SORTING NETWORKS ON 15 AND 16 INPUTS NEED 55 AND 60 COMPARATORS, AND 20 INPUTS NEED 81

TEJASVI SINGH TOMAR

Abstract. Let $S(N)$ be the least number of comparators in a sorting network on N inputs. We prove that $S(16) = 60$, so the sorting network with 60 comparators that Green found in 1969 has the fewest possible comparators. We also prove that $S(15) \geq 55$, which leaves 55 and 56 as the possible values of $S(15)$, and that $S(20) \geq 81$. Together, these results raise the previously known lower bounds on $S(N)$ for $17 \leq N \leq 32$ by two for even $N \geq 20$ and by one otherwise. The lower bounds for 16 and 20 inputs come from a rigidity argument: in a sorting network with one comparator fewer than the bound to be proved, the lower bounds for fewer inputs, among them $S(15) \geq 55$ at 16 inputs and $S(16) \geq 60$ at 20 inputs, fix the comparators through which the two largest and the two smallest values pass, wherever they enter, up to an explicit list of choices, and a computation rules out each choice. At 20 inputs, fixing these comparators also needs a search over the ways in which the two largest values can pass through the network. The proof of $S(15) \geq 55$ follows the method of our computer-assisted proof of $S(14) \geq 50$ and adds new lower bounds of Huffman type on the number of comparators needed to sort a set of vectors. Separately written programs check the computations. In Lean 4 we prove one step of the argument for $S(15) \geq 55$ and derive the three results, assuming only two tables of bounds from our proof of $S(14) \geq 50$ that are certified outside Lean; some steps are evaluated by compiled code rather than by Lean's kernel.

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. Its comparisons do not depend on the data, which makes sorting networks suitable for parallel sorting and for secure multiparty computation [2, Section 1]. 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 $N \leq 8$ [5]. Codish, Cruz-Filipe, Frank and Schneider-Kamp proved $S(9) = 25$ and $S(10) = 29$ by an exhaustive search [3], and Harder proved $S(11) = 35$ and $S(12) = 39$ by a search over sets of vectors with lower bounds of Huffman type [7]. We proved $S(13) = 45$ by a weighted form of the two-value bound and a search over partial networks [12], and $S(14) \geq 50$ by the same bound and a reduction to a list of starts, the possible beginnings of a sorting network with one comparator fewer than the bound to be proved [14]. The two-value bound and the bounds of Huffman type are recalled below. Table 1 lists the values and bounds for $9 \leq N \leq 16$.

Recursive lower bounds on $S(N)$ come from counting the comparators that the largest values must meet. Deleting from a sorting network the comparators met by its largest value from a given input leaves a sorting network with one input fewer [7, Lemma 16]. The paths of the largest value from the N inputs merge into a binary tree with N leaves, so the longest of them meets at least $\lceil \log_2 N \rceil$ comparators; this gives Van Voorhis's bound $S(N) \geq S(N-1) + \lceil \log_2 N \rceil$ [18].

Date: 12 November 2026.

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.

N	9	10	11	12	13	14	15	16
$S(N)$ before this paper	25	29	35	39	45	50–51	54–56	59–60
$S(N)$ with Theorems 1.1 and 1.2	25	29	35	39	45	50–51	**55**–56	**60**

TABLE 1. The least number of comparators in a sorting network on N inputs, for $9 \le N \le 16$: [3] for $N = 9, 10$, [7] for $N = 11, 12$, [12] for $N = 13$, [14] for the lower bounds before this paper for $N = 14$, 15, 16, and [9, Section 5.3.4], [4] for the upper bounds.

Deleting those met by the two largest values from a worst pair of inputs gives the two-value bound $S(N) \ge S(N-2) + P(2, N)$, where $P(2, N)$ is the least number of comparators that the two largest values must meet in the worst case in a network that delivers them to two fixed outputs [12, Lemma 2.3]; we determined $P(2, N)$ for $N \le 32$ in [12, Theorem 1.2]. Huffman's inequality bounds the cost of a set of vectors, the least number of comparators in a network that sorts it, through the costs of smaller sets obtained by pruning [7] (Section 2); we call such bounds of Huffman type.

The smallest known sorting networks on 14, 15 and 16 inputs have 51, 56 and 60 comparators [9, Section 5.3.4], [4]. The network on 16 inputs was found by Green in 1969 [9, Section 5.3.4], [4]. Van Voorhis's bound and $S(14) \ge 50$ imply $S(15) \ge 54$. For 16 inputs, Dobbelaere's table [4] lists the lower bound 57, which rests on Van Voorhis's two-value argument [19]; we proved $S(16) \ge 58$ in [12], which also discusses the status of that argument, and $S(16) \ge 59$ in [14]. For 20 inputs the smallest known sorting network has 91 comparators [4], and the recursion of [12], which takes the larger of Van Voorhis's bound and the two-value bound (Section 12), started from the bounds known before this paper, gives $S(20) \ge 79$ [14, Table 6]. Throughout, a dagger (†) marks a statement whose proof includes a computation.

Theorem 1.1. $S(16) = 60$.†

Theorem 1.2. $S(15) \ge 55$.†

Theorem 1.3. $S(20) \ge 81$.†

Theorem 1.1 shows that no sorting network on 16 inputs has fewer comparators than Green's. Its lower bound follows from Theorem 1.2, $S(14) \ge 50$ and $S(13) = 45$ by a rigidity argument and a computation. Suppose that a sorting network on 16 inputs has 59 comparators. Theorem 1.2 bounds the path of the largest value from each input by $59 - 55 = 4$ comparators, and Kraft's equality for the binary tree in which these paths merge forces all 16 paths to have exactly 4. The same holds for the smallest value. In the same way, $S(14) \ge 50$ fixes the comparators through which the second largest and the second smallest value pass, up to an explicit list of choices, and a finite check, which also uses $S(13) = 45$, excludes every choice. Section 3.5 explains why the argument works at 16 inputs and fails at 14 and 15.

Theorem 1.3 follows from Theorem 1.1 by the same kind of argument. With Theorem 1.1, Van Voorhis's bound gives $S(18) \ge 70$, so in a sorting network on 20 inputs with 80 comparators the two largest values, wherever they enter, meet at most 10 comparators, the least possible worst case, $P(2, 20) = 10$. Up to isomorphism, exactly one configuration of the two largest values, that is, one way in which they can pass through such a network, achieves this. This follows from the search of [12], which a separate implementation repeats here, and from a balance inequality (Section 4) for the one case that the search leaves open. The comparators through which the two largest and the two smallest values pass are then fixed up to an explicit list of choices, and, as at 16 inputs, a computation excludes all of them.

The proof of Theorem 1.2 follows the method of our proof of $S(14) \geq 50$ [14]. In a sorting network on 15 inputs with 54 comparators, $S(14) \geq 50$ fixes the shapes of the trees formed by the paths of the largest and of the smallest value, so that the network begins, up to symmetry, with one of 177 starts. Searches pruned by certified lower bounds rule out the starts after which both trees of paths are finished. In the other starts, one comparator that lies on both trees is still missing, and up to 32 comparators can a priori feed it, that is, precede it on a path into one of its inputs; a search over all choices of these comparators, which settled the corresponding cases at 14 inputs [14], is out of reach here. Direct tests rule out the cases with at most 3 feeding comparators, and new lower bounds of Huffman type, together with a final computation, rule out the others.

The searches and computations write certificates, and separately written programs replay them; the lower bounds on the costs of the sets of vectors of length 9 that the certificates use are certified. The proofs rest on these computations and their replays, except at one step of the proof of Theorem 1.2: the cases in which at least 4 comparators feed the missing comparator are settled by a theorem in Lean, proved partly by compiled code, and there the computation and its replay are cross-checks. Theorems 1.1 to 1.3 are also derived in Lean 4, with some steps run as compiled code. The derivation proves the bounds for sets of vectors of length 9 that it needs, reuses the Lean proofs of $S(13) \geq 45$ [12, 13] and $S(14) \geq 50$ [15], and assumes only the validity of two tables of bounds from the proof of $S(14) \geq 50$, which are certified outside Lean [14]; for Theorem 1.1 it also checks, by Lean's kernel alone, that Green's network sorts. Together, Theorems 1.1 and 1.2 raise the lower bound on $S(N)$ by one for every N from 17 to 32, and Theorem 1.3 raises it by one more for even N from 20 to 32.

Section 2 recalls the model and the tools. Section 3 derives Theorem 1.1 from Theorem 1.2, Section 4 derives Theorem 1.3 from Theorem 1.1, and Sections 5 to 8 prove Theorem 1.2. Section 9 certifies the lower bounds on the costs of sets of vectors of length 9 that the proofs use, Section 10 describes the replays, the trusted components and the controls, and Section 11 the proofs in Lean. Section 12 derives the bounds for 17 to 32 inputs, and Section 13 names the open cases. Appendix A lists the computations, and Appendix B defines the push-out and pull-in bounds, which let the computation of Section 8 stop extending a partial network early, and proves that they are lower bounds for the quantities that the Huffman tests of Section 8 use and that they do not decrease as the partial network grows (Lemma B.1).

2. Preliminaries

This section recalls the model, the notation and the lemmas of [14, Section 2], which build on [12], as far as the paper uses them.

2.1. **Networks, paths and sets.** 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 $i > j$, and for a comparator $g = (i, j)$ we write $\mathrm{hi}(g)$ and $\mathrm{lo}(g)$ for its output leads on the channels i and j, which receive the larger and the smaller value. Every sorting network can be untangled into a sequence of standard comparators of the same length, and any order of the comparators of a network that respects the dependencies gives the same circuit [14, Section 2.1]. The value placed on an input x above all other values follows the *path of the maximum* Π_x, and $\delta(x)$ is the number of comparators on it; the *path of the minimum* Π'_x and $\delta'(x)$ are defined in the same way. Complementing every vector and reversing the order of the channels exchanges the maximum and the minimum; it turns each statement about the largest values into its *mirror statement* about the smallest values, proved by the mirror argument, and we write out only one of the two.

For a sequence P of comparators, $X_P \subseteq \{0,1\}^n$ is the set of 0-1 vectors that it produces. A network *sorts* a set $X \subseteq \{0,1\}^n$ 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 such a network; similar sets (related by a permutation of the channels, possibly combined with complementation) have the same value of s. 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. Following Harder, a network *works* on X if any two vectors of X with the same weight give the same output. A network that sorts X works on it, and the converse holds for the sets that we meet, which are unions of maximal chains of up-steps (*well-behaved* [14, Lemma 2.1(c)]): the images of the vectors of each weight coincide, so the images of each maximal chain form one and the same maximal chain, which fixes the listing of the outputs.

A channel c is *extremal* in X for polarity 0 if the vector e_c with a single 1 on c lies in X, and for polarity 1 if $\mathbf{1} - e_c$ does; $E_\epsilon(X)$ is the set of these channels. For $c \in E_0(X)$ the *pruned set* X/c consists of the restrictions to the other channels of the vectors of X that can be reached from e_c by steps inside X, each changing one coordinate from 0 to 1; for polarity 1 it is defined through complements [12, Definition 5.2]. 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 $(X/x)/y$ [14, Section 2.2]. Pruning does not raise the cost: $s(X/c) \le s(X)$ [14, Lemma 2.1(b)].

2.2. **Runs of values.** Let P be a sequence of k standard comparators on $[n]$. A *run* of r values of polarity 0 places values $a_1 > \cdots > a_r$ on distinct inputs $i_1, \ldots, i_r$ and smaller values on all other inputs; a comparator is *touched* if one of $a_1, \ldots, a_r$ is on one of its input leads, and W denotes the set of touched comparators. The run *ends* on $(x_1, \ldots, x_r)$ if a_l ends on x_l for each l. Runs of polarity 1 use $a_1 < \cdots < a_r$ below all other values. The path of a_l depends only on $i_1, \ldots, i_l$. The following lemma is [14, Lemma 2.3].

Lemma 2.1. *For a run of r values of polarity ϵ through P that ends on $(x_1, \ldots, x_r)$, put $Y_0 = X_P$ and $Y_l = Y_{l-1}/x_l$, pruned for ϵ. Then $x_l \in E_\epsilon(Y_{l-1})$ for each l, and $s(Y_r) \ge S(n-r) - (k - |W|)$.*

If P is a sorting network (of standard comparators, as in the lemma), every Y_l is a set of sorted vectors, so $s(Y_r) = 0$ and the lemma gives the bound used most often in the rest of the paper.

Corollary 2.2 (deletion)**.** *If a sorting network of k standard comparators on $[n]$ touches the comparators of W in a run of r values, then $k - |W| \ge S(n-r)$.*

For $r = 1$ this is [7, Lemma 16], and for $r = 2$ it is the two-value bound of [19] as proved in [12, Lemma 2.3].

2.3. **Trees of paths.** Let U be a network and E a set of input channels whose paths of the maximum in U end on the same output. By the proof of [12, Lemma 7.1], the leads of these paths form a rooted tree. A comparator with two input leads in the tree is a *merge* and one with a single input lead in it a *pass-through*; contracting the vertices with one child gives a binary tree with leaves E, in which the depth $D(x)$ of x is at most $\delta(x)$.

Lemma 2.3 (Kraft's equality)**.** *In this situation $\sum_{x \in E} 2^{-D(x)} = 1$. Hence, if $\delta(x) \le d(x)$ for numbers $d(x)$ with $\sum_{x \in E} 2^{-d(x)} = 1$, then $\delta(x) = D(x) = d(x)$ for every $x \in E$, and every comparator on these paths is a merge.*

Proof. The equality is [12, Lemma 2.5]. Under the hypothesis, $1 = \sum 2^{-d(x)} \le \sum 2^{-\delta(x)} \le \sum 2^{-D(x)} = 1$, and since each term on the left is at most the corresponding term on the right, all terms agree. The number of pass-throughs on the path from x is $\delta(x) - D(x) = 0$. □

The same holds for the paths of the minimum. A lead that is the output of a comparator g lies on no path of the minimum if it is $\mathrm{hi}(g)$, and on no path of the maximum if it is $\mathrm{lo}(g)$, because the minimum leaves every comparator on lo and the maximum on hi [14, proof of Lemma 3.3]. Hence a comparator each of whose two input leads lies on some path of the maximum and on some path of the minimum has two input leads of the network.

2.4. **Lower bounds on costs.** We use the trivial bounds of Lemma 2.1, Huffman's inequality [14, Lemma 2.4]: if $E = E_\epsilon(X) \neq \emptyset$ and $v_x \leq s(X/x)$ for $x \in E$, then $\sum_{x\in E} 2^{v_x} \leq 2^{s(X)}$, and the weighted two-value bound [12, Proposition 6.2], [14, Proposition 2.6]. A *weighted instance* on n channels consists of a budget $L \geq 0$ and a list of scenarios (x, y, v) with $x \neq y$ and integer weights $v \geq 0$. 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 touches at most $L - v$ comparators [12, Definition 6.1]. For $X \subseteq \{0, 1\}^n$, a polarity ϵ, a budget L and weights v_t on some scenarios (x_t, y_t) of (X, ϵ), $I(X, \epsilon; v; L)$ is the instance with these scenarios and weights, read on the complement of X if $\epsilon = 1$. If $0 \leq v_t \leq s(Y_t)$ for the cores Y_t and $I(X, \epsilon; v; L)$ is infeasible, then $s(X) \geq L + 1$. The program `lb2cr` of [12, Section 8] decides weighted instances; its answer *infeasible* is correct whenever all budgets are at most 56 [12, Proposition 8.9], and only this answer is ever used to discard a case.

We use the bound on two levels. A weight v_t may itself be proved by an infeasible instance on the core Y_t, or, when $X = X_P$ for a sequence P of standard comparators and the extreme value of polarity $1 - \epsilon$ is *finished*, $E_{1-\epsilon}(X_P) = \{b\}$, on the set Y_t/b, pruned for polarity $1 - \epsilon$, which has the same cost as Y_t [14, Lemma 2.5]. The weights of such an instance are lower bounds on the costs of its cores, which are sets with $n - 4$ or $n - 5$ channels. In Section 3 the weights of the lower level are exact values of 6-channel sets; in the core bounds of Section 6 and of the test (T3) of Section 7 they are values of 9-channel sets; and in the test (T2) of Section 7 they are trivial weights of runs of 5 values.

3. From 15 to 16 inputs

Theorem 3.1. *If $S(15) \geq 55$, then $S(16) \geq 60$.*[†]

Besides $S(15) \geq 55$, the proof uses $S(14) \geq 50$ [14] and $S(13) = 45$ [12]. Throughout this section, N is a sorting network on 16 inputs with 59 comparators, untangled into a sequence of standard comparators on [16]. By Van Voorhis's bound and $S(15) \geq 55$, a sorting network on 16 inputs has at least 59 comparators, so Theorem 3.1 follows once no such N exists. All the structure below is a property of the circuit of N; when we list N as a sequence, any order that respects the dependencies may be used.

3.1. **The first stage.** A *perfect* tree is a binary tree whose leaves all have the same depth.

