Data Structures Lower Bounds and Popular Conjectures

Pavel Dvořák
Charles University, Prague, Czech Republic

Michal Koucký
Charles University, Prague, Czech Republic

Karel Král
Charles University, Prague, Czech Republic

Veronika Slívová
Charles University, Prague, Czech Republic

Abstract

In this paper, we investigate the relative power of several conjectures that attracted recently lot of interest. We establish a connection between the Network Coding Conjecture (*NCC*) of Li and Li [25] and several data structure problems such as non-adaptive function inversion of Hellman [19] and the well-studied problem of polynomial evaluation and interpolation. In turn these data structure problems imply super-linear circuit lower bounds for explicit functions such as integer sorting and multi-point polynomial evaluation.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Data structures, Circuits, Lower bounds, Network Coding Conjecture

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.39

Related Version *Full Version*: `https://arxiv.org/abs/2102.09294`

Funding The authors were partially supported by Czech Science Foundation GAČR grant #19-27871X. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 823748. The fourth author was supported by Charles University project UNCE/SCI/004.

Acknowledgements We would like to thank to Mike Saks and Sagnik Mukhopadhyay for insightful discussions.

1 Introduction

One of the central problems in theoretical computer science is proving lower bounds in various models of computation such as circuits and data structures. Proving super-linear size lower bounds for circuits even when their depth is restricted is rather elusive. Similarly, proving polynomial lower bounds on query time for certain static data structure problems seems out of reach. To deal with this situation researchers developed various conjectures which if true would imply the sought after lower bounds. In this paper, we investigate the relative power of some of those conjectures. The Network Coding Conjecture (*NCC*) of Li and Li [25] attracted recently lot of attention and it was used to prove various lower bounds such as lower bounds on the size of circuits computing multiplication [3] and the number of IO operations needed for external memory sorting [13].

Another problem that is popular in cryptography is a certain data structure type problem for function inversion [19]. Corrigan-Gibbs and Kogan [9] observed that lower bounds for the function inversion problem imply lower bounds for logarithmic depth circuits. A similar more general observation was made by Viola [35]. In this paper, we establish a new connection between the function inversion problem and NCC. We show that NCC implies certain weak lower bounds for the inversion data structure problem. That in turn implies the same type of circuit lower bounds as given by Corrigan-Gibbs and Kogan [9]. We show that similar results apply to a host of other data structure problems such as the well-studied polynomial evaluation problem or the Finite Field Fourier transform problem. Corrigan-Gibbs and Kogan [9] gave their circuit lower bound for certain apriori undetermined function. We establish the same circuit lower bounds for sorting integers which is a very explicit function. Similarly, we establish a connection between data structure for polynomial evaluation and circuits for multi-point polynomial evaluation. Our results sharpen and generalize the picture emerging in the literature.

The data structures considered in this paper are static, non-adaptive, and systematic, i.e., a very restricted class of data structures for which lower bounds should perhaps be easier to obtain. Such data structure problems have the following structure: Given the input data described by N bits, create a data structure of size s. Then we receive a single query from a set of permissible queries and we are supposed to answer the query while non-adaptively inspecting at most t locations in the data structure and in the original data. The non-adaptivity means that the inspected locations are chosen only based on the query being answered but not on the content of the inspected memory. We show that when $s \geq \omega\big(N/\log\log N\big)$, polynomial lower bounds on t for certain problems would imply super-linear lower bounds on log-depth circuits for computing sorting, multi-point polynomial evaluation, and other problems.

We show that logarithmic lower bounds on t for the data structures can be derived from NCC even in the more generous setting of $s \geq \varepsilon N$ and when inspecting locations in the data structure is for free. This matches the lower bounds of Afshani [3] for certain circuit parameters derived from NCC. One can recover the same type of result they showed from our connection between NCC, data structure lower bounds, and circuit lower bounds. In this regards, NCC seems to be a stronger assumption than that certain functions require large boolean circuits or inefficient data structures. One would hope that for the strongly restricted data structure problems, obtaining the required lower bounds should be within our reach.

Our technique seems applicable to data structure problems that are *involutions* that are inverses of themselves. Although we use a lot of the same technical machinery as the previous papers on NCC our proofs involve new ideas. An interesting aspect of our proofs is that they apply the hypothesized data structure twice in the reductions. This is reminiscent of many quantum algorithms that use Hadamard transform twice in a row.

Organization. The statement of our main results is in Section 4. In the next section, we review the data structure problems we consider. Then we provide a precise definition of Network Coding Conjecture in Section 3. In Section 5 we prove our main result for function inversion. In Section 6 we discuss the connection between data structure and circuit lower bounds for explicit functions. Some of the proofs are left for the full version [12].

2 Data Structure Problems

In this paper, we study lower bounds on *systematic data structures* for various problems – function inversion, polynomial evaluation, and polynomial interpolation. We are given an input $I = \{x_0, \ldots, x_{n-1}\}$, where each $x_i \in [n] = \{0, \ldots, n-1\}$ or each x_i is an element of some field $\mathbb{F}$. First, a data structure algorithm can preprocess I to produce an advice string $\mathbf{a}_I$ of s bits (we refer to the parameter s as *space* of the data structure $\mathcal{D}$). Then, we are given a query q and the data structure should produce a correct answer (what is a correct answer depends on the problem). To answer a query q, the data structure $\mathcal{D}$ has access to the whole advice string $\mathbf{a}_I$ and can make t probes to the input I, i.e., read at most t elements from I. We refer to the parameter t as the query time of the data structure.

We consider non-uniform data structures as we want to provide connections between data structures and non-uniform circuits. Formally, a non-uniform systematic data structure $\mathcal{D}_n$ for an input $I = \{x_0, \ldots, x_{n-1}\}$ is a pair of algorithms $(\mathcal{P}_n, \mathcal{Q}_n)$ with oracle access to I. The algorithm $\mathcal{P}_n$ produces the advice string $\mathbf{a}_I \in \{0,1\}^s$. The algorithm $\mathcal{Q}_n$ with inputs $\mathbf{a}_I$ and a query q outputs a correct answer to the query q with at most t oracle probes to I. The algorithms $\mathcal{P}_n$ and $\mathcal{Q}_n$ can differ for each $n \in \mathbb{N}$.

2.1 Function Inversion

In the function inversion problem, we are given a function $f : [n] \to [n]$ and a point $y \in [n]$ and we want to find $x \in [n]$ such that $f(x) = y$. This is a central problem in cryptography as many cryptographic primitives rely on the existence of a function that is hard to invert. To sum up we are interested in the following problem.

Function Inversion

Input:	A function $f : [n] \to [n]$ as an oracle.
Preprocessing:	Using f, prepare an advice string $\mathbf{a}_f \in \{0,1\}^s$.
Query:	Point $y \in [n]$.
Answer:	Compute the value $f^{-1}(y)$, with a full access to $\mathbf{a}_f$ and using at most t probes to the oracle for f.

