A Post-Quantum Lower Bound for the Distributed Lovász Local Lemma

Sebastian Brandt* Tim Göttlicher†

Abstract

In this work, we study the Lovász local lemma (LLL) problem in the area of *distributed* quantum computing, which has been the focus of attention of recent advances in quantum computing [STOC'24, STOC'25, STOC'25]. We prove a lower bound of $2^{\Omega(\log^* n)}$ for the complexity of the distributed LLL in the quantum-LOCAL model. More specifically, we obtain our lower bound already for a very well-studied special case of the LLL, called sinkless orientation, in a stronger model than quantum-LOCAL, called the randomized online-LOCAL model. As a consequence, we obtain the same lower bounds for sinkless orientation and the distributed LLL also in a variety of other models studied across different research communities.

Our work provides the first superconstant lower bound for sinkless orientation and the distributed LLL in all of these models, addressing recently stated open questions. Moreover, to obtain our results, we develop an entirely new lower bound technique that we believe has the potential to become the first generic technique for proving post-quantum lower bounds for many of the most important problems studied in the context of locality.

*CISPA Helmholtz Center for Information Security

†Saarland University, CISPA Helmholtz Center for Information Security

1 Introduction

In this work, we study the task of proving lower bounds for quantum algorithms in the distributed setting, with a focus on the Lovász local lemma. In recent years, distributed quantum computing has seen a surge of exciting results making considerable progress in a variety of settings, such as in the quantum versions of the CONGEST and Congested Clique models which capture bandwidth-constrained distributed algorithms [25, 41, 45, 62, 63, 65, 67, 80–82] and other models [1, 5, 7, 34, 40, 42, 46, 47, 49, 57]. Some of the most recent of these works study the quantum version of the standard LOCAL model of distributed computation (called quantum-LOCAL), achieving a number of breakthrough results for locally checkable problems[1]. For instance, [34] shows that for approximate graph coloring problems, there is essentially no quantum advantage in the LOCAL model, by proving upper bounds that hold in LOCAL and lower bounds that hold in quantum-LOCAL and match the upper bounds up to a polylogarithmic factor. In contrast, [7] shows that there exist locally checkable problems for which quantum computation helps in the LOCAL model, by proving a separation between the LOCAL and quantum-LOCAL complexities for a locally checkable problem called "iterated GHZ".

Besides the study of quantum advantage, another topic of considerable interest is the relation of quantum-LOCAL to other models related to locality. In [1], the authors show that there are fundamental connections between a variety of such models that have been studied across different fields (and include quantum-LOCAL). Before delving into the details of this work, we introduce the two main models discussed above. For the reader unfamiliar with the quantum world, we already note at this point that no deep understanding of quantum computation is necessary for obtaining our lower bound, since most technical details are examined in a setting that can be studied independently of any quantum considerations.

The LOCAL model. In the *LOCAL model* of distributed computation [66, 74], each vertex (or node) of the input graph G for some given graph problem is considered as a computational entity. These entities can exchange messages via incident edges; more precisely, computation/communication proceeds in synchronous rounds, and in each round each node first sends an arbitrarily large message to each of its neighbors, then receives the messages sent by its neighbors, and finally can perform some (unbounded) internal computation on the information it has accumulated so far. In the beginning of the overall computation, each node v does not know G; node v merely knows its own degree $\deg(v)$, can distinguish between its incident edges via port numbers $1, \ldots, \deg(v)$ uniquely assigned to these edges by v, and knows the total number n of nodes of the input graph and some symmetry-breaking information that depends on whether the considered model is the *deterministic* or *randomized* LOCAL model. In the deterministic LOCAL model, each node is equipped with an ID (in the range from 1 to n^c for some constant c) that is unique across the entire graph; in the *randomized* LOCAL model, each node is assigned a private random bit string of unbounded length. Each node executes the same algorithm that specifies which messages to send and which computations to perform. Each node v also has to decide to terminate at some point, upon which it provides its local part of the solution to the given problem. The (time or round) complexity of an algorithm in the LOCAL model is simply the (worst-case) number of rounds it takes until the last node terminates; the complexity of a problem is the time complexity of an optimal algorithm for that problem. In the randomized LOCAL model, an algorithm is required to produce a correct

[1] Locally checkable problems are problems that can be defined via local (i.e., constant-hop) constraints, i.e., problems for which a claimed solution is correct if and only if the constant-hop neighborhood of each node satisfies some problem-specific local criteria. This problem class contains most of the problems studied in the LOCAL model, such as coloring problems, maximal independent set, maximal matching, and the Lovász local lemma problem.

output with high probability (w.h.p.), i.e., with probability at least $1 - 1/n$.

The quantum-LOCAL model. The *quantum-LOCAL model* (see, e.g., [1, 49]) is essentially the randomized LOCAL model enhanced with quantum communication/computation. More specifically, each node is a quantum computer holding an arbitrarily large number of quantum bits (qubits), which form local states that can be manipulated in the standard quantum manner via unitary transformations. Moreover, instead of exchanging standard messages consisting of bits, in quantum-LOCAL the messages sent and received by the nodes consist of qubits. Finally, the nodes may perform arbitrary quantum measurements. This concludes the description of the quantum-LOCAL model. In a common variant of quantum-LOCAL, called "quantum-LOCAL (shared)", before the communication starts and the nodes receive their initial information, the algorithm may additionally provide the nodes with shared information, including a shared entangled quantum state. We note that the lower bounds that we prove apply to both versions of the model. However, for simplicity, we will mainly consider quantum-LOCAL throughout the paper and avoid mentioning repeatedly that our results also hold in quantum-LOCAL (shared).

A landscape of locality models. Following the results from [1] (and earlier works studying relations between models, such as [3, 49]), a highly interesting picture capturing relations between all kinds of models related to locality has emerged. This landscape includes standard distributed models such as LOCAL or quantum-LOCAL, models based on probability distributions such as non-signaling or bounded-dependence distributions which have been studied extensively across different communities (see, e.g., [5, 49, 58–61]), distributed sequential models such as the SLOCAL model [51], and centralized sequential models such as the dynamic-LOCAL and online-LOCAL models [3]. The aforementioned relations between the models are usually statements of the form "model A is at least as strong as model B" which is to be understood in the sense that an algorithm in model B can be simulated in model A without (asymptotic) loss in runtime and failure probability. An example of a sequence of models with increasing strength is randomized LOCAL, quantum-LOCAL, quantum-LOCAL (shared), non-signaling, randomized online-LOCAL. In fact, the randomized online-LOCAL model can simulate all models considered in the landscape in the above sense, making it the strongest of those models. For the full overview of all models and their relations, we refer the reader to [1], in particular to Fig. 10 therein.

Post-quantum distributed lower bounds and beyond. Keeping the above model landscape in mind, let us turn our attention back to the task of proving post-quantum lower bounds, i.e., LOCAL model lower bounds for locally checkable problems that do not break if we go to quantum-LOCAL. As already indicated in [1], proving lower bounds directly in quantum-LOCAL seems to be beyond the current state-of-the-art techniques. Instead, the known lower bounds that hold in quantum-LOCAL are achieved by proving the respective bounds in stronger models, in particular the non-signaling model (which, due to the aforementioned model relations then directly imply the same lower bound in quantum-LOCAL).

In particular, these bounds include bounds obtained via LOCAL model techniques that also work in the non-signaling model: Via existential graph-theoretic arguments, [34] showed a lower bound of $\Omega(n^{1/\alpha})$ for c-coloring χ-chromatic graphs (where $\alpha := \lfloor \frac{c-1}{\chi-1} \rfloor$) in the non-signaling model and also extended Linial's logarithmic lower bound for coloring trees [66] to this model. Moreover, as observed in [49], lower bounds in the (randomized) LOCAL model that are achieved via indistinguishability arguments such as an $\Omega(\sqrt{\frac{\log n}{\log\log n}})$-round lower bound for maximal matching and maximal independent set [64] and a $\Omega(\frac{\log n}{\log\log n})$-round lower bound for greedy coloring [48] transfer to the non-signaling model (called φ-LOCAL in [49]) and therefore also to the quantum-LOCAL model. For locally checkable problems on constant-degree *rooted trees*, [1] proved a general lifting theorem

showing that any $\omega(\log^* n)$-round lower bound in the (deterministic or randomized) LOCAL model implies an $\Omega(\log\log\log n)$ lower bound (for the same problem) in the randomized online-LOCAL model (and therefore also in quantum-LOCAL). Furthermore, [1] showed an $\Omega(\log n)$-round lower bound for 3-coloring 2-dimensional grids in the randomized online-LOCAL model (building on the same lower bound in the deterministic online-LOCAL model from [31]); note however, that the implications for quantum-LOCAL are subsumed by the aforementioned results by [34].

In general, the currently available techniques for proving lower bounds for locally checkable problems in quantum-LOCAL and stronger models are quite limited: there are lower bound tools such as indistinguishability arguments or existential graph theoretic arguments that are applicable to the quantum-LOCAL and non-signaling models but they do not provide anything close to a lower bound technique that is widely applicable and able to capture the most important problems such as coloring problems or the Lovász local lemma problem. This is in sharp contrast to the situation in the LOCAL model where such a technique is available in the form of the round elimination technique [17, 20] and raises the following fundamental question.

Open Problem 1. *How can we develop a* generic *lower bound technique for locally checkable problems in the quantum-LOCAL model (and stronger models)?*

A natural first candidate for such a technique would be the aforementioned round elimination technique; however, unfortunately, [7] recently showed that round elimination cannot be used to prove lower bounds for quantum-LOCAL. We also remark that, in terms of applicability to many models at once, a lower bound technique as desired in Open Problem 1 would ideally already apply to the strongest of the models in the aforementioned model landscape—the randomized online-LOCAL model.

Fundamental problems. While obtaining a generic lower bound technique is a highly ambitious general goal, there are also many concrete problems for which close to nothing is known w.r.t. proving lower bounds in quantum-LOCAL and beyond. Two of the most important such problems are the $(\Delta+1)$-coloring problem and the Lovász local lemma problem, which are both used as subroutines in countless algorithms for other problems in the LOCAL model (and stronger models).

The $(\Delta+1)$-coloring problems asks to color the nodes of the input graph G with $\Delta+1$ colors such that no two adjacent nodes receive the same color, where Δ denotes the maximum degree of G. In the LOCAL model, a lower bound of $\Omega(\log^* n)$ rounds[2] is known for $(\Delta+1)$-coloring which is tight for the special case of $\Delta=2$, i.e., for the problem of 3-coloring a cycle or path [35, 66]. However, in quantum-LOCAL even the latter special case is completely open: essentially nothing is known besides the fact that 3-coloring on cycles/paths can be solved in $O(\log^* n)$ rounds (which trivially follows from the same upper bound in LOCAL). Fascinatingly, resolving this coloring problem was one of the earliest results in the LOCAL model, proved more than three decades ago, which emphasizes our lack of understanding of lower bounds in quantum-LOCAL.

Lovász local lemma. The other of the two mentioned problems, the Lovász local lemma problem, is a fundamental algorithmic problem that goes back to an existential probabilistic statement proved in 1975 by Erdős and Lovász [43]. This statement, called the Lovász local lemma (LLL), can be phrased as follows.