Theorem 3.2. *Assume $S(15) \geq 55$.*

- *(a) Every path of the maximum and every path of the minimum in N has exactly 4 comparators, all of them merges.*
- *(b) The first comparators of the 16 inputs form a perfect matching F of the inputs, with 8 comparators. Both trees of paths are perfect trees of depth 4, and the comparators of F are their merges of level 1, those whose inputs are leaves; we number the levels of merges from 1 to 4, the root.*
- *(c) Let G_0 consist of the 22 comparators of the two trees: F, the 7 further merges of the tree of the maximum, and the 7 further merges of the tree of the minimum. Then N can be listed as G_0 followed by a sequence R of 37 comparators on the channels $1, \ldots, 14$, and*

after G_0 the channels 15 and 0 hold the largest and the smallest value of every input sequence.

Proof. (a) By Corollary 2.2 with $r = 1$, $59 - \delta(x) \geq S(15) \geq 55$, so $\delta(x) \leq 4$ for every input x. All 16 paths of the maximum end on the output of the maximum, and Lemma 2.3 with $d(x) = 4$ gives $\delta(x) = D(x) = 4$ and no pass-through.

(b) The first comparator g of an input x lies on Π_x and on Π'_x, so it is a merge of both trees by (a), and both of its input leads lie on paths of both kinds. By the remark after Lemma 2.3, both are input leads of N, so g is also the first comparator of its other input. Hence the first comparators pair the inputs. The tree of the maximum is a binary tree with 16 leaves of depth 4, hence perfect, and its level-1 merges are the comparators of F; the same holds for the minimum. A merge of both trees has two input leads of N by the remark after Lemma 2.3, so it lies in F; hence the further merges of the two trees are distinct and $|G_0| = 22$.

(c) A comparator with an input lead on a path of the maximum lies on that path, because the maximum enters every comparator that one of its leads enters. By (a) it is then a merge of the tree of the maximum. Therefore every comparator outside G_0 has no input lead on any path, and its output leads lie on no path either. The input leads of a comparator of G_0 lie on paths, so they are input leads of N or outputs of comparators of G_0. Hence no comparator of G_0 depends on a comparator outside G_0, and N can be listed as G_0 followed by the others. After G_0, the leads that lie on paths of the maximum are the output of the root of its tree, on one channel, and no later comparator touches it; since N sorts, this channel is 15. Likewise, the minimum ends on channel 0. □

Since a standard comparator puts the larger value on the higher of its two channels, G_0 is determined by F and the two trees, taken as trees on the input channels: each merge of the tree of the maximum acts on the largest input channels of its two subtrees, and each merge of the tree of the minimum on the smallest. Renaming the inputs and untangling, as in the proof of Theorem 3.1 (Section 3.4), therefore changes G_0 only by the labelling of the two trees. With $F = \{(2i+1, 2i) : 0 \leq i \leq 7\}$ and the tree of the maximum fixed on the channels $1, 3, \ldots, 15$, the trees of the minimum on the channels $0, 2, \ldots, 14$ fall into 16 orbits under the automorphisms of the tree of the maximum above F, a group of order 128 acting on the 315 perfect trees on 8 leaves. Up to relabelling and reflection, which exchanges the maximum and the minimum, they form 13 classes. We use one representative of each of the 16 orbits; Figure 1 shows G_0 for the first of them.

3.2. **The second stage.** After G_0 the largest value of every input sequence is on channel 15. Consider a run of the two largest values of polarity 0 through G_0, with a_1 on an input i and a_2 on an input $j \neq i$; it ends on $(15, y)$ for some channel y. Let E be the set of these channels y, and for $y \in E$ let $W_2(y)$ be the largest number of comparators of G_0 touched by a run that ends on $(15, y)$. Define E' and W'_2 for the two smallest values in the same way.

Lemma 3.3. *Let l be the level at which the paths of the maximum from i and j meet in the tree of the maximum, so that $l = 1$ if (i, j) or (j, i) lies in F and $l = 4$ if they meet at the root.*

- *(a) If $l \geq 2$, the run ends on the channel of $\mathrm{lo}(c)$, where c is the merge at which the two paths meet, and it touches $3 + l$ comparators of G_0.*
- *(b) If $l = 1$, the run ends on the channel of $\mathrm{hi}(c')$, where c' is the level-2 merge of the tree of the minimum above the comparator of F on i and j, and it touches 5 comparators of G_0.*

Hence $|E| = 11$, every run that ends on $(15, y)$ touches exactly $W_2(y)$ comparators of G_0, W_2 takes the value 5 on eight channels, 6 on two and 7 on one, and $\sum_{y \in E} 2^{W_2(y)-9} = 1$. The eight channels with $W_2(y) = 5$ are those of $\mathrm{lo}(c)$ for the four level-2 merges c of the tree of the maximum and of $\mathrm{hi}(c')$ for the four level-2 merges c' of the tree of the minimum; they form $B = E \cap E'$.

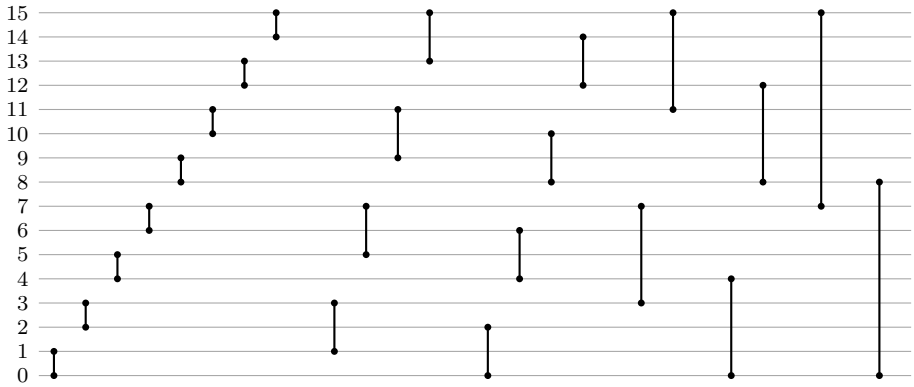


FIGURE 1. The prefix G_0 of representative 0, one comparator per column, with channel 15 at the top: the matching F, then the merges of the tree of the maximum on the odd channels and those of the tree of the minimum on the even channels, level by level. These are 22 of the 32 comparators of the first four layers of Green's network [9, Section 5.3.4, Fig. 49].

Proof. The value a_1 follows Π_i through its 4 comparators. The value a_2 follows Π_j until it meets a_1. (a) If $l \geq 2$, the paths meet at a merge c of the tree of the maximum that is not in F, and a_2 leaves c on $\mathrm{lo}(c)$. This lead lies on no path of the maximum, and on no path of the minimum, since the input leads of c are hi outputs; so no comparator of G_0 touches it again. The touched comparators are the 4 on Π_i and the $l-1$ on Π_j below c. (b) If $l = 1$, a_2 leaves the comparator $f \in F$ on i and j on $\mathrm{lo}(f)$, which lies on a path of the minimum. At the next merge c' of the tree of the minimum, the other input carries the smaller value of another pair, so a_2 leaves on $\mathrm{hi}(c')$, which lies on no path; the touched comparators are the 4 on Π_i and c'. These leads lie on distinct channels: a lead on no path is not touched again by G_0, so after G_0 each of the channels $1, \ldots, 14$ carries $\mathrm{lo}(c)$ for a merge c of the tree of the maximum outside F or $\mathrm{hi}(c')$ for a merge c' of the tree of the minimum outside F. Thus y determines the merge and l, and the counts follow: four channels with $l = 2$, two with $l = 3$, one with $l = 4$, and four $\mathrm{hi}(c')$, with $\frac{8}{16} + \frac{2}{8} + \frac{1}{4} = 1$. The set E' is described in the same way, and the intersection is as stated. □

Theorem 3.4. *Assume $S(15) \geq 55$ and $S(14) \geq 50$, and list N as G_0 followed by R.*

- *(a) For $y \in E$, the path of the maximum of R from y has exactly $9 - W_2(y)$ comparators, all of them merges of the tree $\mathcal{T}_2$ of these paths, and it ends on channel 14. The same holds for E' with the minimum, ending on channel 1, with a tree $\mathcal{T}_2'$.*
- *(b) The first comparators of R on the eight channels of B form a perfect matching S_2 of B, and each comparator of S_2 is a merge of $\mathcal{T}_2$ and $\mathcal{T}_2'$. Besides S_2, each of these trees has 6 merges: 3 that pair its six vertices of depth 3, 2 that pair its four vertices of depth 2, and the root.*
- *(c) Let G_1 consist of S_2 and these $6 + 6$ merges. Then N can be listed as $G_0 G_1 R'$, where R' is a sequence of 21 comparators on the channels $2, \ldots, 13$, and after $G_0 G_1$ the channels $15, 14$ and $0, 1$ hold the two largest and the two smallest values of every input sequence.*

Proof. (a) Take a run that ends on $(15, y)$ and touches $W_2(y)$ comparators of G_0. Channel 15 is not touched by R, and in R the value a_2 starts on y above all values on the channels $1, \ldots, 14$; it follows the path of the maximum of R from y, with $\delta_R(y)$ comparators. The run therefore touches $W_2(y) + \delta_R(y)$ comparators, and Corollary 2.2 with $r = 2$ gives $59 - W_2(y) - \delta_R(y) \geq S(14) \geq 50$, that is, $\delta_R(y) \leq 9 - W_2(y)$. Since N sorts, a_2 ends on channel 14 for every $y \in E$. Lemma 3.3

gives $\sum_y 2^{-(9-W_2(y))} = 1$, and Lemma 2.3, for $U = R$ with the bounds $d(y) = 9 - W_2(y)$, gives the claim.

(b) After a comparator of R that touches a lead on one of these paths, its hi output lies on no path of the minimum and its lo output on none of the maximum. Hence the leads that lie on paths of both kinds are the input leads of R on channels of B. The first comparator of R on a channel of B lies on both kinds of path, so by (a) it is a merge of $\mathcal{T}_2$ and $\mathcal{T}_2'$, and its other input lead lies on paths of both kinds: it is the input lead of R on another channel of B. By (a) and Lemma 3.3, $\mathcal{T}_2$ has eight leaves of depth 4, the channels of B, two of depth 3 and one of depth 2. The comparators of S_2 are the merges that join its leaves of depth 4, and the remaining merges pair the six vertices of depth 3, then the four of depth 2, then the two of depth 1. Lemma 3.3 for E' and W_2' gives $\mathcal{T}_2'$ in the same way. A merge of $\mathcal{T}_2$ and $\mathcal{T}_2'$ has two input leads of R, on channels of B, so it lies in S_2; hence G_1 has 16 comparators and R' has 21.

(c) As in the proof of Theorem 3.2(c), a comparator of R that touches a lead on one of these paths is a merge of $\mathcal{T}_2$ or $\mathcal{T}_2'$ and lies in G_1, so the others touch no such lead and no comparator of G_1 depends on them. After G_1, the paths of the maximum of R from E have all reached channel 14, and a later comparator on channel 14 would lie on them; so channel 14 is not touched again, and neither is channel 1. □

Since every run that ends on $(15, y)$ touches exactly $W_2(y)$ comparators of G_0, every run of the two largest values touches exactly 9 comparators, and deleting them from N would leave a sorting network on 14 inputs with 50 comparators; so $S(14) \geq 51$ alone would already rule N out.

With the channels labelled as above, a choice of G_1 consists of a perfect matching of B for S_2 (one of 105) and, on each side, a pairing of the six vertices of depth 3 and then of the four of depth 2 (one of $15 \cdot 3 = 45$), with each merge placed in standard form. This gives $105 \cdot 45 \cdot 45 = 212{,}625$ *labelled* choices of G_1 for each representative of G_0.

3.3. **The third stage.** After G_0G_1, consider the runs of the three largest values of polarity 0; they end on $(15, 14, z)$ for channels z in a set $E_3 \subseteq \{2, \ldots, 13\}$. Let $W_3(z)$ be the largest number of comparators of G_0G_1 touched by such a run that ends on $(15, 14, z)$, and put

$$K_3 = \sum_{z \in E_3} 2^{W_3(z)-14}.$$

Define E_3', W_3' and K_3' for the three smallest values in the same way.

Theorem 3.5. *Assume $S(15) \geq 55$, $S(14) \geq 50$ and $S(13) \geq 45$, and list N as G_0G_1R'.*

(a) $K_3 \leq 1$ and $K_3' \leq 1$.

(b) If $K_3 = 1$, then for $z \in E_3$ the path of the maximum of R' from z has exactly $14 - W_3(z)$ comparators, all merges of a binary tree with leaves E_3 and these depths, and its root puts the third largest value on channel $13 = \max E_3$. The mirror statement holds if $K_3' = 1$, with $\min E_3' = 2$.

(c) If $K_3 = K_3' = 1$ and $E_3 \cap E_3' = \emptyset$, let G_2 consist of the merges of the two trees of (b). Then N can be listed as $G_0G_1G_2R''$, where R'' acts on the channels $3, \ldots, 12$.

Proof. (a) As in Theorem 3.4(a), with three values: channels 15 and 14 are not touched by R', so a run that ends on $(15, 14, z)$ and touches $W_3(z)$ comparators of G_0G_1 touches $W_3(z) + \delta_{R'}(z)$ comparators in all, where $\delta_{R'}(z)$ is the length of the path of the maximum of R' from z. Corollary 2.2 with $r = 3$ gives $\delta_{R'}(z) \leq 14 - W_3(z)$. These paths end on channel 13, and Lemma 2.3 gives $K_3 \leq \sum_z 2^{-\delta_{R'}(z)} \leq \sum_z 2^{-D(z)} = 1$. (b) If $K_3 = 1$, all these inequalities are equalities. A merge in standard form puts the union of the two sets of leaves on the higher of its two channels, so by induction the root of the tree puts the third largest value on the largest

channel of E_3, which must be channel 13. (c) If E_3 and E'_3 are disjoint, no comparator is a merge of both trees. As before, the comparators of R' outside G_2 touch no lead on these paths, so they can be listed after G_2, and channels 13 and 2 are not touched after the roots. □

3.4. **The finite check.** For a choice of G_0 and G_1 with $K_3 = K'_3 = 1$ and $E_3 \cap E'_3 = \emptyset$, and a choice of the two trees of Theorem 3.5(b), let $k = |G_0G_1G_2|$ and let $M \subseteq \{0,1\}^{10}$ be the set of restrictions to the channels $3, \ldots, 12$ of the vectors of $X_{G_0G_1G_2}$. If $N = G_0G_1G_2R''$, then R'' sorts M, so $s(M) \leq 59 - k$. Every vector of M lies on a maximal chain of $\{0,1\}^{10}$ inside M: map a maximal chain of $\{0,1\}^{16}$ through a suitable input by $G_0G_1G_2$, restrict it to the channels $3, \ldots, 12$ and remove repetitions. Thus M is well-behaved, and so are its cores [12, Lemma 5.4].

Proposition 3.6. *For each of the* 16 *representatives of* G_0 *and each of its* 212,625 *labelled choices of* G_1*, either* $K_3 > 1$ *or* $K'_3 > 1$*, or the following hold:* $K_3 = K'_3 = 1$*;* $E_3 \cap E'_3 = \emptyset$*;* $\max E_3 = 13$ *and* $\min E'_3 = 2$*; on each side exactly one binary tree has the leaves and depths of Theorem 3.5(b); the resulting prefix* $G_0G_1G_2$ *has* $k \in \{42, 43, 44\}$ *comparators; and* $s(M) > 59 - k$*.*[†]

The computation runs over all $16 \cdot 212{,}625 = 3{,}402{,}000$ labelled prefixes, with no reduction by symmetry beyond the choice of the 16 representatives of G_0. For each of them it computes E_3, W_3 and K_3, and their mirrors, by simulating all runs of three values on each side, and checks that $K_3 \leq 1$ and $K'_3 \leq 1$ hold together only when $K_3 = K'_3 = 1$. In the second case it builds the trees, which are unique, and the set M. Table 2 gives the counts. There are 17,334 prefixes $G_0G_1G_2$: 11,403 with $k = 42$, 5,022 with $k = 43$ and 909 with $k = 44$. Their middle sets M form 648 distinct sets, with 96 distinct canonical forms under similarity.

For each of the 648 distinct sets M and the largest budget $L = 59 - k$ among its prefixes, we prove $s(M) > L$ on both polarities by two levels of the weighted two-value bound. The instance $I(M, \epsilon; b; L)$ on 10 channels has all scenarios of (M, ϵ), with weights $b(Y)$ for their cores Y, which have 8 channels, and `lb2cr` finds it infeasible. Each weight is then proved. For every core Y, the instance on Y with budget $b(Y) - 1$ and with weights equal to the exact costs of the 6-channel cores of Y is infeasible for one polarity (in fact for both), so $s(Y) \geq b(Y)$. The exact costs of the 6-channel sets come from a breadth-first search over sets: a set of vectors on 6 channels is a 64-bit mask, the search applies all standard comparators, identifies equal sets, and returns the first layer that contains a set with at most one vector of each weight, that is, the image of the set under a network that works on it. Standard comparators suffice, since untangling only permutes the outputs and keeps this property. A network that sorts a set works on it, so the values of this search are at most the costs s, which is all that the weighted bound needs; by the remark in Section 2 they are equal to them. These sets have 3,871 distinct cores on 8 channels, and these have 4,295 distinct cores on 6 channels, with exact values from 6 to 12. The bound proves $s(M) \geq L + 1$ for every set. Exact 10-channel values, computed with the search engine of Harder's program [8], show that $L + 1$ is attained for some sets, so the bound is sharp there.

