SORTING NETWORKS ON 14 INPUTS NEED 50 COMPARATORS

TEJASVI SINGH TOMAR

Abstract. Let $S(N)$ be the least number of comparators in a sorting network on N inputs. Its value is known for $N \leq 13$, and for 14 inputs it was known that $49 \leq S(14) \leq 51$. We prove $S(14) \geq 50$, so $S(14)$ is 50 or 51; as a consequence, $S(16) \geq 59$. The proof uses $S(13) = 45$ to restrict the paths of the largest and the smallest value in a sorting network on 14 inputs with 49 comparators. After a computation, these restrictions leave a few thousand possible prefixes of such a network, up to symmetry. A computer search shows that none of them can be completed to such a network. The search uses Harder's Huffman bound (arXiv:2012.04400) and the weighted two-value bound from our proof of $S(13) = 45$; several of our programs are modifications of Harder's code. A separately written program checks the certificates, and $S(14) \geq 50$ is also derived in Lean 4 from bounds certified outside Lean; part of the derivation runs as compiled code.

1. Introduction

A comparator network on N inputs is a circuit of comparators, each of which outputs the larger and the smaller of its two input values on two separate leads. A sorting network delivers every input sequence in sorted order. The Bose–Nelson problem [1] asks for $S(N)$, the least number of comparators in a sorting network on N inputs. Floyd and Knuth determined $S(N)$ for $N \leq 8$ [6], Codish, Cruz-Filipe, Frank and Schneider-Kamp proved $S(9) = 25$ and $S(10) = 29$ by an exhaustive search [2], Harder proved $S(11) = 35$ and $S(12) = 39$ by a search over sets of vectors with lower bounds of Huffman type [8], and we proved $S(13) = 45$ by a weighted two-value bound and a search over partial networks [14]. The smallest known sorting networks on 14 inputs have 51 comparators [10, Section 5.3.4], [5]. Van Voorhis's bound $S(N) \geq S(N-1) + \lceil \log_2 N \rceil$ [19] and $S(13) = 45$ give $S(14) \geq 49$ [14, Corollary 1.3]. We rule out 49, so Van Voorhis's bound is not attained for $N = 14$, the first even N for which this happens. Throughout, a dagger (†) marks a statement whose proof includes a computation.

Theorem 1.1. $S(14) \geq 50$.†

Hence $S(14)$ is 50 or 51. We do not know which.

The proof combines the weighted two-value bound and the search of [14] with three main further ingredients: the shape of the trees formed by the paths of the largest and the smallest value, which reduces the problem to a list of starts; a normal form for the comparators that follow a start; and, for as long as an extreme value can still end on several channels, lower bounds from Huffman's inequality, our name for a weaker form of Harder's Huffman bound [8].

The structure of a network with 49 comparators. Suppose that T is a sorting network on 14 inputs with 49 comparators. Since $S(14) \geq 49$, the set of 0-1 vectors produced by a prefix of T cannot be sorted with fewer comparators than follow it. Deleting the comparators on the path of the largest value from an input leaves a sorting network on the other 13 inputs [8, Lemma 16],

Date: 17 November 2026, version 3.

2020 *Mathematics Subject Classification.* Primary 68P10; Secondary 68Q06, 05C05, 68V05.

Key words and phrases. Sorting networks, Bose–Nelson problem, comparator networks, lower bounds, computer-assisted proof.

so each such path meets at most $49 - 45 = 4$ comparators, and the same holds for the smallest value. The paths of the maximum form a tree whose leaves are the 14 inputs, and Kraft's equality leaves room only for two leaves at depth 3 and twelve at depth 4; only the paths from the two leaves at depth 3 can meet one further comparator each (Lemma 3.2). The same holds for the paths of the minimum. Both trees are built from the same first layer of six or seven comparators (Lemma 3.4).

Extending Mericanii's definition of red leads [13, Definition 1] from the paths of the maximum to those of the minimum, we call a lead *red* if a path of the maximum or of the minimum passes through it, and a comparator *blue* if neither of its input leads is red, and *red* otherwise. The red comparators that do not depend on any blue comparator can be applied first (Lemma 4.2). Up to a permutation of the channels and complementation, their output set is the set produced by one of a few thousand sequences of comparators, the *starts* (Theorem 4.4). In most of them both trees are already built: the maximum and the minimum have reached two fixed channels. We call these starts *complete*. In the other starts, which we call *late*, some red comparators wait for an input lead produced by a blue comparator. We show that the rest of such a network can be scheduled in *macro-steps*, each a blue comparator followed by red comparators, each of which touches a channel touched earlier in the macro-step (Lemma 5.1).

The search. For each start we search over the possible continuations, one comparator at a time when both trees are complete and one macro-step at a time otherwise. We keep, for each class of similar sets, a state of least length found so far, and discard a state when a lower bound shows that it needs more comparators than remain. This is the generate-and-prune scheme of [2], except that, as in Harder's search [8, Section 5], lower bounds replace subsumption. Most of these bounds rest on the weighted two-value bound of [14]. Its weights are lower bounds on the *costs* of the *cores*, that is, on the least numbers of comparators that sort them; a core is a set on 11 channels that remains when the paths of the three largest or of the three smallest values are deleted, and in effect a set on 10 channels when the other extreme value has also reached its channel. The bound applies directly once the maximum or the minimum has reached a fixed channel. While it can still end on several channels, Huffman's inequality over these channels combines such bounds, and a last test applies the weighted two-value bound to the whole 14-channel set with weights from Huffman's inequality (Section 6). Every search ends with an empty layer.

Certificates and checks. The searches write certificates. We check them in two ways: by a replay, and by a certification of the lower bounds on costs of cores that the searches use. Certifying the results of an untrusted search in this way goes back to the proof of $S(9) = 25$: Cruz-Filipe and Schneider-Kamp checked its computation with a formally verified program [3], and Cruz-Filipe, Larsen and Schneider-Kamp later extended this work [4]. Harder certified $S(11) \geq 35$ in the same way [7], [8, Sections 7 and 8]. The bound $S(14) \geq 50$ is also derived in Lean 4 from the certified bounds; part of the derivation runs as compiled code. Sections 9 to 11 describe these checks and Table 5 summarizes them; Appendices A and B list the computations and the tests of the programs.

Larger networks. Theorem 1.1 also raises the lower bound on $S(16)$ that [14, Corollary 1.3] gives, from 58 to 59.

Corollary 1.2. $S(16) \geq 59$.[†]

The smallest known sorting network on 16 inputs, due to Green, has 60 comparators [10, Section 5.3.4], so $S(16)$ is 59 or 60. For the other $N \leq 32$ the bounds of [14, Corollary 1.3] do not change (Section 12).

2. Preliminaries

We use the model and the notation of [14], which we recall briefly, and prove three lower bounds that the searches need.

2.1. **Networks and paths.** A comparator network T on a set A of channels is a finite acyclic circuit of comparators. Each comparator g has two input leads and two output leads $\mathrm{hi}(g)$ and $\mathrm{lo}(g)$, which receive the larger and the smaller input value. Each lead is an input lead of T or an output lead of exactly one comparator, and it enters at most one comparator; the leads that enter no comparator are the outputs [14, Section 2.1]. A comparator h is a *predecessor* of g, and g *depends* on h, if a directed path in the circuit leads from an output lead of h to an input lead of g. On channels $[n] = \{0, \ldots, n-1\}$ we write (i, j), for $i \neq j$, for the comparator that puts the maximum on channel i and the minimum on channel j; it is *standard* if $i > j$. A sequence of comparators on channels is a network in this sense, and every sorting network can be untangled into a sequence of standard comparators of the same length [10, Exercise 5.3.4–16]. Untangling processes the comparators from left to right and exchanges two channel labels in the rest of the sequence whenever a comparator is not standard, so it keeps every prefix that consists of standard comparators. Any order of the comparators of a network that respects the dependencies gives the same circuit, and therefore the same network.

The value M placed on an input x above all other values leaves every comparator that it enters on its hi output. Its leads form the *path of the maximum* Π_x, and $\delta(x)$ denotes the number of comparators on it. The network T/x on $A \setminus \{x\}$ is obtained by deleting the leads of Π_x and the comparators that they enter, and joining the other input lead of each such comparator to its lo output [14, Section 2.2]. It has $|T| - \delta(x)$ comparators. Exchanging the roles of hi and lo gives the *path of the minimum* Π'_x, its number of comparators $\delta'(x)$, and the network $T\backslash x$ with $|T| - \delta'(x)$ comparators.

For a sequence P of comparators on $[n]$ we write $X_P = \{P(v) : v \in \{0,1\}^n\}$ for the set of 0-1 vectors that it produces. The map $v \mapsto P(v)$ is monotone and preserves the number of 1s. Hence, if u' arises from u by changing one coordinate from 0 to 1, then $P(u')$ arises from $P(u)$ in the same way [14, proof of Lemma 5.8]. We call such a pair an *up-step*.

2.2. **Sets of vectors.** We use Harder's notions [8] of sorting a set of vectors and its cost (Definition 1), similarity (Definition 3), pruning (Definition 18) and well-behaved sets (Definitions 27 and 28); each pruned set defined below, taken from [14, Definition 5.2], is a subset of the corresponding pruned set of [8]. A network on A *sorts* a set $X \subseteq \{0,1\}^A$ if its outputs can be listed so that every vector of X is mapped to a nondecreasing sequence, and $s(X)$ is the least size of a network that sorts X. Sets related by a permutation of the channels, possibly combined with complementation $v \mapsto \mathbf{1} - v$, are *similar*. A standard comparator (i, j) is *useful* for X if X contains vectors v with $(v_i, v_j) = (0, 1)$ and with $(v_i, v_j) = (1, 0)$; otherwise it leaves X unchanged or acts on it as the exchange of the channels i and j. We write e_c for the vector with a single 1 on channel c and $\mathbf{1}$ for the vector of 1s. A channel c is *extremal* in X for *polarity* 0 if $e_c \in X$, and for polarity 1 if $\mathbf{1} - e_c \in X$. We write $E_\epsilon(X)$ for the set of channels extremal for polarity ϵ. For $c \in E_\epsilon(X)$, the *pruned set* $X/c \subseteq \{0,1\}^{A\setminus\{c\}}$ is defined in [14, Definition 5.2]: for $\epsilon = 0$ it consists of the restrictions to $A \setminus \{c\}$ of the vectors of X that can be reached from e_c by up-steps inside X, and for $\epsilon = 1$ it is obtained by complementation. A *scenario* of (X, ϵ) is a pair (x, y) with $x \in E_\epsilon(X)$ and $y \in E_\epsilon(X/x)$, and its *core* is $(X/x)/y$, all pruned for the polarity ϵ [14, Definition 5.5].

Lemma 2.1. *Let $X \subseteq \{0,1\}^n$.*

(a) $s(\{0,1\}^n) = S(n)$, $s(X) \leq s(X')$ if $X \subseteq X'$, and similar sets have the same value of s.