We want to design an efficient data structure, i.e., make s and t as small as possible. There are two trivial solutions. The first one is that the whole function f^{-1} is stored in the advice string $\mathbf{a}_f$, thus $s = O(n \log n)$ and $t = 0$. The second one is that the whole function f is probed during answering a query $y \in [n]$, thus $t = O(n)$ and $s = 0$. Note that the space s of the data structure is the length of the advice string $\mathbf{a}_f$ in bits but with one oracle-probe x_i the data structure reads the whole $f(x_i)$, thus with n oracle-probes we read the whole description of f, i.e., $n \log n$ bits.

The question is whether we can design a data structure with $s, t \le o(n)$. Hellman [19] gave the first non-trivial solution and introduced a randomized (adaptive) systematic data structure which inverts a function with a constant probability (over the uniform choice of the function f and the query $y \in [n]$) and $s = O\left(n^{2/3} \log n\right)$ and $t = O\left(n^{2/3} \log n\right)$. Fiat and Naor [14] improved the result and introduced a data structure that inverts any function at any point, however with a slightly worse trade-off: $s^3 t = O\left(n^3 \log n\right)$. Hellman [19] also introduced a more efficient data structure for inverting a permutation – it inverts any permutation at any point and $st = O(n \log n)$. Thus, it seems that inverting a permutation is an easier problem than inverting an arbitrary function.

In this paper, we are interested in lower bounds for the inversion problem. Yao [36] gave a lower bound that any systematic data structure for the inversion problem must have $st \ge \Omega(n \log n)$, however, the lower bound is applicable only if $t \le O(\sqrt{n})$. Since then, only slight progress was made. De et al. [10] improved the lower bound of Yao [36] to be applicable for the full range of t. Abusalah et al. [1] improved the trade-off, that for any k it must hold that $s^k t \ge \Omega\left(n^k\right)$. Seemingly, their result contradicts Hellman's trade-off $\left(s = t = O\left(n^{2/3} \log n\right)\right)$ as it implies $s = t \ge n^{k/k+1}$ for any k. However, for Hellman's attack [19] we need that the function can be efficiently evaluated and the functions introduced by Abusalah et al. [1] cannot be efficiently evaluated. There is also a series of papers [17, 30, 11, 8] that study how the probability of successful inversion depends on the parameters s and t. However, none of these results yields a better lower bound than $st \geq \Omega(n \log n)$. Hellman's trade-off is still the best-known upper bound trade-off for the inversion problem. Thus, there is still a substantial gap between the lower and upper bounds.

Another caveat of all known data structures for the inversion is that they heavily use adaptivity during answering queries $y \in [n]$. I.e., probes to the oracle depend on the advice string $\mathbf{a}$ and answers to the oracle probes which have been already made. We are interested in non-adaptive data structures. We say a systematic data structure is *non-adaptive* if all oracle probes depend only on the query $y \in [n]$.

As non-adaptive data structures are weaker than adaptive ones, there is a hope that for non-adaptive data structures we could prove stronger lower bounds. Moreover, the non-adaptive data structure corresponds to circuits computation [31, 32, 34, 9]. Thus, we can derive a circuit lower bound from a strong lower bound for a non-adaptive data structure. Non-adaptive data structures were considered by Corrigan-Gibbs and Kogan [9]. They proved that improvement by a polynomial factor of Yao's lower bound [36] for non-adaptive data structures would imply the existence of a function $F : \{0,1\}^N \to \{0,1\}^N$ for $N = n \log n$ that cannot be computed by a linear-size and logarithmic-depth circuit. More formally, they prove that if a function $f : [n] \to [n]$ cannot be inverted by a non-adaptive data structure of space $O\left(n \log n / \log \log n\right)$ and query time $O(n^{\varepsilon})$ for some $\varepsilon > 0$ then there exists a function $F : \{0,1\}^N \to \{0,1\}^N$ that cannot be computed by any circuit of size $O(N)$ and depth $O(\log N)$. They interpret $r \in \{0,1\}^N$ as n numbers in $[n]$, i.e, $r = (r_1, \dots, r_n) \in \{0,1\}^N$ where each $r_i \in [n]$. The function F is defined as $F(y) = F(y_1, \dots, y_n) = \big(f^{-1}(y_1), \dots . f^{-1}(y_n)\big)$ where $f^{-1}(y_i) = \min\{x \in [n] \mid f(x) = y\}$ and $\min \emptyset = 0$. Informally, if the function f is hard to invert at some points, then it is hard to invert at all points together. Compared to the result of Corrigan-Gibbs and Kogan [9], we provide an explicit function (sorting n-bit integers) which will require large circuits if any of the functions f is hard to invert. A connection similar to Corrigan-Gibbs and Kogan between data structures and circuits was made also by Viola [35].

2.2 Evaluation and Interpolation of Polynomials

In this section, we describe two natural problems connected to polynomials. We consider our problems over a finite field $\mathbb{F}$ to avoid issues with encoding reals.

Polynomial Evaluation over $\mathbb{F}$

Input:	Coefficients of a polynomial $p \in \mathbb{F}[x]$: $\alpha_0, \dots, \alpha_{n-1} \in \mathbb{F}$ (i.e., $p(x) = \sum_{i \in [n]} \alpha_i x^i$)
Preprocessing:	Using the input, prepare an advice string $\mathbf{a}_p \in \{0,1\}^s$.
Query:	A number $x \in \mathbb{F}$.
Answer:	Compute the value $p(x)$, with a full access to $\mathbf{a}_p$ and using at most t probes to the coefficients of p.

Polynomial Interpolation over $\mathbb{F}$

Input:	Point-value pairs of a polynomial $p \in \mathbb{F}[x]$ of degree at most $n-1$: $\big(x_0, p(x_0)\big), \dots, \big(x_{n-1}, p(x_{n-1})\big) \in \mathbb{F} \times \mathbb{F}$ where $x_i \neq x_j$ for any two indices $i \neq j$
Preprocessing:	Using the input, prepare an advice string $\mathbf{a}_p \in \{0,1\}^s$.
Query:	An index $j \in [n]$.
Answer:	Compute j-th coefficient of the polynomial p, i.e., the coefficient of x^j in p, with a full access to $\mathbf{a}_p$ and using at most t probes to the oracle for point-value pairs.

In the paper we often use a version of polynomial interpolation where the points $x_0, x_1, \ldots, x_{n-1}$ are fixed in advance and the input consists just of $p(x_0), p(x_1), \ldots, p(x_{n-1})$. Since we are interested in lower bounds, this makes our results slightly stronger.