Proof of Theorem 3.1. Suppose that N exists. Rename its channels by a permutation that maps F and the two trees of Theorem 3.2 onto those of one of the 16 representatives, and untangle the result. The renamed network sorts with its outputs listed in the renamed order. Untangling keeps its circuit and its input channels, since it only exchanges the labels of the two channels of a comparator in the rest of the sequence [14, Section 2.1], and the sequence of 59 standard comparators that it produces sorts with some listing of its outputs, hence in the natural order, since a standard network leaves every sorted input unchanged. Because standard comparators put the larger value on the higher channel (Section 3.1), in a sequence of standard comparators the channels of every comparator are determined by the circuit and the input channels. Therefore the 22 merges of the two trees of the new network are the comparators of the representative, and we replace N by it. (Renaming alone does not suffice: in N the matching F need not pair adjacent

representative	orbit size	ruled out by K_3 or K_3'	prefixes $G_0G_1G_2$	canonical middle sets
0	1	207,552	5,073	15
1	4	210,230	2,395	29
2	4	211,432	1,193	59
3	16	211,886	739	11
4	64	212,287	338	7
5	64	212,456	169	9
6	2	209,424	3,201	15
7	8	211,556	1,069	30
8	8	212,144	481	40
9	8	211,556	1,069	30
10	32	212,287	338	9
11	32	212,456	169	13
12	8	212,144	481	40
13	32	212,456	169	13
14	16	212,344	281	15
15	16	212,456	169	9

TABLE 2. The finite check of Proposition 3.6, for each representative of G_0. The orbit size is the number of trees of the minimum in its orbit. The third column counts the labelled prefixes G_0G_1 with $K_3 > 1$ or $K_3' > 1$; with the fourth, each row sums to 212,625 labelled choices of G_1. The pairs of representatives $(7, 9)$, $(8, 12)$ and $(11, 13)$ are related by reflection. The last column counts the canonical forms of the middle sets under similarity; all rows together have 96 of them, those of the 648 distinct sets.

channels, so its pairs can be ordered differently by their larger and by their smaller channels, and the renamed G_0 can then contain comparators that are not standard.) By Theorem 3.4 and $S(14) \geq 50$ [14], N can be listed as G_0G_1R' with one of the labelled choices of G_1. By Theorem 3.5(a) and $S(13) = 45$ [12], $K_3 \leq 1$ and $K_3' \leq 1$, so Proposition 3.6 gives $K_3 = K_3' = 1$, disjoint sets E_3 and E_3', and unique trees. By Theorem 3.5(b) and (c), N can be listed as $G_0G_1G_2R''$ with the G_2 of Proposition 3.6, so $s(M) \leq 59 - k$, against Proposition 3.6. □

Proof of Theorem 1.1. Green's network shows $S(16) \leq 60$, and Theorems 1.2 and 3.1 give $S(16) \geq 60$. □

3.5. **Why the argument needs 16 inputs.** Every inequality of the first two stages holds with equality: $59 - 55 = 4 = \log_2 16$, and $59 - 50 = 9 = P(2, 16)$ [12, Theorem 1.2], with $P(2, N)$ as in Section 1. For 14 inputs and 50 comparators the paths of the maximum may have $50 - S(13) = 5$ comparators, Kraft's equality over 14 leaves allows pass-throughs, and the second stage has slack $50 - S(12) - P(2, 14) = 2$; nothing is forced. For 15 inputs and 55 comparators, $S(14) \geq 50$ bounds the paths of the maximum only by 5 comparators, Kraft's equality over 15 leaves again allows pass-throughs, and the second stage has slack $55 - S(13) - P(2, 15) = 1$.

4. From 16 to 20 inputs

This section proves Theorem 1.3. Theorem 1.1 and Van Voorhis's bound give $S(17) \geq 65$, $S(18) \geq 70$, $S(19) \geq 75$ and $S(20) \geq 80$, the second row of bounds of Table 6; these values do not use Theorem 1.3. In this section, let N be a sorting network on 20 inputs with 80 comparators, untangled as before into a sequence of standard comparators on [20]; Theorem 1.3 follows once no such N exists. The argument follows Section 3 stage by stage, with three differences. The

first stage now has slack, and the two trees of paths are forced instead at the second stage, by the computation of the least worst-case cost of two values in [12, Section 4]. The sums of the second stage exclude most trees of the minimum, which never happens at 16 inputs. Finally, the third stage excludes every remaining case by itself (Proposition 4.8), so no middle set is needed.

4.1. **The trees of paths.** By Corollary 2.2 with $r = 1$, every path of the maximum and of the minimum in N has at most $80 - S(19) \leq 5$ comparators. Kraft's equality over 20 leaves of depth at most 5 then leaves room for many trees and for pass-throughs. By Corollary 2.2 with $r = 2$, every run of the two largest values touches at most $80 - S(18) \leq 10$ comparators. Since N sorts, the two largest values always end on the channels 19 and 18, respectively, and by this bound each such run touches at most 10 comparators, the least possible worst case: $P(2, 20) = \lceil \log_2 F(20) \rceil = 10$ [12, Theorem 1.2], where F is Van Voorhis's function [12, Section 2.4], and $F(20) = 1024 = 2^{10}$ is a power of two.

We use the structure theory of [12, Section 3].

- A *top-two correct* network, one that delivers the two largest values to two fixed outputs, has a *reduced* version in which each run meets at most as many comparators: it deletes the pass-throughs whose other input never carries the second value, and the comparators off the tree of the maximum with an input that never carries the second value, and leaves the leads of the two values unchanged otherwise [12, Lemma 3.5].
- The reduced network is described by a *configuration*. This consists of the tree B of the paths of the maximum, the number of pass-throughs on each edge, and, for each merge and each pass-through whose lo output the second value can use, a *source*: a root and a depth in the forest that carries the second value to its output. If $p_2 \leq L$, where p_2 is the largest number of comparators that a run of the two largest values touches, this configuration is *feasible at level* L: every run (x, y) has $\mathrm{cost}(x, y) \leq L$ [12, Theorem 3.8(a)], where $\mathrm{cost}(x, y)$ is the number of comparators that the run starting on (x, y) meets.
- In a configuration Γ, the merges of B and the pass-throughs are the *red nodes*, and the pass-throughs on the edge from a vertex w of B to its parent, or to the output o_M of the maximum if w is the root, are $(w, 1), \ldots, (w, k(w))$, numbered from below; B with the numbers $k(w)$ is the *skeleton* of Γ. Each red node v is either *dead*, which only a pass-through can be, or *live* with a source $(\rho(v), d(v))$: a root $\rho(v)$, which is the output o_m of the second value or a pass-through, and a depth $d(v) \geq 0$. At each root r, that is, at o_m and at every pass-through, the depths of the live red nodes with root r must satisfy Kraft's equality $\sum 2^{-d} = 1$, and the red nodes must satisfy the *acyclicity condition*: the directed graph on them with an arc from each red node to the next red node above it, and an arc from each live red node to its root whenever that root is a pass-through, is acyclic [12, Definition 3.7]. For a red node v let $u(v) = v$ if v is a merge and $u(v) = w$ if $v = (w, i)$, and let $A(v)$ be the set of red nodes above v on the path to o_M, with v excluded; for a pass-through $q = (w, i)$ and a proper ancestor e of w in B, let $\mathrm{cl}(q, e)$ be the number of red nodes from q up to e, with q included and e excluded [12, Section 3.1]. For a live red node v, $\tau_\Gamma(v)$ is defined by the recursion

$$\tau_\Gamma(v) = \begin{cases} d(v) & \text{if } \rho(v) = o_m, \\ d(v) + \tau_\Gamma(q) & \text{if } \rho(v) = q \in A(v), \\ d(v) + \mathrm{cl}(q, e) + \tau_\Gamma(e) & \text{if } \rho(v) = q \notin A(v), \end{cases}$$

where $e = \mathrm{lca}_B(u(q), u(v))$, and $\tau_\Gamma(v) = \infty$ if the recursion reaches a dead node [12, Definition 3.7 and Section 3.2].

For a merge c of B let $a(c) = \mathrm{dep}(c) + 1 + h_1(c) + h_2(c)$, where $\mathrm{dep}(c)$ is the depth of c in B, the root having depth 0, and $h_1(c)$ and $h_2(c)$ are the heights of its two subtrees, a leaf having height 0 [12, Section 2.4]. Let B_{20} be the binary tree on 20 leaves whose root has two children: a perfect tree with 8 leaves, and a vertex with a perfect tree with 8 leaves and a perfect tree with 4 leaves below it. It has 12 leaves of depth 4 and 8 of depth 5, every leaf has a leaf as its sibling, and $\sum_c 2^{a(c)} = 1024 = F(20)$. A dynamic program over the pairs (height, f), as in [12, Section 2.4], shows that it is the only tree on 20 leaves with this value; Proposition 4.1 recovers this within its search.

Proposition 4.1. *The search of* [12, Section 4] *on* 20 *inputs with* $L = 10$*, without the symmetry test, finds exactly one configuration feasible at level* 10 *outside one skeleton,* C^**;* C^* *admits none, by Lemma 4.2 and the bound that follows it. The configuration found has the tree* B_{20} *and no pass-throughs, and it gives every merge* c *the root* o_m *and the depth* $10 - a(c)$*. By* [12, Theorem 3.8(b), Lemma 3.10 and Lemma 3.9] *and the proof of* [12, Proposition 4.1]*, every configuration on* 20 *inputs that is feasible at level* 10 *and has depths at most* 10 *is isomorphic to this one.*[†]

The search runs over all 201 tree shapes of height at most 5 and all placements of pass-throughs with $\delta(x) \le 5$, bounds forced by feasibility at level 10 [12, Theorem 3.8(b) and Lemma 3.10]; up to the automorphisms of B, 102,739 of these skeletons pass the tests (S3) and (P1) of [12, Section 4]. For the tree B_{20} without pass-throughs the depths are forced by Kraft's equality at o_m: the lo output of each merge c must reach o_m within $10 - a(c)$ comparators, and $\sum_c 2^{-(10-a(c))} = F(20)/2^{10} = 1$. Without the symmetry test, the search decides every skeleton within seconds except one: the tree B_{20} with one pass-through above the perfect subtree with 4 leaves and one above the perfect subtree with 8 leaves of depth 4, a skeleton with 2^{17} automorphisms, which we call C^*. We settle C^* by a balance inequality of the forest of sources, which uses nothing but Kraft's inequality at the roots of that forest. In it, $\tau(v) = \tau_{\Gamma}(v)$ for a live red node v, $\tau(v) = \infty$ for a dead one, and $2^{-\infty} = 0$.

Lemma 4.2 (balance)**.** *Every configuration satisfies*

$$\sum_c 2^{-\tau(c)} \;\le\; 1 + \sum_q M_q, \qquad \text{where } M_q = \max_e 2^{-(\mathrm{cl}(q,e)+\tau(e))},$$

c runs over the merges of B, $q = (w, i)$ over the pass-throughs, and e over the proper ancestors of w in B; $M_q = 0$ if w is the root of B.

Proof. Put $R(o_m) = 1$ and $R(q) = 2^{-\tau(q)} + M_q$ for each pass-through q. Let v be a live red node with source $(\rho(v), d(v))$. By the recursion that defines τ, $2^{-\tau(v)} = 2^{-d(v)}$ if $\rho(v) = o_m$; $2^{-\tau(v)} = 2^{-d(v)}\,2^{-\tau(q)}$ if $\rho(v) = q \in A(v)$; and otherwise $2^{-\tau(v)} = 2^{-d(v)}\,2^{-(\mathrm{cl}(q,e)+\tau(e))} \le 2^{-d(v)} M_q$, where $q = \rho(v)$ and $e = \mathrm{lca}_B(u(q), u(v))$ is a proper ancestor of $u(q)$, since $u(q)$ and $u(v)$ are incomparable: otherwise q, which is not in $A(v)$, would be v or lie below v on the path to o_M, and the arcs from q up to v, with the arc from v to its root q, would form a cycle, against the acyclicity condition [12, Definition 3.7]. In every case $2^{-\tau(v)} \le 2^{-d(v)} R(\rho(v))$. Every root is o_m or a pass-through, and the depths of the sources at each root have Kraft sum at most 1. Summing over the live red nodes and grouping them by roots therefore gives

$$\sum_c 2^{-\tau(c)} + \sum_q 2^{-\tau(q)} \;\le\; \sum_r R(r) \;=\; 1 + \sum_q 2^{-\tau(q)} + \sum_q M_q,$$

and the terms $2^{-\tau(q)}$ cancel. □

For C^* at level 10, write e for the root of B_{20}, e_1 for its child above the perfect subtrees with 8 and 4 leaves, and q_1, q_2 for the pass-throughs above the 4-leaf subtree and above the other 8-leaf

subtree. Every leaf has $\delta = 5$. The budget of a merge c is $\sigma(c) = 10 - \max(\delta(x) + \mathrm{pre}(y, c))$, the maximum over the leaves x and y below different children of c, where $\mathrm{pre}(y, c)$ is the number of red nodes on the path from y up to c, with c excluded [12, Section 3.1]. It is 1 at e, 2 at e_1, 3 at the roots of the two 8-leaf subtrees, 4 at the five merges with exactly 4 leaves below them, and 5 at the ten cherries. Feasibility at level 10 gives $\tau(c) \le \sigma(c)$ at every merge [12, Lemma 3.12]. No pass-through lies above e, and every other vertex is below it, so the root of the source of e is o_m, $\tau(e) = d(e) \le 1$ and $\alpha = 2^{-\tau(e)}$ is 1 or 1/2; also $\beta = 2^{-\tau(e_1)} \ge 1/4$. The left side of Lemma 4.2 is therefore at least $\alpha + \beta + 2 \cdot 2^{-3} + 5 \cdot 2^{-4} + 10 \cdot 2^{-5} = \alpha + \beta + 7/8$. On the right, $\mathrm{cl}(q_1, e_1) = 1$, $\mathrm{cl}(q_1, e) = 2$ and $\mathrm{cl}(q_2, e) = 1$, so the right side is $1 + \max(\beta/2, \alpha/4) + \alpha/2$. The left side exceeds the right side by at least $\alpha/2 + \beta - 1/8 - \max(\beta/2, \alpha/4)$, which is at least 1/4 in both cases: $\alpha/2 + \beta/2 - 1/8$ when $\beta/2 \ge \alpha/4$, and $\alpha/4 + \beta - 1/8$ otherwise. So no configuration with the skeleton C^* is feasible at level 10. The bound uses only $\tau(e)$, $\tau(e_1)$ and the budgets of the other 17 merges, whatever the other sources are. At the largest values allowed, $\tau(e) = 1$ and $\tau(e_1) = 2$, the left side is at least 13/8 and the right side is 11/8.

Section 10.3 lists the programs that confirm this.

Lemma 4.3. *In the configuration of Proposition 4.1, every leaf x of B_{20} is the starting point of a run of cost exactly* 10*.*

Proof. Let c be the parent of x, and let y be a deepest leaf of the subtree at the sibling of x, at height h. The largest value from x meets $\mathrm{dep}(c) + 1$ comparators. The second value from y meets h comparators before c, and then the forest, with depth $10 - a(c)$, where $a(c) = \mathrm{dep}(c) + 1 + 0 + h$. The sum is 10. □

Theorem 4.4. *Assume $S(18) \ge 70$.*[†]

(a) Both trees of paths of N are copies of B_{20}, and no path of the maximum or of the minimum contains a pass-through.

(b) The first comparators of the 20 *inputs form a perfect matching F of the inputs, with* 10 *comparators, and these are the level-*1 *merges of both trees.*

(c) Let G_0 consist of the 28 *comparators of the two trees: F, the* 9 *further merges of the tree of the maximum, and the* 9 *further merges of the tree of the minimum. Then N can be listed as G_0 followed by a sequence R of* 52 *comparators on the channels* $1, \ldots, 18$*. After G_0 the channels* 19 *and* 0 *hold the largest and the smallest value of every input.*

Proof. (a) By $S(18) \ge 70$, every run of the two largest values touches at most 10 comparators (Section 4.1). Let N^* be the reduced version of N. Its configuration is feasible at level 10 and gives every live red node a depth at most 10 [12, Theorem 3.8(a)], so by Proposition 4.1 it is the configuration found there, up to an isomorphism of trees, which preserves costs [12, Lemma 3.9]. So the tree of the maximum of N^* is a copy of B_{20} without pass-throughs. The network N arises from N^* by putting back deleted comparators. A deleted pass-through lies on the path of the maximum of every input x below it, so every run with the largest value on x meets it. Take such an x, and the run of Lemma 4.3. In N^* it meets exactly 10 comparators [12, proof of Theorem 3.8(a)]. Apart from the deleted comparators, the two values meet the same comparators in N and in N^* [12, proof of Lemma 3.5], so in N the run meets these 10 comparators and also the deleted pass-through. That is at least $11 > 10$ comparators, a contradiction. So no pass-through was deleted, and the tree of the maximum of N is that of N^*.