(b) *Let T sort X with outputs $o_0, \dots, o_{n-1}$, and let $c \in E_\epsilon(X)$. For $\epsilon = 0$ the path Π_c ends on o_{n-1}, and T/c sorts X/c. For $\epsilon = 1$ the path Π'_c ends on o_0, and $T\backslash c$ sorts X/c. In particular $s(X/c) \le s(X)$.*

(c) *All sets obtained from $\{0,1\}^n$ by standard comparators, similarities and pruning are* well-behaved*, that is, unions of maximal chains of up-steps.*

Proof. Apart from polarity 1 in (b), these are Lemmas 5.1, 5.3 and 5.4 of [14], whose proofs use Lemma 2, Corollary 6, Lemma 20, Lemma 35 and Corollary 36 of [8]. For polarity 1 in (b), let T^* be T with hi and lo exchanged at every comparator. It sorts the complement of X with the outputs listed in reverse order, its paths of the maximum are the paths of the minimum of T, and T^*/c is $T\backslash c$ with hi and lo exchanged. Apply [14, Lemma 5.3] to T^*. □

Our cores are obtained by pruning several channels in turn. The next lemma describes the result.

Lemma 2.2. *Let $X \subseteq \{0,1\}^n$ and $1 \le r \le n-1$, let $Y_0 = X$, and let $Y_l = Y_{l-1}/x_l$ with $x_l \in E_0(Y_{l-1})$ for $l = 1, \dots, r$. Then Y_r consists of the restrictions to $[n] \setminus \{x_1, \dots, x_r\}$ of the vectors of X that can be reached from $e_{x_1} + \cdots + e_{x_r}$ by up-steps inside X. Conversely, if $x_1, \dots, x_r$ are distinct channels and X contains the vectors $e_{x_1} + \cdots + e_{x_l}$ for $l = 1, \dots, r$, then $x_l \in E_0(Y_{l-1})$ for each l.*

Proof. For $l = 1$ the first claim is the definition of X/x_1. The vectors reachable from $e_{x_1} + \cdots + e_{x_{l-1}}$ have 1s on $x_1, \dots, x_{l-1}$, so restriction is injective on them, and it maps up-steps to up-steps in both directions. A vector of Y_{l-1} that can be reached from e_{x_l} inside Y_{l-1} therefore lifts to a vector of X that can be reached from $e_{x_1} + \cdots + e_{x_l}$ inside X, and conversely, which gives the induction step. For the second claim, $e_{x_1} + \cdots + e_{x_l}$ arises from $e_{x_1} + \cdots + e_{x_{l-1}}$ by one up-step, so by the first claim its restriction e_{x_l} lies in Y_{l-1}. □

Let P be a sequence of standard comparators on $[n]$ and ϵ a polarity. A channel x lies in $E_0(X_P)$ exactly when some path of the maximum in P ends on x, because only the vectors e_i are mapped to vectors with a single 1. For $x \in E_0(X_P)$ let $w^0_P(x)$ be the largest number of comparators on a path of the maximum in P that ends on x, and define $w^1_P(x)$ for $x \in E_1(X_P)$ in the same way with the paths of the minimum. We write w_x when P and ϵ are clear.

2.3. **Trivial bounds.** The next lemma extends the trivial bound of [14, Lemma 5.8] from two values to any number of values.

Lemma 2.3. *Let P be a sequence of k standard comparators on $[n]$, let $1 \le r \le n-1$, and let $i_1, \dots, i_r$ be distinct channels. Run P with values $a_1 > \cdots > a_r$ on $i_1, \dots, i_r$ and smaller values on the other inputs, let x_l be the channel on which a_l ends, and let W be the set of comparators that have one of $a_1, \dots, a_r$ on an input lead. Put $Y_0 = X_P$ and $Y_l = Y_{l-1}/x_l$, pruned for polarity 0. Then $x_l \in E_0(Y_{l-1})$ for $l = 1, \dots, r$, and*

$$s(Y_r) \;\ge\; S(n-r) - (k - |W|).$$

The same holds for polarity 1, with $a_1 < \cdots < a_r$ below all other values.

Proof. Let $P_0 = P$ and $P_l = P_{l-1}/i_l$. We show by induction on l that a_l is the largest value in the run of P_{l-1} on the inputs other than $i_1, \dots, i_{l-1}$, and that P_{l-1} computes on its leads the values that P computes on the leads that do not carry $a_1, \dots, a_{l-1}$. For $l = 1$ this is clear. The network P_l arises from P_{l-1} by deleting the path of its maximum a_l; at each deleted comparator the other value leaves on lo, where the join takes it, so P_l computes the values of P_{l-1} on the leads that do not carry a_l [8, proof of Lemma 16]. A comparator of P whose input leads carry a_l and some a_j with $j < l$ lies on the path of a_j in P_{j-1} and is therefore deleted in P_j, so the comparators that

a_l enters in P_{l-1} are exactly the comparators of P that a_l enters while no larger a_j is on the other input lead. Summing over l, the networks $P_0, \dots, P_{r-1}$ lose $|W|$ comparators in all, and P_r has $k - |W|$ comparators on the inputs $[n] \setminus \{i_1, \dots, i_r\}$ and its outputs on $[n] \setminus \{x_1, \dots, x_r\}$.

The same argument with 0-1 inputs shows, for every $z \in \{0,1\}^{[n]\setminus\{i_1,\dots,i_r\}}$, that $P(e_{i_1} + \cdots + e_{i_r} + z)$ has 1s on $x_1, \dots, x_r$ and agrees with $P_r(z)$ on the other channels: on an input with a 1 on i_l, the leads of the path of the maximum of P_{l-1} from i_l carry 1, because the networks are monotone. Starting from e_{i_1} and adding the 1s on $i_2, \dots, i_r$ and then those of z one at a time gives a chain of up-steps whose images form a chain of up-steps in X_P from $P(e_{i_1}) = e_{x_1}$ through $e_{x_1} + \cdots + e_{x_l}$ for each l to $P(e_{i_1} + \cdots + e_{i_r} + z)$. By Lemma 2.2, $x_l \in E_0(Y_{l-1})$ for each l, and $P_r(z)$, the restriction of $P(e_{i_1} + \cdots + e_{i_r} + z)$, lies in Y_r for every z. If a network U sorts Y_r, then P_r followed by U sorts $\{0,1\}^{[n]\setminus\{i_1,\dots,i_r\}}$, so $s(Y_r) + k - |W| \geq S(n-r)$ by Lemma 2.1(a). The polarity 1 follows by complementation, as in the proof of [14, Lemma 5.8]. □

For $r = 1$ the lemma gives, for each $x \in E_\epsilon(X_P)$,

$$s(X_P/x) \;\geq\; S(n-1) - (k - w_x), \tag{1}$$

by choosing an input whose path of the extreme value ends on x and meets w_x comparators. For $r = 3$, $n = 14$ and a scenario (y, z) of $(X_P/x, \epsilon)$, we call

$$w(x; y, z) = \max\bigl(0,\ \max\{35 - (k - |W|)\}\bigr)$$

the *trivial weight* of (y, z), where the inner maximum runs over the runs of three values of polarity ϵ that end on (x, y, z), in this order, and is omitted if there is no such run. By Lemma 2.3 and $S(11) = 35$ [8], it is a lower bound on the cost of the core $((X_P/x)/y)/z$.

2.4. **Harder's Huffman bound.** The bound of [8, Theorem 26] compares the cost of a set with the costs of its pruned sets. We use the following weaker form of it, which we call *Huffman's inequality*.

Lemma 2.4. *Let $X \subseteq \{0,1\}^n$ with $E = E_\epsilon(X) \neq \emptyset$, and let $v_x \leq s(X/x)$ for $x \in E$. Then $\sum_{x\in E} 2^{v_x} \leq 2^{s(X)}$. In particular, if $s(X) \leq L$, then $\sum_{x\in E} 2^{v_x - L} \leq 1$.*

Proof. For $\epsilon = 0$, let T sort X with $s(X)$ comparators. By Lemma 2.1(b), the paths Π_x with $x \in E$ end on the same output. The proof of [14, Lemma 7.1], applied to the inputs in E, shows that these paths form a rooted tree; contracting its vertices with one child gives a binary tree with leaves E, in which the depth $D(x)$ of x is at most $\delta(x)$. By Kraft's equality [14, Lemma 2.5], $\sum_{x\in E} 2^{-D(x)} = 1$. The network T/x sorts X/x, so $\delta(x) \leq s(X) - s(X/x) \leq s(X) - v_x$. Hence $1 \geq \sum_x 2^{-\delta(x)} \geq \sum_x 2^{v_x - s(X)}$. For $\epsilon = 1$ use the paths of the minimum. □

With the bounds (1), Lemma 2.4 yields the following inequality, which we call the *excess filter*. If a sequence P of k standard comparators on [14] satisfies $s(X_P) \leq 49 - k$, then

$$\sum_{x\in E_\epsilon(X_P)} 2^{w_P^\epsilon(x)} \;\leq\; 16 \qquad \text{for } \epsilon = 0, 1. \tag{2}$$

Indeed, $v_x = 45 - (k - w_x)$ is a lower bound on $s(X_P/x)$ by (1) and $S(13) = 45$ [14], and $2^{v_x - (49-k)} = 2^{w_x - 4}$.

2.5. **A finished extreme stays finished.** The cores in our instances are pruned three times for one polarity. When the other extreme is finished, it remains an extremal channel of every core. The equality in the next lemma is, up to complementation, the case of a single prunable channel in [8, Lemma 53]; we prove it directly for the pruned sets of [14].

Lemma 2.5. *Let P be a sequence of standard comparators on $[n]$ with $E_1(X_P) = \{b\}$. Let $1 \le r \le n-2$, let $Y_0 = X_P$, and let $Y_l = Y_{l-1}/x_l$ with $x_l \in E_0(Y_{l-1})$ for $l = 1, \dots, r$. Then $b \notin \{x_1, \dots, x_r\}$ and $b \in E_1(Y_r)$, and the set $C = Y_r/b$ on $n-r-1$ channels, pruned for polarity 1, satisfies $s(C) = s(Y_r)$. The same holds with the polarities exchanged.*

Proof. By Lemma 2.2 it suffices to reach $\mathbf{1} - e_b$ from $e_{x_1} + \cdots + e_{x_r}$ by up-steps inside X_P. The vector $e_{x_1} + \cdots + e_{x_r}$ lies in X_P, so it is $P(u)$ for an input u with r ones. Choose a channel j with $u_j = 0$. A chain of up-steps from u to $\mathbf{1} - e_j$ is mapped to a chain of up-steps inside X_P from $e_{x_1} + \cdots + e_{x_r}$ to $P(\mathbf{1} - e_j)$, which has a single 0, on a channel of $E_1(X_P) = \{b\}$. Hence $\mathbf{1} - e_b$ lies above $e_{x_1} + \cdots + e_{x_r}$, so that $b \notin \{x_1, \dots, x_r\}$, and its restriction lies in Y_r, so that $b \in E_1(Y_r)$.