Let $\mathbb{F} = \mathrm{GF}(p^k)$ denote the Galois Field of p^k elements. Let n be a divisor of $p^k - 1$. It is a well-known fact that for any finite field $\mathbb{F}$ its multiplicative group $\mathbb{F}^*$ is cyclic (see e.g. Serre [28]). Thus, there is a primitive n-th root of unity $\sigma \in \mathbb{F}$ (that is an element σ such that $\sigma^n = 1$ and for each $1 \le j < n$, $\sigma^j \ne 1$). Pollard [27] defines the *Finite Field Fourier transform* (FFFT) (with respect to σ) as a linear function $\mathrm{FFFT}_{n,\sigma} : \mathbb{F}^n \to \mathbb{F}^n$ which satisfies:

$$\mathrm{FFFT}_{n,\sigma}(\alpha_0, \ldots, \alpha_{n-1}) = (\beta_0, \ldots, \beta_{n-1}) \text{ where}$$
$$\beta_i = \sum_{j \in [n]} \alpha_j \sigma^{ij} \qquad \text{for any } i \in [n]$$

The inversion $\mathrm{FFFT}^{-1}_{n,\sigma}$ is given by:

$$\mathrm{FFFT}^{-1}_{n,\sigma}(\beta_0, \ldots, \beta_{n-1}) = (\alpha_0, \ldots, \alpha_{n-1}) \text{ where}$$
$$\alpha_i = \frac{1}{n} \sum_{j \in [n]} \beta_j \sigma^{-ij} \qquad \text{for any } i \in [n]$$

Here, $\frac{1}{n} = \left(\sum_{i=1}^{n} 1\right)^{-1}$ over $\mathbb{F}$. In our theorems we always set n to be a divisor of $|\mathbb{F}| - 1 = p^k - 1$ thus n modulo p is non-zero and the inverse exists. Observe, that $\mathrm{FFFT}^{-1}_{n,\sigma} = \frac{1}{n}\mathrm{FFFT}_{n,\sigma^{-1}}$.

FFFT is the finite field analog of Discrete Fourier transform (DFT) which works over complex numbers. The FFT algorithm by Cooley and Tukey [7] can be used for the case of finite fields as well (as observed by Pollard [27]) to get an algorithm using $O(n \log n)$ field operations (addition or multiplication of two numbers). Thus we can compute $\mathrm{FFFT}_{n,\sigma}$ and its inverse in $O(n \log n)$ field operations.

It is easy to see that $\mathrm{FFFT}_{n,\sigma}$ is actually evaluation of a polynomial in multiple special points (specifically in $\sigma^0, \ldots, \sigma^{n-1}$). We can also see that it is a special case of interpolation by a polynomial in multiple special points since $\mathrm{FFFT}^{-1}_{n,\sigma} = \frac{1}{n}\mathrm{FFFT}_{n,\sigma^{-1}}$. We provide an NCC-based lower bound for data structures computing the polynomial evaluation. However, we use the data structure only for evaluating a polynomial in powers of a primitive root of unity. Thus, the same proof yields a lower bound for data structures computing the polynomial interpolation.

There is a great interest in data structures for polynomial evaluation in a cell probe model. In this model, some representation of a polynomial $p = \sum_{i \in [n]} \alpha_i x^i \in \mathbb{F}[x]$ is stored in a table $\mathcal{T}$ of s_{cell} cells, each of w bits. Usually, w is set to $O\big(\log|\mathbb{F}|\big)$, that we can store an element of $\mathbb{F}$ in a single cell. On a query $x \in \mathbb{F}$ the data structure should output $p(x)$ making at most t_{cell} probes to the table $\mathcal{T}$. A difference between data structures in the cell probe model and systematic data structures is that a data structure in the cell probe model is charged for any probe to the table $\mathcal{T}$ but a systematic data structure is charged only for probes to the input (the coefficients α_i), reading from the advice string $\mathbf{a}_p$ is for free. Note that, the coefficients α_i of p do not have to be even stored in the table $\mathcal{T}$. There are again two trivial solutions. The first one is that we store a value $p(x)$ for each $x \in \mathbb{F}$ and on a query $x \in \mathbb{F}$ we probe just one cell. Thus, we would get $t_{\mathsf{cell}} = 1$ and $s_{\mathsf{cell}} = |\mathbb{F}|$ (we assume that we can store an element of $\mathbb{F}$ in a single cell). The second one is that we store the coefficients of p and on a query $x \in \mathbb{F}$ we probe all cells and compute the value $p(x)$. Thus, we would get $t_{\mathsf{cell}} = s_{\mathsf{cell}} = n$.

Let $k = \log |\mathbb{F}|$. Kedlaya and Umans [22] provided a data structure for the polynomial evaluation that uses space $n^{1+\varepsilon} \cdot k^{1+o(1)}$ and query time $\log^{O(1)} n \cdot k^{1+o(1)}$. Note that, $n \cdot k$ is the size of the input and k is the size of the output.

The first lower bound for the cell probe model was given by Miltersen [26]. He proved that for any cell probe data structure for the polynomial evaluation it must hold that $t_{\mathsf{cell}} \geq \Omega\big(k/\log s_{\mathsf{cell}}\big)$. This was improved by Larsen [23] to $t_{\mathsf{cell}} \geq \Omega\big(k/\log(s_{\mathsf{cell}} w/nk)\big)$, that gives $t_{\mathsf{cell}} \geq \Omega(k)$ if the data structure uses linear space $s_{\mathsf{cell}} \cdot w = O(n \cdot k)$. However, the size of $\mathbb{F}$ has to be super-linear, i.e., $|\mathbb{F}| \geq n^{1+\Omega(1)}$, and it is not known if the bound holds for smaller fields, e.g., of linear size. Data structures in a bit probe model were studied by Gál and Miltersen [15]. The bit probe model is the same as the cell probe model but each cell contains only a single bit, i.e., $w = 1$. They studied succinct data structures that are data structures such that $s_{\mathsf{cell}} = (n + r) \cdot k$ for $r < o(n)$. Thus, the succinct data structures are related to systematic data structures but still, the succinct data structures are charged for any probe (as any other data structure in the cell probe model). Note that a succinct data structure stores only a few more bits than it is needed due to information-theoretic requirement. Gál and Miltersen [15] showed that for any succinct data structure in the bit probe model it holds that $r \cdot t_{\mathsf{cell}} \geq \Omega(n \cdot k)$. We are not aware of any lower bound for systematic data structures for the polynomial evaluation.

Larsen et al. [24] also gave a log-squared lower bound for dynamic data structures in the cell probe model. Dynamic data structures also support updates of the polynomial p which usually impacts their query time.

There is a great interest in algorithmic questions about the polynomial interpolation such as how fast we can interpolate polynomials [16, 5, 18], how many probes we need to interpolate a polynomial if it is given by oracle [6, 20], how to compute the interpolation in a numerically stable way over infinite fields [29] and many others. However, we are not aware of any results about data structures for the interpolation, i.e., when the interpolation algorithm has an access to some precomputed advice.