(b) By (a), the argument of Theorem 3.2(b) applies: the first comparators pair the inputs and are level-1 merges of both trees, and a merge of both trees lies in F, so the further merges of the two trees are distinct. In B_{20} every leaf has a leaf as its sibling, so the level-1 merges are exactly these 10 comparators.

(c) By (a) every comparator touched by the maximum is a merge of its tree, so, by the argument of Theorem 3.2(c), N can be listed as G_0 followed by the others. After the root of the tree, the lead of the maximum is touched again by no comparator; since N sorts, it ends on channel 19, and likewise the minimum ends on channel 0. □

The argument of Theorem 3.2(b) shows more generally that if no path of the maximum or of the minimum of a sorting network contains a pass-through, then its first comparators pair the inputs; so a sorting network on an odd number of inputs has a pass-through on some path of the maximum or of the minimum. For 27 inputs and 119 comparators, the lower bound on $S(27)$ in Table 6, the second stage is also an equality: $S(25) \geq 108$ (Table 6) and $119 - 108 = 11 = \lceil \log_2 F(27) \rceil$, with $F(27) = 2^{11}$. But configurations with a pass-through are feasible at level 11 even under the bound $119 - 114 = 5$ on the paths given by $S(26) \geq 114$ (Table 6): the 509 skeletons on the 26 tree shapes with 27 leaves and height at most 5 carry 90 feasible configurations, 88 of them with a pass-through. Two separate programs agree on this, and one of them decides the two skeletons left open by the other [16]. So the trees are not forced there.

As in Section 3.1, G_0 is determined by F and the two trees on the input channels, and renaming and untangling changes it only by the labelling of the trees. Take the matching $F = \{(2i+1, 2i) : 0 \leq i \leq 9\}$ and the tree of the maximum fixed on the channels $1, 3, \ldots, 19$. There are 28,350 trees of the minimum on the channels $0, 2, \ldots, 18$ with the shape of B_{20}. One program counts 495 orbits among them under the automorphisms of the tree of the maximum above F, a group of order 128, and 295 classes up to relabelling and reflection, as in Section 3.1; every tree is treated separately below.

4.2. **The second stage.** After G_0 the largest value of every input is on channel 19. Define E, W_2 and, for the two smallest values, E' and W_2' as in Section 3, with channel 19 in place of 15, and put

$$K_2 = \sum_{y \in E} 2^{W_2(y)-10}, \qquad K_2' = \sum_{y \in E'} 2^{W_2'(y)-10}.$$

At 16 inputs Lemma 3.3 gave the analogous sums the value 1 for every G_0. Here they depend on the tree of the minimum. As in Lemma 3.3, after G_0 the channels $1, \ldots, 18$ carry, one on each, the leads $\mathrm{lo}(c)$ of the 9 merges c of the tree of the maximum outside F and the leads $\mathrm{hi}(c)$ of the 9 merges c of the tree of the minimum outside F. A run whose two values meet at such a merge c of the tree of the maximum ends on $\mathrm{lo}(c)$ and touches at most $a(c)$ comparators of G_0, with equality for some run; over these 9 merges $\sum_c 2^{a(c)-10} = 25/32$. A run of the two inputs of a comparator of F ends on $\mathrm{hi}(c')$, where c' is the level-2 merge of the tree of the minimum above it, and touches $1 + \delta(x)$ comparators of G_0, where $\delta(x) \in \{4, 5\}$ is the depth of the two inputs in the tree of the maximum. The runs of both comparators of F below c' end on $\mathrm{hi}(c')$, so on the channel of $\mathrm{hi}(c')$ the value of W_2 is 1 plus the larger of their two depths. Hence $|E| = |E'| = 14$, the set $B = E \cap E'$ has 10 channels (as in Section 3; from here on B denotes this set, not the tree of a configuration), and $K_2 = (30 + n)/32$, where $n \in \{2, 3, 4\}$ is the number of level-2 merges of the tree of the minimum that have an input of depth 5 in the tree of the maximum; so $K_2 \in \{1, 33/32, 17/16\}$, and $K_2 = 1$ exactly when these merges pair the four pairs of inputs of depth 5 among themselves; K_2' takes the same values, with the roles of the two trees exchanged.

Theorem 4.5. *Assume $S(18) \geq 70$, and list N as G_0 followed by R.*

(a) $K_2 \leq 1$ and $K_2' \leq 1$.

(b) Suppose $K_2 = K_2' = 1$. For $y \in E$, the path of the maximum of R from y has exactly $10 - W_2(y)$ comparators, all of them merges of the tree $\mathcal{T}_2$ of these paths, and it ends on channel 18; likewise for E' and the minimum, ending on channel 1, with a tree $\mathcal{T}_2'$. The

first comparators of R on the channels of $B = E \cap E'$ form a perfect matching S_2 of B. Each comparator of S_2 is a merge of $\mathcal{T}_2$ and $\mathcal{T}_2'$, and it pairs two channels that have the same depth in $\mathcal{T}_2$ and the same depth in $\mathcal{T}_2'$. Besides S_2, the merges of each of these trees pair its vertices level by level, from the deepest.

(c) Let G_1 consist of S_2 and these merges. Then N can be listed as G_0G_1R', where R' is a sequence of 31 comparators on the channels $2, \dots, 17$. After G_0G_1 the channels $19, 18$ and $0, 1$ hold the two largest and the two smallest values of every input.

Proof. (a) The argument of Theorem 3.4(a), with the channels 19 and 18 in place of 15 and 14, Theorem 4.4(c) in place of Theorem 3.2(c), and $80 - W_2(y) - \delta_R(y) \geq S(18) \geq 70$ from Corollary 2.2, gives $\delta_R(y) \leq 10 - W_2(y)$ for every $y \in E$, and these paths end on channel 18. Lemma 2.3 then gives $K_2 = \sum_y 2^{-(10-W_2(y))} \leq \sum_y 2^{-\delta_R(y)} \leq \sum_y 2^{-D(y)} = 1$, and $K_2' \leq 1$ in the same way.

(b) If $K_2 = 1$, Lemma 2.3 with $d(y) = 10 - W_2(y)$ gives $\delta_R(y) = 10 - W_2(y)$ for every $y \in E$, and every comparator on these paths is a merge; they form the tree $\mathcal{T}_2$, whose root puts a_2 on channel 18. In the same way $K_2' = 1$ gives $\mathcal{T}_2'$. The argument of Theorem 3.4(b) shows that the first comparators of R on the channels of B form a perfect matching S_2 of B, that each of them is a merge of both trees, and that every merge of both trees lies in S_2. Each comparator of S_2 joins two leaves of both trees, which as siblings have the same depth in each tree, and the other merges of each tree join its vertices level by level, as in any binary tree.

(c) The argument of Theorem 3.4(c), with channel 18 in place of 14, shows that N can be listed as G_0G_1R' and that the channels 18 and 1 are not touched after G_1; by Theorem 4.4(c), neither are the channels 19 and 0. Each tree has $|E| - 1 = |E'| - 1 = 13$ merges, 5 of them in S_2, so G_1 has $5 + 8 + 8 = 21$ comparators and R' has $80 - 28 - 21 = 31$. □

Proposition 4.6. *Of the* 28,350 *trees of the minimum, exactly* 270 *have $K_2 = K_2' = 1$. Every other tree has $K_2 > 1$ or $K_2' > 1$; the sums take only the values* 1, 33/32 *and* 17/16. *For each of the* 270 *trees the choices of G_1 in Theorem 4.5 number $3 \cdot 135^2 = 54{,}675$ (for* 54 *trees), $9 \cdot 135^2 = 164{,}025$ (for* 81*) or $45 \cdot 135^2 = 820{,}125$ (for* 135*),* 126,955,350 *in all.*†

The 270 trees form 41 orbits; by the count of the program above, they lie in 31 of the 295 classes.

4.3. **The third stage and the finite check.** After G_0G_1, define E_3, W_3 and their mirrors as in Section 3, with channels $19, 18$ and $0, 1$, and put

$$K_3 = \sum_{z \in E_3} 2^{W_3(z)-15}, \qquad K_3' = \sum_{z \in E_3'} 2^{W_3'(z)-15}.$$

With $S(17) \geq 65$, the argument of Theorem 3.5(a) gives $K_3 \leq 1$ and $K_3' \leq 1$. The channels 19 and 18 are not touched by R', and Corollary 2.2 with $r = 3$ gives $\delta_{R'}(z) \leq 80 - 65 - W_3(z)$.

The finite check uses the following observation. It makes each choice of G_1 cheap to evaluate; direct simulation gives the same values (Sections 10.3 and 10.5).

Lemma 4.7. *Fix G_0 and S_2, and let T and T' consist of the merges of $\mathcal{T}_2$ and of $\mathcal{T}_2'$ in G_1 outside S_2. Then T and T' act on disjoint sets of channels.*

(a) In a run of the three largest values through G_0S_2T, the two largest values end on the channels 19 *and* 18*, which T' does not touch.*

(b) In T' the third largest value a_3 meets only merges of $\mathcal{T}_2'$. At each of them the other input carries a smaller value, so a_3 leaves on the hi *output.*

(c) *Hence its path through T', and the number of comparators of T' that it touches, depend only on the channel on which it enters T', and*

$$W_3(z) = \max_{z_0}\big(w(z_0) + t(z_0)\big),$$

where $w(z_0)$ is the largest number of comparators of G_0S_2T touched by a run whose third value ends on z_0, and $t(z_0)$ is the number of comparators of T' on the path from z_0. The maximum runs over the z_0 whose path through T' ends on z.

The mirror statement holds for K'_3, with T and T' exchanged.

Proof. The channels of T are those of the vertices of $\mathcal{T}_2$: the channels of $E \setminus B$ and the hi outputs of S_2. Those of T' are the channels of $E' \setminus B$ and the lo outputs of S_2. These sets are disjoint, so T and T' commute. After T the two largest values are on the channels 19 and 18 by Theorem 4.5. In T' the value a_3 is the largest value present on the channels of T', so it leaves every comparator of T' that it enters on the hi output. □

Proposition 4.8. *For each of the* 270 *trees of the minimum of Proposition 4.6 and each of its choices of G_1, $K_3 \geq 2$ and $K'_3 \geq 2$.*[†]

No reduction by symmetry is used: the computation covers all 270 trees, not only one per orbit, and all choices of G_1. For each tree and each S_2 it simulates all runs of three values through G_0S_2T and G_0S_2T' on each side, and then evaluates every pair (T, T') by Lemma 4.7. For every choice it also checks that the runs of two values end on the channels $19, 18$ and $0, 1$, with at most 10 touched comparators and exactly 10 for the worst. The counts are in Table 3. Both sums lie between 2 and 3, and each attains 2 for every tree; no choice of G_1 has $K_3 \leq 1$ or $K'_3 \leq 1$.

choices of S_2	orbits	trees	choices of G_1 per tree	choices of G_1	with $K_3 \leq 1$ or $K'_3 \leq 1$
3	9	54	54,675	2,952,450	0
9	16	81	164,025	13,286,025	0
45	16	135	820,125	110,716,875	0
all	41	270		126,955,350	0

TABLE 3. The finite check of Proposition 4.8, grouped by the number of choices of S_2. A choice of G_1 consists of S_2 and one tree of merges on each side, 135 on each side for every tree of the minimum. The table per orbit is in the archived data [16].

Proof of Theorem 1.3. Suppose that N exists. By Theorem 4.4 and $S(18) \geq 70$, after renaming and untangling as in the proof of Theorem 3.1, its prefix G_0 is given by F, the fixed tree of the maximum and one of the 28,350 trees of the minimum. By Theorem 4.5(a) that tree has $K_2 \leq 1$ and $K'_2 \leq 1$, so by Proposition 4.6 it is one of the 270 and $K_2 = K'_2 = 1$. By Theorem 4.5(b) and (c), N can be listed as G_0G_1R' for one of its choices of G_1, and then $K_3 \leq 1$ and $K'_3 \leq 1$ by $S(17) \geq 65$ (Section 4.3), against Proposition 4.8. □

5. Sorting networks on 15 inputs with 54 comparators

In this section and the next three, T denotes a sorting network on 15 inputs with 54 comparators. Recall from Section 1 that $S(15) \geq 54$, so no sorting network on 15 inputs is smaller. We follow Sections 3 to 6 of [14], which treat 14 inputs and 49 comparators, and state the results for 15 inputs and 54 comparators, with $S(14) \geq 50$ in place of $S(13) = 45$. We list T as a sequence $g_1, \ldots, g_{54}$ of standard comparators on [15] and write X_j for the set produced by $g_1, \ldots, g_j$.

Lemma 5.1. *(a) $s(X_j) = 54 - j$, and g_{j+1} is useful for X_j, for $0 \le j < 54$.*

(b) Every prefix satisfies the excess filter $\sum_{x \in E_\epsilon(X_j)} 2^{w_j^\epsilon(x)} \le 16$ *for $\epsilon = 0, 1$, where $w_j^\epsilon(x)$ is the largest number of comparators on a path of the extreme value of polarity ϵ that ends on x.*

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

Proof. (a) The comparators after g_j sort X_j, and a shorter network sorting X_j, placed after $g_1, \ldots, g_j$ and untangled, would give a sorting network with fewer than 54 comparators, against $S(15) \ge 54$; a comparator that is not useful would leave s unchanged [14, Lemma 3.1]. In (b) the trivial bound $s(X_j/x) \ge S(14) - (j - w_x) \ge 50 - (j - w_x)$ of Lemma 2.1, Huffman's inequality and $s(X_j) = 54 - j$ from (a) give $\sum_x 2^{w_x - 4} \le 1$. Part (c) is also Corollary 2.2 with $r = 1$: $54 - \delta(x) \ge S(14) \ge 50$. □

Lemma 5.2. *The binary tree of the paths of the maximum has* 14 *leaves at depth* 4 *and one leaf x at depth* 3*. The only pass-through that a path of the maximum can contain lies on the edge from x to its parent, so there is at most one. The same holds for the minimum, with a leaf z at depth* 3*.*

Proof. Since $D(x) \le \delta(x) \le 4$ by Lemma 5.1(c), Kraft's equality gives $\sum_x (2^{4 - D(x)} - 1) = 16 - 15 = 1$, with each term in $\{0, 1, 3, 7, 15\}$, so exactly one term is 1. Every edge other than the one above x, including the edge above the root, has a leaf of depth 4 below it, and a pass-through there would give that leaf more than 4 comparators. □

Lemma 5.3. *The set F of comparators of T whose two input leads are input leads of T has* 7 *comparators. They join the* 14 *leaves of depth* 4 *of both trees in pairs of siblings, so the two trees have the same* 7 *sibling pairs, and $x = z$ is the input that F misses.*

Proof. The sibling of a leaf of depth 4 is a leaf of depth 4, with no pass-through on either edge, so their parent lies in F. This gives 7 comparators of F for each tree, and since F is a matching of 15 inputs, $|F| \le 7$; so both trees use the same 7. □

Unlike in the 14-input case, there is therefore no second case: every sorting network on 15 inputs with 54 comparators has this structure.

5.1. **The starts.** A lead of T is *red* if it lies on a path of the maximum or of the minimum, and *blue* otherwise; a comparator is red if one of its input leads is red, and blue otherwise [14, Section 4.1]. The red comparators of T that do not depend on any blue comparator form the *red prefix* of T, and T can be listed as its red prefix followed by the other comparators [14, Lemma 4.2]. The first comparator g of the unpaired input x lies on both trees. It is not a merge of both, since both of its input leads would then lie on a path of the maximum and on a path of the minimum, hence be input leads of T (Section 2), and g would lie in F. Hence g is a merge of the tree of the maximum and the pass-through of the tree of the minimum, or the reverse, or the pass-through of both. In the first two cases the red comparators are the 7 of F and the $7 + 7$ further merges of the two trees, 21 in all, and none of them depends on a blue comparator; the red prefix consists of all 21, and after it both extreme values have reached their final channels. In the third case there are 22 red comparators. Call the other input lead of g its *port*; since $g \notin F$, the port is not an input lead. If the port is produced by a red comparator, the red prefix consists of all 22; if it is produced by a blue comparator, it consists of the 15 red comparators that do not depend on g: the 7 of F and, in each tree, the 4 merges that are not on the path from x. A sequence is *complete* if after it each extreme value has a single extremal channel, and *late* if it is not complete and has an *abstract completion* [14, Section 4.2]: a nonempty sequence of moves on the pairs of path lengths at its extremal channels that keeps the excess filter for

both polarities and ends with one extremal channel for each polarity, where a move models a red comparator, and one on two extremal channels requires one of them to have been produced by an earlier move. By [14, Lemma 4.3], if the red prefix of a sorting network on 15 inputs with 54 comparators is not complete, the red comparators that follow it give an abstract completion of it with one move each.