Lemma 1.1 (LLL)**.** *Let $\mathcal{E}=\{E_1,\dots,E_k\}$ be a finite set of probabilistic events, d a positive integer, and $p<1$ a nonnegative real number such that, for each $1\le i\le k$,*

[2]The expression $\log^* n$ denotes the smallest number of times the log-function has to be applied iteratively to n to obtain a value that is at most 1.

1. *the event E_i is mutually independent of all other E_j except up to d many, and*

2. *the probability that E_i occurs is upper bounded by p.*

Then, if $4pd \leq 1$, the probability that none of the events in $\mathcal{E}$ occurs is positive.

The criterion $4pd \leq 1$ under which Lemma 1.1 holds was subsequently slightly relaxed by works of Spencer [77] and Shearer [76].

The *algorithmic LLL* (or *constructive LLL*) is the algorithmic problem of finding a point in the probability space of the LLL setting that avoids all of the events in $\mathcal{E}$. Initiated by Beck [16], the algorithmic LLL has received considerable attention since then (see, e.g., [4, 26, 36, 55, 70–73, 78]), including a celebrated result by Moser and Tardos presenting an efficient randomized algorithm for solving the algorithmic LLL.

The distributed LLL. In the distributed setting, the algorithmic LLL is modeled as a graph problem as follows. Let $\mathcal{X} = \{X_1, \ldots, X_\ell\}$ be a finite set of mutually independent random variables, $\mathcal{E} = \{E_1, \ldots, E_k\}$ a set of probabilistic events defined on $\mathcal{X}$, and $p < 1$ a nonnegative real number such that, for each of the events in $\mathcal{E}$, the probability that it occurs is upper bound by p. Then, the dependency graph G of this LLL instance (which constitutes the input graph to the distributed LLL), is defined by setting the node set to be $\mathcal{E}$ and connecting two nodes E_i, E_j by an edge if there is a common random variable from $\mathcal{X}$ that both events depend on. Let d be an upper bound for the maximum degree of G, and assume that p and d satisfy some LLL criterion (e.g., $4pd \leq 1$) that guarantees the existence of a point in the probability space avoiding all events.

In the distributed LLL, the task is to find a solution to the algorithmic LLL in the respectively considered distributed setting (e.g., in LOCAL or quantum-LOCAL). More precisely, each node with corresponding event E_i has to output an assignment to all variables from $\mathcal{X}$ that E_i depends on such that

1. E_i does not occur under this assignment, and

2. all nodes assigning a value to the same random variable assign the same value to the random variable.

Following the aforementioned result by Moser and Tardos, which also yields an $O(\log^2 n)$-round algorithm in the LOCAL model, the distributed LLL was studied in an abundance of works [20, 21, 23, 29, 30, 32, 33, 38, 39, 44, 50, 51, 56, 69, 75], improving the state of the art in a variety of settings, for different LLL criteria. For instance, in the randomized LOCAL model, the current state of the art is $O(\log d \cdot \log_{1/(ep(d+1))} n)$ rounds [33] for the (most popular) LLL criterion $ep(d+1) < 1$ and $O(\min\{\log_{1/p} n, d/\log d + \log^{O(1)} \log n\})$ rounds [33, 38] for polynomial LLL criteria, i.e., criteria of the form $p \cdot d^c \in O(1)$ for some constant c (which have been considered frequently in the distributed context). To the best of our knowledge, no better upper bounds are known for any of the stronger models from [2, Fig. 10, arxiv version]. On the lower bound side, the current state-of-the-art complexity bound in the randomized LOCAL model is $\Omega(\log \log n)$ rounds [20] (which holds for a wide variety of LLL criteria), which is achieved by showing this lower bound for a special case of the algorithmic LLL, called sinkless orientation.

Sinkless orientation. The sinkless orientation problem, introduced in [20] in the distributed setting, asks to orient the edges of the input graph such that no node of degree at least 3 is a sink, i.e., such that each node of degree at least 3 has at least one outgoing edge. More specifically, the output for each node v is an orientation of the edges incident to v such that at least one of these edges is oriented away from v and adjacent nodes agree on the orientation of the connecting edge.

When phrasing sinkless orientation in the language of LLL, each edge is considered to be a random variable (whose values are the two orientations of the considered edge) and the event associated with each node is that the node is a sink and has at least 3 incoming edges. Thereby, finding a solution to sinkless orientation is equivalent to solving this particular LLL problem.

If we allow that the input graph contains nodes of degree precisely 3, then no LLL criterion guaranteeing the existence of a solution is satisfied (even though a solution to sinkless orientation always exists, also in this case). However, as soon as we consider graphs with maximum degree $\Delta \geq 4$ where every node of degree at least 3 has degree precisely Δ, sinkless orientation becomes a special case of LLL as setting $p := 1/2^\Delta$ and $d := \Delta$ satisfies, e.g., the LLL criterion $4pd \leq 1$. In fact, sinkless orientation satisfies the very restrictive *exponential* LLL criterion $p \cdot 2^d \leq 1$, which is especially interesting from a lower bound perspective, as any lower bound for LLL proved under this criterion also applies to all more permissive criteria, including all the commonly used criteria. When discussing our results, we will implicitly assume this exponential LLL criterion without explicitly repeating this fact. We also remark that the restriction to the aforementioned graph class makes the bounds proved in this work only stronger as our objective is to prove lower bounds.

In contrast to the situation for the distributed LLL, an upper bound of $O(\log \log n)$ rounds matching the lower bound is known in the randomized LOCAL model. While no better upper bounds are known for the models sandwiched between the randomized LOCAL model and the randomized online-LOCAL model, in the randomized online-LOCAL model itself an upper bound of $O(\log \log \log n)$ is known, following from [51, 53].

The importance of the distributed LLL and sinkless orientation. In the LOCAL model, the distributed LLL and sinkless orientation have been of paramount importance for the development of entire lines of research in the LOCAL model. Below we discuss a few of the highlights.

The aforementioned (randomized) lower bound of $\Omega(\log \log n)$ (which was lifted to a *deterministic* lower bound of $\Omega(\log n)$ by [30]), together with matching randomized and deterministic upper bounds of $O(\log \log n)$ and $O(\log n)$, respectively, for sinkless orientation yielded the first proof that there exist locally checkable problems with provably *intermediate* complexities on constant-degree graphs. Following this discovery, a long line of research (see, e.g., [6, 8, 13–15, 22, 27, 28, 32, 54, 75]) mapped out the landscape of locally checkable problems on constant-degree graphs in a variety of settings, amongst others culminating in an almost complete understanding of the possible complexities that such problems can exhibit on trees and general (constant-degree) graphs. One particularly intriguing result in this line of research is a randomized speedup result by [32]: in this work, the authors show that any problem in the aforementioned setting that admits a sublogarithmic-time algorithm can be solved as fast as the distributed LLL (with a polynomial LLL criterion). This result implies a gap in the randomized complexity landscape which future work may be able to widen further by proving better upper bounds on the complexity of the distributed LLL.

However, the perhaps most important consequence of the aforementioned lower bound proved for the distributed LLL and sinkless orientation has been the development of a new lower bound technique in the LOCAL model called *round elimination*. Building on the lower bound proof in [20] which uses a specific version of round elimination fine-tuned to sinkless orientation, [17] generalized the used approach to a generic lower bound technique that, in principle, is applicable to any locally checkable problem. Using this general version of round elimination, a variety of lower bounds for some of the most fundamental problems studied in the LOCAL model have been obtained [9–12, 24, 29], emphasizing the importance of developing generic techniques. Round elimination has even been used to prove a gap in the aforementioned complexity landscape [54] and to prove a lower bound in the LOCAL model that is instrumental for showing quantum advantage for a locally checkable problem [7].

Lower bounds for the distributed LLL and sinkless orientation in quantum-LOCAL. In contrast to the lower bound situation in the LOCAL model, in the quantum-LOCAL model (and any stronger model), no super-constant lower bounds for the distributed LLL or sinkless orientation were known prior to our work. In fact, even obtaining *large constant* lower bounds for sinkless orientation (and therefore also the distributed LLL) in quantum-LOCAL and stronger models seemed out of reach of current techniques and was stated as an open problem in [79]. We obtain the following fundamental open problem.

Open Problem 2. *Prove superconstant lower bounds for the distributed LLL and sinkless orientation in the quantum-LOCAL model (and stronger models).*

Solving Open Problem 2 would also make significant progress on the more long-term question of fully determining the complexity of sinkless orientation in models related to quantum-LOCAL such as the non-signaling model [37, Question 7].

1.1 Our Contributions

We prove the following lower bound result in the strongest model discussed above, the randomized online-LOCAL model. For a formal introduction to the randomized online-LOCAL model, we refer the reader to Section 1.2.

Theorem 1.2. *The complexity of sinkless orientation in the randomized online-LOCAL model is in* $2^{\Omega(\log^* n)}$*. Moreover, this lower bound holds already on trees with maximum degree* 4 *in which no node has degree* 3*.*

As discussed above, sinkless orientation is a special case of the algorithmic LLL for the class of trees mentioned in Theorem 1.2. Hence, the lower bound from Theorem 1.2 also applies to the distributed LLL. Moreover, the lower bounds for sinkless orientation and the distributed LLL naturally apply also in any model that is at least as weak as the randomized online-LOCAL model, thereby providing lower bounds across models studied in different fields. Recalling the model overview in [1, Fig. 10, arxiv version], we can summarize the concrete implications of Theorem 1.2 as follows.

Corollary 1.3. *The complexities of sinkless orientation and the distributed LLL (even with exponential LLL criterion* $p \cdot 2^d \leq 1$*) are in* $2^{\Omega(\log^* n)}$*, in any model that is at least as weak as the randomized online-LOCAL model, which includes, e.g., the non-signaling model, the bounded-dependence model, and the deterministic dynamic-LOCAL model. In particular, the complexity of the distributed LLL in the quantum-LOCAL model is in* $2^{\Omega(\log^* n)}$*.*

This bound constitutes the first superconstant lower bound for the distributed LLL and sinkless orientation in any of the models mentioned in Corollary 1.3 and in particular solves Open Problem 2. Moreover, it makes substantial first progress on [37, Question 7].

While $2^{\Omega(\log^* n)}$ might appear as a strange complexity, we would not be surprised if our lower bound is actually asymptotically tight (in the exponent) for sinkless orientation in the randomized and deterministic online-LOCAL models and perhaps also beyond. On a high level, the reason for this is that the way in which our lower bound instances are constructed appear to not be wasteful at all (asymptotically speaking) w.r.t. the number of nodes used in the construction, and the bound of $2^{\Omega(\log^* n)}$ naturally emerges from it via the fact that if n nodes are required in our construction to prove a lower bound of k, then exponential in n many nodes are required in our construction to prove a lower bound of $2k$.

