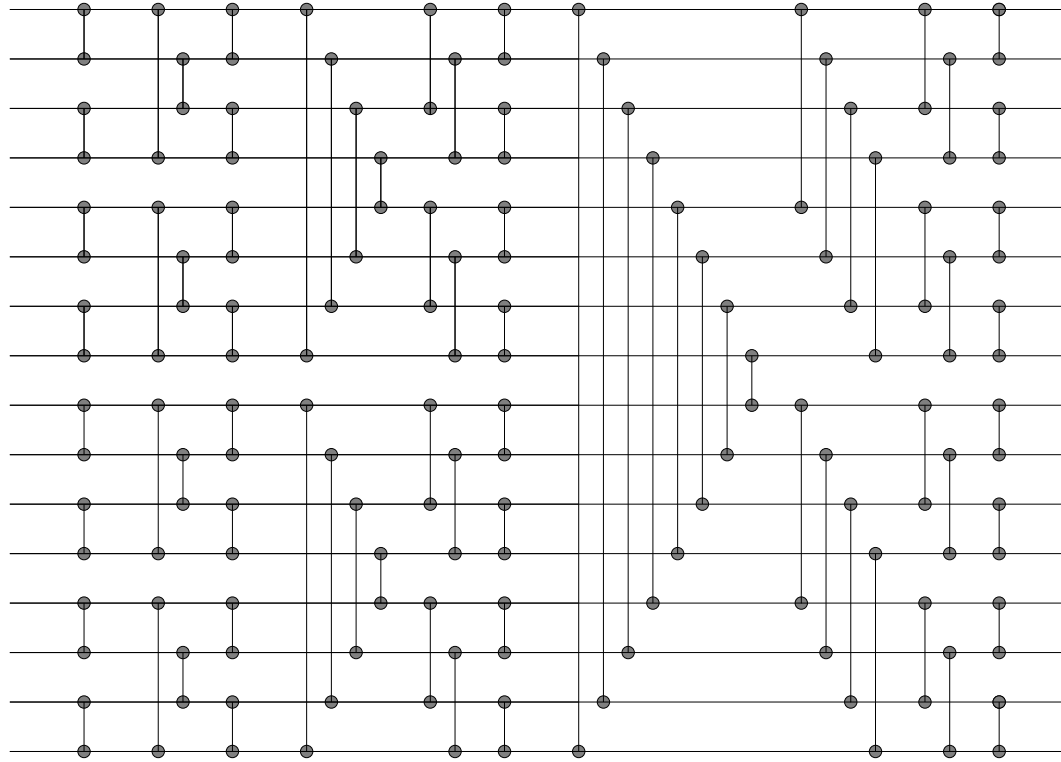
After omitting the bottom 2 lines: **size** 15 and **depth** 6.

Example – $n = 6$



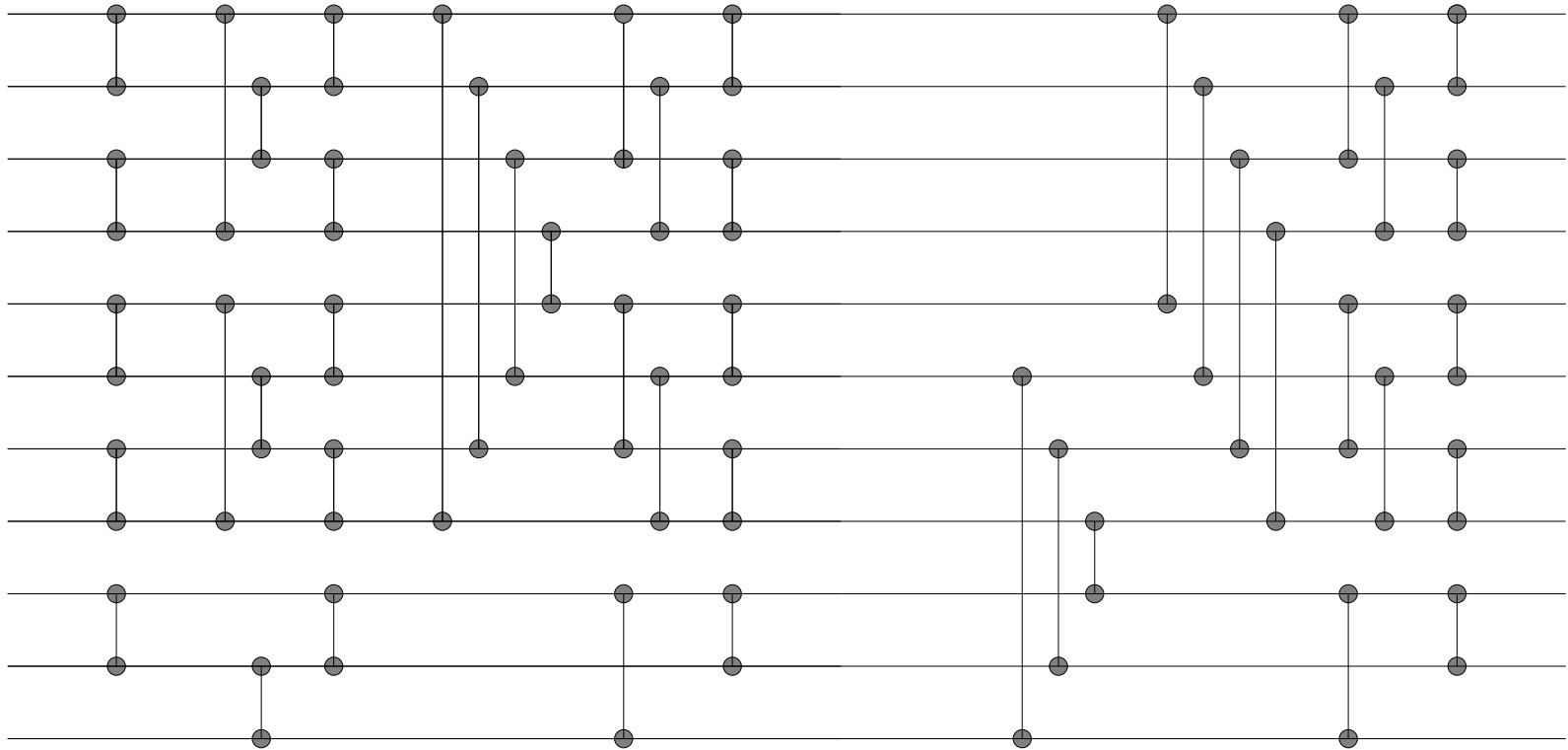
After omitting redundant comparators: **size** 14 and **depth** 6.