The red-prefix search of [14, Section 4.3] builds layers of sequences of standard comparators, beginning with the empty sequence. The children of a sequence P are the sequences Pg in which g touches an extremal channel of X_P, is useful for X_P and keeps the excess filter of Lemma 5.1(b), and each layer keeps one child from each similarity class. A *start* is a complete or late sequence of the layers of least length among those with similar sets, and the *budget* of a start Q is $54 - |Q|$.

Theorem 5.4. *The red-prefix search yields* 177 *starts:* 151 *complete starts,* 26 *with* 21 *comparators and budget* 33 *and* 125 *with* 22 *comparators and budget* 32*, and* 26 *late starts with* 15 *comparators and budget* 39.† *If a sorting network on* 15 *inputs with* 54 *comparators exists, then some start Q with budget β satisfies $s(X_Q) \le \beta$ if Q is complete, and, if Q is late, there is a sequence R of β standard comparators such that QR is a sorting network and every red comparator of R depends on a blue comparator of R; then the red prefix of QR is Q.*

Proof. List the network as its red prefix followed by the other comparators. Following the red prefix comparator by comparator, each prefix is similar to a sequence kept by the search, because each comparator is red, so it touches an extremal channel of the set before it [14, Lemma 4.1], and it is useful and keeps the excess filter by Lemma 5.1, and the rest of the network can be transplanted onto that sequence, by applying the similarity to its comparators, exchanging maximum and minimum if the similarity complements, and untangling [14, Section 4.3]. At the end of the red prefix the sequence is complete, or it is late by [14, Lemma 4.3], and minimality of the network forces it to be a start of the same length. This is the proof of [14, Theorem 4.4], with Lemmas 5.1 to 5.3 in place of [14, Lemmas 3.1 to 3.4] and abstract completions whose path lengths stay at most 4 and whose sums stay at most 16. □

The search keeps 17,856 sequences; the useful comparators that touch an extremal channel extend them in 1,128,900 ways, of which 1,029,011 violate the excess filter.

6. The complete starts

The searches for the complete starts are those of [14, Section 6], with 15 channels and target 54. A *state* is a sequence P of standard comparators on [15] that begins with the start, its children are the states Pg with g useful for X_P, and its budget is $54 - |P|$. After a complete start no useful comparator touches the channel t of the maximum or the channel b of the minimum, so every state is complete and its children act on the 13 channels other than t and b.

A state P with k comparators is *cut* if a test shows $s(X_P) \ge 55 - k$. For a polarity ϵ, let t and b now denote the channels of the extreme values of polarity ϵ and of the other polarity; for a scenario (y, z) of $(X_P/t, \epsilon)$, the core $Z = ((X_P/t)/y)/z$ has 12 channels. The other extreme is finished, so by [14, Lemma 2.5] the 11-channel set Z/b has $s(Z/b) = s(Z)$. The weight of (y, z) is the larger of two lower bounds on $s(Z)$:

- the *trivial weight* $\max(0, \max_W(39 - (k - |W|)))$, the inner maximum over the runs of three values that end on (t, y, z), and 0 if there is no such run, which is a lower bound by Lemma 2.1 and $S(12) = 39$ [7];
- a *core bound*, a lower bound on $s(Z/b)$ from the weighted two-value bound on 11 channels, whose weights are values of the 9-channel cores of Z/b, exact or lower bounds.

The test is (C2) of [14, Section 6.2]: the weighted instance $I(X_P/t, \epsilon; v; 54 - k)$ on 14 channels is infeasible; then $s(X_P) \ge s(X_P/t) \ge 55 - k$ by the weighted two-value bound, as in [14,

Lemma 6.2]. The searches keep one state of least length for each class of similar sets, which is sound because a cut proves $s(X_P) \geq 55 - k$, a property of the set and the length that similar sets share. Every claim of a core bound is certified (Section 9).

Theorem 6.1. *Let Q be a complete 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) $|X_P| > 16$ for every $P \in \mathcal{K}$;
(iii) for every $P \in \mathcal{K}$ and every child P' of P, some state P'' with $|P''| \leq |P'|$ and a set similar to $X_{P'}$ is cut or lies in $\mathcal{K}$.

Then $s(X_Q) > \beta$.

Proof. Suppose that $s(X_Q) \leq \beta$. By (i), and since similar sets have the same value of s, some $P \in \mathcal{K}$ satisfies $s(X_P) \leq 54 - |P|$; choose such a P with $s(X_P)$ least. A set that is sorted without comparators consists of nondecreasing vectors for some listing of the channels, and there are 16 of these, so $s(X_P) \geq 1$ by (ii). Let g be the first comparator of a sequence of $s(X_P)$ standard comparators that sorts X_P, which exists because untangling a network with $s(X_P)$ comparators that sorts X_P only relabels its outputs. It is useful for X_P, since otherwise it would leave X_P unchanged or act on it as the exchange of its two channels, and the rest of the sequence would sort a set similar to X_P with fewer comparators. Thus $P' = Pg$ is a child of P, and $s(X_{P'}) \leq s(X_P) - 1 \leq 54 - |P'|$. Let P'' be the state given by (iii). Then $s(X_{P''}) = s(X_{P'}) \leq 54 - |P'| \leq 54 - |P''|$, so P'' is not cut, since a cut proves $s(X_{P''}) \geq 55 - |P''|$. Hence $P'' \in \mathcal{K}$, with $s(X_{P''}) \leq 54 - |P''|$ and $s(X_{P''}) < s(X_P)$, against the choice of P. For 14 inputs the statement is part of [14, Theorem 6.3]. □

Condition (ii) holds for every state with at most 53 comparators: X_P contains a vector of each of the weights $0, \ldots, 15$, and $|X_P| = 16$ would mean that it contains exactly one, so that P works on $\{0,1\}^{15}$ and hence sorts it (Section 2), against $S(15) \geq 54$.

Proposition 6.2. *For every complete start Q with budget β, the certificate of its search gives a finite set $\mathcal{K}$ of states that satisfies* (i)–(iii) *of Theorem 6.1; hence $s(X_Q) > \beta$.*[†]

All 151 searches die out; Table 4 summarizes them. Of the 125 starts with budget 32, all but one are refuted in the first layer, that is, no child of the start survives, and the search of the other ends in the second; of the 26 starts with budget 33, 20 are refuted in the first layer, and the searches of the others end in the fourth layer (two) or in the sixth (four). The certificates record 18,340 cuts by instances on 14 channels with weights above the trivial ones, and these use 234,627 core bounds on 202,675 distinct 11-channel sets.

budget	starts	last layer	states expanded	children	classes	cut records
32	125	2	126	9,495	4,710	4,709
33	26	6	282	20,520	13,887	13,631
all	151		408	30,015	18,597	18,340

TABLE 4. The searches for the complete starts. "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; "classes" counts their similarity classes in each search after the cuts with trivial weights, and every class is cut by a recorded instance or kept.

7. The late starts: cones

Let Q be one of the 26 late starts. If Q is the red prefix of a sorting network T on 15 inputs with 54 comparators, as for the start given by Theorem 5.4, then by Section 5 the first comparator g of the unpaired input x is the pass-through of both trees and its port is produced by a blue comparator, Q consists of the 15 red comparators of T that do not depend on g and does not touch x, and the red comparators outside Q are g and the $3+3$ merges on the paths from x, which all depend on g. A *cone* is a nonempty sequence C of comparators on the eight channels that carry no red lead after Q (the *middle* channels of Section 8), each of which is a predecessor of the last one or the last one itself; for an output p of the last comparator of C, let g be the comparator on x and p, and M the $3+3$ merges on the paths from x, in the form given at the start of Section 8; the complete state $QCgM$ is *admissible* if every comparator of CgM is useful for the set produced before it.

Lemma 7.1. *T can be listed as $QCgMR$, where C is a cone whose last comparator produces the port of g, M consists of the $3+3$ merges on the paths from x, and R consists of blue comparators. Hence $s(X_{QCgM}) = 54 - (15 + |C| + 7) = 32 - |C|$, and $QCgM$ is complete. Every comparator of $QCgMR$ is useful for the set produced before it, by Lemma 5.1(a); in particular $QCgM$ is admissible.*

Proof. List T in macro-steps after Q by [14, Lemma 5.1]: $QU_1 \cdots U_p$, where the macro-step U_j consists of a blue comparator h_j followed by red comparators, each of which touches a channel touched by an earlier comparator of U_j. No red comparator outside Q can be placed before g, since all of them depend on g, and the port of g is produced by some h_i. By the construction in the proof of [14, Lemma 5.1], g is placed in U_i, and so are the six merges, each of which becomes ready once g and the merges below it are placed. Thus $U_1, \ldots, U_{i-1}$ and $U_{i+1}, \ldots, U_p$ are single blue comparators and $U_i = h_i g M$. Let A be the set of the h_j with $j \le i$ that are predecessors of the port of g, and D the set of the other h_j with $j < i$. If h_l uses an output of $h_j \in D$, then $h_l \notin A$, since otherwise h_j would be in A; and nothing in A, g or M depends on D. Therefore Q, A in order, g, M, D and $h_{i+1}, \ldots, h_p$ list the same circuit. A comparator of A, g or M that follows a comparator $d \in D$ in T shares no channel with d, since it would then depend on d; so the new list keeps the order of the comparators on each channel and consists of the same standard comparators on the same channels. Put $C = A$; its comparators are blue and come before g, so they act on the eight channels that carry no red lead after Q. The value of s follows from Lemma 5.1(a). □

Hence each late start is refuted once $s(X_{QCgM}) > 32 - |C|$ holds for every cone C and for each choice of the port, among the two outputs of the last comparator of C, for which $QCgM$ is admissible. Nothing else bounds the cone, since before g every comparator is blue and the trivial bound with one value gives exactly the budget. The number of complete states grows more than tenfold from one size of the cone to the next in Table 5, so a search over all cones, as for the late starts of [14], is out of reach. We prove these inequalities at the root, without searching below the complete state, by four tests, each of which proves $s(X_P) \ge 55 - k$ for a complete state P with k comparators and the extreme channel t of a polarity ϵ:

- (T1) the instance $I(X_P/t, \epsilon; v; 54 - k)$ of Section 6 with trivial weights is infeasible;
- (T2) the same with *raised* trivial weights: for a scenario (y, z) of $(X_P/t, \epsilon)$ with trivial weight v, the instance on the 12-channel core $Z = ((X_P/t)/y)/z$ whose weights are the trivial weights $\max(0, 29 - (k - |W|))$ of runs of five values, and 0 if there is no such run, by Lemma 2.1 and $S(10) = 29$ [3], is infeasible at budget v, which proves $s(Z) \ge v + 1$; this is repeated while it succeeds, up to a fixed number of times;
- (T3) the same with core bounds, as in Section 6;

(T4) the Huffman tests (H2) and (H4) of Section 8, which we refer to by those names.

Proposition 7.2. *For every late start and every cone C with at most 3 comparators, each complete state $QCgM$, for either port, that is admissible satisfies $s(X_{QCgM}) > 32 - |C|$.*[†]

There are 212,359 admissible complete states with cones of at most 3 comparators, over the 26 late starts and both ports, counted once per produced set for each late start and cone size; cones that differ only by exchanging adjacent comparators on disjoint channels produce the same set. Table 5 shows how the tests divide them; (T2) is applied with up to three raises and then, on the states that remain, with up to six. Since a cut depends only on the set and the number of comparators, it suffices to cut one state of each pair of a similarity class and a length; the 986 states left for (T3) form 321 such pairs.

$\|C\|$	states	cut by (T1)	cut by (T2)	cut by (T3)
1	1,452	1,315	52	85
2	16,389	14,535	1,606	248
3	194,518	187,086	6,779	653
all	212,359	202,936	8,437	986

TABLE 5. The complete states of the late starts with cones of at most 3 comparators, over all 26 late starts and both ports, and the test that cuts each of them first.

8. The late starts: cones with at least four comparators

We keep the notation of Section 7, and relabel the channels so that $x = 14$, keeping the order of the other channels; since Q does not touch x, it stays standard. The comparators of T that touch x need not stay standard; untangling them changes the sets produced only by a permutation of the channels, which preserves costs, usefulness and the answers of the tests, so g and M may be taken in the standard form below. Seen from x, the tree of the maximum consists of three subtrees P_1, P_2, P_4 with 1, 2 and 4 sibling pairs, which the path from x meets in this order. The start Q contains every merge of these subtrees; after Q their maxima are on leads L_1, L_2, L_4, and the merges of x with them are the three merges of the maximum in M. The tree of the minimum gives subtrees P'_1, P'_2, P'_4 and leads L'_1, L'_2, L'_4 in the same way. The other 8 channels are the *middle* channels after Q: the channels of $\mathrm{lo}(c)$ for the merges c of Q above F in P_2 and P_4, and of $\mathrm{hi}(c)$ for those in P'_2 and P'_4. Let u_4 be the channel of $\mathrm{lo}(c)$ for the root c of P_4, and v_4 that of $\mathrm{hi}(c')$ for the root c' of P'_4. For a cone C on the middle channels and a port p, the complete state $QCgM$ has $g = (14, p)$, the merges $(14, L_1)$, $(14, L_2)$, $(14, L_4)$ of the maximum, and the merges of the minimum, which combine L'_1, L'_2, L'_4 in turn with the channel of the running minimum, starting on p. It has $k = 22 + |C|$ comparators and budget $32 - |C|$.

The argument for these states has three steps. The test (H2) of Section 8.1 is Huffman's inequality for the paths that the second largest value still has to travel. When the port carries no class of either polarity, the sum of (H2) for each polarity grows with the potential of that polarity. This potential is 16 after Q and never decreases along the cone, so (H2) cuts unless the cone keeps both potentials at 16; when the port carries a class, (H2) cuts at once (Lemmas 8.3 and 8.4 and their mirror statements). The cones of admissible states that keep both potentials and have a dead port can be reordered into at most nine comparators on channels that carry classes, followed by a network N on channels that no class reaches, at most 24 comparators in all (Lemma 8.5). For them the sums of (H2) are exactly at the threshold, so a cut has to use runs of

more than two values; (H4) uses the third and fourth largest values. The push-out and pull-in bounds of Appendix B are lower bounds, for one kind of port each, on the numbers $H(y)$ of Section 8.5, any of which above 1 makes (H4) cut; they are computed from a partial network N and do not decrease as N grows (Lemma B.1), so a computation can stop extending N early, and the states that remain are cut by (H4) or by the raised trivial weights of (T2) (Proposition 8.6).

8.1. **Huffman tests.** Let P be a complete state with k comparators and budget $\beta = 54 - k$, let ϵ be a polarity, and let t be the channel of the extreme value of polarity ϵ. Let $Y = E_\epsilon(X_P/t)$, let $Z_y = E_\epsilon((X_P/t)/y)$ for $y \in Y$, and let $U_{yz} = E_\epsilon(((X_P/t)/y)/z)$ for $z \in Z_y$, all pruned for ϵ. With $W_2(y)$, $W_3(y,z)$ and $W_4(y,z,u)$ the largest numbers of comparators touched by runs of 2, 3 and 4 values of polarity ϵ that end on (t,y), (t,y,z) and (t,y,z,u), put $v_2(y) = \max(0, 45-(k-W_2))$, $v_3(y,z) = \max(0, 39-(k-W_3))$ and $v_4(y,z,u) = \max(0, 35-(k-W_4))$, where $v_r = 0$ if no run has these ends, and

$$b_3(y,z) = \max\Big(v_3, \Big\lceil \log_2 \sum_{u \in U_{yz}} 2^{v_4} \Big\rceil\Big), \qquad b_2(y) = \max\Big(v_2, \Big\lceil \log_2 \sum_{z \in Z_y} 2^{b_3(y,z)} \Big\rceil\Big),$$

with an empty sum leaving the first term. The test (H2) cuts P if $\sum_{y \in Y} 2^{v_2(y)} > 2^\beta$, and the test (H4) if $\sum_{y \in Y} 2^{b_2(y)} > 2^\beta$.