The bound $s(C) \le s(Y_r)$ is Lemma 2.1(b). For the converse, every vector of X_P other than $\mathbf{1}$ has a 0 on b: if $u \ne \mathbf{1}$ and $u_j = 0$, then $P(u) \le P(\mathbf{1} - e_j) = \mathbf{1} - e_b$. By the description above, so does every vector of Y_r other than $\mathbf{1}$. The complement $\bar{Y}_r$ is well-behaved by Lemma 2.1(c), and all its vectors other than $\mathbf{0}$ have a 1 on b, so a maximal chain inside $\bar{Y}_r$ through such a vector passes through e_b. Hence $\bar{Y}_r/b$ consists of the restrictions of all vectors of $\bar{Y}_r$ other than $\mathbf{0}$, and C consists of the restrictions of the vectors of Y_r with a 0 on b. A network that sorts C, with b listed as the lowest output, therefore sorts Y_r, and $s(Y_r) \le s(C)$. □

2.6. **Weighted instances.** A *weighted instance* on n channels consists of a budget $L \ge 0$ and a list of scenarios (x, y, v) with $x \ne y$ and integer weights $v \ge 0$. It is *feasible* if some comparator network on the n channels has two distinct outputs o_M and o_m such that, for each listed (x, y, v), the run with the largest value on x and the second largest on y ends with these two values on o_M and o_m and meets at most $L - v$ comparators with one of the two values on an input lead [14, Definition 6.1]. For $X \subseteq \{0,1\}^n$, a polarity ϵ and weights v_t on scenarios (x_t, y_t) of (X, ϵ), let $I(X, \epsilon; v; L)$ be the instance with these scenarios and weights; for $\epsilon = 1$ it is read on the complement of X.

Proposition 2.6 ([14, Proposition 6.2])**.** *Let $X \subseteq \{0,1\}^n$, let ϵ be a polarity, and let $0 \le v_t \le s(Y_t)$ for scenarios (x_t, y_t) of (X, ϵ) with cores Y_t. If $I(X, \epsilon; v; L)$ is infeasible, then $s(X) \ge L + 1$.*

The proposition holds for every n, and for any choice of scenarios of (X, ϵ): dropping a scenario only removes a condition. The program `lb2cr` of [14, Section 8] decides weighted instances. It answers *infeasible* only if a search over the valid configurations of [14, Definition 7.9] finds none, and by [14, Proposition 8.9] such an answer is correct whenever all budgets are at most 56. That proposition places no condition on the number of channels, and the program accepts instances on up to 30 channels. It may also answer *feasible*, or *unknown* when a limit on its work is reached. Throughout, only the answer infeasible discards anything, so the limits of the programs affect only how many states are discarded.

3. Sorting networks on 14 inputs with 49 comparators

In this section and the next two, T denotes a sorting network on 14 inputs with 49 comparators. Since $S(14) \ge 49$ [14, Corollary 1.3], no sorting network on 14 inputs is smaller. When we speak of prefixes we list the comparators of T in an order that respects the dependencies, untangled into a sequence $g_1, \dots, g_{49}$ of standard comparators on [14], and we write X_j for the set produced by $g_1, \dots, g_j$. The paths of the maximum and of the minimum, and everything defined from them, are properties of the circuit and do not depend on this order.

Lemma 3.1. *Let $g_1, \dots, g_{49}$ be a sorting network on [14] consisting of standard comparators.*

(a) $s(X_j) = 49 - j$, and g_{j+1} is useful for X_j, for $0 \le j < 49$.

(b) Every prefix $g_1, \dots, g_j$ satisfies the excess filter (2).

(c) $\delta(x) \le 4$ *and* $\delta'(x) \le 4$ *for every input* x.

Proof. (a) The comparators $g_{j+1}, \dots, g_{49}$ sort X_j, and a network with fewer comparators that sorts X_j, placed after $g_1, \dots, g_j$, would sort $\{0,1\}^{14}$ with fewer than 49 comparators. If g_{j+1} were not useful, X_{j+1} would be X_j or arise from it by exchanging two channels, and Lemma 2.1(a) would give $s(X_{j+1}) = s(X_j)$. (b) follows from (a) and (2). (c) The set X_{49} consists of the 15 sorted vectors, its only channel extremal for polarity 0 is the top channel 13, and $w^0(13)$ is the largest $\delta(x)$; so (b) gives $2^{\delta(x)} \le 16$. The same argument applies to the minimum. □

Part (c) is the one-value bound $S(14) \ge S(13) + \lceil \log_2 14 \rceil$ read backwards: deleting Π_x from T leaves a sorting network on 13 inputs with $49 - \delta(x)$ comparators [8, Lemma 16].

3.1. **The two trees.** By [14, Lemma 3.1], the leads of the paths of the maximum form a rooted tree whose root is the output that receives the maximum and whose leaves are the 14 input leads. A comparator with two input leads in this tree is a *branch node*, and one with exactly one input lead in it is a *pass-through*. Contracting the vertices with one child gives a binary tree B whose leaves are the inputs and whose internal vertices are the 13 branch nodes. Each pass-through lies on the edge from a vertex u of B to its parent, or above the root, and $k(u)$ denotes the number of pass-throughs on that edge. If $D(x)$ is the depth of the input x in B, then $\delta(x)$ is $D(x)$ plus the number of pass-throughs on the path from x to the output of the maximum. Applied to the minimum, the same lemma gives a tree with a binary tree B', branch nodes and pass-throughs of the minimum, the numbers $k'(u)$ and the depths $D'(x)$.

Lemma 3.2. *The tree* B *has* 12 *leaves at depth* 4 *and two leaves* x_1, x_2 *at depth* 3. *The path from each input to the output of the maximum contains at most one pass-through, and none if the input has depth* 4. *Hence* $k(u) > 0$ *only for vertices* u *all of whose leaves are among* x_1, x_2. *The same holds for* B', *with two leaves* z_1, z_2 *at depth* 3.

Proof. By Lemma 3.1(c), $D(x) \le \delta(x) \le 4$. Kraft's equality [14, Lemma 2.5] gives $\sum_x 2^{-D(x)} = 1$, that is, $\sum_x (2^{4-D(x)} - 1) = 16 - 14 = 2$, where each term lies in $\{0, 1, 3, 7, 15\}$. Hence two terms equal 1 and the others 0. The number of pass-throughs on the path from x is $\delta(x) - D(x) \le 4 - D(x)$. □

Let F be the set of comparators of T whose two input leads are input leads of T. Since each input lead enters at most one comparator, F is a matching of the inputs.

Lemma 3.3. *A lead lies on a path of the maximum and on a path of the minimum if and only if it is an input lead. A comparator is a branch node of both trees if and only if it lies in* F.

Proof. A lead that is not an input lead is $\mathrm{hi}(g)$ or $\mathrm{lo}(g)$ for exactly one comparator g. The minimum leaves g on $\mathrm{lo}(g)$ and the maximum on $\mathrm{hi}(g)$, so $\mathrm{hi}(g)$ lies on no path of the minimum and $\mathrm{lo}(g)$ on no path of the maximum. Every input lead lies on both kinds of path. A comparator is a branch node of both trees exactly when both of its input leads lie on paths of both kinds. □

Lemma 3.4. *The matching* F *has six or seven comparators. Six of them join the twelve leaves of depth* 4 *of* B *in pairs of siblings, and six join the twelve leaves of depth* 4 *of* B' *in the same way. If* $|F| = 7$, *the seventh joins* x_1 *and* x_2, *and* z_1 *and* z_2 *are joined by a comparator of* F. *If* $|F| = 6$, *then* $\{x_1, x_2\} = \{z_1, z_2\}$ *is the set of the two inputs that* F *does not cover.*

Proof. The sibling of a leaf of depth 4 in B is a leaf of depth 4, since no leaf is deeper. By Lemma 3.2 neither edge carries a pass-through, so the parent is a comparator whose input leads are the two input leads, and it lies in F. This pairs the twelve leaves of depth 4 by six comparators of F, and $|F| \le 7$ because F is a matching. The same argument pairs the leaves of depth 4 of B' by six comparators of F. If $|F| = 7$, the comparator of F outside the first six joins

the remaining inputs x_1 and x_2, and the comparator of F outside the second six joins z_1 and z_2. If $|F| = 6$, both groups of six are all of F, so the inputs that F misses are x_1, x_2 and also z_1, z_2. □

In the *main case*, $|F| = 7$ and neither tree has a pass-through. The part of B above F is then a binary tree whose leaves are the seven comparators of F, with the comparator on x_1, x_2 at depth 2 and the other six at depth 3. Only one shape has these depths, and the same holds for B'. The $7 + 6 + 6 = 19$ comparators of the two trees depend only on each other, and after them the maximum and the minimum of every input lie on two fixed channels t and b. The remaining 30 comparators act on the other 12 channels. Up to permutation and complementation of the channels, the sets produced by these 19 comparators fall into 26 classes (Table 1). Outside the main case, the edges above x_1 and x_2, and above their parent if they are siblings, can carry pass-throughs, whose second input lead may come from the other tree or from a blue comparator, and the first layer may have only six comparators. The next section treats all cases together.

4. The starts

4.1. **The red prefix.** A lead of T is *red* if it lies on a path of the maximum or on a path of the minimum, and *blue* otherwise. A comparator is *red* if one of its input leads is red, and *blue* if both are blue. For a set $X \subseteq \{0,1\}^{14}$ we call the channels of $E_0(X) \cup E_1(X)$ *extremal* and the others *middle* channels. A sequence P of standard comparators on [14] is *complete* if $E_0(X_P)$ and $E_1(X_P)$ have one channel each.

Lemma 4.1. *Let $g_1, \dots, g_m$ be standard comparators on [14] and $j < m$. The lead on channel c after $g_1, \dots, g_j$ lies on a path of the maximum if and only if $c \in E_0(X_j)$, and on a path of the minimum if and only if $c \in E_1(X_j)$, where X_j is the set produced by $g_1, \dots, g_j$. In particular g_{j+1} is red if and only if it touches an extremal channel of X_j, and blue if and only if both of its channels are middle channels.*

Proof. On the input e_x the single 1 follows Π_x, so the lead on c lies on Π_x exactly when $g_1, \dots, g_j$ map e_x to e_c. Only the vectors e_x are mapped to vectors with a single 1. The minimum is treated in the same way with the inputs $\mathbf{1} - e_x$. □

Let G_0 be the set of red comparators of T that do not depend on any blue comparator.

Lemma 4.2. *(a) Every predecessor of a comparator of G_0 lies in G_0. Hence T can be listed, as in Section 3, as P_0 followed by R_0, where P_0 lists the comparators of G_0 and R_0 the others.*

(b) Every comparator of P_0 is red, and every red comparator of R_0 depends on a blue comparator of R_0.

Proof. (a) Let $g \in G_0$ and let h be a predecessor of g. If h were blue, g would depend on a blue comparator. So h is red, and since every comparator that h depends on is a predecessor of g, none of them is blue. Hence $h \in G_0$. An order that respects the dependencies can therefore list G_0 first. (b) The first claim is the definition. A red comparator outside G_0 depends on a blue comparator, and since G_0 contains no blue comparator, that comparator lies in R_0. □

In the main case of Section 3, G_0 consists of the 19 comparators of the two trees, and R_0 of the other 30, all of them blue.

4.2. **Abstract completions.** The red comparators that follow G_0 are constrained by the excess filter. We record only what this filter sees. Let P be a sequence of standard comparators on [14]. The *profile* of P is the multiset of triples $(\alpha_c, \beta_c, 0)$ over the extremal channels c of X_P, where $\alpha_c = 1 + w^0_P(c)$ if $c \in E_0(X_P)$ and $\alpha_c = 0$ otherwise, and $\beta_c = 1 + w^1_P(c)$ if $c \in E_1(X_P)$ and $\beta_c = 0$ otherwise.