Towards a generic post-quantum lower bound technique? The approach for proving our lower bound is based on an entirely new lower bound technique we develop in this work. While the proof requires a lot of technical details, the actual graph constructions we use are based on conceptually simple graph operations, which provide an easy starting point for extensions and generalizations. This is somewhat similar to the lower bound for sinkless orientation and the distributed LLL proved in the LOCAL model [20], which, due to its *conceptual* simplicity gave rise to the generic lower bound technique of round elimination. In fact, there is another example of a similar historical behavior: a (conceptually elegant) lower bound technique from descriptive combinatorics that turned out to be applicable also to the LOCAL model, called Marks' technique [68], was essentially[3] developed in the context of sinkless orientation and has been subject to generalizations later [18, 19]. In general, it seems that finding a conceptually simple lower bound approach for sinkless orientation is a highly promising first step towards providing a much more widely applicable lower bound technique in the respectively considered model and thereby towards solving Open Problem 1. We believe that our lower bound approach constitutes such an approach and hope that it will serve as a foundation for many more post-quantum lower bounds for other locally checkable problems.

1.2 Our Approach

In this section, we give a bird's eye view of our approach for proving Theorem 1.2. As, at this point, the details of the considered model become relevant, we finally introduce the randomized online-LOCAL model formally.

The randomized online-LOCAL model. For simplicity, we first describe the deterministic online-LOCAL model, and then explain the minor changes required to obtain the randomized online-LOCAL model. In the deterministic online-LOCAL model, different from the setting of the LOCAL model, the nodes of the input graph G are presented to the algorithm one after another in an adversarial order. Each time a node v is presented, the (deterministic) algorithm obtains full information about the L-hop neighborhood of v, for some parameter L (that, if considered for a class of graphs instead of for a single graph, is a function $L(n)$ of the number n of nodes of the input graphs). More specifically, the algorithm obtains complete information about the topology of the L-hop neighborhood of v and about the degree of (and the port numbers at) each node in this neighborhood (including those at distance precisely L), and for each node w in this neighborhood it receives the information which nodes that it has seen previously are identical to w.[4] Upon receiving this information, the algorithm has to decide on the output for v before the next node and its L-hop neighborhood are revealed. The algorithm is correct if the global output induced by the choices for each node is a correct solution to the considered problem. We emphasize that the algorithm is not memory-constrained: it can remember all information that it receives during the execution. The complexity (or locality) of such an algorithm is the aforementioned function $L(n)$ (w.r.t. worst-case input instances consisting of an input graph and an ordering of the nodes), i.e., the parameter indicating the radius of the neighborhood around a presented node that is given to the algorithm.

In the randomized online-LOCAL model, the algorithm has an unlimited source of randomness it can use for making its decisions. Moreover, the adversary is oblivious, i.e., it has to select the input

[3]Formally, the problem considered in [68] is Δ-coloring, to which there exists a simple reduction from sinkless orientation in the considered setting. This close connection also ensures that the same approach considered in that paper works directly for sinkless orientation.

[4]One can imagine that the algorithm labels each node it sees with a unique identifier and can recognize those identifiers when it sees such a node again. However, we emphasize that, in contrast to the LOCAL model, the deterministic (and also the randomized) online-LOCAL model are formally defined without identifiers.

instance before the algorithm generates its random bits. Furthermore, an algorithm is considered to solve a given problem if it produces a correct solution w.h.p., i.e., with probability at least $1 - 1/n$, on any input instance. Consequently, the complexity of a problem in the randomized online-LOCAL model is the complexity of an optimal algorithm that produces a correct solution w.h.p.

For readers familiar with the SLOCAL model [52], we remark that the (deterministic and randomized) online-LOCAL models can be seen as versions of the (deterministic, resp. randomized) SLOCAL model with global memory. Moreover, for discussing an online-LOCAL algorithm, it will be convenient to be able to refer to the nodes and edges the algorithm has already "seen" at some point of the execution.

Definition 1.4 (seeing a node/edge)**.** Let L denote the complexity of a (deterministic or randomized) online-LOCAL algorithm $\mathcal{A}$ that is executed on some graph G. Consider any point of the execution, and let v and $e = \{u, u'\}$ denote a node and an edge of G, respectively. We say that $\mathcal{A}$ *has seen* v if there is a node $w \in V(G)$ that has been presented to $\mathcal{A}$ (up to the considered point of the execution) and has distance at most L from v. We say that $\mathcal{A}$ *has seen* e if there is a node $w \in V(G)$ that has been presented to $\mathcal{A}$ and has distance at most L from both u and u'.

Moreover, we will use the terms *query* and *query sequence* to refer to the presentation of a node by the adversary and the sequence of presented nodes, respectively. Now we are set to present an informal overview of our approach.

High-level overview. To obtain the lower bound from Theorem 1.2, intuitively, we first prove the lower bound in the *deterministic* online-LOCAL model and then adapt the construction to also work in the *randomized* online-LOCAL model. While we will not explicitly refer to the deterministic online-LOCAL model in the formal proof provided in Sections 2 and 3, we will present the high-level overview here by following the aforementioned intuition. We remark that the lifting of the deterministic lower bound to the randomized lower bound can also be achieved in a simple black-box fashion by applying [1, Lemma 7.6, arxiv version]; we hope that our white-box approach will be useful for future applications of the developed technique by yielding additional insights.

Now imagine we want to prove a lower bound of L for the complexity of sinkless orientation in the *deterministic* online-LOCAL model. Obtaining such a lower bound is equivalent to showing that any algorithm with complexity at most $L - 1$ fails on some input graph. Hence, assume for a contradiction that there exists some algorithm $\mathcal{A}$ solving sinkless orientation with a complexity of at most $L - 1$. We will explicitly construct some input instance on which $\mathcal{A}$ fails. For an illustration of the initial part of our argumentation see Figure 1, which illustrates the case $L = 2$ and $\Delta = 3$ (in which $\mathcal{A}$ has complexity at most $L - 1 = 1$).

A sequence of contradictory configurations. In essence, our argumentation starts from an (imagined) incorrect local output and then works its way backwards to how $\mathcal{A}$ can be actually forced to produce such an incorrect output. Our starting point is a configuration representing the simplest possible incorrect output: two adjacent nodes that both choose the connecting edge to be oriented outwards. However, why would the node v that is presented to $\mathcal{A}$ later than the other of the two not choose to orient the connecting edge inwards (thereby agreeing with the decision of the adjacent node) and orient another incident edge outwards? Well, imagine that all neighbors of v were presented to $\mathcal{A}$ before v and all of them chose to orient the edge connecting them to v towards v. This is a new configuration that enforces that the aforementioned contradictory configuration is produced by $\mathcal{A}$, no matter which incident edge v chooses to orient outwards.

In the same way as we obtained a new contradictory configuration from the initial one, we can now continue to iteratively transform an obtained configuration C into a new configuration C' that forces $\mathcal{A}$ to produce C. To guarantee that $\mathcal{A}$ cannot avoid to produce C when presented with a

suitable node in C', we create C' from C in the following way: we choose a node u that, in C, is a leaf node that has chosen to output its incident edge away from u and then consider as C' the configuration obtained from C by first undoing u's output decision (which, in Figure 1, corresponds to turning the edge u had chosen to orient outwards into an undirected edge) and then taking Δ copies of C and identifying all copies of u in those copies. Here Δ is some parameter that will correspond to the maximum degree of the graph obtained at the end of our construction. Now, when presented with u in C', Algorithm $\mathcal{A}$ cannot avoid to produce C.

Reflection and split operations. By the described operation—which we call the *reflection operation*—, we can transform a contradictory configuration into a new one. However, across a sequence of configurations produced by applying the reflection operation, the number of nodes that have already decided on their output grows instead of decreases while, ideally, we would like them to reach 0 so that we can take the obtained graph as our input graph and the sequence of the nodes at the center of the reflection operations as query sequence (in reverse order). To this end, we introduce another operation, called the *split operation*, which essentially consists of removing a node from a considered configuration C that is farther away from any node in C that has already decided on its output than the complexity of $\mathcal{A}$. The justification for this removal is that, due to the mentioned distance property of the removed node, Algorithm $\mathcal{A}$ cannot have seen the removed node, which implies that the behavior of $\mathcal{A}$ is independent of whether we include the node or remove it. Moreover, the removal disconnects the considered configuration into connected subconfigurations that (again due to the aforementioned distance property) are "independent" of each other w.r.t. the behavior of $\mathcal{A}$; hence we can iterate on each connected subconfiguration separately. Due to the symmetry generated by the reflection operation, by choosing the node to remove (amongst those nodes that satisfy the distance property) suitably, we can ensure that the removal performed by the split operation will result in isomorphic connected subconfiguration, which allows us to reduce attention to a single of those subconfigurations, implicitly assuming that whatever further operations we perform on that subconfiguration, we also perform on all isomorphic subconfigurations.

Deriving an input instance. Now imagine a sequence of reflection and split operations that ends up with an empty graph. By reversing the sequence, i.e., by starting with the empty graph and traversing the sequence of reflection and split operations in reverse order, where, for a split operation, we *add* the removed node and, for a reflection operation, present the node at the center of the operation to $\mathcal{A}$, we obtain both a graph and a query sequence consisting of (a subset of) the nodes of the obtained graph. This graph and this query sequence will now essentially constitute our input instance for which $\mathcal{A}$ will fail. The reason why $\mathcal{A}$ fails on this instance is that the construction ensures that $\mathcal{A}$ produces precisely the contradictory configurations discussed above in reverse order until it reaches the incorrect output of the initial configuration.[5] The input instance derived for the case $L = 2$ and $\Delta = 3$ corresponding to Figure 1 is illustrated in Figure 2.

The construction tree. On a high level, the above discussion reduces the task of proving the desired lower bound in the deterministic online-LOCAL model to the task of finding a suitable sequence of reflection and split operations that results in the empty graph. Note that the applicability of the split operation (which we would like to apply as often as possible to reduce the size of the

[5]There is a hidden subtlety in our argumentation: So far, we have implicitly assumed that we can decide *which* edge $\mathcal{A}$ orients outwards from a node that is queried, based on the fact that *any* decision will lead to the previous configuration in the above sequence of configurations. In principle, while, for each considered step, $\mathcal{A}$ cannot avoid producing the configuration we desire, it can choose a different edge to orient outwards than we intend in the overall construction. However, this freedom of choice of $\mathcal{A}$ is of cosmetic nature: due to the local isomorphisms in our construction, we can essentially continue the construction of our input instance in a locally isomorphic manner (compared to the presented fixed construction), no matter which edge $\mathcal{A}$ chooses.

considered graph(s)) is limited due to the required distance property. In fact, it is far from clear that a sequence as discussed above resulting in the empty graph even exists.

Our construction of such a sequence is conceptually simple although there are many technical details that have to be taken care of. The key object that we use to obtain such a sequence is what we call a *construction tree*. A construction tree encodes which (reflection/split) operations to perform at which nodes in which order. More precisely, the definition of a construction tree already ensures that the sequence of operations that can be read off of the construction tree is "good" in various regards. Amongst others, the construction tree satisfies a number of properties related to guaranteeing that the graph obtained at the end of the sequence of operations is empty and that the sequence of graphs obtained during the sequence of operations has lots of symmetries. (We remark that, for technical reasons, in our proof we do not remove nodes when performing a split operation, but instead *mark* them; hence, formally, the guarantee that the obtained graph is empty translates to the guarantee that all nodes of the obtained graph are marked.)