Lemma 8.1. *If* (H2) *or* (H4) *cuts P, then $s(X_P) \geq 55 - k$.*

Proof. By Lemma 2.1 with $S(13) \geq 45$, $S(12) = 39$ and $S(11) = 35$, the numbers v_2, v_3 and v_4 are lower bounds on the costs of the corresponding pruned sets. Huffman's inequality, applied to each pruned set over its extremal channels, gives $s((X_P/t)/y) \geq b_2(y)$ and $s(X_P/t) \geq \log_2 \sum_y 2^{b_2(y)}$, and $s(X_P) \geq s(X_P/t)$ [14, Lemma 2.1(b)]. □

Every cut of (H2) is a cut of (H4). Replacing Z_y and U_{yz} by the channels on which runs end gives the *run form* of the two tests; such a channel is extremal in the pruned set, because a run that ends on (t,y,z,u) maps the input with 1s on its values to $e_t + e_y + e_z + e_u$ and its sub-runs give a chain of up-steps from e_t [14, Lemma 2.2]. The sums of the run form are therefore at most those above, and a cut of the run form is a cut. For the second value nothing changes, since $y \in Y$ exactly when $e_t + e_y$ lies in X_P (for polarity 1, $\mathbf{1} - e_t - e_y$), that is, when a run of two values ends on (t,y). For a complete state $QCgM$ of a late start with $|C| \leq 28$, no v_2 is clamped, since $k - 45 = |C| - 23 \leq 5 \leq W_2$: the second value meets at least one comparator without the largest, namely its comparator of F, the comparator g when it starts on x, or a merge of the minimum when the two values share a sibling pair. Hence $v_2(y) - \beta = W_2(y) - 9$, which does not depend on $|C|$.

8.2. **Classes of runs and the potential.** We consider polarity 0; polarity 1 is the mirror image, with hi and lo, maximum and minimum, and the leads L_j and L'_j exchanged. In $QCgM$ the largest value meets exactly 4 comparators from every input and never lies on a middle channel after Q. For a run of the two largest values, the comparators touched by the second value but not by the largest are *exclusive*.

Lemma 8.2. *After Q, the second largest value lies on a middle channel exactly when the two largest values start in the same subtree P_2 or P_4 but in different sibling pairs, or in the same sibling pair, unless that pair is the one of P'_1. The largest number of exclusive comparators over the runs that leave it on a middle channel d is $E(d) = 2$ for $d = u_4$ and $E(d) = 1$ for six of the other middle channels, and no run leaves it on v_4. Hence $\sum_d 2^{E(d)} = 16$.*

Proof. If the two values start in different sibling pairs of P_j, they meet at a merge of Q in P_j, and the second value leaves it on the lo output, a middle channel; before that it has met only

the comparator of its own pair, and in addition the merge at the next level when they meet at the root of P_4, whose lo output is on u_4. If they start in the same pair, the second value leaves the comparator of F on its lo output and, unless the pair is that of P'_1, meets one merge of Q in P'_2 or P'_4, where it is the larger value and leaves on the hi output, a middle channel other than v_4. In all other runs it reaches a lead L_j or L'_1, or it starts on x, which Q does not touch and which is not a middle channel. □

Inside a cone the second largest value is the largest value on the middle channels, so it leaves every comparator of the cone on the hi output. Runs that leave the second value on a middle channel after Q, and whose second values are on the same lead at the same time, follow the same path through the rest of the cone; we call them a *class*, with E the largest number of exclusive comparators of its runs. A comparator of the cone that meets one class raises its E by 1, and one that meets two classes merges them, with E one more than the larger of the two. Since $2^{e+1} > 2^e$ and $2^{\max(a,b)+1} \geq 2^a + 2^b$, with equality only for $a = b$, the *potential* $\Phi_0 = \sum_{\text{classes}} 2^E$, which is 16 after Q by Lemma 8.2, never decreases along a cone, and it stays 16 exactly when every comparator that meets a class merges two classes with equal E. The mirror construction defines Φ_1. A lead is *dead* if it carries no class of either polarity.

8.3. **Dead ports and ports that carry a class.**

Lemma 8.3. *Let C be a cone with port p, and suppose that no class of polarity 0 is on p after C. Then the sum of the run form of* (H2) *for polarity 0 is at least $2^\beta(\Phi_0/32 + 1/2)$, with equality if $|C| \leq 28$. Consequently,* (H2) *cuts unless $\Phi_0 = 16$.*

Proof. The comparators of gM do not touch the middle channels other than p. Thus the runs of a class on such a channel end there, and the largest touch count among them is $4 + E$: four comparators for the largest value and E exclusive ones. These channels contribute $\sum 2^{4+E-9} = \Phi_0/32$ times 2^β. Following both values through Q, g and M in each of the remaining cases shows that the runs whose second value never lies on a middle channel end on L_1, L_2 and L_4 with largest touch counts 5, 6 and 7: the largest value starts on x or in another subtree than the second value, or the second value starts on x, and the second value reaches L_j or channel 14 and loses there to the largest value. The runs that start in the sibling pair of P'_1 leave the second value on L'_1, where the first merge of the minimum moves it to the channel $\max(p, L'_1)$, with touch count 5. These end channels are distinct from each other and from the middle channels other than p, and contribute $\frac{1}{16} + \frac{1}{8} + \frac{1}{4} + \frac{1}{16} = \frac{1}{2}$ times 2^β. With $|C| \leq 28$ no bound v_2 is clamped at 0; otherwise clamping only raises the sum. □

Lemma 8.4. *If p is a* hi *output and a class of polarity 0 lies on p after C, then* (H2) *cuts for polarity 0.*

Proof. Let E_p be the value of E of the class on p. The last comparator of C met this class, so $E_p \geq 2$. Take a run of the class with E_p exclusive comparators. Its two values met in Q, so its largest value is on L_j for some j, and channel 14 holds a smaller value. Its second value therefore enters g, moves to channel 14, passes the merges with the L_i before L_j, loses at $(14, L_j)$ and ends on L_j, with $W_2 = 4 + E_p + e_j$, where $e_1 = 1$, $e_2 = 2$, $e_4 = 3$. Compared with the proof of Lemma 8.3, the term $2^{E_p}/32$ of the class leaves the middle, and the term 2^{e_j-5} of L_j rises to at least $2^{E_p+e_j-5}$. The sum is therefore at least

$$2^\beta\Big(\frac{\Phi_0}{32} + \frac{1}{2} + \frac{2^{e_j}(2^{E_p} - 1) - 2^{E_p}}{32}\Big) \;\geq\; 2^\beta\Big(1 + \frac{2^{E_p} - 2}{32}\Big) \;>\; 2^\beta,$$

by $\Phi_0 \geq 16$, $e_j \geq 1$ and $E_p \geq 2$. □

The mirror statements hold for polarity 1, with the same counts: a class of polarity 1 on a lo port follows the running minimum and ends on the hi output of the merge with the lead L'_j that holds the smallest value, and the runs on the sibling pair of P_1 end on L_1 with touch count 5. A hi output carries no class of polarity 1 and a lo output none of polarity 0. Thus, for a hi port, polarity 1 falls under the mirror of Lemma 8.3, and for a lo port polarity 0 falls under Lemma 8.3 itself. Hence a state survives (H2) on both polarities only if its port is dead and its cone *conserves* both potentials, $\Phi_0 = \Phi_1 = 16$. Let $\mathcal{F}$ be the set of cones that conserve both potentials and whose last comparator has a dead output.

8.4. **The conserving family.** After Q, the six middle channels other than u_4 and v_4 carry one class of each polarity, with $E = 1$; the channel u_4 carries one class of polarity 0 only and v_4 one class of polarity 1 only, both with $E = 2$. After a comparator of a cone, its hi output carries only classes of polarity 0 and its lo output only classes of polarity 1. Hence a comparator of a cone in $\mathcal{F}$ is of one of four types: D, on two of the six channels described above that no comparator of the cone has touched yet; M_0, on two leads that carry classes of polarity 0 only, with equal E, whose lo output is then dead; M_1, the mirror, whose hi output is dead; and X, on two dead leads. Every other comparator raises Φ_0 or Φ_1. Two sequences of comparators are *equivalent* if one arises from the other by repeatedly exchanging adjacent comparators on disjoint channels.

Lemma 8.5. *Every cone in $\mathcal{F}$ of an admissible state is equivalent to a cone C_0N, where the touching part C_0 consists of at most 3 comparators of type D and $m \le 6$ of types M_0 and M_1, and N is a network of type X on the m dead channels of C_0 whose comparators are useful and are predecessors of the port, which is an output of the last comparator of N, or of C_0 if N is empty. The network N has at most $\binom{m}{2}$ comparators, so every such cone has at most 24 comparators.*

Proof. A comparator with a dead input and an input that carries a class is not of the four types, so inside $\mathcal{F}$ dead leads meet only dead leads, and a dead lead stays dead. Dead leads arise only as the lo output of M_0 and the hi output of M_1. Each D uses two of the six channels, so there are at most 3. An M_0 merges two leads that carry classes of polarity 0 only, namely u_4 and hi outputs of D and M_0; their number starts at 1, rises by one at each D, falls by one at each M_0 and stays at least 1, since the hi output of an M_0 is again such a lead; so $\#\mathrm{M}_0 \le \#\mathrm{D} \le 3$, and $\#\mathrm{M}_1 \le \#\mathrm{D}$ by the mirror argument. Thus $m \le 6$. At every moment the dead channels are used only by comparators of type X and the others only by touching comparators, so every comparator of type X commutes with every later touching comparator, and moving them all to the end gives the same circuit.

A comparator of type X is useful for the set before it exactly when it is useful for the projection of that set onto the dead channels, since usefulness depends only on the two channels of the comparator. For $Y \subseteq \{0,1\}^m$ let $\mathrm{Inv}(Y)$ be the set of pairs $a > b$ of channels such that some $v \in Y$ has $v_a = 0$ and $v_b = 1$. A standard comparator (i, j) that is useful for Y removes (i, j) from Inv. For $k > i$, the pair (k, i) lies in Inv afterwards only if (k, i) or (k, j) did before, and (k, j) only if both did; for $l < j$, the pair (j, l) lies in Inv afterwards only if (i, l) or (j, l) did before, and (i, l) only if both did; every other pair lies in Inv afterwards only if it did before. Thus $|\mathrm{Inv}|$ drops by at least 1 at each useful comparator, and a sequence of useful comparators on m channels has at most $\binom{m}{2}$ comparators. Hence $|C| \le 3 + 6 + \binom{6}{2} = 24$. □

Equivalent cones have the same circuit, so they produce the same set and the same runs and receive the same answers from every test. Usefulness is unchanged too since, as in the proof of Lemma 8.5, it depends only on the projection of the current set onto the two channels of the comparator, and in a cone the last comparator is the only one on which no other depends, so it stays last. Fix an order of the 28 pairs of middle channels. The enumeration lists the touching part and the network N separately, each only as a sequence in which a comparator that shares

no channel with its predecessor in the same part comes later in this order; the first comparator of N is not compared with the last of the touching part. The lexicographically least sequence of the equivalence class of the touching part, followed by that of the equivalence class of N, has this property and lists an equivalent cone.

8.5. **Push-out and pull-in bounds.** For admissible states with cones in $\mathcal{F}$ and a dead port the cones have at most 24 comparators, so by Lemma 8.3 the sum of the run form of (H2) is exactly 2^β, and the run form of (H4), hence (H4), cuts polarity 0 when

$$H(y) = \sum_z 2^{b_3(y,z)-v_2(y)} > 1 \quad \text{for some } y,$$

the sum over the ends z of runs, with b_3 of the run form. Since $v_2(y) = \beta + W_2(y) - 9 \geq 4$ is not clamped, $2^{v_3-v_2} \geq T_3(z) := 2^{W_3-W_2-6}$ and $2^{v_4-v_2} \geq b(z,u) := 2^{W_4-W_2-10}$, so $H(y) \geq \sum_z \max(T_3(z), \mathrm{rnd}(\sum_u b(z,u)))$, where $\mathrm{rnd}(a)$ is the least power of 2 that is at least a.

Appendix B defines, for each polarity, two quantities that a partial network N already determines, each a lower bound on H for one kind of port, the *push-out bound* Ψ and the *pull-in bound* $\bar{\Psi}$. They come from a *frozen evaluation* of the runs, in which the values that lie on dead channels when g begins stay where they are and the others continue through g and M with a dead channel in place of the port; we write $\Psi_0, \bar{\Psi}_0$ and $\Psi_1, \bar{\Psi}_1$ for the two polarities, and H_0, H_1 for those of H. By Lemma B.1, $H_0 \geq \Psi_0$ and $H_1 \geq \bar{\Psi}_1$ if C_0N is a cone in $\mathcal{F}$ and the port is the lo output of the last comparator of N, $H_0 \geq \bar{\Psi}_0$ and $H_1 \geq \Psi_1$ if it is the hi output, and for a fixed touching part the four bounds do not decrease when a comparator of type X is appended to N. Hence, if a partial network N has $\max_y \Psi_0 > 1$ or $\max_y \bar{\Psi}_1 > 1$, and also $\max_y \bar{\Psi}_0 > 1$ or $\max_y \Psi_1 > 1$, then (H4) cuts every completion of N, whichever output of its last comparator is the port.

8.6. **The computation.**

Proposition 8.6. *For every late start and every cone C with at least 4 comparators, each complete state $QCgM$, for either port, that is admissible satisfies $s(X_{QCgM}) > 32 - |C|$.*[†]

Proof. By Lemmas 8.3 and 8.4 and their mirror statements, (H2) cuts every state whose cone is not in $\mathcal{F}$ or whose port is not dead. For the cones in $\mathcal{F}$, a computation enumerates, for each late start, the touching parts and then the networks N in the normal form of Lemma 8.5, in the order above and with useful comparators only, and lists only admissible states. It prunes N, by Lemma B.1, as soon as each kind of port has a bound above 1 for some y: for a lo port Ψ_0 or $\bar{\Psi}_1$, for a hi port $\bar{\Psi}_0$ or Ψ_1. It visits 11,138,216 partial networks, of which 8,769,051 are pruned. The computation leaves 730,583 states to the tests, the 727,139 not settled by the bounds and the 3,444 with an empty network; of these, 726,269 are cut by (H4) and the other 4,314 by the raised trivial weights of (T2), with at most three raises. A cone in $\mathcal{F}$ whose network N is empty ends with a comparator of type M_0 or M_1 and has 2, 3 or 6 comparators. Those with 2 or 3 are covered by Proposition 7.2, and those with 6, which have three dead channels, are listed by the computation as states with an empty network, all cut by (H4). These counts are those of the program that produced the certificates; Section 10 explains how the proposition is proved and checked. □

Proof of Theorem 1.2. Suppose that a sorting network on 15 inputs with 54 comparators exists. Theorem 5.4 gives a start Q. If Q is complete, Proposition 6.2 gives $s(X_Q) > \beta$, against Theorem 5.4. If Q is late, Lemma 7.1, applied to the network QR of Theorem 5.4, whose red prefix is Q, gives a cone C and an admissible complete state $QCgM$ with $s(X_{QCgM}) = 32 - |C|$, against Propositions 7.2 and 8.6. Hence no sorting network on 15 inputs has 54 comparators; since $S(15) \geq 54$, it follows that $S(15) \geq 55$. □

9. Certifying the 9-channel values

The core bounds of Section 6 and of the test (T3) are lower bounds on the costs of 11-channel sets, each proved by an instance of the weighted two-value bound on 11 channels whose weights are values of 9-channel sets. The certificates list, for every such claim, the values of the 9-channel cores of all its scenarios on both polarities, and the checker of Section 10 decides each instance again with `lb2cr` from these values. We call these values the *facts*; each is a lower bound $s(K) \geq v$ for a set K on 9 channels. The certificates of the 151 complete starts and the root certificates of (T3) contain facts for 2,332,160 distinct sets, in 1,120,969 classes up to similarity; no set and no class carries two values. The check of a claim uses only the facts of the polarity on which its instance is infeasible. Each fact is certified by the first of three routes that applies, each of which proves a value at least the listed one:

- 1,102,718 facts follow from a step of Harder's certificate for $S(11) \geq 35$ [6], every step of which is checked by Lean's kernel [12, Section 14.2];
- 227,491 follow from an entry of the table of 9-channel values of [13], whose entries are proved by Lean's kernel;
- 1,001,951 are computed again by the search engine of Harder's program [8], run separately through a wrapper that reports a mismatch unless the computed value is at least the listed one, and it reports none.

A step of the certificate or an entry of the table is a pair (Y, v) stating that every network that works on Y has at least v comparators, and a separate program checks that a similarity maps Y into K; then $s(K) \geq v$, as in [14, Section 9]. With the facts certified, every claim of a core bound in the certificates rests only on `lb2cr` and these routes.

10. Replay and trusted components

10.1. **The proof of $S(15) \geq 55$.** A second program enumerates the configurations of the two trees allowed by Lemmas 5.2 and 5.3 and the cases for g and its port, builds the red prefix for each, and reduces the sets with Harder's canonical form [8]; it gives the same 177 classes with the same lengths as the red-prefix search of Theorem 5.4.

The replays use programs that were written from the format of the certificates and from the mathematics of Sections 5 to 8, without the code that produced the certificates, the *producer*; they share only `lb2cr` with it. We call such programs *independent*.