An *abstract state* is a finite multiset of triples (α, β, τ) of integers with $\alpha, \beta \geq 0$, $(\alpha, \beta) \neq (0, 0)$ and $\tau \in \{0, 1\}$. It is *valid* if $\alpha \leq 5$ and $\beta \leq 5$ for all its triples, $\sum 2^{\alpha-1} \leq 16$ over the triples with $\alpha > 0$, and $\sum 2^{\beta-1} \leq 16$ over those with $\beta > 0$. It is *finished* if exactly one triple has $\alpha > 0$ and exactly one has $\beta > 0$. A *move* removes either one triple, or two triples at least one of which has $\tau = 1$. Let a and b be the largest first and second entries of the removed triples. The move adds the triple $(a + 1, 0, 1)$ if $a > 0$ and the triple $(0, b + 1, 1)$ if $b > 0$. An *abstract completion* of P is a nonempty sequence of moves that leads from the profile of P through valid states to a finished state.

A move with one triple models a red comparator between an extremal channel and a middle channel, and a move with two triples a red comparator between two extremal channels. The value $\tau = 1$ marks a channel whose lead was produced by a comparator after P.

Lemma 4.3. *Let Q followed by R be a sorting network on* [14] *with* 49 *standard comparators in which every red comparator of R depends on a blue comparator of R. If Q is not complete, then Q has an abstract completion with one move for each red comparator of R.*

Proof. We follow R one comparator at a time. Before each comparator, let the abstract state be the multiset of triples $(\alpha_c, \beta_c, \tau_c)$ over the extremal channels c of the current set, with α_c and β_c defined from the current prefix as in the profile, and with $\tau_c = 1$ exactly when the lead on c is an output of a comparator of R. At the start this is the profile of Q. A blue comparator acts on two middle channels and does not change the state, since its output leads are blue. A red comparator g on a middle channel and an extremal channel c carries the paths of the maximum through c on to $\mathrm{hi}(g)$ and those of the minimum on to $\mathrm{lo}(g)$, each with one more comparator. This is the move with the triple of c. A red comparator g on two extremal channels c and d is the move with their two triples: the longest path of the maximum that ends on $\mathrm{hi}(g)$ meets $\max(w^0(c), w^0(d)) + 1$ comparators, and likewise for the minimum. In this case g depends on a blue comparator h of R, so one of its input leads is an output of h or of a comparator that depends on h. That comparator lies in R, and the lead is red because both input leads of g are red. Hence one of the two triples has $\tau = 1$. By Lemma 3.1(b), every state reached is valid. The final set consists of the 15 sorted vectors and has one extremal channel for each polarity, so the final state is finished. The profile of Q is not finished, because Q is not complete, so at least one move occurs. □

4.3. **The search for starts.** The *red-prefix search* builds layers $L_0, L_1, \ldots$ of sequences of standard comparators on [14], with L_0 consisting of the empty sequence. For $P \in L_k$, every standard comparator g that touches an extremal channel of X_P, is useful for X_P, and is such that P followed by g satisfies the excess filter (2), gives the child Pg. The layer L_{k+1} contains one child from each similarity class of the sets of the children. A sequence P in the layers is *late* if it is not complete and has an abstract completion. A *start* is a sequence Q in the layers that is complete or late and has the least length among the complete and late sequences in the layers whose sets are similar to X_Q. Its *budget* is $\beta = 49 - |Q|$.

Sequences with similar sets are interchangeable in the following sense. Let P and Q be sequences of standard comparators, let σ be a permutation of the channels, possibly combined with complementation, that maps X_P onto X_Q, and let P followed by R be a sorting network. Apply σ to the channels of each comparator of R, exchange the roles of maximum and minimum if σ complements, and untangle. The result is a sequence R^σ of $|R|$ standard comparators such

that Q followed by R^σ is a sorting network; we say that R^σ is *transplanted* onto Q. The circuit of R^σ is that of R, and σ maps the extremal channels of each set produced along P followed by R to the extremal channels of the corresponding set along Q followed by R^σ, with the polarities exchanged if σ complements. So the red comparators, the blue comparators and the dependencies of R correspond to those of R^σ.

Theorem 4.4. *The red-prefix search yields* 5,271 *starts, of which* 4,394 *are complete and* 877 *are late.*[†] *If a sorting network on* 14 *inputs with* 49 *comparators exists, then some start Q with budget β satisfies:*

(a) if Q is complete, then $s(X_Q) \le \beta$;
(b) if Q is late, then there is a sequence R of β standard comparators such that Q followed by R is a sorting network and every red comparator of R depends on a blue comparator of R.

Proof. The counts come from the computation described below. Let T be listed as P_0 followed by R_0 as in Lemma 4.2, and let $m = |P_0|$. We show by induction on $j \le m$ that some $Q_j \in L_j$ has a set similar to the set produced by the first j comparators of P_0, and that the rest of T can be transplanted onto Q_j. For $j = 0$ take the empty sequence. Given Q_j, let T_j be Q_j followed by the transplanted rest, and let g be the comparator of T_j after Q_j. It corresponds to the $(j+1)$-th comparator of P_0, so it is red. By Lemma 3.1, applied to T_j, it is useful for X_{Q_j}, and $Q_j g$ satisfies the excess filter. Hence $Q_j g$ is a child of Q_j, and L_{j+1} contains a sequence Q_{j+1} whose set is similar to $X_{Q_j g}$. The rest of T_j after $Q_j g$ can be transplanted onto Q_{j+1}.

For $j = m$ let R_m be the rest of T transplanted onto Q_m. Every red comparator of R_m depends on a blue comparator of R_m, by Lemma 4.2(b). If Q_m is complete, then $s(X_{Q_m}) = s(X_{P_0}) = 49 - m$ by Lemma 3.1(a). Otherwise Lemma 4.3 gives an abstract completion of Q_m, so Q_m is late. Let Q be the start in the similarity class of X_{Q_m}. If $|Q| < m$, transplanting R_m onto Q would give a sorting network with fewer than 49 comparators. Hence $|Q| = m$, so $Q = Q_m$ because L_m holds one sequence of each class, and $\beta = 49 - m$. In the complete case $s(X_Q) = s(X_{Q_m}) = \beta$. In the late case, transplanting R_m onto Q gives the sequence R of (b). □

Table 1 lists the starts by length and budget, separated by the size of the first layer F of their prefix. Each complete start with budget 30 belongs to the main case. In the search, 4,402 complete sequences occur in the layers, because eight similarity classes appear at two lengths; the start keeps the shorter one. The search examines 299,796 sequences and 15,133,001 children, of which 13,828,697 violate the excess filter, and it meets 5,370,612 red comparators that are not useful.

Two programs computed the list of starts. The first runs the red-prefix search with its own canonical form for sets up to similarity. The second enumerates the configurations of the two trees that Lemmas 3.2 to 3.4 allow, together with a case analysis of the comparators that belong to both trees and of the second input leads of their pass-throughs, builds the comparators of G_0 for each configuration, and reduces the resulting 145,305 sets with Harder's canonical form [9]. Both programs give the same 5,271 classes with the same budgets, and so does a third construction of the 26 starts of the main case. Theorem 4.4 rests on the first program only.

Remark 4.5. The sorting network on 14 inputs with 51 comparators in Dobbelaere's list [5] has the structure of the main case: its first layer has seven comparators, its trees have no pass-throughs, its 19 red comparators depend on no blue comparator, and the set they produce is similar to the set of one of the 26 starts of the main case. Its other 32 comparators are blue. The script in the directory `dobbelaere_check` of [17] repeats this check.

		complete starts		late starts	
length $\|Q\|$	budget β	$\|F\| = 7$	$\|F\| = 6$	$\|F\| = 7$	$\|F\| = 6$
10	39				7
11	38				8
12	37				15
13	36				8
14	35				97
15	34			26	461
17	32			35	
18	31			220	
19	30	26			
20	29	295	54		
21	28	1,304	647		
22	27		2,068		
all		1,625	2,769	281	596

TABLE 1. The 5,271 starts by length and budget. F is the set of comparators of the start whose two input leads are input leads of the network (Lemma 3.4).

5. Macro-steps

After the red prefix of a late start, the remaining comparators of a hypothetical network can be put into a form that the search can follow.

Lemma 5.1. *Let Q followed by R be a sorting network on* [14] *with* 49 *standard comparators in which every red comparator of R depends on a blue comparator of R, and let $h_1, \ldots, h_p$ be the blue comparators of R in their order in R. Then the comparators of R can be listed as $M_1 M_2 \cdots M_p$, where the* macro-step *M_i consists of h_i followed by red comparators, each of which touches a channel touched by an earlier comparator of M_i, and Q followed by $M_1 \cdots M_p$ is a sorting network with the same comparators. In this network the two channels of h_i are middle channels of the set produced before h_i, and every comparator of M_i after h_i touches an extremal channel of the set produced before it.*

Proof. We place the comparators of R one at a time after Q. For $i = 1, \ldots, p$ we place h_i, and then, as long as some red comparator of R that is not yet placed has all its predecessors placed, we place one such comparator; these form the rest of M_i.

We claim that every comparator that precedes h_i in R is placed before h_i. Let g precede h_i in R. If g is blue, it is some h_l with $l < i$. If g is red, it depends on a blue comparator of R, which precedes g and is therefore some h_l with $l < i$; in particular $i \geq 2$. Its predecessors in Q are placed at the start, and its other predecessors precede g in R, so by induction on the position of g they are placed before h_i. When M_{i-1} ends, g therefore has all its predecessors placed, and since M_{i-1} ends only when no such red comparator remains, g has been placed. The same induction shows that every red comparator after h_p is placed in M_p. So every comparator of R is placed, each after all its predecessors, and the new order gives the same circuit.

Let g be a red comparator placed in M_i after h_i. Before h_i was placed, some predecessor of g was not placed: for $i = 1$ because g depends on a blue comparator, and for $i \geq 2$ because otherwise g would have been placed in M_{i-1}. Since every placed comparator has its predecessors placed, some comparator whose output lead is an input lead of g was not placed before h_i. It is therefore h_i or an earlier comparator of M_i, and the lead between it and g lies on a channel

that both of them touch. The statements about middle and extremal channels follow from Lemma 4.1. □

The red comparators of R give the moves of an abstract completion of Q (Lemma 4.3). For each of the 877 late starts, a search over all abstract completions finds none with more than 12 moves, so a hypothetical network adds at most 12 red comparators to a late start. The proof of Theorem 1.1 does not use this bound.

6. The searches and their cuts

6.1. **States and children.** A *state* is a sequence P of standard comparators on [14] with $|P| \le 49$. Its *budget* is $49 - |P|$. There are two kinds of search, one for complete starts and one for late starts, which differ in the children of a state.

- In the search for a complete start, the children of P are the states Pg, where g is a standard comparator that is useful for X_P.
- In the search for a late start, the children of P are the states $Phc_1 \cdots c_l$ with $l \ge 0$, where h is a standard comparator on two middle channels of X_P that is useful for X_P, and each c_i is a standard comparator that is useful for the set produced before it, touches an extremal channel of that set, and touches a channel touched by $h, c_1, \ldots, c_{i-1}$. Moreover, the states $Phc_1 \cdots c_j$ with $j < l$ satisfy the excess filter (2).

A state has finitely many children of either kind, by the following lemma.