Deriving the lower bound in the deterministic online-LOCAL model. Perhaps most importantly, based on the structure of the construction tree, we can define a transformation $F(\cdot)$ that takes a construction tree as input and returns a graph that can be shown to be another (usually much larger) construction tree. This transformation is carefully designed to satisfy a key property that lies at the heart of our lower bound approach: if T is a construction tree that yields a sequence of reflection and split operations that results in the empty (respectively fully marked) graph and obeys the aforementioned distance property with parameter $L-1$ (describing the complexity of the considered algorithm $\mathcal{A}$), then $F(T)$ is a construction tree that yields a sequence that results in the empty graph and obeys the aforementioned distance property with parameter $2L-1$. In other words, if the lower bound that we obtain by using construction tree T is L, then the lower bound that we obtain by using construction tree $F(T)$ is $2L$.

Moreover, we show that if we compare the number of nodes of the input graph that we obtain by using construction tree T with the number of nodes of the input graph that we obtain by using construction tree $F(T)$, then the latter is exponential in the former. In other words, by increasing the size of the input graph exponentially, we increase the obtained lower bound by a factor of 2, which results in the desired lower bound of $2^{\Omega(\log^* n)}$ (though so far only for the *deterministic* online-LOCAL model).

Lifting the lower bound to the randomized online-LOCAL model. In order to lift the obtained lower bound to the *randomized* setting, we modify, for each deterministic lower bound instance, both the input graph and the query sequence by a sequence of adaptations each of which corresponds to a presented node. More specifically, consider any of the lower bound instances constructed for the deterministic online-LOCAL model. We essentially transform this instance into a lower bound instance for the randomized online-LOCAL model by going through the query sequence and, each time a node is presented to the algorithm, tweaking both the overall graph and the remainder of the query sequence in a manner depending on the *most probable* behavior of the considered randomized algorithm w.r.t. which edge(s) incident to the presented node the algorithm will orient outwards. In essence, these modifications guarantee that we obtain the same sequence of contradicting configurations as in the deterministic lower bound proof until we reach an incorrect output, with the difference that for each presented node we may now lose a factor of Δ in the probability that the behavior is indeed as we need for our lower bound. (Note that we do not lose more than a factor of Δ due to considering the "most probable" behavior of the considered randomized algorithm, which occurs with probability at least $1/\Delta$.)

While overall, this results in our contradiction occurring with only a small probability, by increasing the number n of nodes suitably (by adding sufficiently many nodes to the considered input

graph) we can make sure that this small probability is still larger than $1/n$, thereby guaranteeing that the considered algorithm fails with probability more than $1/n$, yielding a contradiction. For this, an exponential increase in the number of nodes suffices, which yields a lower bound of $2^{\Omega(\log^*(\log n))}$ which is identical to $2^{\Omega(\log^* n)}$. Hence, asymptotically (in the exponent), we obtain the same lower bound for the randomized online-LOCAL model as for the deterministic online-LOCAL model.

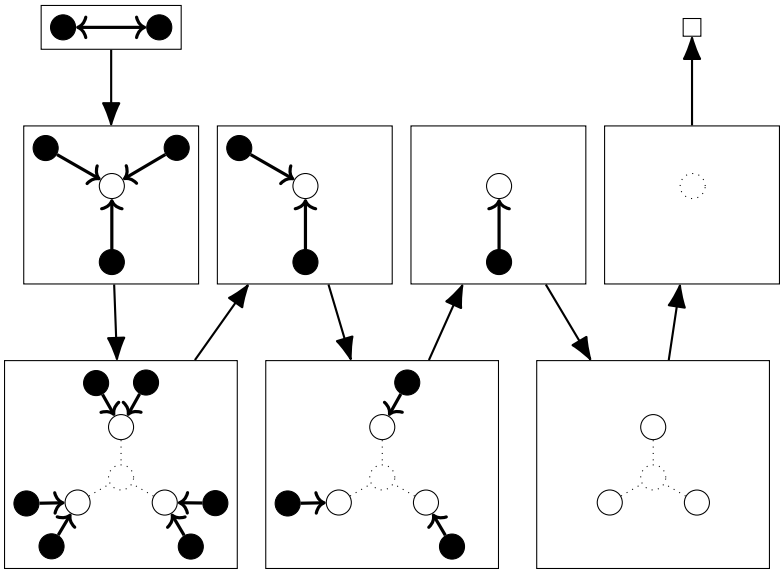

Figure 1: A high-level overview of our approach in the deterministic online-LOCAL model for the case $L = 2$ and $\Delta = 3$. Each box displays a configuration as discussed in the high-level overview in Section 1.2. A downward arrow between configuration corresponds to a reflection operation, an upward arrow to a split operation, where in the source configuration the node to be removed is indicated by a dotted circle and in the target configuration only one of the isomorphic connected subconfigurations of the resulting configuration is displayed. The final (small) box represents the empty configuration.

1.3 Roadmap

In Section 2, we collect a number of ingredients on which our lower bound proof builds. More specifically, Section 2 contains the following parts. In Section 2.1 we develop the notion of a construction tree. In Section 2.2 we make use of this general definition to define a concrete sequence of construction trees that we will use to derive our lower bound proof. To obtain this sequence, we introduce a graph transformation and prove that it preserves the property of being a construction tree. We also introduce the "initial tree" from which we obtain the aforementioned sequence via iterative application of the aforementioned transformation and bound the number of nodes of the trees in the sequence. In Section 2.3, we define two edge labelings for construction trees that, in particular, will be useful for keeping track of the different components obtained from the removal of nodes discussed in the above high-level overview. Moreover, we prove a number of technical lemmas about these labelings. In Section 2.4, we define an input tree that is already very close to the input graph that we will present to the algorithm in Section 3 and prove its well-definedness. In Section 2.5, we introduce the notion of distance-D-correctness and prove the key lemma that if a construction tree

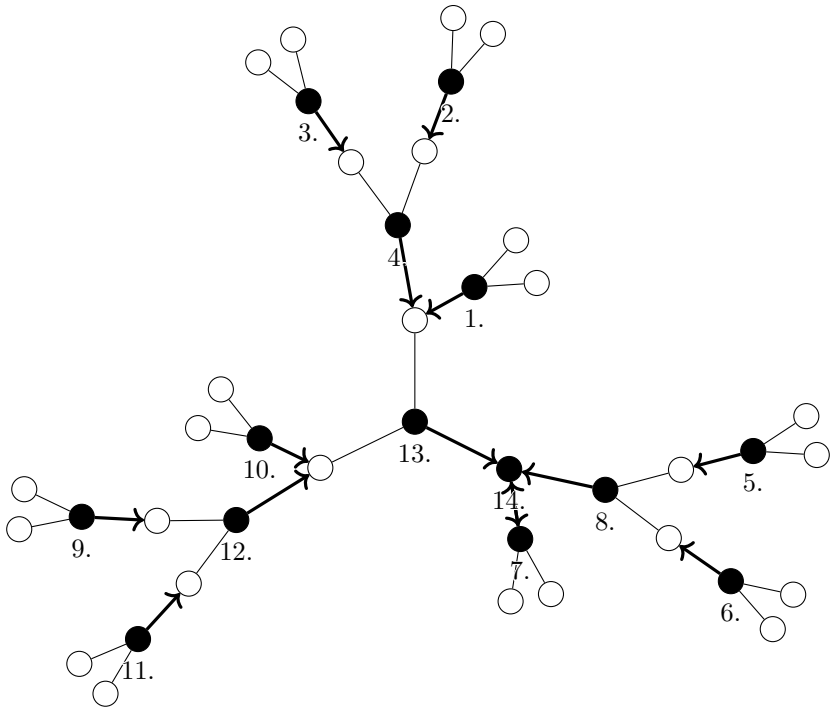

Figure 2: A slightly simplified overview of the input graph and query sequence obtained from the approach illustrated in Figure 1 for $L = 2$ (i.e., considering an algorithm $\mathcal{A}$ with complexity $L - 1 = 1$) and $\Delta = 3$. The solid nodes represent the first nodes in the query sequence, in the order displayed below the nodes. For each solid node one incident edge is displayed as being oriented away from the node, illustrating the part of the decision of $\mathcal{A}$ for the respective node that is relevant for obtaining the desired contradiction. Note that each time $\mathcal{A}$ has to make a decision for a solid node v, the situation regarding the connected component containing v of the subgraph that $\mathcal{A}$ has already seen at this point in time is symmetric, essentially allowing us to choose which incident edge we want $\mathcal{A}$ to orient outwards. With the 14th presented node, $\mathcal{A}$ will create an incorrect output. We remark that, unfortunately, the small examples given in this figure and Figure 1 are misleading in their simplicity; some assumptions that one is tempted to make by looking at the figures do actually not hold when considering larger L. For instance, the aforementioned symmetry does not hold in general; however it holds if one ignores the parts of the connected component that are "behind" a node that is oriented towards the considered node v. We also note that the aforementioned "slight simplification" of Figure 2 merely consists in omitting some nodes that are incident to the displayed nodes and guarantee that every displayed node has degree $\Delta = 3$ (which is relevant due to the precise definition of the deterministic online-LOCAL model, in particular regarding which information the algorithm receives).

T is distance-D-correct, then the tree obtained from T by the aforementioned graph transformation is distance-$(2D)$-correct. Since the property that a construction tree is distance-D-correct roughly corresponds to the property that the input instance obtained from the construction tree requires complexity D to be solved (in the deterministic online-LOCAL model), this key lemma, together with the bound on the size of the construction trees discussed in Section 2.2, provides the basis for the analysis of the complexity lower bound. Finally, in Section 3, we prove our lower bound by combining the above ingredients with new ingredients that are specifically targeted towards obtaining lower bound instances that are good against *randomized* online-LOCAL algorithms.

2 Ingredients for the Lower Bound Construction

In this section, we will develop a variety of tools and constructions that are essential for our overall lower bound approach. Besides defining the objects we need in the approach, we will also prove a number of lemmas that collect important properties of said objects.

We start by recalling some basic graph-theoretic definitions and notations.

Definition 2.1. For a graph G, we denote the set of nodes of G by $V(G)$ and the set of edges of G by $E(G)$, and write $G = (V(G), E(G))$. A *(maximal) connected component* C of G is a connected subgraph of G such that, for any two nodes $v \in V(C), w \in V(G) \setminus V(C)$, there is no path in G connecting v and w. Let $V' \subseteq V(G)$ be a subset of the nodes of G. The (sub)graph *induced by* V' is the subgraph of G with node set V' in which any two nodes of V' are connected if and only if they are connected in G.

For two nodes v, w of a rooted tree T, we call w a *descendant* of v (and equivalently v an *ancestor* of w) if there is a directed path from w to v in T. This path may be of length 0, i.e., any node is descendant and ancestor of itself. If there is neither a directed path from v to w, nor a directed path from w to v, then we call v and w *incomparable*. A *leaf (node)* of a (rooted) tree is a node of degree 1. For a directed edge $e = (v, w)$, we call v the *tail* of e and w the *head* of e.

In order to distinguish between the two ports numbers assigned to the same edge e with endpoints v and w, we will refer to them as the *port number assigned by v (resp. w) to e* or simply as the *port number at v (resp. w)*.