- *Complete starts.* The replay is that of [14, Section 10.1], extended to 15 channels. For each of the 151 searches it regenerates all children of the kept states with its own canonical form for sets, recomputes extremal channels, pruned sets, scenarios and trivial weights, checks every listed core, and decides every recorded instance with `lb2cr`. It decides every claim of a core bound again from the listed 9-channel values, all of which are certified. It refutes all 151 starts, 35 of them with trivial weights only. A separate program checks that the first state of each certificate is its start, with the same comparators, and that each start has exactly one certificate.
- *Cones with at most* 3 *comparators.* An independent enumeration, based on the definition of a cone, finds the same complete states, with the same sets for every late start and every size of the cone. Independent implementations of (T1) and (T2), run with the producer's settings on these states, cut as many of them by each test as Table 5 lists, size by size, and leave exactly the same states. A third program maps each of these onto one of the 321 representatives with the same number of comparators, by a permutation of the channels possibly combined with complementation, found without the producer's canonical form, and checks each image bit for bit. The representatives are replayed in

root mode: the replay requires a valid cut for every listed state, with its exact prefix, and decides each of the 1,117 claims of (T3) again from certified 9-channel values.

- *Cones with at least* 4 *comparators.* Proposition 8.6 is proved by the Lean theorem `Sortnet.S15.FCN.fclosure` of Section 11; the producer's computation and an independent replay are cross-checks. The replay's bounds give the same values, to the printed precision of 10^{-6}, at all listed partial networks, its tests cut all listed states, and its search, with the same pruning, finds no state outside the list other than those whose cone has 2 or 3 comparators and an empty network on dead channels, which Proposition 7.2 covers. An independent enumeration of the family $\mathcal{F}$, based on the normal form, lists the producer's states at sizes 4 to 8, and independent implementations of (H2), (H4) and (T2) cut every state of these sizes.

10.2. **The proof of $S(16) = 60$.** The finite check of Proposition 3.6 is repeated, up to the middle sets, by an independent program based on the statements of Section 3 and a specification of the checks. For all 16 representatives, again without symmetry reduction, it confirms the conclusions of Lemma 3.3, enumerates the labelled choices of G_1, recomputes K_3 and K_3', the forced trees and the finality of the channels 0 to 2 and 13 to 15, and obtains the counts in the last three columns of Table 2; for representative 0, the output sets of its prefixes $G_0G_1G_2$ agree, as a multiset of hashes of the full sets, with those of a run of the original program without symmetry reduction, and its 128 distinct middle sets agree with that run's bit for bit. By a permutation of the channels 3 to 12, possibly with complementation, it maps each of the 648 distinct middle sets explicitly onto one of the 96 canonical sets and checks the image bit by bit; the canonical sets come from a run of the original program with symmetry reduction, and the same two-level bound refutes each of them; in every case the budget of the set is at most the budget at which its class is refuted. The exact costs of the 4,295 sets on 6 channels are computed again by a second program with a different method, a memoized depth-first recursion with a witness network for each value; all values agree. As a further check, an exact search for 8-channel sets written from scratch computed the costs of all 4,884 cores on 8 channels of the 648 distinct middle sets and of the 96 canonical sets, the 3,871 of Section 3.4 together with those of the canonical sets, each with a witness network; with these values `lb2cr` refutes every middle set on both polarities.

10.3. **The proof of $S(20) \geq 81$.** Another independent program, written in the same way, repeats the computations of Section 4 [16]. It confirms the tree B_{20}, its forced depths and the runs of Lemma 4.3; it reproduces the 28,350 trees of the minimum and the joint distribution of K_2 and K_2', and, compared as networks, the 270 trees with $K_2 = K_2' = 1$; it obtains the number of choices of G_1 in Proposition 4.6, with the checks of the runs of two values described after Proposition 4.8; and it finds $K_3 \geq 2$ and $K_3' \geq 2$ for every choice, computing the third stage twice, by its own form of Lemma 4.7 and by a memoized direct simulation of every prefix G_0G_1. It also checks $\lceil \log_2 F(N) \rceil = 10$ for $17 \leq N \leq 20$ against [12, Table 1] and recomputes the bounds for 17, 18 and 19 inputs. For the search of Proposition 4.1, the published programs `abs9f` and `abs9g` were rebuilt and gave the same outputs; for the skeletons other than C^* this only reruns the same code.

Three programs confirm without a symmetry reduction that C^* admits no configuration feasible at level 10. The first is `abs9f` itself, with its depth-first search split into 256 shares, two of which were split again into 96 shares each, so that 446 shares together cover the search, as in [12, Section 4.4]. The second is an exact solver written independently for this check: it enumerates the sources of the pass-throughs, of the root of B_{20} and of the merges on which the costs through the ports depend, and decides the remaining sources as an exact packing problem [16]. The second also finds, on the same tree without pass-throughs, exactly the configuration of Proposition 4.1 with its forced depths. The third, a separate implementation of the whole search written for the

check of the remark on 27 inputs in Section 4, decides all 102,739 skeletons at level 10, finds exactly one feasible configuration, and finds none with the skeleton C^* [16]. With the symmetry test of [12, Section 4.3], the program `abs9g` gives the same result for the whole search in a few seconds.

10.4. **Trusted components.** The proofs of Theorems 1.1 to 1.3 rest on Sections 2 to 9, on the computations (the starts, the searches for the complete starts, the tests of the cones, the finite checks at 16 and 20 inputs and the search of Proposition 4.1) and their replays, and, for Proposition 8.6, on the Lean theorem of Section 11. These rely on the following components, which they do not check.

(1) *The program* `lb2cr`, for each answer infeasible; it is proved correct for all budgets used here (Section 2 and [12, Sections 7 and 8]).
(2) *The search engine of Harder's program* [8], for the facts of the third route of Section 9; Harder proved its correctness for well-behaved sets [7], and our sets are well-behaved [14, Lemma 2.1(c)].
(3) *The lower bounds* $S(10) \geq 29$, $S(11) \geq 35$, $S(12) \geq 39$, $S(13) \geq 45$ and $S(14) \geq 50$, which enter through trivial weights and the deletion argument: $S(14)$ through the starts and the deletion argument, $S(12)$ through the trivial weights of the complete starts, and $S(13)$, $S(12)$, $S(11)$ and $S(10)$ through the tests of the late starts; the proof of Theorem 3.1 uses $S(14)$ and $S(13)$.
(4) *Harder's certificate and the table of* [13], for the facts derived from them; both are checked by Lean's kernel.
(5) *The published programs* `abs9f` *and* `abs9g` *of* [12], for the search of Proposition 4.1; a separate implementation of the search repeats it for all skeletons, and an independent exact solver also settles C^* (Section 10.3).

Beyond these, we trust, in the usual way, the program that computes the starts, the replays, the programs that check the maps, the compilers and the hardware, and, for Proposition 8.6, Lean's kernel and the compiled checks of Section 11, but not the programs that produced the certificates, the search programs or their instance solvers other than `lb2cr`. Section 11 states what the proofs in Lean trust.

10.5. **Tests and controls.** The following checks run the programs on known cases and on planted errors, and compare them with separate computations.

- After the first 49 comparators of Green's network, the channels 3 to 12 need exactly 11 more comparators, the number Green uses, and the weighted two-value bound with the exact costs of the 8-channel cores as weights gives 11 as well.
- The breadth-first search for the exact costs of 6-channel sets returns 3, 5, 9 and 12 on the full cubes of 3, 4, 5 and 6 channels, the values of $S(3)$ to $S(6)$, and reaches 19,334 sets from the cube of 6 channels; the exact search for 8-channel sets returns $S(3)$ to $S(7)$.
- At 20 inputs, Lemma 4.7 agrees with direct simulation of the whole prefix G_0G_1 on 8,505 random choices, and the direct simulation agrees with a separate program, which reads E_3 off the set of the prefix, on 3,510 further choices. The search of [12, Section 4] reproduces the single configuration of 16 inputs at level 9.
- The red-prefix search and the program that enumerates the configurations of the two trees both reproduce the 5,271 starts of the 14-input case.
- On the known sorting network on 15 inputs with 56 comparators, with target 56, the Huffman tests of Section 8 cut none of its prefixes that have a single extremal channel.
- The second implementation of the raised trivial weights agrees with the original one on 2,272 stored states and catches planted errors. The replay of the complete starts rejects

a certificate with a raised weight, and in root mode it catches a deleted line, a raised weight and an extra state. Removing half of the certified 9-channel facts, or lowering every value by one, makes the replay report uncertified claims.

- For the closure, bounds recomputed from scratch at 4,467 nodes, every 2,500th listed partial network of each start, agree with the listed ones, and all 16,897 states of sizes 6 to 8 on which the run form of (H4) is known to fail lie among the states left open by the bounds.

11. Proofs in Lean

Theorems 1.1 to 1.3 are proved in Lean 4 [10] with Mathlib [11], in an extension of the development of our proof of $S(14) \geq 50$ [15]. The definitions of networks and of sorting are those of [12, Section 14.1], and the statements are

```
def S15Statement : Prop := forall net : Network 15, net.Sorts -> 55 <= net.length
def S16Statement : Prop := forall net : Network 16, net.Sorts -> 60 <= net.length
def S20Statement : Prop := forall net : Network 20, net.Sorts -> 81 <= net.length
```

The Lean proof of $S(14) \geq 50$, `Sortnet.S14.S14_ge_50_N`, has two hypotheses: `coreTableC.Valid`, which states 74,705 lower bounds on the costs of 10-channel sets, and `(Wed.TLf i).Valid` for the 877 late starts of the 14-input case, which states the bounds on 10- and 11-channel sets that their certificates use; both are certified outside Lean [14, Sections 9 and 11]. Table 7 lists the five final theorems. Each has exactly these two hypotheses and no others: `S15_ge_55_N` gives `S15Statement`, `S16_ge_60_N` gives `S16Statement`, `S16_eq_60_N` gives `S16Statement` together with a sorting network on 16 inputs with 60 comparators, `S20_ge_81_N` gives `S20Statement`, and `table6Lo_N` gives the bounds of Table 6 for $17 \leq N \leq 27$. For example,

```
theorem S15_ge_55_N : coreTableC.Valid ->
    (forall i in Wed.done, (Wed.TLf i).Valid) -> S15Statement
theorem S16_of_S15 : S15Statement -> S14Statement -> S13Statement ->
    S16Statement
```

and `S16_ge_60_N` follows from `S15_ge_55_N` by `S16_of_S15` and the Lean theorems for 14 and 13 inputs. The five final theorems lie in five modules; the modules of `S16_ge_60_N`, `S16_eq_60_N`, `S20_ge_81_N` and `table6Lo_N` state `S15_ge_55_N` again under the same name, and the last three also `S16_ge_60_N`, so the modules are built and imported one at a time, and importing two of them together fails with a duplicate declaration.

Besides the three standard axioms `propext`, `Classical.choice` and `Quot.sound`, each final theorem depends on one axiom for each computation that runs as compiled code through `native_decide`, and on no other axiom; Table 7 gives their numbers. Relying on these axioms means trusting Lean's compiler, the C compiler and the GMP library in addition to Lean's kernel. The upper bound in Theorem 1.1 is checked by Lean's kernel alone: the development defines Green's network, `Sortnet.S16.greenNet`, and proves that it sorts (`Sortnet.S16.green_sorts`) and has 60 comparators (`Sortnet.S16.green_length`) with standard axioms only, and three separate checkers of Lean proofs, `leanchecker`, `lean4lean` and `nanoda`, accept these proofs. The bounds $S(11) \geq 35$ and $S(13) \geq 45$ come from [12, 13], and $S(14) \geq 50$ comes from [15]. The bound $S(10) \geq 29$ comes from the kernel check in [13], whose modules the project reuses unchanged, of a certificate for 10 channels made with Harder's program, and $S(12) \geq 39$ follows from $S(11) \geq 35$ by Van Voorhis's bound [18] (`Sortnet.S15.lb39`).

The values of 9-channel sets read by the checkers enter as a table of 725,768 sets, each with a lower bound on its cost: 723,763 for the complete starts and 2,005 more for the late starts. The table has fewer entries than the facts of Section 9: each claim is checked on one polarity and reads one value for each scenario, while the certificates list both polarities. The theorem `Sortnet.S15.K9.TA.valid9_T9_all2`, which has no hypotheses, proves the whole table. Its

N	13	14	15	16	17	18	19	20	21	22
$P(2,N)$	9	9	9	9	10	10	10	10	11	11
[14, Table 6]	45	50	54	59	64	69	74	79	85	90
with Theorems 1.1 and 1.2	45	50	**55**	**60**	**65**	**70**	**75**	**80**	**86**	**91**
with Theorem 1.3	45	50	55	60	65	70	75	**81**	86	**92**
N	23	24	25	26	27	28	29	30	31	32
$P(2,N)$	11	11	11	11	11	12	12	12	12	12
[14, Table 6]	96	101	107	112	118	124	130	136	142	148
with Theorems 1.1 and 1.2	**97**	**102**	**108**	**113**	**119**	**125**	**131**	**137**	**143**	**149**
with Theorem 1.3	97	**103**	108	**114**	119	**126**	131	**138**	143	**150**

TABLE 6. Lower bounds on $S(N)$ from $S(13) = 45$ and $S(14) \geq 50$, Van Voorhis's bound and the two-value bound with the values $P(2,N) = \lceil \log_2 F(N) \rceil$ of [12, Theorem 1.2 and Table 1]. The first row of bounds is that of [14, Table 6]. The second adds Theorems 1.1 and 1.2; its bold entries exceed the first row by one, and the entry for $N = 16$ is the exact value. The third adds Theorem 1.3; its bold entries exceed the second row by one. The entries for $17 \leq N \leq 27$ are also derived in Lean from $S(16) \geq 60$ (Section 11).

theorem	proves	compiled checks
`S15_ge_55_N`	`S15Statement`	269,587
`S16_ge_60_N`	`S16Statement`	269,608
`S16_eq_60_N`	`S16Statement` and a sorting network with 60 comparators	269,608
`S20_ge_81_N`	`S20Statement`	269,641
`table6Lo_N`	Table 6 for $17 \leq N \leq 27$	269,644
`Sortnet.S15.tierA`	the starts at 15 inputs	0
`Sortnet.S15.TBN.tierB_N`	the complete starts	224,086
`Sortnet.S15.TL.tierL_NB`	the late starts	569
`Sortnet.S15.FCN.fclosure`	Proposition 8.6	754
`Sortnet.S15.K9.TA.valid9_T9_all2`	the table of 9-channel values	1,078
`Sortnet.S14.wedN_all`	the late starts at 14 inputs	41,376
`Sortnet.S13_ge_45`	$S(13) \geq 45$	1,724
`S16_of_S15`	Theorem 3.1	21
`Sortnet.S20.S20_of_S16`	`S20Statement` from `S16Statement`	33
`Sortnet.T6.table6Lo_of_S16`	Table 6 for $17 \leq N \leq 27$ from $S(16) \geq 60$	3

TABLE 7. The final theorems in Lean with all the compiled checks they depend on, and their parts with the compiled checks that each adds. `S15_ge_55_N` uses the first seven parts; `S16_ge_60_N` and `S16_eq_60_N` add `S16_of_S15`, `S20_ge_81_N` adds `Sortnet.S20.S20_of_S16` to these, and `table6Lo_N` adds `Sortnet.S20.S20_of_S16` and `Sortnet.T6.table6Lo_of_S16`.

entries come from Harder's certificate for $S(11) \geq 35$ (318,545 entries), from the table of [13] (56,484), and from a search in Lean whose steps a proved checker verifies (350,739). The validity of the table of [13], `Sortnet.S15.T9K.t9k_valid`, and the links to it are checked by Lean's kernel alone, and `leanchecker` accepts them; the links to Harder's certificate and the steps of the search are 1,078 compiled checks. Outside Lean, 371 modules of this proof, those of the search

and of its checker, are accepted by three separate kernels, `leanchecker`, `lean4lean` and `nanoda`, and the 729 modules that assemble the table or carry its data are accepted by `leanchecker` alone; a deliberately unsound control module is rejected in each run.

The proof of $S(15) \geq 55$ has three parts, composed in `Sortnet.S15.mainN_T9all2`. The first, `Sortnet.S15.tierA`, derives `S15Statement` from $S(14) \geq 50$ and the refutation of all 177 starts; as for 14 inputs, the kernel checks a certificate of the red-prefix search in place of the computation of Theorem 5.4, and this part uses only the standard axioms. The second, `Sortnet.S15.TBN.tierB_N`, refutes the 151 complete starts with their certificates. The kernel checks the certificates, the links between them and every lemma; it recomputes from the certificate data the instance of every recorded cut and of every claim of a core bound and checks that it equals the recorded one. Only the refutations of these instances run as compiled code, through `native_decide` on the solver of [12]: 21,411 instances of (C2), namely the 18,340 recorded cuts of Section 6 and 3,071 cuts with trivial weights that form new classes, and 202,675 claims of core bounds, one for each distinct 11-channel set. The 9-channel values that these claims read are those of `Sortnet.S15.K9.TA.valid9_T9_all2`. The third, `Sortnet.S15.TL.tierL_NB`, refutes the late starts. Its cones with at most 3 comparators are enumerated by compiled code whose completeness is proved and are cut at the root, partly through records converted from the root certificates of (T3). Its cones with at least 4 comparators are refuted by `Sortnet.S15.FCN.fclosure`, which enumerates them again in the normal form of Lemma 8.5, computes exact integer versions of the bounds of Section 8, reads none of the producer's data, and cuts every leaf by tests whose soundness is proved in Lean, with one compiled check for each of 29 chunks of each late start. Lean also proves the statements on which this enumeration rests: every cone that (H2) does not cut lies in $\mathcal{F}$ (Lemmas 8.3 and 8.4; `Sortnet.S15.FC.DP.inFOfNotH2`), every cone in $\mathcal{F}$ has the normal form of Lemma 8.5 (`Sortnet.S15.FC.reached_of_inF`), the bounds of Lemma B.1 hold (`Sortnet.S15.FC.R3.pushOutStmt` and `Sortnet.S15.FC.R3.PI.pullIn_stmt`), and an accepted run of the checker refutes every cone in normal form (`Sortnet.S15.FC.closureConcl_of_run`).