3 Network Coding

We prove our conditional lower bounds based on the Network Coding Conjecture. In network coding, we are interested in how much information we can send through a given network. A *network* consists of a graph $G = (V, E)$, positive capacities of edges $c : E \to \mathbb{R}^+$ and k pairs of vertices $(s_0, t_0), \ldots, (s_{k-1}, t_{k-1})$. We say a network $R = \big(G, c, (s_i, t_i)_{i\in[k]}\big)$ is *undirected* or *directed (acyclic)* if the graph G is undirected or directed (acyclic). We say a network is *uniform* if the capacities of all edges in the network equal to some $q \in \mathbb{R}^+$ and we denote such network as $\big(G, q, (s_i, t_i)_{i\in[k]}\big)$.

A goal of a coding scheme for directed acyclic network $R = \big(G, c, (s_i, t_i)_{i\in[k]}\big)$ is that at each target t_i it will be possible to reconstruct an input message w_i which was generated at the source s_i. The coding scheme specifies messages sent from each vertex along the outgoing edges as a function of received messages. Moreover, the lengths of the messages sent along the edges have to respect the edge capacities.

More formally, each source s_i of a network receives an input message w_i sampled (independently of the messages for the other sources) from the uniform distribution $\mathbf{W}_i$ on a set W_i. Without loss of generality we can assume that each source s_i has an in-degree 0 (otherwise we can add a vertex s_i' and an edge (s_i', s_i) and replace s_i by s_i'). There is an alphabet Σ_e for each edge $e \in E(G)$. For each source s_i and each outgoing edge $e = (s_i, u)$ there is a function $f_{s_i,e} : W_i \to \Sigma_e$ which specifies the message sent along the edge e as a function of the received input message $w_i \in W_i$. For each non-source vertex $v \in V, v \neq s_i$ and each outgoing edge $e = (v, u)$ there is a similar function $f_{v,e} : \prod_{e'=(u',v)} \Sigma_{e'} \to \Sigma_e$ which specifies the message sent along the edge e as a function of the messages sent to v along the edges incoming to v. Finally, each target t_i has a decoding function $d_i : \prod_{e'=(u',t_i)} \Sigma_{e'} \to W_i$. The coding scheme is executed as follows:

1. Each source s_i receives an input message $w_i \in W_i$. Along each edge $e = (s_i, u)$ a message $f_{s_i,e}(w_i)$ is sent.
2. When a vertex v receives all messages $m_1, \dots, m_a$ along all incoming edges (u', v) it sends along each outgoing edge $e = (v, u)$ a message $f_{v,e}(m_1, \dots, m_a)$. As the graph G is acyclic, this procedure is well-defined and each vertex of non-zero out-degree will eventually send its messages along its outgoing edges.
3. At the end, each target t_i computes a string $\tilde{w}_i = d_i(m'_1, \dots, m'_b)$ where m'_j denotes the received messages along the incoming edges (u', t_i). We say the encoding scheme is *correct* if $\tilde{w}_i = w_i$ for all $i \in [k]$ and any input messages $w_0, \dots, w_{k-1} \in W_0 \times \cdots \times W_{k-1}$.

The coding scheme has to respect the edge capacities, i.e., if $\mathbf{M}_e$ is a random variable that represents a message sent along the edge e, then $H(\mathbf{M}_e) \leq c(e)$, where $H(\cdot)$ denotes the Shannon entropy. A *coding rate* of a network R is the maximum r such that there is a correct coding scheme for input random variables $\mathbf{W}_0, \dots, \mathbf{W}_{k-1}$ where $H(\mathbf{W}_i) = \log |W_i| \geq r$ for all $i \in [k]$. A network coding can be defined also for directed cyclic networks or undirected networks but we will not use it here.

Network coding is related to multicommodity flows. A multicommodity flow for an undirected network $\bar{R} = \big(\bar{G}, c, (s_i, t_i)_{i\in[k]}\big)$ specifies flows for each commodity i such that they transport as many units of commodity from s_i to t_i as possible. A flow of the commodity i is specified by a function $f^i : V \times V \to \mathbb{R}_0^+$ which describes for each pair of vertices (u, v) how many units of the commodity i are sent from u to v. Each function f^i has to satisfy:

1. If u, v are not connected by an edge, then $f^i(u, v) = f^i(v, u) = 0$.
2. For each edge $\{u, v\} \in E(\bar{G})$, it holds that $f^i(u, v) = 0$ or $f^i(v, u) = 0$.
3. For each vertex v that is not the source s_i or the target t_i, it holds that what comes to the vertex v goes out from the vertex v, i.e.,

$$\sum_{u\in V} f^i(u, v) = \sum_{u\in V} f^i(v, u).$$

4. What is sent from the source s_i arrives to the target t_i, i.e.,

$$\sum_{u\in V} f^i(s_i, u) - f^i(u, s_i) = \sum_{u\in V} f^i(u, t_i) - f^i(t_i, u).$$

Moreover, all flows together have to respect the capacities, i.e., for each edge $e = \{u, v\} \in E(\bar{G})$ it must hold that $\sum_{i\in[k]} f^i(u, v) + f^i(v, u) \leq c(e)$. A *flow rate* of a network $\bar{R}$ is the maximum r such that there is a multicommodity flow $F = (f^0, \dots, f^{k-1})$ that for each i transports at least r units of the commodity i from s_i to t_i, i.e., for all i, it holds that $\sum_{u\in V} f^i(u, t_i) - f^i(t_i, u) \geq r$. A multicommodity flow for directed graphs is defined similarly, however, the flows can transport the commodities only in the direction of edges.

Let R be a directed acyclic network of a flow rate r'. It is clear that for a coding rate r of R it holds that $r \geq r'$. As we can send the messages without coding and thus reduce the encoding problem to the flow problem. The opposite inequality does not hold: There is a directed network $R = \big(G, c, (s_i, t_i)_{i\in[k]}\big)$ such that its coding rate is $\Omega\big(|V(G)|\big)$-times larger than its flow rate as shown by Adler et al. [2]. Thus, the network coding for directed networks provides an advantage over the simple solution given by the maximum flow. However, such a result is not known for undirected networks. Li and Li [25] conjectured that the network coding does not provide any advantage for undirected networks, thus for any undirected network $\bar{R}$, the coding rate of $\bar{R}$ equals to the flow rate of $\bar{R}$. This conjecture is known as *Network Coding Conjecture* (NCC) and we state a weaker version of it below.

For a directed graph $G = (V, E)$ we denote by $\mathsf{un}(G)$ the undirected graph $(V, \bar{E})$ obtained from G by making each directed edge in E undirected (i.e., replacing each $(u, v) \in E(G)$ by $\{u, v\}$). For a directed acyclic network $R = \big(G, c, (s_i, t_i)_{i\in[k]}\big)$ we define the undirected network $\mathsf{un}(R) = \big(\mathsf{un}(G), \bar{c}, (s_i, t_i)_{i\in[k]}\big)$ by keeping the source-target pairs and capacities the same, i.e, $c\big((u, v)\big) = \bar{c}\big(\{u, v\}\big)$.

▶ **Conjecture 1** (Weaker NCC). *Let R be a directed acyclic network, r be a coding rate of R and $\bar{r}$ be a flow rate of $\mathsf{un}(R)$. Then, $r = \bar{r}$.*

This conjecture was used to prove a conditional lower bound for sorting algorithms with an external memory [13] and for circuits multiplying two numbers [3].