2.1 The Construction Tree

In this section, we define one of the crucial building blocks of our approach, which we call a *construction tree*. A construction tree can be thought of as a rooted tree with certain properties that, roughly speaking, encodes how to construct a hard lower bound instance, regarding both the input graph and the query sequence (that will force an algorithm to have at least a certain complexity). Please note that this high-level description is somewhat inaccurate in that the precise construction of a lower bound instance will depend on the considered algorithm; however, such a lower bound instance for a concrete algorithm will be achieved by tweaking the aforementioned general "hard lower bound instance".

Before we can introduce construction trees formally, we need to introduce a number of additional definitions.

Topological considerations. We first collect some definitions that are relevant for defining the topological structure of a construction tree. The following definition introduces a balance property that construction trees will satisfy.

Definition 2.2 (b-balanced tree)**.** Let T be a rooted tree with root r. For each positive integer i, we denote the set of vertices of T that are at distance precisely $i-1$ of r by L_i (and will call such a set a *layer*). For instance, $L_1 = \{r\}$.

Now, let $b \geq 3$ be an integer. We call T *b-balanced* if

1. every node of T has precisely 0, 1 or b children, and
2. for each positive integer i and any two nodes $v, w \in L_i$, the number of children of v and w is identical.

We call a node a *reflect node* if it has precisely one child, and a *split node* otherwise. We denote the set of all reflect nodes by V_R and the set of all split nodes by V_S. Moreover, we call a split node an

internal split node if it has precisely b children, and a *leaf (split) node* otherwise (i.e., if it has no children).

We call a layer L_i in a b-balanced rooted tree a *reflect layer* if it consists only of reflect nodes and a *split layer* if it consists only of split nodes. Moreover, we call L_i an *internal split layer* if it consists only of internal split nodes and a *leaf (split) layer* if it consists only of leaf split nodes.

We observe that for each internal split node of a b-balanced rooted tree, the b subtrees hanging from its b children are isomorphic rooted trees. Moreover, it follows from the definition of a b-balanced rooted tree that each nonempty layer of a b-balanced rooted tree is either a reflect layer or a split layer, and each split layer is either an internal split layer or a leaf split layer (of which there is precisely one). For simplicity we will disregard empty layers in the remainder of the paper, considering a layer L_i to be a layer of a b-balanced rooted tree if and only if it is nonempty.

Next, we introduce a second property that construction trees will satisfy. This property ensures that the layers from Definition 2.2 are "well-nested", regarding whether they are reflect or split layers.

Definition 2.3 (well-nested)**.** Let T be a b-balanced rooted tree T and let $i_{\max}$ denote the index of the leaf split layer (i.e., the layers of T are precisely $L_1, \dots, L_{i_{\max}}$). We denote the set of indices of reflect layers by $\mathcal{I}_{\text{reflect}}$ (i.e., $\mathcal{I}_{\text{reflect}} := \{i \mid 1 \leq i \leq i_{\max}, L_i \text{ is a reflect layer}\}$) and the set of indices of split layers by $\mathcal{I}_{\text{split}}$ (i.e., $\mathcal{I}_{\text{split}} := \{i \mid 1 \leq i \leq i_{\max}, L_i \text{ is a split layer}\}$). We call T *well-nested* if

1. the number of split layers of T equals the number of reflect layers of T,
2. $1 \in \mathcal{I}_{\text{reflect}}$ and $i_{\max} \in \mathcal{I}_{\text{split}}$, and
3. there is a bijection $\varphi : \mathcal{I}_{\text{reflect}} \to \mathcal{I}_{\text{split}}$ such that
   (a) $\varphi(1) = i_{\max}$,
   (b) for each $i \in \mathcal{I}_{\text{reflect}}$, we have $i < \varphi(i)$, and
   (c) for each $i, j \in \mathcal{I}_{\text{reflect}}$ with $i < j < \varphi(i)$, we have $\varphi(i) > \varphi(j)$.

The following lemma guarantees that the bijection φ from Definition 2.3 is unique.

Lemma 2.4. *Let T be a well-nested b-balanced rooted tree. Then there is precisely one bijection φ as described in Definition 2.3.*

Proof. Let φ be an arbitrary bijection with the properties described in Definition 2.3. Consider some arbitrary $i \in \mathcal{I}_{\text{reflect}}$. Let $j_{\min} \in \{i+1, \dots, i_{\max}\}$ denote the smallest index j satisfying that the set $\{L_i \dots, L_j\}$ contains equally many reflect and split layers. We note that $j_{\min}$ exists and is contained in $\mathcal{I}_{\text{split}}$, by the fact that T is well-nested. Moreover, we claim that $\varphi(i)$ must be identical to $j_{\min}$.

Assume for a contradiction that this is not the case. Consider first the case that $\varphi(i) > j_{\min}$. Then, by the properties of φ described in Definition 2.3, we have $i < \varphi^{-1}(j_{\min}) < j_{\min}$. If the set $\{L_{\varphi^{-1}(j_{\min})}, \dots, L_{j_{\min}}\}$ contains equally many reflect and split layers, then so does the set $\{L_i, \dots, L_{\varphi^{-1}(j_{\min})-1}\}$, contradicting the minimality of $j_{\min}$. Hence, assume that $\{L_{\varphi^{-1}(j_{\min})}, \dots, L_{j_{\min}}\}$ does not contain equally many reflect and split layers. If it contains more reflect layers than split layers, then by the pigeonhole principle there must be some index $\varphi^{-1}(j_{\min}) < k < j_{\min}$ of a reflect layer satisfying $\varphi(k) > j_{\min}$, violating Property 3c in Definition 2.3 for $\varphi^{-1}(j_{\min})$ and k. If $\{L_{\varphi^{-1}(j_{\min})}, \dots, L_{j_{\min}}\}$ contains more split layers than reflect layers, then, similarly, there must be some index $\varphi^{-1}(j_{\min}) < k < j_{\min}$ of a split layer satisfying $\varphi^{-1}(k) < \varphi^{-1}(j_{\min})$, violating Property 3c in Definition 2.3 for $\varphi^{-1}(k)$ and $\varphi^{-1}(j_{\min})$.

Now consider the second case, i.e., that $\varphi(i) < j_{\min}$. Then, analogously to above, we obtain that the set $L_i, \dots, L_{\varphi(i)}$ does not contain equally many reflect and split layers (as otherwise the minimality of $j_{\min}$ would be violated). An analogous argumentation to above then yields that there is some $i < k < \varphi(i)$ satisfying $\varphi(i) < \varphi(k)$ or $\varphi^{-1}(k) < \varphi(i)$, yielding a contradiction to Property 3c in Definition 2.3 in either case. This concludes the proof by contradiction and proves the claim that $\varphi(i) = j$.

Since i and φ were chosen arbitrarily, we obtain that there is at most one bijection φ as described in Definition 2.3. As, by definition, there exists such a bijection due to the well-nestedness of T, it follows that there is exactly one such bijection φ. □

Now, Lemma 2.4 gives rise to the following definition which indexes reflect and split layers.

Definition 2.5 (j-th reflect/split layer)**.** Let T be a well-nested b-balanced rooted tree. Let φ be the unique bijection from Lemma 2.4. Let $i_1 < i_2 < \dots < i_{|\mathcal{I}_{\text{reflect}}|}$ be the elements of $\mathcal{I}_{\text{reflect}}$, ordered by size. Then, for each $1 \leq j \leq |\mathcal{I}_{\text{reflect}}|$, we set $\mathcal{R}_j := L_{i_j}$. We call $\mathcal{R}_j$ the *j-th reflect layer (of T)*. Moreover, for each $1 \leq j \leq |\mathcal{I}_{\text{reflect}}|$, we set $\mathcal{S}_j := L_{\varphi(i_j)}$ and call $\mathcal{S}_j$ the *j-th split layer (of T)*.

For any node v in a layer L_i, we call i the *layer index* of v. Similarly, for any node v in a reflect layer $\mathcal{R}_j$ or split layer $\mathcal{S}_k$, we call j the *reflect index* or k the *split index*, respectively, of v. If $L_i = \mathcal{R}_j$, we call i the *layer index* of $\mathcal{R}_j$ and j the *reflect index* of L_i. Similarly, if $L_i = \mathcal{S}_k$, we call i the *layer index* of $\mathcal{S}_k$ and k the *split index* of L_i.

Note that, for indices $j < k$, the j-th split layer $\mathcal{S}_j$ might be at greater depth from the root r than the k-th split layer $\mathcal{S}_k$ (whereas for reflect layers, $\mathcal{R}_k$ is at greater depth from r than $\mathcal{R}_j$, for all $j < k$ for which $\mathcal{R}_j$ and $\mathcal{R}_k$ exist).

Label considerations. After the above topological considerations, we turn our attention now towards tree labelings since the definition of a construction tree will also involve node labels. Our next definition collects a number of useful notions related to strings/labels.

Definition 2.6 (final substring/independent/clearing)**.** Let $b \geq 3$ be an integer. We denote by X_b the set of all strings of the form $x_j x_{j-1} \dots x_0$ where j is a nonnegative integer, $x_0 \in \{1, 2\}$, and, for all $1 \leq i \leq j$, we have $x_i \in \{1, 2, \dots, b, *\}$, i.e., $X_b = \{x_j x_{j-1} \dots x_0 \mid j \in \mathbb{Z}_{\geq 0}, x_0 \in \{1, 2\}, x_i \in \{1, 2, \dots, b, *\} \text{ for all } 1 \leq i \leq j\}$. Please note that throughout the paper, we will index strings from X_b with indices starting at 0 (starting at the right-most symbol). Moreover, for simplicity, we will write the concatenation of strings and symbols in a compact way, e.g., for some string $x = x_j \dots x_0$ and some symbol z, we write xz to denote the string $x_j \dots x_0 z$ and zx to denote the string $z x_j \dots x_0$.

Consider some string $x = x_j x_{j-1} \dots x_0 \in X_b$. For each $0 \leq i \leq j$, we call x_i the i-th symbol of x. In particular, x_0 is the 0-th symbol of x. Moreover, we define the *length* $\text{len}(x)$ of x to be $j + 1$. For two strings $x, x' \in X_b$ we call x a *final substring* of x' if $\text{len}(x) \leq \text{len}(x')$ and $x_i = x'_i$ for each $0 \leq i \leq \text{len}(x) - 1$. For a finite subset Y of X_b, we define the length $\text{len}(Y) := \max_{y \in Y} \text{len}(y)$ of Y to be the maximum length of a string contained in Y. Furthermore, we define X_b^- as the subset of X_b containing all strings that do not contain the symbol $*$ (i.e., that just use the symbols $1, \dots, b$).

Now consider some finite subset Y of X_b. We call Y *independent* if, for any two strings $y, y' \in Y$ satisfying $y \neq y'$, neither of y and y' is a final substring of the other. Finally, consider some finite subset Y' of X_b^-. We call Y' *clearing* if, for any string $x \in X_b^-$ of length $\text{len}(x) \geq \text{len}(Y')$, there exists a string $y' \in Y'$ that is a final substring of x.

Now we are ready to finally define a construction tree. Essentially, a construction tree is a well-nested balanced rooted tree with a node labeling that satisfies a list of carefully selected properties. For an illustration of a construction tree, see Figure 3.