The implication `S16_of_S15` runs the finite check of Proposition 3.6 as 21 closed Boolean checks over fixed data, each by `native_decide`: the certificate that gives the 16 representatives of G_0, the witnesses of the second stage, three tables of bounds, and the test of the third stage on every labelled choice of G_1 for each representative. The forcing of the choices of G_1 and the links between the checks and the statement are proved with the standard axioms. For the exact 6-channel values only the lower bounds are needed: a passing breadth-first search gives a lower bound for every network that works on the set (`Sortnet.S16.bfs_sound` and `Sortnet.S16.lb_of_std`), and one of the 21 checks runs the search for the 407 cores that the other checks look up, with values from 6 to 12. The bound $S(20) \geq 81$ is the theorem `Sortnet.S20.S20_of_S16`, which derives `S20Statement` from `S16Statement`. In it, $S(17) \geq 65$, $S(18) \geq 70$ and $S(19) \geq 75$ follow from $S(16) \geq 60$ by Van Voorhis's bound, proved in Lean; the search of Proposition 4.1, which uses the balance inequality of Lemma 4.2, and the checks of Propositions 4.6 and 4.8 run as 33 compiled checks. The Lean version of Lemma 4.2, `Sortnet.S20.Cf.bal_core`, is proved with the standard axioms and also covers the weak form of a configuration, with Kraft's inequality in place of the equality and without acyclicity; a cycle, like a dead source, then gives $\tau = \infty$. The bounds of Table 6 for $17 \leq N \leq 27$ are derived as well: `Sortnet.T6.table6Lo_of_S16` derives them from $S(16) \geq 60$, by Van Voorhis's bound, checked by the kernel, at odd N and at $N = 18$, and at the other even N by Theorem 1.3 and one two-value search at each of $N = 22$, 24 and 26. The entries for $28 \leq N \leq 32$ rest on the values $P(2, N) \geq 12$ for $28 \leq N \leq 32$ of [12, Theorem 1.2], which are not proved in Lean.

12. Consequences for larger networks

By [12, Lemma 2.3], $S(N) \geq S(N-2) + P(2,N)$, and $P(2,N) = \lceil \log_2 F(N) \rceil$ for $3 \leq N \leq 32$ [12, Theorem 1.2], where F is Van Voorhis's function (Section 4.1). Together with Van Voorhis's bound, the recursion of [12, Section 15] gives lower bounds on $S(N)$ for larger N from the known values, taking the larger of the two bounds. Started from $S(13) = 45$, $S(14) \geq 50$, $S(15) \geq 55$ and $S(16) = 60$, it gives the second row of bounds in Table 6. For $N = 17$, for example, it gives $\max(60+5,\, 55+10) = 65$. For $15 \leq N \leq 32$ these bounds are one more than those of [14, Table 6]. Theorem 1.3 raises the bound for $N = 20$ by one more, and through the two-value bound every second bound above it: $S(22) \geq \max(86+5,\, 81+11) = 92$, and so on up to $S(32) \geq 150$.

13. Concluding remarks

The cases of 14 and 15 inputs remain open; at both, the known bounds leave slack at the first two stages (Section 3.5). Deciding these two cases seems to need a search like that of Sections 6 to 8 for networks with one comparator more than those excluded so far, that is, with 50 comparators on 14 inputs and 55 on 15.

Appendix A. Computations

Table 8 lists the computations in vCPU-hours. A vCPU-hour is one virtual processor for one hour of wall-clock time; the first column counts the processors of each machine for the time that the computation held it, and the second the processor time measured while the programs ran. Where the records show them, reruns and lost runs are included.

Appendix B. The push-out and pull-in bounds

The number $H(y)$, the formulas $T_3(z) = 2^{W_3 - W_2 - 6}$ and $b(z,u) = 2^{W_4 - W_2 - 10}$, and the rounding rnd are those of Section 8.5; below, $T_3(z)$ and $b(z,u)$ are computed from the touch counts of the frozen evaluation. Let C_0 be a touching part, a sequence of comparators of types D, M_0 and M_1 as in Lemma 8.5, and let N be a nonempty sequence of comparators of type X on the dead channels of C_0, not necessarily a cone, with last comparator c_N. Fix, for each touching part, a dead channel p_0 as a stand-in port. The *frozen evaluation* of a run follows its values through Q, C_0 and N; the values that lie on dead channels when g begins are *frozen* and stay where they are, and the others continue through g and M with p_0 in place of the port, carrying a value below all of them. It gives touch counts and ends, and from them numbers $T_3(z)$ and $b(z,u)$ in the same way. The ends of frozen values are recorded by their dead channel and those of the other values by their channel after gM, and the two kinds are kept apart even when they are the same channel. The ends of the merges of the minimum correspond, by their role, to those in the real state through a fixed bijection between $\{p_0, L_1', L_2', L_4'\}$ and $\{p, L_1', L_2', L_4'\}$, and y is read through this bijection. We write TGT for the ends of values that are not frozen and end on a channel of gM other than 14, and MID for the ends of values that are not frozen and end on a middle channel that gM does not touch. The *push-out bound* is

$$\Psi(y) = \sum_z \max\bigl(T_3(z),\ \mathrm{rnd}(T_4^{\mathrm{s}}(z))\bigr), \qquad T_4^{\mathrm{s}}(z) = \sum_u w(z,u)\, b(z,u),$$

with $w(z,u) = \frac{1}{2}$ if z is frozen and $u \in \mathrm{TGT}$, and $w(z,u) = 1$ otherwise. The *pull-in bound* is

$$\bar{\Psi}(y) = \sum_{z \text{ frozen}} \max\bigl(T_3(z), T_4^{\mathrm{p}}(z)\bigr) + \sum_{z \text{ not frozen}} \gamma(z) \max\bigl(T_3(z), \mathrm{rnd}(T_4^{\mathrm{l}}(z))\bigr),$$

computation	vCPU-hours	busy vCPU-hours
starts at 15 inputs	0.03	0.02
searches for the complete starts	1,251.2	839.4
cones with at most 3 comparators	19.3	14.2
cones with at least 4 comparators: the producer's closure	33.6	3.3
conserving family $\mathcal{F}$ at sizes 4 to 8, for the checks	271.2	199.6
certification of the 9-channel facts	$\approx$2,845	2,187.1
replays of the proof of $S(15) \geq 55$	74.0	65.2
finite check at 16 inputs	5.4	3.5
its replays	2.5	0.2
search and finite check at 20 inputs	$\leq$1.4	—
its replays	2.0	1.3
proofs in Lean: builds and rebuilds	61,750.4	—
their external re-check	2,550.5	—

TABLE 8. Computations, in vCPU-hours held and busy vCPU-hours. The row of the conserving family includes a superseded run and an unfinished exploration whose results are not used. The row of the cones with at most 3 comparators includes 10,400 larger cones that are not needed, and the busy hours of the root certificates of (T3), whose machine hours are in the row "searches for the complete starts". The search and finite check at 20 inputs ran as single-threaded jobs without a meter; their held hours are an upper bound from wall-clock times and time limits, and their busy hours were not recorded. The check of the remark on 27 inputs in Section 4 and its replay took 0.4 vCPU-hours. The rows of the proofs in Lean give held hours from the start and stop records of the machines, an upper bound that includes retries, reruns and idle time; the builds for the table of 9-channel values before the last build window account for 38,974.7 of them. Their busy hours were not recorded as a whole.

where $T_4^{\mathrm{p}}(z)$ sums $b(z,u)$ over the frozen u and the $u \in \mathrm{MID}$, without rounding, $T_4^{\mathrm{l}}(z)$ sums $w'(u)b(z,u)$ with $w'(u) = \frac{1}{2}$ for $u \in \mathrm{TGT}$ and 1 otherwise, and $\gamma(z) = \frac{1}{2}$ for $z \in \mathrm{TGT}$ and 1 otherwise. The mirror construction gives Ψ_1 and $\bar{\Psi}_1$ for polarity 1; we write Ψ_0, $\bar{\Psi}_0$ for the bounds above, and H_0, H_1 for the two polarities of H.

Lemma B.1. *(a) If C_0N is a cone in $\mathcal{F}$ and the port is the* lo *output of c_N, then $H_0(y) \geq \Psi_0(y)$ and $H_1(y) \geq \bar{\Psi}_1(y)$ for every y.*

(b) If C_0N is a cone in $\mathcal{F}$ and the port is the hi *output of c_N, then $H_0(y) \geq \bar{\Psi}_0(y)$ and $H_1(y) \geq \Psi_1(y)$ for every y.*

(c) For a fixed touching part, the four bounds do not decrease when a comparator of type X is appended to N.

We prove Lemma B.1 for polarity 0; the mirror argument gives polarity 1. The touching part C_0 and the network N are fixed, and runs, touch counts and ends are as in Section 8; we call the r-th largest value of a run its *rank-r* value. A *row* is the set of terms of $H(y)$ or of a bound with a given z, and a *cell* is the term $b(z,u)$ inside it. We use four facts.

(F1) The values of ranks 1 and 2 never lie on a dead lead, and the path of the rank-r value depends only on the values of rank less than r. Inside N the rank-3 value is the largest value present, so it leaves every comparator on the hi output; the rank-4 value leaves on the hi output unless the rank-3 value is on the other input.

(F2) At a comparator of type X, every run with a value on one of its inputs gains one touch.

(F3) Compare two evaluations of gM that differ only in one additional value entering at the port, with at most three other values present. At each step the channels that carry values in the first evaluation are those of the second and one more channel e, so every comparator touched in the second is touched in the first. While the partner of e carries a value, both outputs carry values in the first evaluation and only the hi output in the second, so e moves to the lo output: it follows the port and then the channel of the running minimum, whose partners in gM are 14, L'_1, L'_2 and L'_4. At most three of them carry a value, so some comparator on the path of e has an empty partner in both evaluations; it is touched only in the first. Hence the first touch count exceeds the second by at least 1.

(F4) For $x, a, a_1, a_2 \geq 0$: $\max(x, 2a) \geq a + x/2$ and $\max(2a_1, 2a_2) \geq a_1 + a_2$.

If no value enters gM at the port, the channel of the running minimum carries a value below all values of the run through g and the merges of the minimum, in the frozen evaluation and in the real state alike. Each merge of the minimum then sends a value on L'_j to its hi output and is touched exactly when L'_j carries a value, so the touches agree and the ends correspond through the fixed bijection above. The ends of frozen values are the same dead channels in both.

Proof of Lemma B.1(a) *for polarity* 0. Let the port be the lo output of c_N. By (F1) no rank-3 value is on it after c_N, and a rank-4 value is on it only if the rank-3 value was on the other input of c_N, so that the rank-3 value is frozen on the channel of the hi output. Call these runs *affected*. The other runs have the same touches and ends in the real state as in the frozen evaluation. In an affected run the rank-4 value enters gM; by (F3) its run gains at least one touch, and it ends on a channel of TGT, while the rank-3 value stays frozen. Thus, in the row of that frozen rank-3 end, the frozen cell at the port, with weight 1 in T_4^{s}, is replaced by at least doubled values on cells of TGT, where unaffected cells, with weight $\frac{1}{2}$, may also lie. By (F4), the sum over these cells of $\max(x, 2a)$ is at least $a + \frac{1}{2}\sum x$. Every other row is unchanged. The real sum over u in every row is therefore at least $T_4^{\mathrm{s}}(z)$, the terms $T_3(z)$ are unchanged, and rnd is monotone, so $H_0(y) \geq \Psi_0(y)$. □

Proof of Lemma B.1(b) *for polarity* 0. Let the port be the hi output of c_N. Two kinds of runs are affected: those whose rank-3 value is on the port, and those whose rank-4 value is on the port while the rank-3 value is elsewhere. In both, by (F3), the value on the port gains at least one touch and ends on a channel of TGT. In the first kind, split the runs by the real end $z^* \in$ TGT of the rank-3 value. Each part keeps its cells with frozen u or $u \in$ MID at distinct u, because frozen values stay and values in MID are not touched by gM, and every such cell and T_3 are at least doubled. At z^* the real row is at least each of its value over the runs not of the first kind and twice the part's value, so by (F4) it is at least the part's value plus half the former; and the former is at least $\max(T_3(z^*), \mathrm{rnd}(T_4^{\mathrm{l}}(z^*)))$ by the argument for the second kind below. Summing over the parts gives at least $\max(T_3, T_4^{\mathrm{p}})$ of the frozen row at the port plus $\gamma(z)\max(T_3(z), \mathrm{rnd}(T_4^{\mathrm{l}}(z)))$ for these rows. In the second kind, in a frozen row the cell at the port moves to a cell of TGT, at least doubled, outside the support of T_4^{p}; in a row that is not frozen it lands on a cell of weight $\frac{1}{2}$ in T_4^{l}, and (F4) applies. Rows that receive nothing keep their value, and every rounding acts on a real sum that is at least the bounded one. Hence $H_0(y) \geq \bar{\Psi}_0(y)$. □

Proof of Lemma B.1(c). The stand-in port depends only on the touching part. The weights depend only on whether a value is frozen and whether its end lies in TGT or MID, which a comparator of type X does not change. Frozen values move as in (F1), and rows or cells merge only at a comparator that touches all merged runs, by (F2). When rows z_1 and z_2 merge into z, the new T_3 is $2\max(T_3(z_1), T_3(z_2))$ and every kept cell sum at least doubles the larger of the two sums, so the new row value is at least twice the largest of the four parts and hence at least

the sum of the two old row values. The same holds after rounding, since $\mathrm{rnd}(2a) = 2\,\mathrm{rnd}(a)$, and for the unrounded maximum in the frozen rows of $\bar{\Psi}$. Cells that merge inside a row satisfy $2\max(b_1, b_2) \geq b_1 + b_2$ at equal weight. Rows that are not frozen change only through their cells with frozen u. □

Data and code availability

The certificates, the programs that produced and checked them, and their records are archived as [16]; the Lean proofs, with their data and build records, are archived as [17]. They use Lean 4.35.0-rc3 and Mathlib at commit `c55e6e7`.

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, *Twenty-five comparators is optimal when sorting nine inputs (and twenty-nine for ten)*, arXiv:1405.5754 [cs.DM], 2014.

[3] 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.

[4] B. Dobbelaere, *Smallest and fastest sorting networks for a given number of inputs*, web page, `https://bertdobbelaere.github.io/sorting_networks.html`, consulted 2026-11-01.

[5] 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.

[6] J. Harder, *Certificate of the minimal size of 11-channel sorting networks*, Zenodo, 2019, dataset, doi:10.5281/zenodo.4108365.

[7] J. Harder, *An answer to the Bose–Nelson sorting problem for 11 and 12 channels*, arXiv:2012.04400v3 [cs.DS], 2022.

[8] J. Harder, *Lower size bounds for sorting networks*, source code, 2020, `https://github.com/jix/sortnetopt`, commit `0b5d09c`.

[9] D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, 2nd ed., Addison-Wesley, Reading, MA, 1998.

[10] 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.

[11] 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.

[12] T. S. Tomar, *Sorting networks on 13 inputs need 45 comparators*, version 3, Zenodo, 2026, doi:10.5281/zenodo.23620051; all versions, doi:10.5281/zenodo.23504358.

[13] T. S. Tomar, *A kernel-checked Lean proof that sorting networks on 13 inputs need 45 comparators*, Zenodo, 2026, doi:10.5281/zenodo.23603571.

[14] T. S. Tomar, *Sorting networks on 14 inputs need 50 comparators*, version 2, Zenodo, 2026, doi:10.5281/zenodo.23718361; all versions, doi:10.5281/zenodo.23622888.

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

[16] T. S. Tomar, *Certificates and programs for "Sorting networks on 15 and 16 inputs need 55 and 60 comparators, and 20 inputs need 81"*, Zenodo, 2026, doi:10.5281/zenodo.24156536.

[17] T. S. Tomar, *Lean proofs for "Sorting networks on 15 and 16 inputs need 55 and 60 comparators, and 20 inputs need 81"*, Zenodo, 2026, doi:10.5281/zenodo.24156534.

[18] D. C. Van Voorhis, *An improved lower bound for sorting networks*, IEEE Trans. Comput. **C-21** (1972), no. 6, 612–613.

[19] 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.

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