Lemma 6.1. *Let P be a state, and let $c_1, \ldots, c_l$ be standard comparators such that each c_i touches an extremal channel of the set produced before it, and $Pc_1 \cdots c_j$ satisfies the excess filter for every $j < l$. Then $l \le 57$.*

Proof. For a sequence P' let $\Phi(P') = \sum_\epsilon \sum_{x \in E_\epsilon(X_{P'})} 2^{w^\epsilon_{P'}(x)}$ and $N(P') = |E_0(X_{P'})| + |E_1(X_{P'})|$. Channel 13 is extremal for polarity 0 and channel 0 for polarity 1 in every set produced by standard comparators, since these comparators fix the sorted vectors, so $\Phi(P') \ge N(P') \ge 2$. Let g be a comparator on channels c and d that touches an extremal channel. After g, the channel of $\mathrm{hi}(g)$ is extremal for polarity 0 exactly when c or d was, with w^0 one more than the larger of their values, and it is not extremal for polarity 1; symmetrically for $\mathrm{lo}(g)$ (Lemma 4.1). If for some polarity exactly one of c and d is extremal, Φ grows by at least 1. Otherwise, for some polarity both are extremal; then N drops by at least 1 and Φ does not drop. Since N never grows, at most 30 of the comparators $c_1, \ldots, c_{l-1}$ raise Φ, which stays at most 32 by the excess filter, and at most 26 lower N, which stays at least 2. Hence $l - 1 \le 56$. □

After a complete start, no comparator that touches the channel t of the maximum or the channel b of the minimum is useful, because every vector other than $\mathbf{0}$ has a 1 on t and every vector other than $\mathbf{1}$ has a 0 on b. So in the first kind of search all children act on the 12 middle channels, and every state is complete.

6.2. **The cuts.** A state P is *cut* if one of the tests below shows that $s(X_P) \ge 50 - |P|$, that is, that X_P cannot be sorted within the budget. Let $k = |P|$, $L = 49 - k$, $X = X_P$, let ϵ be a polarity, and let $E = E_\epsilon(X)$ and $E' = E_{1-\epsilon}(X)$. For $x \in E$ and a scenario (y, z) of $(X/x, \epsilon)$, the *core* $Z_x(y, z) = ((X/x)/y)/z$ is a set on 11 channels. A *core bound* for it is a lower bound u, certified as in Section 9, on one of two costs:

- if $E' = \{b\}$, on the cost $s(Z_x(y, z)/b)$ of the 10-channel set obtained by pruning b for polarity $1 - \epsilon$, which equals $s(Z_x(y, z))$ by Lemma 2.5; the searches use either values computed with a modified version of Harder's program [9] (exact values of these sets or lower bounds on them) or lower bounds from the weighted two-value bound on 10 channels;

- on $s(Z_x(y,z))$ itself; the searches obtain these bounds from the weighted two-value bound on 11 channels.

The *weight* of (y,z) is $v_x(y,z) = \max(w', u)$, where u is a core bound or 0, and w' is the trivial weight $w(x;y,z)$ or, if $E' = \{b\}$, the trivial weight of a core Z'' at another state P'' of the same search whose opposite extreme is also finished, $E_{1-\epsilon''}(X_{P''}) = \{b''\}$, such that Z''/b'' is similar to $Z_x(y,z)/b$. By Lemmas 2.1(a), 2.3 and 2.5, every weight satisfies $v_x(y,z) \le s(Z_x(y,z))$. For a budget L' we write $I_x(L')$ for the weighted instance $I(X/x, \epsilon; v_x; L')$ on 13 channels, with all or some of the scenarios of $(X/x, \epsilon)$, as Section 2 allows.

(C1) *Excess:* $\sum_{x\in E} 2^{w_P^\epsilon(x)} > 16$.
(C2) *Two values after the maximum:* $E = \{t\}$ and $I_t(L)$ is infeasible.
(C3) *Huffman:* there are numbers v_x for $x \in E$ with $\sum_{x\in E} 2^{v_x - L} > 1$, such that each v_x equals $45 - (k - w_P^\epsilon(x))$, or exceeds it and $I_x(v_x - 1)$ is infeasible.
(C4) *Two values on the whole set:* $I(X, \epsilon; h; L)$ is infeasible, where the instance has the scenarios (x,y) of (X,ϵ) such that $(X/x)/y$ has an extremal channel for ϵ, with the weights

$$h(x,y) = \Big\lceil \log_2 \sum_{z\in E_\epsilon((X/x)/y)} 2^{v_x(y,z)} \Big\rceil .$$

In (C2) and (C3) the weights may all be trivial. In the certificates the tests (C2), (C3) and (C4) are called MID, HUFF and TOP2, and (C3) is called HUFF11 when $|E'| \ge 2$, which is the case where the core bounds are of the second kind.

Lemma 6.2. *If one of the tests* (C1) *to* (C4) *holds for P and a polarity, then $s(X_P) \ge 50 - |P|$.*

Proof. Suppose that $s(X) \le L$. Then (C1) fails by (2). If $E = \{t\}$, then $s(X/t) \le s(X) \le L$ by Lemma 2.1(b), and since the weights are lower bounds on the costs of the cores, Proposition 2.6 shows that $I_t(L)$ is feasible; so (C2) fails. In (C3), each v_x is a lower bound on $s(X/x)$: either by (1) and $S(13) = 45$, or because $I_x(v_x - 1)$ is infeasible and Proposition 2.6 gives $s(X/x) \ge v_x$. Lemma 2.4 then gives $\sum_x 2^{v_x - L} \le 1$, so (C3) fails. In (C4), Lemma 2.4 applied to the set $(X/x)/y$, whose pruned sets $((X/x)/y)/z$ cost at least $v_x(y,z)$, gives $\sum_z 2^{v_x(y,z)} \le 2^{s((X/x)/y)}$, and since the exponent on the right is an integer, $h(x,y) \le s((X/x)/y)$. The sets $(X/x)/y$ are the cores of the scenarios of (X,ϵ), so Proposition 2.6 on 14 channels shows that the instance of (C4) is feasible. □

6.3. **Extinct searches.** A search keeps a finite set of states. The following theorem states what it must satisfy to refute its start.

Theorem 6.3. *Let Q be a start with budget β, and let $\mathcal{K}$ be a finite set of states such that*

(i) some state in $\mathcal{K}$ has at most $|Q|$ comparators and a set similar to X_Q;
(ii) $|X_P| > 15$ for every $P \in \mathcal{K}$;
(iii) for every $P \in \mathcal{K}$ and every child P' of P, in the search of the kind of Q, some state P'' with $|P''| \le |P'|$ and a set similar to $X_{P'}$ is cut or lies in $\mathcal{K}$.

Then Q satisfies neither (a) *nor* (b) *of Theorem 4.4.*

Proof. A set produced by a network has at least one vector of each weight $0, \dots, 14$, and it can be sorted without comparators exactly when it has 15 vectors. By Lemma 2.1(a), similar sets have the same cost.

Let Q be complete and suppose that $s(X_Q) = s \le \beta$. Let U be a network with s comparators that sorts X_Q, and untangle Q followed by U into Q followed by standard comparators $g_1, \dots, g_s$. Let X_j be the set produced by Q followed by $g_1, \dots, g_j$. As in Lemma 3.1(a), $s(X_j) = s - j$ and g_{j+1} is useful for X_j. We show by induction on j that some $P \in \mathcal{K}$ with $|P| \le |Q| + j$ has a set similar to X_j. For $j = 0$ this is (i). Let P and a similarity σ from X_j onto X_P be given.

Apply σ to g_{j+1}, exchanging maximum and minimum if σ complements. The result is useful for X_P, and so is its standard version g' on the same two channels, which gives the child $P' = Pg'$. Its set is similar to X_{j+1}, because a comparator that is not standard acts as the standard one followed by the exchange of its two channels. Now $s(X_{P'}) = s - j - 1 \le \beta - j - 1 \le 49 - |P'|$, so by Lemma 6.2 no state with at most $|P'|$ comparators and a set similar to $X_{P'}$ is cut. By (iii) some $P'' \in \mathcal{K}$ with $|P''| \le |P'| \le |Q| + j + 1$ has a set similar to X_{j+1}. For $j = s$, the set X_s has 15 vectors, and so does the set of the corresponding state of $\mathcal{K}$, contradicting (ii).

Let Q be late and suppose that R is as in Theorem 4.4(b). Order R into macro-steps $M_1 \cdots M_p$ by Lemma 5.1, and let X_i be the set produced by Q followed by $M_1 \cdots M_i$ and k_i the length of this sequence. By Lemma 3.1(a), applied to the reordered network, $s(X_i) = 49 - k_i$, and every comparator is useful for the set before it. We show by induction on i that some $P \in \mathcal{K}$ with $|P| \le k_i$ has a set similar to X_i; for $i = 0$ this is (i). Given such P and a similarity σ, apply σ to the comparators of M_{i+1}, exchanging maximum and minimum if σ complements, and untangle. This gives a child P' of P, because σ maps middle channels to middle channels and extremal channels to extremal channels, and it preserves usefulness and the channels touched. Untangling preserves them too: if a comparator c of M_{i+1} touches a channel a and d is the last earlier comparator of M_{i+1} that touches a, the exchanges of labels made between d and c do not move the image of a, except the one at d itself, which keeps it on a channel of d. The states between P and P' satisfy the excess filter: each has a set similar to a set produced along the reordered network, whose cost is 49 minus the length there and hence at most 49 minus its own length, and (2) applies. The set of P' is similar to X_{i+1}, and $|P'| = |P| + |M_{i+1}| \le k_{i+1}$, so $s(X_{P'}) = 49 - k_{i+1} \le 49 - |P'|$, and as before no state with at most $|P'|$ comparators and a set similar to $X_{P'}$ is cut. By (iii), some $P'' \in \mathcal{K}$ with $|P''| \le k_{i+1}$ has a set similar to X_{i+1}. For $i = p$ the set X_p has 15 vectors, contradicting (ii) as before. □

Since $S(14) \ge 49$, condition (ii) holds for every state with at most 48 comparators, because a state whose set has 15 vectors is a sorting network. Our searches keep only states with at most 45 comparators. The replay of Section 10 checks (ii) directly as well.

Proof of Theorem 1.1. By [14, Corollary 1.3], $S(14) \ge 49$. Suppose that a sorting network on 14 inputs with 49 comparators exists. Theorem 4.4 gives a start Q that satisfies (a) or (b). For every start, the certificates of Sections 7 and 8 provide a set $\mathcal{K}$ with the properties (i) to (iii), as the replay of Section 10 and the replays with certified bounds of Section 9 confirm. This contradicts Theorem 6.3. Hence $S(14) \ge 50$. □

6.4. **The certificates.** The search for a start proceeds in layers: layer l consists of the states kept after l comparators in the search for a complete start, and after l macro-steps in the search for a late start. The children of the states of a layer are tested in turn. A child is dropped when a state with a similar set and the same or a smaller length was kept or cut before; otherwise it is tested, and it is kept unless it is cut. So the set $\mathcal{K}$ of all kept states satisfies (iii) by construction. For each start, the certificate lists the kept states of each layer with their lengths. It contains one record for each child cut by (C4), and for each child cut by (C2) or (C3) with at least one weight above the trivial weight of the state itself, a core bound or a trivial weight taken from another state. The record gives the polarity, the budget, the test, the numbers v_x in (C3), and for each scenario of each instance its channels, its trivial weight, the weight used and the core as a bitmap. A weight above the trivial weight is a *claim* $s(Y) \ge v$ about the core Y, or about the 10-channel set obtained from it by pruning the finished extreme. The other cuts, by (C1) and by (C2) or (C3) with the trivial weights of the state itself, are not recorded: the replay finds them again. The search ends when a layer is empty.