4 NCC Implies Data Structure Lower Bounds

In this paper, we provide several connections between lower bounds for data structures and other computational models. The first connection is that the Network Coding Conjecture (Conjecture 1) implies lower bounds for data structures for the permutation inversion and the polynomial evaluation and interpolation. Assuming NCC, we show that a query time t of a non-adaptive systematic data structure for any of the above problems satisfies $t \geq \Omega\big(\log n/ \log\log n\big)$, even if it uses linear space, i.e., the advice string $\mathbf{a}$ has size $\varepsilon n \log n$ for sufficiently small constant $\varepsilon > 0$. Formally, we define $t_{\mathsf{Inv}}(s)$ as a query time of the optimal non-adaptive systematic data structure for the permutation inversion using space at most s. Similarly, we define $t^{\mathbb{F}}_{\mathsf{Eval}}(s)$ and $t^{\mathbb{F}}_{\mathsf{Interp}}(s)$ for the polynomial evaluation and interpolation over $\mathbb{F}$.

▶ **Theorem 2.** *Let $\varepsilon > 0$ be a sufficiently small constant. Assuming NCC, it holds that $t_{\mathsf{Inv}}(\varepsilon n \log n) \geq \Omega\big(\log n/ \log\log n\big)$.*

▶ **Theorem 3.** *Let $\mathbb{F}$ be a field and n be a divisor of $|\mathbb{F}| - 1$. Let $s = \varepsilon n \log |\mathbb{F}|$ for a sufficiently small constant $\varepsilon > 0$. Then assuming NCC, it holds that $t^{\mathbb{F}}_{\mathsf{Eval}}(s), t^{\mathbb{F}}_{\mathsf{Interp}}(s) \geq \Omega\big(\log n/ \log\log n\big)$.*

Note that by Theorem 2, assuming NCC, it holds that $s \cdot t \geq \Omega\big(n \log^2 n/ \log\log n\big)$ for $s = \varepsilon n \log n$ and $t = t_{\mathsf{Inv}}(s)$. The same holds for $t^{\mathbb{F}}_{\mathsf{Eval}}$ and $t^{\mathbb{F}}_{\mathsf{Interp}}$ by Theorem 3. Thus, these conditional lower bounds cross the barrier $\Omega(n \log n)$ for $s \cdot t$ given by the best unconditional lower bounds known for the function inversion [36, 10, 1, 17, 30, 11, 8] and the lower bound for the succinct data structures for the polynomial evaluation by Gál and Miltersen [15]. Note that our lower bound does not contradict Hellman's attack [19] for the permutation inversion, as his data structure is heavily adaptive.

Our lower bound for the data structure for the polynomial evaluation and interpolation is applicable even for linear size field (i.e., linear number of queries). Larsen's lower bound for the data structure for the polynomial evaluation [23] is applicable only for superlinear fields, i.e., $|\mathbb{F}| \geq n^{1+\Omega(1)}$. We give the result for the polynomial evaluation here as it has an analogous proof as the lower bound for the polynomial interpolation and it might illustrate a more general phenomenon that our technique might be applicable to a broader class of functions that contains an involution as a subproblem.

To prove Theorems 2 and 3, we build on the technical machinery of Farhadi et al. [13]. The proof can be divided into two steps:

1. From a data structure for the problem we derive a network R with $O(tn)$ edges such that R admits an encoding scheme that is correct on a large fraction of the inputs. This step is distinct for each problem and the reduction for the function inversion is shown in Section 5, the reduction for the polynomial problems is left to the full version [12]. This step uses new ideas and interestingly, it uses the data structure twice in a sequence.
2. If there is a network R with dn edges that admits an encoding scheme which is correct for a large fraction of inputs, then $d \geq \Omega\big(\log n / \log\log n\big)$. This step is common to all the problems. It was implicitly proved by Farhadi et al. [13] and Afshani et al. [3].

5 NCC Implies a Weak Lower Bound for the Function Inversion

In this section, we prove Theorem 2 that assuming NCC, any non-adaptive systematic data structure for the permutation inversion requires query time at least $\Omega\big(\log n / \log\log n\big)$ even if it uses linear space. Let $\mathcal{D}$ be a data structure for inverting permutations of a linear space $s = \varepsilon n \log n$, for sufficiently small constant $\varepsilon < 1$, with query time $t = t_{\mathsf{Inv}}(s)$. Recall that $t_{\mathsf{Inv}}(s)$ is a query time of the optimal non-adaptive systematic data structure for the permutation inversion using space s. From $\mathcal{D}$ we construct a directed acyclic network $R = \big(G, c, (s_i, t_i)_{i\in[n]}\big)$ and an encoding scheme of a coding rate $\log n$. By Conjecture 1 we get that the flow rate of $\mathsf{un}(R) = \big(\bar{G}, c, (s_i, t_i)_{i\in[n]}\big)$ is $\log n$ as well. We prove that there are many source-target pairs of distance at least $\Omega(\log_t n)$. Since the number of edges of $\bar{G}$ will be $O(tn)$ and flow rate of $\mathsf{un}(R)$ is $\log n$, we are able to derive a lower bound $t \geq \Omega\big(\log n / \log\log n\big)$.

We will design the network based on the *probe graph* of the data structure. By the probe graph of the data structure, we understand a graph with n input vertices corresponding to possible oracle probes and n output vertices corresponding to possible data structure queries. Each query vertex is connected to the vertices of oracle probes executed by the data structure when answering that query. Here, we use the non-adaptivity of the studied data structures as the probe graph does not depend on the stored data but only on the data structure itself. Our construction will utilize two copies of the probe graph connected in a sequence. The network will have input vertices $s_0, \ldots, s_{n-1}$ and output vertices $u_0, \ldots, u_{n-1}$ where each target t_i is set to $u_{i+b \mod n}$ for a suitable constant b. The input vertices $s_0, \ldots, s_{n-1}$ correspond to the oracle vertices of the first copy of the probe graph. (See Fig. 1 for an illustration.)

We will feed distinct values $x_0, \ldots, x_{n-1} \in [n]$ to the input vertices which then send them to the query vertices of the first copy of the probe graph. Values $x_0, \ldots, x_{n-1}$ define a permutation $f(i) = x_i$. Each query vertex j of the first copy of the probe graph can determine $f^{-1}(j)$ if it is provided with the advice string $\mathbf{a}_f$ of the data structure corresponding to f. (We will fix the most common advice string $\mathbf{a}_f$ and restrict ourselves to inputs $x_0, \ldots, x_{n-1}$ consistent with it.) Each query vertex j can also determine the value of a newly defined function $h(j) = f^{-1}(j) + b$ which it sends along its outgoing edges. The second copy of the data structure serves to invert the function h similarly to inverting f. The oracle vertices of the second copy of the probe graph coincide with the query vertices of the first copy. The query vertices of the second copy of the probe graph are the output vertices $u_0, \ldots, u_{n-1}$. Hence, the query vertex $i + b$ of the second copy will be used to determine $h^{-1}(i + b) = x_i$. Thus, x_i is directed from s_i to $t_i = u_{i+b}$.