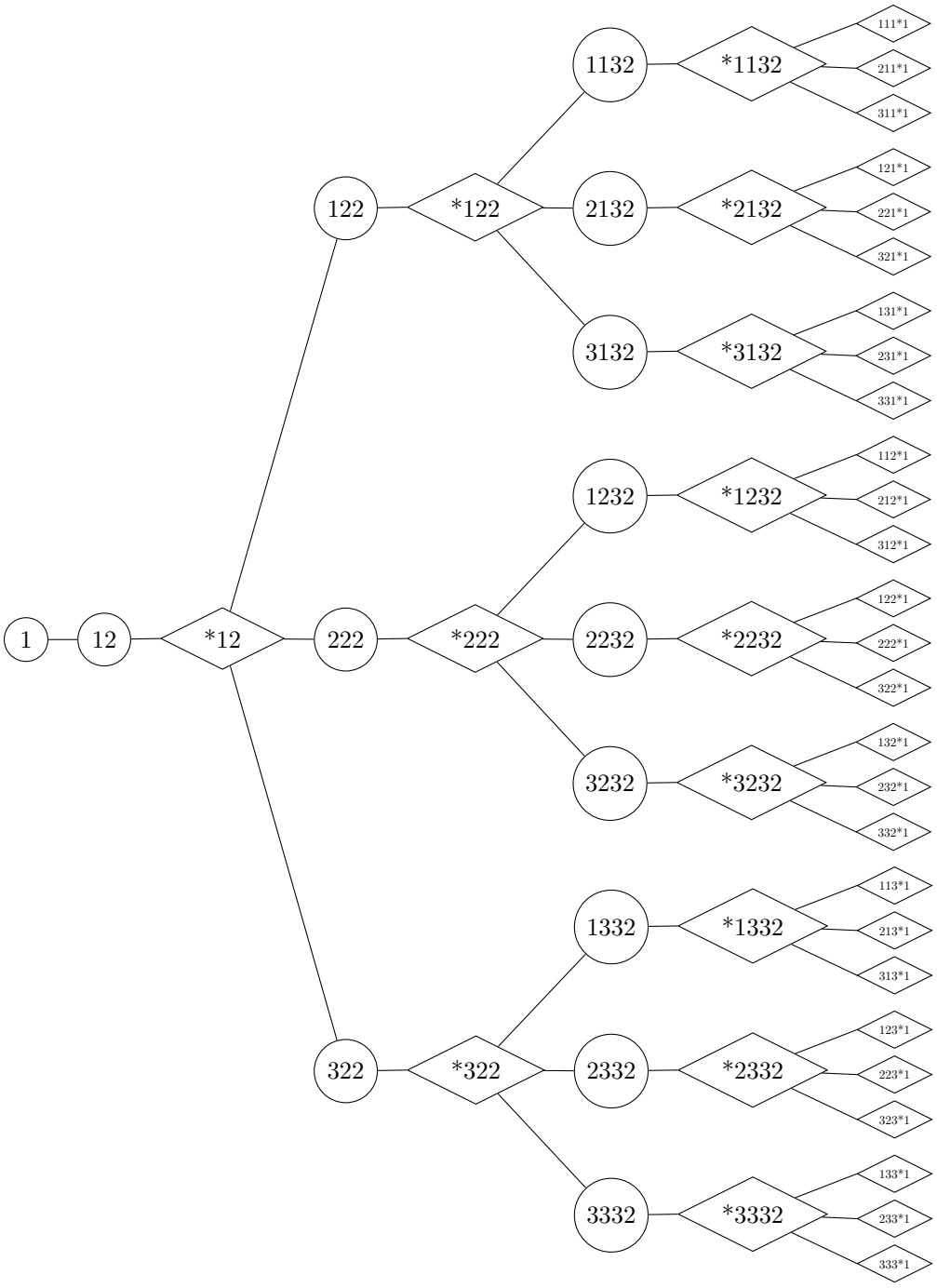

Figure 3: A b-ary construction tree for $b = 3$. Circles and diamonds represent reflect nodes and split nodes, respectively. The string inside a circle/diamond is the label of the respective node.

Definition 2.7 (solid/construction tree)**.** Let $b \geq 3$ be an integer. Let T be a well-nested b-balanced rooted tree with root r. Consider a labeling of the nodes of T with labels from X_b, i.e., a function $\ell : V(T) \to X_b$. We call ℓ *solid* if it satisfies the following properties.

1. $\ell(r) = 1$.

2. For each $i \geq 1$ and each reflect node v in reflect layer $\mathcal{R}_i$, the length of label $\ell(v)$ is i.

3. For any two reflect nodes $v, w \in V_R$ with $v \neq w$, we have $\ell(v) \neq \ell(w)$.

4. For each reflect node v, label $\ell(v)$ does not contain the symbol $*$, i.e., $\ell(v) \in X_b^-$.

5. The set $\{\ell(v) \mid v \in V_R\}$ of labels of all reflect nodes is independent and clearing.

6. For each $i \geq 1$ and each split node v in layer L_i, the length of label $\ell(v)$ is larger by precisely 1 than the number of indices $1 \leq j < i$ such that L_j is a reflect layer.

7. Consider any split node v. Let i be the split index of v and let w be the unique ancestor of v in reflect layer $\mathcal{R}_i$. Then the i-th symbol of $\ell(v)$ is $*$ and the 0-th to $(i-1)$-th symbols of $\ell(v)$ are the same as in $\ell(w)$. Moreover, $\ell(v)$ contains the symbol $*$ precisely once.

8. Consider any internal split node v and let i be the split index of v. Let $w_1, \ldots, w_b$ denote the b children of v. Then there exists a permutation $\rho : \{1, \ldots, b\} \to \{1, \ldots, b\}$ such that, for each $1 \leq j \leq b$ and each descendant u of w_j (including w_j itself), the i-th symbol of $\ell(u)$ is $\rho(j)$.

We call a pair (T, ℓ) a *(b-ary) construction tree* if T is a well-nested b-balanced rooted tree and ℓ a solid labeling of T. For simplicity, we may omit ℓ and use T instead of (T, ℓ).

Moreover, for each internal split node v in some layer $\mathcal{S}_i$ and each $1 \leq j \leq b$, call the child w_j of v satisfying that the i-th symbol of $\ell(w_j)$ is j the *j-th child of v*. (Note that all those children exist due to Property 8.)

The following observation collects some simple properties of construction trees.

Observation 2.8. *Let T be a construction tree. Then the following properties hold.*

1. *The root r of T is a reflect node; the leaves of T are split nodes.*

2. *The 0-th symbol of the label of a node of T is 1 if and only if the node is the root or a leaf; the 0-th symbol of the label of a node of T is 2 otherwise.*

3. *For each node $v \in V(T)$, the length of $\ell(v)$ is larger by precisely 1 than the number of reflect nodes on the path from v to r that are distinct from v.*

Proof. Property 1 follows from the well-nestedness of T. Property 2 follows from the well-nestedness of T and Properties 1, 5, and 7 in Definition 2.7. Property 3 follows from Properties 2 and 6 in Definition 2.7. □

2.2 A Sequence of Construction Trees

While Section 2.1 took care of defining the general concept of a construction tree, the goal of the current section is to define a sequence $T_2, T_3, \ldots$ of infinitely many concrete construction trees (with a growing number of nodes) that we will use later for our lower bound construction. For defining this sequence, we will develop a transformation $F(\cdot)$ that takes as input a construction tree and returns a construction tree. Before defining $F(\cdot)$, we introduce the useful notion of a variant of a given string obtained by a certain padding operation.

Definition 2.9 (padded version of a string)**.** Let $b \geq 3$, $i \geq 1$, and $j \geq 0$ be integers. Let $x \in X_b$ be a string of length i and $\mathcal{K}$ a subset of $\{0, \ldots, i+j-1\}$ of size $|\mathcal{K}| = j$. Let $p_0 < p_1 < \cdots < p_{i-1}$ denote the elements of $\{0, \ldots, i+j-1\} \setminus \mathcal{K}$. Then the *$\mathcal{K}$-padded version of x* is the string y with

symbols from the set $\{1, \ldots, b, *, \square\}$ given by setting $y_{p_q} := x_q$, for each $0 \leq q \leq i-1$, and $y_k := \square$, for each $k \in \mathcal{K}$. In other words, y is the string obtained from x by spreading out the symbols of x such that they occupy the positions whose index is not contained in $\mathcal{K}$ (but in $\{0, \ldots, i+j-1\}$) and filling the positions whose index is contained in $\mathcal{K}$ with the special symbol $\square$.

Now we are set to define $F(\cdot)$.

Definition 2.10 (transformation $F(\cdot)$)**.** Let T be a b-ary construction tree. Recall the definition of the j-th child of a split node in Definition 2.7. Consider the unique depth-first search traversal of T where, for each split node and any $1 \leq i < j \leq b$, the i-th child of the split node is visited before the j-th child of the split node. Let $Q = (v_1, \ldots, v_{2|V_R|})$ be the (chronologically ordered) sequence of reflect nodes visited by the traversal where each reflect node appears twice, once when it is visited for the first time (when the node is left towards its unique child) and once when it is visited for the second time (when the node is left towards its parent, unless it is the root). Note that Observation 2.8 ensures that no reflect node is a leaf of T, which implies that each reflect node is indeed visited twice. Furthermore, for simplicity, we may consider the same node occurring in two different places in Q as two different objects.

Call two nodes in Q *twins* if they correspond to the first and second appearance of the same node of T. Call a node in Q *early* if it is the first appearance of a node from T in Q and *late* if it is the second appearance. If v_i is the first appearance of a reflect node $u \in V(T)$ in Q and v_i is the j-th early node in Q, then we call j the *early rank* of both u and v_i. Moreover, for each $1 \leq i \leq 2|V_R|$, let Λ_i denote the number of late nodes in $\{v_1, \ldots, v_{i-1}\}$.

We now define $F(T)$ as follows. First we consider the topology of $F(T)$. We define the node set of $F(T)$ by $V(F(T)) := \bigcup_{1 \leq i \leq 2|V_R|} L_i^F$ where L_i^F is a set of b^{Λ_i} (so far isolated) nodes. To obtain the edge set $E(F(T))$ of $F(T)$, connect, for each $1 \leq i \leq 2|V_R| - 1$, each node in L_i^F with

1. precisely one node in L_{i+1}^F if v_i is an early node, and
2. precisely b nodes in L_{i+1}^F if v_i is a late node,

in either case such that each node in L_{i+1}^F is connected to precisely one node in L_i^F and the tail of the connecting edge is the respective endpoint in L_{i+1}^F and the head the respective endpoint in L_i^F. This is possible (and uniquely defines the topology of $F(T)$ up to isomorphism) as, by design, $|L_{i+1}^F| = |L_i^F|$ if v_i is an early node, and $|L_{i+1}^F| = b \cdot |L_i^F|$ if v_i is a late node.