7. The complete starts

After a complete start every state P of the search has $E_0(X_P) = \{t\}$ and $E_1(X_P) = \{b\}$, because the children act on the middle channels only. The test (C2) therefore applies to both polarities, and its core bounds can be exact values of 10-channel sets. The test (C1) cannot cut such a state, since the paths that end on t and b are those of the start. The test (C3) cuts only states that (C2) cuts, and (C4) is not used.

For each child, the programs first decide the two instances of (C2) with trivial weights. If neither is infeasible, they raise weights to lower bounds on the costs of the 10-channel cores: exact values or bounds proved by a modified version of Harder's program [9], or the weighted two-value bound on 10 channels with exact values of 8-channel sets. They decide the instances again and repeat while the instances stay feasible and the bounds improve. A computation that exceeds a time limit leaves its weight unchanged, and the certificate records the weight used. The programs decide the instances with `lb2cr` or with solvers of their own, and the replay decides every recorded instance again with `lb2cr`. As in [14, Section 9], a weight may also be the trivial weight of a core at another state of the same search, under the similarity condition of Section 6.2; such a weight exceeds the trivial weight of the state itself. The replay of Section 10 computes only the trivial weights of each state and counts any larger weight as a claim, and these claims are certified as in Section 9. Table 2 summarizes the searches.

budget	starts	last layer	children	classes	cut records	claims
27	2,068	1	129,125	113,602	0	0
28	1,951	5	124,258	107,681	6,851	60,335
29	349	6	28,234	21,742	7,345	44,481
30	26	7	12,889	8,108	4,559	17,994

Table 2. The searches for the complete starts, as regenerated by the replay of Section 10. "Last layer" is the largest number of comparators after the start at which some state of the group is kept, plus one. "Children" counts the children of all kept states, and "classes" their similarity classes in each search. "Claims" counts the distinct pairs of a set and a weight above its trivial weight in the cut records; each set is a core with its finished extreme pruned, on 10 channels.

All 2,068 starts with budget 27 are refuted in the first layer by trivial weights alone. In the main case the searches end after one to seven layers, with at most 14 states kept in a layer.

8. The late starts

The 877 late starts have budgets 31 to 39. Their searches proceed by macro-steps. While both extremes have several extremal channels, only the tests (C1), (C3) and (C4) apply. The trivial bound $45 - (k - w_x)$ of a channel that no comparator has touched yet is far from tight, and (C3) gains most from raising the bounds of the channels with large w_x, which carry large terms in the Huffman sum. In (C3) with $|E'| \geq 2$ the core bounds are lower bounds on the costs of 11-channel sets obtained by the weighted two-value bound on 11 channels [14, Section 9], with exact values of 9-channel sets or, where a limit on the work was reached, lower bounds on them. Many states that survive (C3) satisfy the Huffman inequality with equality. Then a network completing them would have no pass-through on that side and would meet every bound exactly, and (C4) exploits this. Once one extreme is finished, (C3) for the other extreme uses exact values of 10-channel sets or lower bounds on them, and (C3) for the finished extreme, with its single channel, uses bounds on 11-channel cores. Once both are finished, (C2) applies as for the complete starts.

The search programs follow macro-steps with at most 16 comparators after the blue one; the replay of Section 10 follows all of them, and Lemma 6.1 bounds their length. Table 3 gives the corresponding data for the late starts.

budget	starts	last layer	kept states	records (C2)/(C3)	records (C4)
31	220	7	2,742	96,299	0
32	35	11	16,582	327,231	0
34	487	26	4,059,138	2,180,324	12,998
35	97	11	30,520	1,000,766	1,197
36–39	38	10	41,256	1,159,315	17,385

TABLE 3. The searches for the late starts, as regenerated by the replay of Section 10. "Last layer" is the largest number of macro-steps after which some state of the group is kept, plus one. "Kept states" are summed over the layers and the starts.

In 485 of the 877 searches no record is needed: every child is cut by (C1), cut by (C2) or (C3) with trivial weights, or matched with a kept state. Among them are 450 of the 461 starts with budget 34 whose first layer has six comparators. Their searches are the longest, with up to 26 macro-steps and up to 46,003 states kept in a layer. The other late searches end after at most 11 macro-steps, with at most 1,812 states kept in a layer, and 96 of them use (C4).

9. Certifying the costs of cores

The claims of the cut records are lower bounds $s(Y) \geq v$ on sets Y with 10 or 11 channels. Each claim that the replays use is certified by a derivation in Harder's formal system for certificates [8, Section 7.1], extended by one rule. Each of the four rules below states a lower bound for a set from lower bounds for smaller or similar sets:

(R1) if Y has two distinct vectors of the same weight, then $s(Y) \geq 1$;
(R2) the weighted two-value bound (Proposition 2.6): if $I(Y, \epsilon; u; v-1)$ is infeasible, with all scenarios of (Y, ϵ) and weights u that are proved lower bounds for their cores, then $s(Y) \geq v$;
(R3) Huffman's inequality (Lemma 2.4): if $E_\epsilon(Y) \neq \emptyset$ and $B_c \leq s(Y/c)$ for all $c \in E_\epsilon(Y)$, then $s(Y) \geq \lceil \log_2 \sum_c 2^{B_c} \rceil$;
(R4) the successor rule: if Y has two distinct vectors of the same weight, and every standard comparator maps Y to Y, to the image of Y under the exchange of its two channels, or to a set Y' with $s(Y') \geq v-1$, then $s(Y) \geq v$;

As Harder's system allows, derivations may also use Lemma 2.1(a) for similar sets and subsets. Rules (R1) and (R4) are its pigeonhole and successor rules [8, Lemmas 9 and 12], in the form of the lemmas `unsorted_bound` and `suc_bound` of the verified checker of [9] (`checker/verified/Sorting_Network.thy`). Rule (R3) is a weaker form of the Huffman rule of that system, and (R2) is the rule that we add. As in [8, Lemma 12], the successor rule holds because an optimal network that sorts Y, once untangled, begins with a standard comparator that is useful for Y and is followed by a network that sorts its image. A derivation that uses only (R2) is a single instance on 10 or 11 channels whose weights are costs of cores with 8 or 9 channels.

At the bottom of the derivations are *facts*, lower bounds $s(K) \geq v$ for sets K with 7, 8 or 9 channels, mostly exact values. They were computed by a modified version of Harder's program [9], changed to run faster and to report a lower bound that its search has proved when it reaches a limit on its work. The original program [9], run separately, confirmed all facts on 7

and 8 channels and about half of those on 9 channels; each other fact follows from a step of the certificate [7] for $S(11) \geq 35$ or from an entry of the table of [15], which holds 3,624,738 values on 9 channels. The step or entry used for a fact on K is a pair (Y, v) of a 9-channel set Y that a similarity maps into K and a number v. Following the verified checker of [9] (the definition `pls_bound` in `checker/verified/Sorting_Network.thy`), we say that a network *works* on Y if any two vectors of Y with the same weight give the same output. For every entry of the table, the theorem `S2KF.Final.t9ValidK_uncond` of [15] states that every network that works on Y has at least v comparators. A step of the certificate states the same, and every step is checked by Lean's kernel [14, Section 14.2]. A network that sorts Y works on Y, so $s(Y) \geq v$, and $s(K) \geq v$ by Lemma 2.1(a). A separate program checks the maps.

The certificates were produced by programs that we do not trust, and checked by a program that shares no code with them: it recomputes every pruned set, scenario, extremal channel and map itself, and it calls `lb2cr` for every infeasible instance. A replay of the cut records then uses only certified bounds: each claimed weight is replaced by the largest bound that the checker proved for the core or for a similar set and that does not exceed the claimed weight, but not by less than the trivial weight, and every recorded instance must remain infeasible. For the complete starts this replay is a second program that replays the whole search with certified bounds. It also uses trivial weights of cores at other states, under the similarity condition of Section 6.2; without them it would not refute 44 of the complete starts. For the late starts it is the replay of Section 10 in a mode that replaces each claim on a 10-channel set by its certified bound and uses each claim on an 11-channel set as claimed; every claim on an 11-channel set was certified at its full value. For the complete starts, and for the claims of the late starts on 10-channel sets, only the claims needed for the replays were certified at their full value. Both replays with certified bounds accept. The second program refutes all 4,394 complete starts, and the replay of Section 10 in its certified mode refutes the 392 late starts that have claims; for the other late starts the two modes coincide. Table 4 gives the numbers of claims, certificate lines and facts.

item	number
claims on 10-channel sets, certified	2,273,305
claims on 11-channel sets, certified	43,863,506
certificate lines accepted by the checker	1,775,084,961
facts on 7 channels	12,170,156
facts on 8 channels	820,274,780
facts on 9 channels	10,656,747
confirmed by Harder's original program	5,526,091
derived from Harder's certificate	4,876,800
derived from the table of [15]	253,856

TABLE 4. Certification of the claims of the cut records. The lines of the claim certificates are counted without duplicates within each certification run and summed over the runs.

10. Replay and trusted components

10.1. **Replay.** The certificates of all searches were checked by a program written without the code of the searches, from the format of the certificates, the specification of the moves and tests, and the mathematics of Sections 5 and 6. It has its own generator of children, which follows the definitions of Section 6, its own canonical form for sets up to similarity, and its own computation

of extremal channels, pruned sets, scenarios and trivial weights. It shares only `lb2cr` with the searches.

For each start and each layer, the replay regenerates all children of the kept states that satisfy the excess filter; the others are cut by (C1). It keeps the least length for each similarity class of children. Each class must be kept in this or an earlier layer at a length not exceeding its own, or be cut for some state of the class with no greater length in this or an earlier layer. Such a cut is a valid record, or a cut by (C1), or by (C2) or (C3) with the trivial weights of that state, which the replay recomputes.

For each record, the replay recomputes the set of the child, its extremal channels and its scenarios with their trivial weights. It checks that each listed scenario is a scenario, that every weight is at least its trivial weight and that every listed core is the core of its scenario, pruned at the opposite extreme if that extreme is finished, and it decides every instance with `lb2cr`. In (C3) it recomputes the Huffman sum. In (C4) it requires every scenario to be listed and recomputes the weights $h(x, y)$. The answer unknown counts as a failure. The replay also checks that every kept state has more than 15 vectors. A separate pass over all certificates checks that each start has exactly one certificate, and that the first state of the certificate has the length of the start and a set similar to the start's set. The replay accepts the certificates of all 5,271 starts, and no kept state is sorted. Every budget that the replay passes to `lb2cr` is at most 37, every budget that the checker of the claim certificates passes is at most 29, and every budget that the replay of the complete starts with certified bounds passes is at most 30; all lie in the range of [14, Proposition 8.9]. For the tests (C2) and (C4) this also follows from the construction: a child of length k has budget $49 - k \leq 38$, because every start has at least 10 comparators.