The above construction gives a network R with an encoding scheme E that is correct only on a substantial fraction of all possible inputs. Namely on inputs $x_0, \ldots, x_{n-1} \in [n]$ which are distinct and consistent with the fixed advices. This forces correlations among messages received by the sources. However, the Network Coding Conjecture requires independently sampled messages for each source. To overcome this issue we use the technique introduced by Farhadi et al. [13] to augment R so that it admits an encoding scheme for independent messages. We provide technical details next.

Let $R = \big(G, c, (s_i, t_i)_{i\in[k]}\big)$ be a directed acyclic network. Let each source receive a binary string of length r as its input message, i.e., each $W_i = \{0,1\}^r$. If we concatenate all input messages w_i we get a string of length $r \cdot k$, thus the set of all possible inputs for an encoding scheme for R corresponds to the set $\mathcal{I} = \{0,1\}^{rk}$. We say an encoding scheme is correct on an input $\bar{w} = (w_0, \dots, w_{k-1}) \in \mathcal{I}$ if it is possible to reconstruct all messages w_i at appropriate targets. An (ε, r)-*encoding scheme* is an encoding scheme which is correct on at least $2^{(1-\varepsilon)rk}$ inputs in $\mathcal{I}$.

We say a directed network $R = \big(G, c, (s_i, t_i)_{i\in[k]}\big)$ is (δ, d)-*long* if for at least δk source-target pairs (s_i, t_i), it holds that distance between s_i and t_i in $\mathsf{un}(G)$ is at least d. Here, we measure the distance in the undirected graph $\mathsf{un}(G)$, even though the network R is directed. The following lemma is implicitly used by Farhadi et al. [13] and Afshani et al. [3].

▶ **Lemma 4** (Implicitly used in [13, 3]). *Let $R = \big(G, r, (s_i, t_i)_{i\in[k]}\big)$ be a (δ, d)-long directed acyclic uniform network for $\delta > \frac{5}{6}$ and sufficiently large $r \in \mathbb{R}^+$. Assume there is an (ε, r)-encoding scheme for R for sufficiently small ε. Then assuming NCC, it holds that $\frac{|E(G)|}{k} \geq \delta' \cdot d$, where $\delta' = \frac{\delta - 5/6}{10}$.*

Now we are ready to prove a conditional lower bound for the permutation inversion. For the proof we use the following fact which follows from well-known Stirling's formula:

▶ **Fact 1.** *The number of permutations $[n] \to [n]$ is at least $2^{n\log n - 2n}$.*

▶ **Theorem 2.** *Let $\varepsilon > 0$ be a sufficiently small constant. Assuming NCC, it holds that $t_{\mathsf{Inv}}(\varepsilon n \log n) \geq \Omega\big(\log n / \log\log n\big)$.*

Proof Sketch. Let $\mathcal{D} = \mathcal{D}_n$ be the optimal data structure for the inversion of permutation on $[n]$ using space $\varepsilon n \log n$. We set $t = t_{\mathsf{Inv}}(\varepsilon n \log n)$. We will construct a directed acyclic uniform network $R = \big(G, r, (s_i, t_i)_{i\in[n]}\big)$ where $r = \log n$. Let $\varepsilon' = 2 \cdot \varepsilon + \frac{2}{q} + \frac{2}{\log n}$ for sufficiently large q so that we could apply Lemma 4. The network R will admit an (ε', r)-encoding scheme E. The number of edges of G will be at most $2tn$ and the network R will be $\big(\frac{9}{10}, d\big)$-long for $d = \frac{1}{2}\log_{qt} n$. Thus, by Lemma 4 we get that

$$2t = \frac{2tn}{n} \geq \Omega\left(\log_{qt} n\right),$$

from which we can conclude that $t \geq \Omega\big(\log n / \log\log n\big)$. Thus, it remains to construct the network R and the scheme E.

First, we construct a graph G' which will yield the graph G by deleting some edges. The graph G' has three layers of n vertices: a source layer A of n sources $s_0, \dots, s_{n-1}$, a middle layer M of n vertices $v_0, \dots, v_{n-1}$ and a target layer B of n vertices $u_0, \dots, u_{n-1}$. The targets $t_0, \dots, t_{n-1}$ of R will be assigned to the vertices $u_0, \dots, u_{n-1}$ later.

We add edges according to the data structure $\mathcal{D}$: Let $Q_j \subseteq [n]$ be the set of oracle probes which $\mathcal{D}$ makes during the computation of $f^{-1}(j)$, i.e., for each $i \in Q_j$, the query j probes the oracle f for $f(i)$. As $\mathcal{D}$ is non-adaptive, the sets Q_j are well-defined. For each $j \in [n]$ and $i \in Q_j$ we add edges (s_i, v_j) and (v_i, u_j). We set a capacity of all edges to $r = \log n$. This finishes the construction of G', see Fig. 1 for illustration of the graph G'.

The graph G' has exactly $2tn$ edges. Moreover, the vertices of the middle and the target layer have in-degree at most t as the incoming edges correspond to the oracle probes made by $\mathcal{D}$. However, some vertices of the source and the middle layer might have large outdegree,

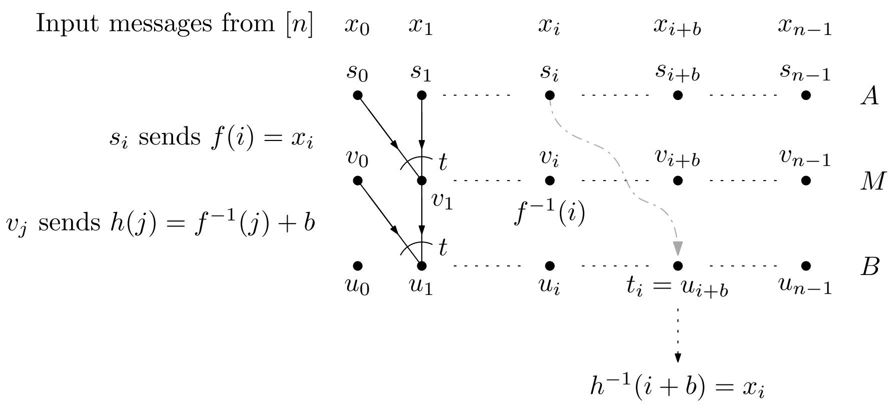
Input messages from $[n]$
x_0 x_1 x_i x_{i+b} x_{n-1}
s_0 s_1 s_i s_{i+b} s_{n-1}
A
s_i sends $f(i) = x_i$
v_0 t v_i v_{i+b} v_{n-1}
M
v_1 $f^{-1}(i)$
v_j sends $h(j) = f^{-1}(j) + b$
t
B
u_0 u_1 u_i $t_i = u_{i+b}$ u_{n-1}
$h^{-1}(i + b) = x_i$

Figure 1 A sketch of the graph G' and encoding scheme E.

which is a problem that might prevent the network R to be $\left(\frac{9}{10}, d\right)$-long. Thus, we need to remove edges adjacent to high-degree vertices. Let $W \subseteq V(G')$ be the set of vertices of out-degree larger than qt. We remove all edges incident to W from G' to obtain the graph G. (For simplicity, we keep the degree 0 vertices in G). Thus, the maximum degree of G is at most qt. Since the graph G' has $2tn$ edges, it holds that $|W| \leq \frac{2}{q} \cdot n$.