Observe that if v_i is early, then each node in L_i^F is a reflect node, and if v_i is late, then each node in L_i^F is a split node (for each $1 \leq i \leq 2|V_R|$). For each $1 \leq j \leq |V_R|$, set $\mathcal{R}_j^F := L_i^F$ where $1 \leq i \leq 2|V_R|$ is the index such that v_i has early rank j. Moreover, for each $1 \leq j \leq |V_R|$, set $\mathcal{S}_j^F := L_{i'}^F$ where $1 \leq i' \leq 2|V_R|$ is the index such that, for the index i satisfying that v_i and $v_{i'}$ are twins, we have $\mathcal{R}_j^F = L_i^F$. Note that the $\mathcal{R}_j^F$ are precisely the reflect layers of $F(T)$ and the $\mathcal{S}_j^F$ the split layers of $F(T)$. Note further that the fact that sequence Q is obtained from a depth-first search traversal guarantees that $F(T)$ is a well-nested b-balanced rooted tree (which in particular implies that notions such as the split index of a node are well-defined in $F(T)$).

For defining the labels of the nodes of $F(T)$, we start by defining, for each reflect layer and each split layer, the set of labels of the nodes in the respective layer. Consider any $1 \leq i \leq 2|V_R|$, and let $\mathcal{K} \subseteq \{1, \ldots, |V_R|\}$ be the set of indices k such that $\mathcal{S}_k^F$ has layer index $< i$.

Consider first the case that L_i^F is a reflect layer, and let j be the reflect index of L_i^F. Then we define the labels of the nodes in L_i^F to be precisely those strings in X_b^- of length j that can be obtained from the $\mathcal{K}$-padded version of $\ell(v_i)$ by replacing the special symbols $\square$ with symbols from $\{1, \ldots, b\}$ in all possible ways. (Note that, by the definitions of Q and $\mathcal{K}$ and Property 2

of Definition 2.7, we have that $\mathcal{K}$ is a subset of $\{0, \ldots, |\mathcal{K}| + \text{len}(\ell(v_i)) - 1\}$, which implies that the $\mathcal{K}$-padded version of $\ell(v_i)$ is well-defined.) For instance, for $b = 2$, $j = 5$, $K = \{2, 3\}$, and $\ell(v_i) = 211$, we obtain that the labels for the nodes in L_i^F are 21111, 21211, 22111, and 22211.[6]

Now consider the case that L_i^F is a split layer, and let j be the number of reflect layers with a strictly smaller layer index than L_i^F. Then we define the labels of the nodes in L_i^F to be precisely those strings in X_b of length $j + 1$ that can be obtained from the $\mathcal{K}$-padded version of $*\ell(v_i)$ by replacing the special symbols $\square$ with symbols from $\{1, \ldots, b\}$ in all possible ways.

Note that, in either case, the size of the obtained set of labels is precisely the number of nodes in L_i^F since there are precisely $|\mathcal{K}|$ split layers with layer index $< i$ (which implies that the number of nodes in L_i^F is $b^{|\mathcal{K}|}$). In either case, we will assign each of the labels to precisely one node in L_i^F. It remains to decide for each layer which node is assigned which of the labels.

We will do so in a way that satisfies the following property. Let $v \in L_i^F$ be a node of $F(T)$ (where $1 \leq i \leq 2|V_R| - 1$) and let $\mathcal{K}$ be defined as above. Then, if v is a reflect node, it is connected to the unique node $w \in L_{i+1}^F$ satisfying $\ell(v)_p = \ell(w)_p$ for each $p \in \mathcal{K}$; if v is a split node, it is connected to the b nodes $w \in L_{i+1}^F$ for which $\ell(v)_p = \ell(w)_p$ for each $p \in \mathcal{K}$. In fact, the b-balancedness of T implies that there is precisely one way (up to isomorphism) to assign labels such that this property is satisfied. We define the label assignment of $F(T)$ to be this assignment.

We will prove in Lemma 2.12 that the tree $F(T)$ is indeed a construction tree. However, before doing so, we will state a useful observation that in particular presents in a slightly more digestible way than Definition 2.10 how to infer the labels of nodes in some layer of $F(T)$ from the "corresponding" node in T. The observation follows directly from Definition 2.10 (and in particular from the fact that Q is obtained from T in a depth-first search manner). Recall the definition of the early rank of a node from Definition 2.10.

Observation 2.11. *Let v be a reflect node of T with reflect index i (which, by Property 2 of Definition 2.7, in particular implies that $\text{len}(v) = i$). Let $\mathcal{J}$ be the set of indices j such that there is an ancestor $w \neq v$ of v in T that is a reflect node and has early rank j. Let $j_1 < j_2 < \cdots < j_{i-1}$ denote the elements of $\mathcal{J}$. Let a be the early rank of v (which, by the definition of Q in Definition 2.10, in particular implies that $j < a$, for each $j \in \mathcal{J}$). Let h be the early rank of the node with greatest early rank in the maximal subtree of T hanging from v. Let v_p denote the first occurrence of v in Q and v_q the second (which in particular implies that v_p and v_q are twins and that $p < q$). Let $\mathcal{K}_p \subseteq \{1, \ldots, |V_R|\}$, resp. $\mathcal{K}_q \subseteq \{1, \ldots, |V_R|\}$, be the set of indices k such that $\mathcal{S}_k^F$ has layer index $< p$, resp. $< q$, in $F(T)$. Then, the following hold.*

1. *The sets $\mathcal{J}$, $\mathcal{K}_p$, and $\{a\}$ are pairwise disjoint, and their union equals $\{1, \ldots, a\}$.*
2. *The sets $\mathcal{J}$, $\mathcal{K}_q$, and $\{a\}$ are pairwise disjoint, and their union equals $\{1, \ldots, h\}$.*
3. *The $\mathcal{K}_p$-padded version of $\ell(v) = \ell(v_p)$ is the string y of length a satisfying $y_0 = \ell(v)_0$, $y_{j_d} = \ell(v)_d$, for each $1 \leq d \leq i - 1$, and $y_k = \square$, for each $k \in \mathcal{K}_p$.*
4. *The $\mathcal{K}_q$-padded version of $\ell(v) = \ell(v_q)$ is the string y of length $h + 1$ satisfying $y_0 = \ell(v)_0$, $y_{j_d} = \ell(v)_d$, for each $1 \leq d \leq i - 1$, $y_a = *$, and $y_k = \square$, for each $k \in \mathcal{K}_q$.*
5. *The labels of the nodes in reflect layer $L_p^f = \mathcal{R}_a^F$ of $F(T)$ are precisely those strings y of length a that satisfy $y_0 = \ell(v)_0$, $y_{j_d} = \ell(v)_d$, for each $1 \leq d \leq i - 1$, and $y_k \in \{1, \ldots, b\}$, for each $k \in \mathcal{K}_p$.*

[6] This is just for the purpose of illustration; we do not claim that this case can actually occur in $F(T)$.

6. *The labels of the nodes in split layer $L_q^f = \mathcal{S}_a^F$ of $F(T)$ are precisely those strings y of length $h+1$ that satisfy $y_0 = \ell(v)_0$, $y_{j_d} = \ell(v)_d$, for each $1 \leq d \leq i-1$, $y_a = *$, and $y_k \in \{1, \ldots, b\}$, for each $k \in \mathcal{K}_q$.*

Lemma 2.12. *Let T be a b-ary construction tree. Then $F(T)$ is a b-ary construction tree.*

Proof. As already observed in Definition 2.10, $F(T)$ is a well-nested b-balanced rooted tree. It remains to show that it also satisfies the properties listed in Definition 2.7.

Properties 1, 2, 3, 4, and 6 follow directly from the definition of $F(T)$.

Next, observe that (the definition of $\mathcal{K}$ and) the property used in Definition 2.10 for deciding for each layer which node is assigned which of the labels that have been reserved for that layer implies directly that $F(T)$ satisfies Property 8 of Definition 2.7.

For proving Property 7, consider some arbitrary split node $u \in V(F(T))$. Let a be the split index of u, and w the unique ancestor of u in reflect layer $\mathcal{R}_a^F$. Let p be the layer index of w, and q the layer index of u. From Definition 2.10, we obtain that there must be some reflect node v of T such that the first occurrence of v in Q is v_p and the second occurrence v_q. Now, by Observation 2.11 and Property 8 of Definition 2.7 for $F(T)$, it follows that the a-th symbol of $\ell(u)$ is $*$ and the 0-th to $(a-1)$-th symbols of $\ell(u)$ are the same as in $\ell(w)$. Moreover, by Definition 2.10, Property 4 of Definition 2.7 (for T), and the fact that v is a reflect node, we obtain that $\ell(u)$ contains the symbol $*$ precisely once, which concludes the proof that $F(T)$ satisfies Property 7 of Definition 2.7.

Now, we turn towards proving Property 5, starting with the independence. Assume for a contradiction that the set of labels of all reflect nodes of $F(T)$ is not independent, and let u and w be two reflect nodes of $F(T)$ such that $\ell(u)$ is a final substring of $\ell(w)$. By Property 3 of Definition 2.7 for $F(T)$, we know that $\ell(u) \neq \ell(w)$, which in particular implies that u and w are in different layers in $F(T)$ (as the labels of the nodes in any fixed layer have the same length). Let i_u be the layer index of u in $F(T)$ and i_w the layer index of w. Recall the definition of $Q = (v_1, \ldots, v_{2|V_R|})$ from Definition 2.10.

Let u' be the reflect node in T such that v_{i_u} is the first occurrence of u' in Q, and w' the reflect node in T such that v_{i_w} is the first occurrence of w' in Q (which in particular implies that $u' \neq w'$). Let v' denote the common ancestor of u' and w' in T that has the largest distance from the root r. We consider three cases.

First, consider the case that $u' \neq v' \neq w'$. Then v' is a split node. Let j be the split index of v'. By Property 8 of Definition 2.7, we obtain that $\mathrm{len}(\ell(u')) \geq j+1$, $\mathrm{len}(\ell(w')) \geq j+1$ and $\ell(u')_j \neq \ell(w')_j$. Moreover, by the well-nestedness of T, there is a (unique) ancestor of v' in reflect layer L_j. Call this ancestor $\hat{v}$, and let k denote the early rank of $\hat{v}$ in Q. Then, by Observation 2.11 and the fact that $\hat{v}$ is an ancestor of both u' and w' in T (with $u' \neq \hat{v} \neq w'$), we obtain $\ell(u)_k = \ell(u')_j$ and $\ell(w)_k = \ell(w')_j$. Since $\ell(u')_j \neq \ell(w')_j$ (as observed above), it follows that $\ell(u)_k \neq \ell(w)_k$, yielding a contradiction to the fact that $\ell(u)$ is a final substring of $\ell(w)$.

Next, consider the case that $u' = v'$. Let j be the reflect index of u'. Since $u' \neq w'$ and $\mathrm{len}(\ell(u')) = j$ (by Property 2 of Definition 2.7), it follows, by Property 5 of Definition 2.7, that there is some $j' \leq j-1$ such that $\ell(u')_{j'} \neq \ell(w')_{j'}$. Let $\hat{v}$ denote the unique ancestor of u' in reflect layer $L_{j'}$, and let k denote the early rank of $\hat{v}$ in Q. Then, analogously to the first case, we obtain $\ell(u)_k = \ell(u')_{j'} \neq \ell(w')_{j'} = \ell(w)_k$, yielding the desired contradiction.