10.2. **Trusted components.** The proof of Theorem 1.1 rests on Sections 2 to 6, on the computation of the starts, on the replay and on the certification of the claims. These rely on the following components, which they do not check.

(1) *Harder's original program* [9], which recomputed every fact that is not derived from the certificate [7] or from the table of [15]. Harder proved that the algorithm of this program computes the cost of every well-behaved set [8, Section 5 and Corollary 56], and the sets we pass to it are well-behaved (Lemma 2.1(c)). The implementation is not verified, and unlike the bound $S(11) \geq 35$ (Section 1), the facts that we take from the program have no certificate.
(2) *The program* `lb2cr`, for each answer infeasible. Its correctness is proved in [14, Sections 7 and 8] for all budgets that occur here, as noted above.
(3) *The lower bounds* $S(11) \geq 35$ *and* $S(13) \geq 45$, which enter through the trivial weights, the excess filter and the bounds $45 - (k - w_x)$ in (C3), and $S(14) \geq 49$, which follows from $S(13) \geq 45$ by the one-value bound. The first two are proved in Lean [14]; only these directions of the known values are used.
(4) *Harder's certificate for* $S(11) \geq 35$ *and the table of values on* 9 *channels of* [15], for the facts derived from them. Both are checked by Lean's kernel [14, 15].

Beyond these, we trust the program that computes the starts, the replay of Section 10 and its separate pass over the certificates, the checker of the claim certificates, the program that checks the maps from the certificate and from the table, the replays with certified bounds and the scripts that match their claims to the certificates, the compilers and the hardware. We do not trust the search programs, their instance solvers, the programs that compute exact values of 10-channel sets and bounds on 11-channel sets, or the producers of the certificates. The replay of the complete starts with certified bounds belongs to the same code base as the programs that produce the claims and their certificates. Table 5 summarizes the checks, Section 11 states what the proofs in Lean trust, and Appendix B lists tests of the methods and programs.

check	scope	outcome
replay with the claimed weights (Section 10)	the certificates of all 5,271 searches	all accepted
checker of the claim certificates (Section 9)	2,273,305 claims on 10-channel and 43,863,506 on 11-channel sets	1,775,084,961 lines accepted, none rejected
facts (Section 9)	843,101,683 lower bounds on sets with 7 to 9 channels	each confirmed by Harder's original program or derived from Harder's certificate or the table of [15]
replays with certified bounds (Section 9)	the 4,394 complete starts and the 392 late starts with claims	all refuted
Lean, `S14_ge_50_N` (Section 11)	the reduction and all searches	$S(14) \geq 50$ from the certified tables; part runs as compiled code
Lean, `wedK` (Section 11)	the reduction and the searches for the complete starts	proved by the kernel alone, from $S(11) \geq 35$, $S(13) \geq 45$, the 10-channel table and the refutation of the late starts

TABLE 5. The checks of the proof of Theorem 1.1 and their outcomes.

11. Proofs in Lean

We formalized the reduction of Sections 3 to 5 and the searches for all starts in Lean 4 [11] with Mathlib [12]. The definitions of networks and of sorting are those of [14, Section 14.1]; the file that contains them is byte-identical to the one used there. A network on n channels is a list of comparators (i, j) with $i < j$ that put the minimum on channel i, and it sorts if its output is nondecreasing on every input vector of natural numbers. There are two main theorems:

```
theorem Sortnet.S14.S14_ge_50_N : coreTableC.Valid ->
    (forall i in Wed.done, (Wed.TLf i).Valid) -> S14Statement
theorem Sortnet.S14.wedK : WedK coreTableC
```

where `WedK coreTableC` unfolds to

```
S11Statement -> S13Statement -> coreTableC.Valid ->
  (forall s in starts, s.complete = false -> StartRefuted s) -> S14Statement
```

Here `S14Statement` states that every sorting network on 14 channels has at least 50 comparators, and `S13Statement` and `S11Statement` state the bounds 45 and 35 for 13 and 11 channels. The proof of `S14_ge_50_N` applies the theorems that prove these two bounds in the published Lean proof of [14], which runs some of its computations as compiled code [16]; that proof was built from its published files together with ours, and Lean checks that its two statements are the same expressions as `S13Statement` and `S11Statement`. The list `starts` contains the 5,271 starts, and `Wed.done` lists the indices of the 877 late starts. For a complete start, `StartRefuted` says that every list of comparators that works on its set, in the sense of Section 9, is longer than the budget; for a late start it says the same of the completions in macro-step form in which each macro-step has at most 16 comparators after the blue one. The table `coreTableC` has 79,144 entries, of which 74,705 are pairs (Y, v) with a 10-channel set Y; the other entries claim nothing. `coreTableC.Valid` says that for each of these pairs every list of comparators that works on Y has at least v comparators. For a late start i, `Wed.TLf i` is the table of the bounds on 10- and 11-channel sets that its certificate uses, and `Valid` has the same meaning; 392 late starts have a nonempty table, with 45,833,317 pairs in total. For the sets of these tables the condition is equivalent to $s(Y) \geq v$, by an argument that is not part of the formalization. A network that sorts a set works on it. Conversely, let a network work on such a set Y. All vectors of Y of

weight w have a common output o_w. The set Y is well-behaved by Lemma 2.1(c), so it contains a maximal chain of up-steps, and the network maps up-steps to up-steps. Hence $o_0, o_1, \ldots$ is a maximal chain of up-steps, and listing the channels in the reverse of the order in which they become 1 makes every o_w nondecreasing, so the network sorts Y.

The proofs have three parts. The first is the theorem `Sortnet.S14.tierA`, which derives `S14Statement` from `S13Statement` and the refutation of every start. It formalizes Lemmas 3.1, 4.2, 4.3 and 5.1 and Theorem 4.4. In place of the computation in Theorem 4.4, the kernel checks a certificate of the red-prefix search. It lists kept states of depths 0 to 22. Every child that passes the filters, that is, every child by a red and useful comparator that satisfies the excess filter, is mapped by a similarity onto a kept state of the next layer, and no child passes at depth 22. Every kept state has a set similar to that of a start with at most as many comparators, or its abstract state lies in a set of abstract states that is closed under moves to valid states and contains no finished state. The starts to which the certificate refers are checked to be those of the list `starts`. Unlike the proof of Theorem 1.1, the formalization uses a bound on the abstract completions, weaker than the one in Section 5: the kernel checks that no late start has an abstract completion with more than 16 moves. Hence a completion of a late start adds at most 16 red comparators, and it has the macro-step form of the statement.

The second part refutes the complete starts with their certificates. Its cuts by (C2) are decided by the search over configurations whose soundness is proved in [14], and every weight above a trivial weight is an entry of `coreTableC` or one of the 4,396 trivial weights of cores at other states that the certificates use; the kernel checks these. There are two versions of this part. In the one used by `S14_ge_50_N`, the search runs as compiled code for 1,985 of the largest instances. In the one used by `wedK`, the kernel checks every computation.

The third part refutes the late starts. Their certificates are converted into a form that a checker written in Lean reads. For every macro-step from a kept state it gives a similarity map onto a kept state of any layer with at most as many comparators, or a cut. A cut is decided with trivial weights, or it is a record that is checked once, at its own state, and applies to every state with a similar set and at least as many comparators. The soundness of the checker is proved in Lean: it follows every macro-step with at most 16 comparators after the blue one whose states pass the excess filter, and it decides every recorded instance with the same search over configurations, whose soundness is proved in [14] for at most 13 channels and here for 14 channels, as (C4) requires. The weights above trivial weights are the entries of `Wed.TLf i`, which are certified claims for the same set or a similar set. These checks run as compiled code in 39,391 pieces, and only `S14_ge_50_N` uses them; `wedK` keeps the refutations of the late starts as a hypothesis.

Besides `propext`, `Classical.choice` and `Quot.sound`, `S14_ge_50_N` depends on 43,100 axioms, one for each computation that runs as compiled code: 1,985 in the second part, 39,391 in the third and 1,724 in the published proof of $S(13) \geq 45$. They trust Lean's compiler, the C compiler and the GMP library in addition to Lean's kernel. The theorem `wedK` depends only on the three standard axioms. Outside the build, `leanchecker`, which replays the 160,853 constants on which `wedK` depends through Lean's kernel, and `lean4lean`, an independent implementation of Lean's type theory, accept all of them; the independent checker `nanoda` accepts all of them except 46 constants of one module, on which it gives no verdict. Appendix A gives the sizes and processor times of the builds.

The tables are defined in Lean. Of the 74,705 pairs of `coreTableC`, 68,821 are claims certified as in Section 9 and 5,884 are certified claims on similar sets. The trivial weights of the second part are not part of the hypothesis, because the kernel checks them. The Lean project, its data and the records of its builds are archived as [18].

12. Consequences for larger networks

Proof of Corollary 1.2. By Van Voorhis's two-value bound [20], $S(N) \geq S(N-2) + P(2,N)$, where $P(2,N)$ is the least worst-case number of comparators met by the two largest values in a network that delivers them to fixed outputs. Mericanii found the published proof of Van Voorhis's lower bound on $P(2,N)$ invalid [13]; for the two-value bound we use the proof of [14, Lemma 2.3]. By [14, Theorem 1.2], $P(2,16) = \lceil \log_2 F(16) \rceil = 9$. With Theorem 1.1 this gives $S(16) \geq 50 + 9 = 59$. □

The lower bound $P(2,16) \geq 9$ used here follows from $P(2,13) \geq 9$, which is proved in Lean [14], and the monotonicity of $P(2,N)$ [14, Lemma 2.4]. For the other values $15 \leq N \leq 32$, we start the recursion of [14, Section 15], which takes the larger of the one-value bound and the two-value bound, from $S(13) \geq 45$ and $S(14) \geq 50$. For $N = 15$ it gives $\max(50+4,\ 45+9) = 54$, as before. For $N = 17$ it gives $\max(59+5,\ 54+10) = 64$, and for $N = 18$ it gives $\max(64+5,\ 59+10) = 69$; these are the values of [14, Corollary 1.3]. From $N = 19$ on, both terms involve only unchanged values. So among $15 \leq N \leq 32$ only the bound for $N = 16$ changes. Table 6 lists the resulting lower bounds.

N	13	14	15	16	17	18	19	20	21	22
lower bound on $S(N)$	45	**50**	54	**59**	64	69	74	79	85	90
N	23	24	25	26	27	28	29	30	31	32
lower bound on $S(N)$	96	101	107	112	118	124	130	136	142	148

Table 6. Lower bounds on $S(N)$ from $S(13) = 45$, Theorem 1.1, the one-value bound and the two-value bound with $P(2,N) = \lceil \log_2 F(N) \rceil$ [14, Theorem 1.2]. The bold entries exceed those of [14, Corollary 1.3].

13. Concluding remarks

Theorem 1.1 rules out sorting networks on 14 inputs with 49 comparators, and Corollary 1.2 those on 16 inputs with 58. Whether networks with 50 comparators on 14 inputs or with 59 on 16 inputs exist remains open; the smallest known networks have 51 and 60 comparators.