Example – $n = 11$



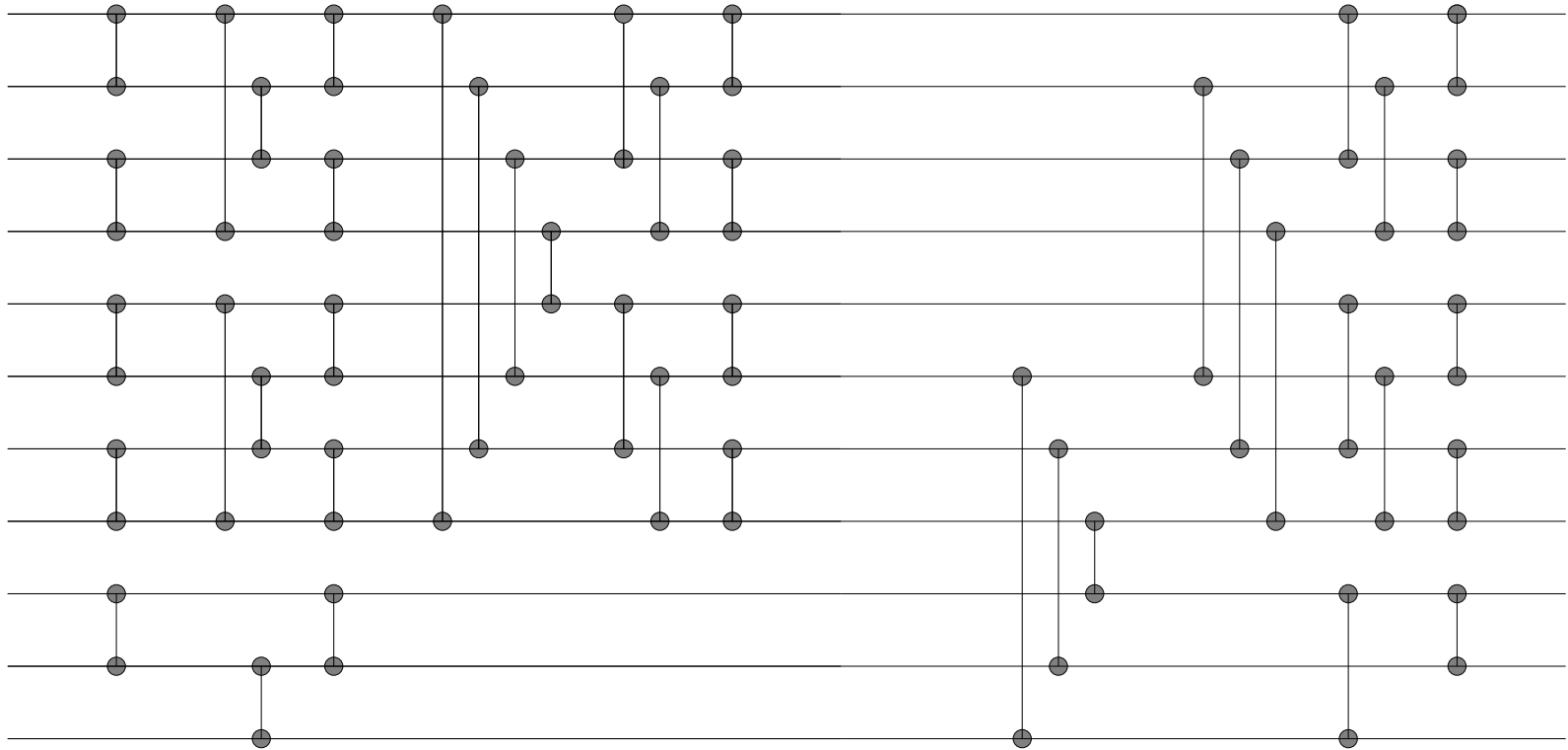
Sorting network for $n = 16$: size 80 and depth 10.

Example – $n = 11$



After omitting the bottom 5 lines: **size** 46 and **depth** 10.

Example – $n = 11$



After omitting redundant comparators: **size 43** and **depth 10**.