Now, we assign the targets of R in such a way that R is $\left(\frac{9}{10}, d\right)$-long. Let C_v be the set of vertices of G which have distance at most d from v in $\mathsf{un}(G)$. Since the maximum degree of G is at most qt and $d = \frac{1}{2}\log_{qt} n$, for each $v \in V(G)$, $|C_v| \leq 2\sqrt{n}$. It follows from an averaging argument that there is an integer b such that there are at least $n - 2\sqrt{n}$ sources s_i with distance at least d from u_{i+b} in $\mathsf{un}(G)$. (Here the addition $i + b$ is modulo n.) We fix one such b and set $t_i = u_{i+b}$. For n large enough, it holds that $n - 2\sqrt{n} \geq \frac{9}{10} \cdot n$. Thus, the network R is $\left(\frac{9}{10}, d\right)$-long.

It remains to construct the (ε', r)-encoding scheme E for R (see Fig. 1 for a sketch of the encoding E). We only sketch the construction here, the full proof is in the full version [12]. At vertices of the middle layer we compute the inversion of f, i.e., at a vertex v_j we compute $f^{-1}(j)$ using the data structure $\mathcal{D}$. To do that we need to fix the advice string $\mathbf{a}_f$ (of at most $\varepsilon n \log n$ bits) and values on the vertices in W (the high degree vertices).

We define a function $h : [n] \to [n]$ as $h(j) = f^{-1}(j) + b$. Thus, at each vertex v_j we are able to compute the value $h(j)$. By the anologous strategy, we compute the inverse $h^{-1}(\ell)$ at each vertex u_ℓ (again using the data structure $\mathcal{D}$, now for the function h). It can be showed that $h^{-1}(i + b) = x_i$, thus we can reconstruct x_i at each vertex $t_i = u_{i+b}$. Overall, we fix at most $(2\varepsilon + \frac{2}{q}) \cdot n \log n$ bits (the advice strings $\mathbf{a}_f$ and $\mathbf{a}_h$, and the computed values at the vertices in W). Since the sheme E is correct on all inputs encoding a permutation and consistent with the fixing, the scheme E is (ε', r)-encoding scheme for $\varepsilon' = 2\varepsilon + \frac{2}{q} + \frac{2}{\log n}$. ◀

6 Strong Lower Bounds for Data Structures and Lower Bounds for Boolean Circuits

In this section, we study a connection between non-adaptive data structures and boolean circuits. We are interested in circuits with binary AND and OR gates, and unary NOT gates. (See e.g. [21] for background on circuits). Corrigan-Gibbs and Kogan [9] describe a connection between lower bounds for non-adaptive data structures and lower bounds for boolean circuits for a special case when the data structure computes function inversion. They show that we would get a circuit lower bound if any non-adaptive data structure using $O(N^\varepsilon)$ queries must use at least $\omega\big(N/\log\log N\big)$ bits of advice (for some fixed constant $\varepsilon > 0$). They define a *boolean operator* to be a family of functions $(F_N)_{N\in\mathbb{N}}$ for $F_N\colon \{0,1\}^N \to \{0,1\}^N$ represented by boolean circuits with N input and N output bits and constant fan-in gates. A boolean operator is said to be an *explicit operator* if the decision problem whether the j-th output bit of F_N is equal to one is in the complexity class NP.

▶ **Theorem 5** (Corrigan-Gibbs and Kogan [9], Theorem 6 (in contraposition))**.** *If every explicit operator F_N has fan-in-two boolean circuits of size $O(N)$ and depth $O(\log N)$ then, for every $\varepsilon > 0$, there exists a family of strongly non-adaptive black-box algorithms that inverts all functions $f\colon [n] \to [n]$ using $O\big(n\log n/\log\log n\big)$ bits of advice and $O(n^\varepsilon)$ online probes.*

To prove their theorem Corrigan-Gibbs and Kogan [9] use the common bits model of boolean circuits introduced by Valiant [31, 32, 33]. Valiant proves that for any circuit there is a small cut, called *common bits*, such that each output bit is connected just to few input bits. Corrigan-Gibbs and Kogan [9] use the common bits of the given circuit to create a non-adaptive data structure by setting the advice string to the content of common bits and the queries are to those function values which are still connected to the particular output after removing the common bits. Using Valiant's result directly one can obtain the following corollaries (see the full version [12] for details).

▶ **Corollary 6.** *Let $\mathcal{S} = \big\{p^k \mid p \text{ is a prime}, k \in \mathbb{N}, k \neq 0\big\}$ be the set of all sizes of finite fields. For each $n \in \mathcal{S}$, let $\mathbb{F}_n = GF(n)$ and σ_n be a primitive $(n-1)$-th root of unity (thus a generator of the multiplicative group $\mathbb{F}_n^*$). If there is a circuit family computing $FFFT_{n-1,\sigma_n}$ (over $\mathbb{F}_n$) of size $O(n\log n)$ and depth $O(\log n)$ (where each input and output number is represented by $\log|\mathbb{F}_n|$ bits) then for every $\varepsilon > 0$ there is a family of non-adaptive data structures $\{\mathcal{D}_n\}_{n\in\mathcal{S}}$ where $\mathcal{D}_n$ uses advice of size $O\big(n\log n/\log\log n\big)$ and on a query $j \in [n-1]$ outputs the j-th output of $FFFT_{n-1,\sigma_n}$ using $O(n^\varepsilon)$ queries to the input.*

To put the corollary in a counter-positive way: if for some $\varepsilon > 0$, there are no non-adaptive data structures for polynomial interpolation, polynomial evaluation, or FFFT with an advice of size $o\big(n\log n/\log\log n\big)$ that use $O(n^\varepsilon)$ queries to the input then there are no linear-size circuits of logarithmic depth for FFFT.

In Theorem 2, resp. Theorem 3, we prove a conditional lower bound for permutation inversion, resp. polynomial evaluation and polynomial interpolation, of the form, that a non-adaptive data structure using $\varepsilon n\log n$ bits must do at least $\Omega\big(\log n/\log\log n\big)$ queries. It is not clear if assuming NCC we can get a sufficiently strong lower bound which would rule out non-adaptive data structures with sublinear advice string using $O(n^\varepsilon)$ oracle queries.