Appendix A. Computations

Table 7 lists the computations in thread-hours; a thread-hour is one hardware thread for one hour of wall-clock time. The machines had 32 to 384 hardware threads. The thread-hours of a machine are its number of hardware threads times the time between its start and its stop, so they include idle time and runs that produced no certificate, and they are upper bounds. Where a machine served several rows, its hours are split between them by time windows, so the split is approximate. Where it was measured, the table also gives the busy thread-hours: for the searches, the processor time that the programs recorded for their threads; for the certification, the per-minute processor utilization of the machines; and for the proofs in Lean, the processor time of the builds and of the check outside the build.

The proofs in Lean consist of 43,983 modules, whose compilation took 1,274.5 hours of processor time: 5.2 hours for the first part of Section 11, 109.8 and 191.0 hours for the two versions of the second, and 968.4 hours for the third. The check of `wedK` outside the build took 993.8 hours, and the build of the published proof of $S(13) \geq 45$ for `S14_ge_50_N` took 1,098.6 hours.

computation	thread-hours	busy thread-hours
complete starts with budgets 27 to 29	1,443	466
complete starts with budget 30	2,807	
late starts with budget 34 and $\lvert F\rvert = 6$	1,333	917
late starts with $\lvert F\rvert = 7$	18,580	
late starts with budgets 35 to 39	31,968	
other measurements and calibrations	2,163	
list of starts, replay and its tests	10,372	
certification of the claims, facts	33,723	22,510
proofs in Lean	28,856	3,367
total	131,245	

TABLE 7. Computations, in thread-hours of the machines and, where they were measured, busy thread-hours. The row for budget 30 includes exploratory runs and the calibrations at $N = 10$ and $N = 12$.

APPENDIX B. TESTS OF THE METHODS AND PROGRAMS

The value $S(6) = 12 = S(5) + \lceil \log_2 6 \rceil$ makes $N = 6$ the smallest case in which the one-value bound is attained with the same Kraft slack, two, as it would be by a sorting network on 14 inputs with 49 comparators, so the analogues of Sections 3 to 5 can be tested on all optimal sorting networks on 6 inputs; the first rows of Table 8 do this.

CHANGES IN VERSION 3

Version 3 corrects the attribution of Jannis Harder's work and fixes three places that used the wrong pronoun for Harder; the results and their proofs are unchanged.

(1) The abstract and Sections 1, 2, 6, 9 and 10.2 credit Harder's work [7, 8, 9] where we use it: notions, lemmas, search, Huffman bound, certificate for $S(11) \geq 35$, formal system for certificates, verified checker and program. Section 10.2 also says that Harder's algorithm is proved correct but its implementation is not verified. Section 1 credits the generate-and-prune scheme of [2].
(2) The data statement names the programs derived from sortnetopt, records our request for permission to distribute the patches, and gives the licences.
(3) New references: Van Voorhis's two-value bound and Mericanii's finding on the published proof of Van Voorhis's bound on $P(2, N)$ (Section 12), Mericanii's red leads and the certified proofs of $S(9) = 25$ (Section 1), and the archived release of sortnetopt.
(4) We now cite version 2 of the data record, which carries the same corrections.
(5) The paper now has an acknowledgements section.

DATA AND CODE AVAILABILITY

The list of starts, the certificates of all searches, the programs that computed the list and replayed the certificates with their logs, including those of the replays with certified bounds (`claim_certificates/replay_certified` and `replay/logs/replays_certified`), the cut records with their claims, the lists of facts, the producer and the checker of the claim certificates, the witnesses for the facts derived from Harder's certificate or from the table of [15] with their checker, a ledger with one line for each start, and the script of Remark 4.5 are archived on Zenodo as [17] (doi:10.5281/zenodo.24299514). Of the claim certificates, the files for the weighted two-value bound are archived; the rest is regenerated by the archived producer from the archived

test	scope	outcome
red prefixes of optimal networks at $N = 6$	all 13,635 sorting networks on 6 inputs with 12 comparators, one from each class under exchanges of adjacent independent comparators	each red prefix in the list of starts for $N = 6$
macro-step form at $N = 6$	the same networks, 54,855 macro-steps	every macro-step generated, no state cut (138,690 instances)
late searches at $N = 6$	all 13 late starts for $N = 6$	a sorted state reached exactly for the 4 starts that extend to an optimal network
test (C4) at $N = 6$	all 177,255 prefixes of the networks above, exact core values	none of 354,510 instances infeasible
complete-start search with exact cores at $N = 10$	the 19 starts of the main case, budget $S(10) - 13 = 16$	the 12 starts that extend to optimal networks keep states through all 16 layers; the other 7 die
the same at $N = 12$	the 32 starts of the main case, budget $S(12) - 16 - 1 = 22$	all die by layer 4
test (C2) with exact cores on a sorting network with 51 comparators found by a heuristic search	its prefixes with 20 to 51 comparators, both polarities, budget $51 - k$	no state cut
test (C4) on 8 such networks	their prefixes with 12 to 50 comparators, both polarities, budget $51 - k$	none of the 14-channel instances infeasible; raising weights makes feasible ones infeasible
generators of children	the search program and the replay, first layer of 5 late starts, for 3 of them with the same work limit of `lb2cr`	identical sets of children that survive the tests with trivial weights
replay with single corruptions	6 copies of the certificate of one start, each with one record or kept state deleted or one budget, trivial weight, weight or core changed; 66 corruptions of 33 valid synthetic records (C4)	each rejected
`lb2cr` and the instance solver of the Lean proof of [14]	the 251,551 infeasible instances that the replay decides for the complete starts and the 11,000,259 that it decides for 25 late starts, all on at most 13 channels	same answers, except that the Lean solver did not finish 6,464 instances of one late start; the Lean proof of that start decides other instances
the same, on 14 channels	the 31,580 instances of (C4) in the certificates	same answers; the solver is proved sound in Lean for at most 14 channels

TABLE 8. Tests of the methods and programs. The files of the last two tests are in the directory `lean_s14/solver_check` of [18].

input claims of each group, and the record gives a checksum for each group against which the regenerated lines can be compared. The witnesses for the facts derived from the certificate [7] give the numbers of the steps they use and nothing else from it; the certificate is under CC BY 4.0 and is not part of [17]. The Lean proofs, with their data and the records of their builds, are archived as [18] (doi:10.5281/zenodo.23632985); its directory `lean_s14/solver_check` holds the comparisons of `lb2cr` with the Lean solver in Table 8. This paper is doi:10.5281/zenodo.24299513.

Several of our programs are modifications of Jannis Harder's program sortnetopt [9], among them a search program, the programs that compute exact values and bounds for cores, the producers of the claim certificates and the replay of the complete starts with certified bounds; the README of [17] lists them. The data record [17] contains our changes only as patches against commit `0b5d09c`, and its build instructions fetch the original code. The sortnetopt repository has no licence, so we have asked Harder for permission to distribute these patches; the request was pending when this version was published. The context lines and the removed lines in the patches are Harder's code and are not covered by our licences. Our own code is under the Apache License 2.0, and our text and data under CC BY 4.0.

Acknowledgements

We thank Jannis Harder for publishing sortnetopt, its verified checker and the certificate for $S(11) \geq 35$. Several of our programs are modifications of sortnetopt, and the certification in Section 9 uses the original program, the certificate, and two lemmas and a definition of the verified checker.

References

[1] R. C. Bose and R. J. Nelson, *A sorting problem*, J. ACM **9** (1962), no. 2, 282–296, doi:10.1145/321119.321126.
[2] M. Codish, L. Cruz-Filipe, M. Frank, and P. Schneider-Kamp, *Sorting nine inputs requires twenty-five comparisons*, J. Comput. System Sci. **82** (2016), no. 3, 551–563, doi:10.1016/j.jcss.2015.11.014.
[3] L. Cruz-Filipe and P. Schneider-Kamp, *Optimizing a certified proof checker for a large-scale computer-generated proof*, Intelligent Computer Mathematics (CICM 2015), Lecture Notes in Comput. Sci., Springer, 2015, pp. 55–70, doi:10.1007/978-3-319-20615-8_4.
[4] L. Cruz-Filipe, K. S. Larsen, and P. Schneider-Kamp, *Formally proving size optimality of sorting networks*, J. Automat. Reason. **59** (2017), no. 4, 425–454, doi:10.1007/s10817-017-9405-9.
[5] B. Dobbelaere, *Smallest and fastest sorting networks for a given number of inputs*, web page, `https://bertdobbelaere.github.io/sorting_networks.html`, consulted 2026-10-18.
[6] R. W. Floyd and D. E. Knuth, *The Bose–Nelson sorting problem*, A Survey of Combinatorial Theory (J. N. Srivastava et al., eds.), North-Holland, Amsterdam, 1973, pp. 163–172, doi:10.1016/B978-0-7204-2262-7.50020-X.
[7] J. Harder, *Certificate of the minimal size of 11-channel sorting networks*, Zenodo, 2019, dataset, doi:10.5281/zenodo.4108365.
[8] J. Harder, *An answer to the Bose–Nelson sorting problem for 11 and 12 channels*, arXiv:2012.04400v3 [cs.DS], 2022.
[9] J. Harder, *sortnetopt: lower size bounds for sorting networks*, source code; we use commit `0b5d09c` (`https://github.com/jix/sortnetopt`). Release v1, which predates that commit, is archived on Zenodo, 2020, doi:10.5281/zenodo.4139152.
[10] D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, 2nd ed., Addison-Wesley, Reading, MA, 1998.
[11] L. de Moura and S. Ullrich, *The Lean 4 theorem prover and programming language*, Automated Deduction – CADE 28, Lecture Notes in Comput. Sci., vol. 12699, Springer, 2021, pp. 625–635, doi:10.1007/978-3-030-79876-5_37.
[12] The mathlib Community, *The Lean mathematical library*, Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020), ACM, 2020, pp. 367–381, doi:10.1145/3372885.3373824.
[13] Mericanii, *Twenty tabulated lower bounds for sorting networks rest on an invalid proof: an audit of Van Voorhis (1972), and a partial repair*, draft, August 2026, `https://mericanii.com/work/s13/paper-correction.pdf`, consulted 2026-11-17.
[14] T. S. Tomar, *Sorting networks on 13 inputs need 45 comparators*, version 3, Zenodo, 2026, doi:10.5281/zenodo.23620051.
[15] T. S. Tomar, *A kernel-checked Lean proof that sorting networks on 13 inputs need 45 comparators*, Zenodo, 2026, doi:10.5281/zenodo.23603571.
[16] T. S. Tomar, *Certificate and verifier for "Sorting networks on 13 inputs need 45 comparators"*, Zenodo, 2026, doi:10.5281/zenodo.23504361.

[17] T. S. Tomar, *Certificates and programs for "Sorting networks on 14 inputs need 50 comparators"*, version 2, Zenodo, 2026, doi:10.5281/zenodo.24299514.

[18] T. S. Tomar, *Lean proofs for "Sorting networks on 14 inputs need 50 comparators"*, Zenodo, 2026, doi:10.5281/zenodo.23632985.

[19] D. C. Van Voorhis, *An improved lower bound for sorting networks*, IEEE Trans. Comput. **C-21** (1972), no. 6, 612–613, doi:10.1109/TC.1972.5009021.

[20] D. C. Van Voorhis, *Toward a lower bound for sorting networks*, Complexity of Computer Computations (R. E. Miller and J. W. Thatcher, eds.), Plenum Press, New York, 1972, pp. 119–129, doi:10.1007/978-1-4684-2001-2_12.

ORCID: `https://orcid.org/0000-0003-4668-3639`