▶ **Corollary 7.** *We say that a circuit $C_n\colon \{0,1\}^{n\lceil\log n\rceil} \to \{0,1\}^{n\lceil\log n\rceil}$ sorts its input if on an input viewed as n binary strings $x_1, x_2, \ldots, x_n \in \{0,1\}^{\lceil\log n\rceil}$ outputs the strings sorted lexicographically. If there is a circuit family $(C_n)_{n\in\mathbb{N}}$, where $C_n\colon \{0,1\}^{n\lceil\log n\rceil} \to \{0,1\}^{n\lceil\log n\rceil}$ sorts its inputs, and each circuit C_n is of size $O(n\log n)$ and depth $O(\log n)$ then for every $\varepsilon > 0$, for every permutation $f : [n] \to [n]$ there is a non-adaptive data structure for inverting f that uses advice of size $O\big(n\log n/\log\log n\big)$ and $O(n^\varepsilon)$ queries.*

The works of Farhadi et al. [13] and Asharov et al. [4] connect the NCC conjecture directly to lower bounds for sorting. Their work studies sorting n numbers of $k+w$ bits by their first k bits. Namely Asharov et al. [4] show that NCC implies that constant fan-in constant fan-out circuits must have size $\Omega\big(nk(w - \log(n) + k)\big)$ whenever $w > \log(n) - k$ and $k \leq \log n$. This is incomparable to our results as we have $w = 0$.

References

1 Hamza Abusalah, Joël Alwen, Bram Cohen, Danylo Khilko, Krzysztof Pietrzak, and Leonid Reyzin. Beyond hellman's time-memory trade-offs with applications to proofs of space. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 357–379. Springer, 2017. doi:10.1007/978-3-319-70697-9_13.

2 Micah Adler, Nicholas J. A. Harvey, Kamal Jain, Robert Kleinberg, and April Rasala Lehman. On the capacity of information networks. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, page 241–250, USA, 2006. Society for Industrial and Applied Mathematics.

3 Peyman Afshani, Casper Benjamin Freksen, Lior Kamma, and Kasper Green Larsen. Lower Bounds for Multiplication via Network Coding. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:12, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.10.

4 Gilad Asharov, Wei-Kai Lin, and Elaine Shi. Sorting short keys in circuits of size o (n log n). In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2249–2268. SIAM, 2021.

5 Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 301–309, New York, NY, USA, 1988. Association for Computing Machinery. doi:10.1145/62212.62241.

6 Michael Clausen, Andreas Dress, Johannes Grabmeier, and Marek Karpinski. On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theor. Comput. Sci.*, 84(2):151–164, July 1991. doi:10.1016/0304-3975(91)90157-W.

7 James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

8 Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John Steinberger. Random oracles and non-uniformity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2018 Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 227–258. Springer Verlag, 2018. 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2018 ; Conference date: 29-04-2018 Through 03-05-2018. doi:10.1007/978-3-319-78381-9_9.

9 Henry Corrigan-Gibbs and Dmitry Kogan. The function-inversion problem: Barriers and opportunities. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography*, pages 393–421, Cham, 2019. Springer International Publishing.

10 Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and prgs. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 649–665, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

11 Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 473–495, Cham, 2017. Springer International Publishing.

12 Pavel Dvořák, Michal Koucký, Karel Král, and Veronika Slívová. Data structures lower bounds and popular conjectures, 2021. arXiv:2102.09294.

13 Alireza Farhadi, MohammadTaghi Hajiaghayi, Kasper Green Larsen, and Elaine Shi. Lower bounds for external memory integer sorting via network coding. In *Proceedings of the 51st*

Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, page 997–1008, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316337.

14 Amos Fiat and Moni Naor. Rigorous time/space trade-offs for inverting functions. *SIAM J. Comput.*, 29(3):790–803, 1999. doi:10.1137/S0097539795280512.

15 Anna Gál and Peter Bro Miltersen. The cell probe complexity of succinct data structures. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Automata, Languages and Programming*, pages 332–344, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

16 Joachim von zur Gathen and Jrgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, USA, 3rd edition, 2013.

17 Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput.*, 35(1):217–246, 2005. doi:10.1137/S0097539704443276.

18 Dima Grigoryev, Marek Karpinski, and Michael Singer. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. *SIAM J. Comput.*, 19:1059–1063, December 1990. doi:10.1137/0219073.

19 M. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980. doi:10.1109/TIT.1980.1056220.

20 Gábor Ivanyos, Marek Karpinski, Miklos Santha, Nitin Saxena, and Igor E. Shparlinski. Polynomial interpolation and identity testing from high powers over finite fields. *Algorithmica*, 80(2):560–575, 2018. doi:10.1007/s00453-016-0273-1.

21 Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

22 K. S. Kedlaya and C. Umans. Fast modular composition in any characteristic. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 146–155, 2008. doi:10.1109/FOCS.2008.13.

23 Kasper Green Larsen. Higher cell probe lower bounds for evaluating polynomials. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, page 293–301, USA, 2012. IEEE Computer Society. doi:10.1109/FOCS.2012.21.

24 Kasper Green Larsen, Omri Weinstein, and Huacheng Yu. Crossing the logarithmic barrier for dynamic boolean data structure lower bounds. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 978–989, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3188745.3188790.

25 Zongpeng Li and Baochun Li. Network coding: The case of multiple unicast sessions. *Proceedings of the 42nd Allerton Annual Conference on Communication, Control, and Computing*, October 2004.

26 Peter Bro Miltersen. On the cell probe complexity of polynomial evaluation. *Theor. Comput. Sci.*, 143(1):167–174, 1995. doi:10.1016/0304-3975(95)80032-5.

27 John M Pollard. The fast fourier transform in a finite field. *Mathematics of computation*, 25(114):365–374, 1971.

28 Jean-Pierre Serre. *A course in arithmetic*, volume 7. Springer Science & Business Media, 2012.

29 A. Smoktunowicz, I. Wróbel, and P. Kosowski. A new efficient algorithm for polynomial interpolation. *Computing*, 79(1):33–52, February 2007. doi:10.1007/s00607-006-0185-z.

30 Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 205–223, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

31 Leslie G Valiant. Graph-theoretic arguments in low-level complexity. In *International Symposium on Mathematical Foundations of Computer Science*, pages 162–176. Springer, 1977.

32 Leslie G Valiant. Exponential lower bounds for restricted monotone circuits. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 110–117, 1983.

33 Leslie G Valiant. Why is boolean complexity theory difficult. *Boolean Function Complexity*, 169(84-94):4, 1992.

34 Emanuele Viola. On the power of small-depth computation. *Found. Trends Theor. Comput. Sci.*, 5(1):1–72, 2009.

35 Emanuele Viola. Lower bounds for data structures with space close to maximum imply circuit lower bounds. *Theory of Computing*, 15(18):1–9, 2019. `doi:10.4086/toc.2019.v015a018`.

36 A. C.-C. Yao. Coherent functions and program checkers. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 84–94, New York, NY, USA, 1990. Association for Computing Machinery. `doi:10.1145/100216.100226`.