Finally, for the case that $w' = v'$, we obtain a contradiction analogously to the case $u' = v'$. This concludes the proof by contradiction and shows that the set of labels of all reflect nodes of $F(T)$ is independent.

We continue by showing that this set is also clearing. Let p denote the number of reflect nodes of T and q the number of reflect layers of T. Observe that, by the construction of $F(T)$, the number of reflect layers of $F(T)$ is also equal to p. By Property 2 of Definition 2.7 for $F(T)$, we obtain that

the length of the set of labels of all reflect nodes of $F(T)$ (as defined in Definition 2.6) is also equal to p. Consider an arbitrary string $z \in X_b^-$ of length $\geq p$.

We define a sequence $W = (w_1, w_2, \dots)$ of nodes of T and a sequence $Y = (y^{(1)}, y^{(2)}, \dots)$ of strings as follows. Set $w_1 := r$ and $y^{(1)} := z_0$ (which in particular implies that $w_1 \in \mathcal{R}_1$, by Observation 2.8). Then recursively do the following for $i = 1, 2, \dots, q-1$. If $\ell(w_i) = y^{(i)}$, then terminate the recursion and set $w := w_i$ and $y := y^{(i)}$; otherwise do the following. Let $\mathcal{J} \subseteq \{1, \dots, q\}$ be the set of all indices j such that the layer index of $\mathcal{S}_j$ is strictly greater than the layer index of $\mathcal{R}_i$ and strictly smaller than the layer index of $\mathcal{R}_{i+1}$. Set $y^{(i+1)} := z_k y^{(i)}$ where k is the early rank of w_i. Define w_{i+1} to be the unique descendant of w_i in reflect layer $\mathcal{R}_{i+1}$ satisfying that the j-th symbol of $\ell(w_{i+1})$ is equal to $y_j^{(i+1)}$, for each $j \in \mathcal{J}$. (Such a descendant exists (and is unique) due to Property 8 of Definition 2.7.) If the recursion is not terminated before completing all recursion steps, set $w := w_q$ and $y := y^{(q)}$.

Let h be the early rank of w. We claim that there exists a node u in reflect layer $\mathcal{R}_h^F$ of $F(T)$ such that $\ell(u)$ is a final substring of z.

Towards proving this claim, we first show that $\ell(w) = y$. If the above recursion terminates before completing all recursion steps, this follows directly from the definition of the recursive process. Hence, assume that the recursion does not terminate before completing all recursion steps, which implies that $w = w_q$ and $y = y^{(q)}$. By Property 2 of Definition 2.7, it follows that $\mathrm{len}(y) = q = \mathrm{len}(\{\ell(v) \mid v \in V_R\})$ (where V_R denotes the set of reflect nodes of T). Hence, by Property 5 of Definition 2.7, there must be some reflect node $\hat{w}$ such that $\ell(\hat{w})$ is a final substring of y.

If $\hat{w}$ does not lie on the path from r to w, then there must be some split node that is an ancestor of both w and $\hat{w}$ (since w is contained in the final reflect layer $\mathcal{R}_q$); let w' denote the split node of this kind with largest distance from the root r. But then, for the split index a of w', the construction of y (together with Property 8) guarantees that $\ell(\hat{w})_a \neq y_a$, contradicting the fact that $\ell(\hat{w})$ is a final substring of y. Hence, assume that $\hat{w}$ lies on the path from r to w, and let $\hat{i}$ be the reflect index of $\hat{w}$. By the definition of the recursive process, we obtain that $\hat{w} = w_{\hat{i}}$ and $\ell(\hat{w}) = y^{\hat{i}}$, which implies that the recursion terminates after the recursion step for $i = \hat{i} - 1$. As the recursion does not terminate before completing all recursion steps, we obtain that $\hat{i} = q$ and $\ell(w) = \ell(\hat{w}) = y^{\hat{i}} = y$.

Let g denote the reflect index of w and recall that h denotes the early rank of w. From the definition of the recursive process above, it follows that the reflect nodes of T that are ancestors of w are precisely $r = w_1, w_2, \dots, w_g = w$, ordered by increasing distance from the root r. Let h_d denote the early rank of w_d, for each $1 \leq d \leq g-1$. By Observation 2.11, the fact that $\ell(w) = y$, and the definition of the $y^{(i)}$ in the recursive process, we obtain that the labels of the nodes in reflect layer $\mathcal{R}_h^F$ of $F(T)$ are precisely those strings $x \in X_b^-$ of length h that satisfy $x_0 = y_0 = z_0$ and $x_{h_d} = y_d = z_{h_d}$, for each $1 \leq d \leq g-1$. Since $h \leq p$, it follows that there exists some node $u \in \mathcal{R}_h^F$ such that $\ell(u)$ is a final substring of z. This shows that the set of labels of all reflect nodes of $F(T)$ is clearing and concludes the proof that $F(T)$ satisfies Property 5 of Definition 2.7. □

The Construction Tree Sequence. Now we are set to finally define the desired sequence $T_2, T_3, \dots$ of construction trees (where the fact that we start with index 2 is for technical reasons). Please note that the construction tree from Figure 3 is actually T_2 for $\Delta = 3$.

Definition 2.13. We define T_2 as follows. Tree T_2 has $\Delta^3 + 2\Delta^2 + 2\Delta + 3$ nodes, with labels

- $1, 12, *12$,
- $j22, *j22$ for each $1 \leq j \leq \Delta$,

- $kj32, *kj32$ for each $1 \leq j, k \leq \Delta$, and
- $pkj*1$ for each $1 \leq j, k, p \leq \Delta$.

The node with label 1 is the root of T_2, and the edges of T_2 are as follows, where for simplicity, we represent a node by its label.

- $(1, 12), (12, *12)$,
- $(*12, j22), (j22, *j22)$, for each $1 \leq j \leq \Delta$,
- $(*j22, kj32), (kj32, *kj32)$ for each $1 \leq j, k \leq \Delta$, and
- $(*kj32, pkj*1)$ for each $1 \leq j, k, p \leq \Delta$.

Moreover, for any integer $i \geq 3$, we set $T_i := F(T_{i-1})$.

The reflect nodes of T_2 are those that do not contain the symbol $*$; the split nodes are those that do contain the symbol $*$. It is straightforward to verify that T_2 is a well-nested Δ-balanced rooted tree with a solid labeling, i.e., that T_2 is a Δ-ary construction tree. By Lemma 2.12, this implies that T_i is a Δ-ary construction tree, for any integer $i \geq 2$.

Bounding the number of nodes. Next we bound the number of nodes of the trees T_i. To this end, we introduce the following notation, capturing certain power towers in a convenient format.

Definition 2.14. Let j, k, and $\Delta \geq 3$ be positive integers. We denote the power tower of height[7] j with the top exponent being k and the other $j-1$ entries being Δ by $P_\Delta(j, k)$. (For instance, $P_\Delta(3, 5) = \Delta^{\Delta^5}$.)

Now, for each positive integer i, we upper bound the number of nodes of T_i.

Lemma 2.15. *For each integer $i \geq 2$ (and any $\Delta \geq 3$), the number of nodes of T_i is at most $P_\Delta(i, \Delta + 1)$.*

Proof. From the definition of the T_i (and the fact that, due to its well-nestedness, any construction tree contains at least as many split nodes as reflect nodes), we obtain directly that, if k denotes the number of nodes of tree T_i, then T_{i+1} has at most $\Delta^{k/2}$ leaf nodes. This in turn implies that T_{i+1} has at most $2\Delta^{k/2}$ split nodes, which implies that the total number of nodes of T_{i+1} is at most $4\Delta^{k/2}$, due to the well-nestedness of T_{i+1}. Note that, for any $k \geq 4$ (and any $\Delta \geq 3$), we have $\Delta^{k/2} > 4$, which implies $\Delta^k > 4\Delta^{k/2}$. Together with the fact that the number of nodes of T_2 is $\Delta^3 + 2\Delta^2 + 2\Delta + 3$ (and the fact that $\Delta^3 + 2\Delta^2 + 2\Delta + 3 \leq \Delta^4 \leq \Delta^{\Delta+1}$ for any $\Delta \geq 3$), the above discussion implies the lemma statement. □

In the following, we show that the number of nodes of the T_i is not far away from the upper bound presented in Lemma 2.15. Please note that this is not necessary for obtaining the lower bound we present in this work; the purpose of the following lemma is to imply that we cannot obtain a stronger lower bound (asymptotically in the exponent) by improving the analysis in the proof of Lemma 2.15.

Lemma 2.16. *For each integer $i \geq 2$ (and any $\Delta \geq 3$), the number of nodes of T_i is at least $P_\Delta(\lfloor i/2 \rfloor, \Delta + 1)$.*

[7]We say that a power tower has height j if it consists of j entries; e.g., the power tower 5^{6^7} has height 3.

Proof. We start by observing that the definition of $F(\cdot)$ ensures that if k denotes the number of nodes in the last reflect layer of some construction tree T_i, then the number of nodes in the last reflect layer of T_{i+1} is at least Δ^{k-1} (as, in T_{i+1}, there must be at least $k-1$ split layers with smaller layer index than the last reflect layer). Since $\Delta^{k-1} - 1 \geq k$ (for any $k \geq 2$ and $\Delta \geq 3$), we obtain that the number of nodes in the last reflect layer of T_{i+2} is at least Δ^k. As the number of reflect nodes in the last reflect layer of T_2 is $\Delta^2 \geq \Delta + 1$, the lemma statement follows. □

2.3 Auxiliary Labelings of Construction Trees

Before describing (in Section 2.4) how the information stored in a construction tree is used to create an actual input instance, we introduce two auxiliary labelings of construction trees that will be highly useful for proving statements about the input instance and related objects. More specifically, we introduce two auxiliary labelings of the edges of the construction tree T and collect some basic properties of the auxiliary labelings. On a high level, these two labelings capture information about the structure and labels of a certain sequence of objects that we will use to derive the input tree from the construction tree in Section 2.4.

We start by introducing the first of the two labelings, which we call ψ. Recall the definition of X_b^- from Definition 2.6.

Definition 2.17 (edge labeling ψ)**.** Let T be a Δ-ary construction tree. We define an edge labeling $\psi : E(T) \to 2^{X_\Delta^-}$ that assigns to each edge of T a finite (and possibly empty) subset of X_Δ^- via the following inductive process.

Let e be the edge whose head is the root r of T. (Note that Observation 2.8 asserts that r is a reflect node, which implies that r has only one incoming edge.) Then we set $\psi(e) := \{i2 \mid 1 \leq i \leq \Delta\}$.

Now, let $e = (u, v)$ be any edge of T not connected to the root and assume that the unique edge $e' = (v, w)$ with tail v has already been labeled w.r.t. ψ. If v is a reflect node, then we set $\psi(e) := \{iz \mid 1 \leq i \leq \Delta, z \in \psi(e'), z \neq \ell(v)\}$. If v is a split node, then we set $\psi(e) := \{z \mid z \in \psi(e'), z_j = i\}$ where j is the index such that the j-th symbol of $\ell(v)$ is $*$ (which is unique by Property 7 of Definition 2.7), and $i \in \{1, \ldots, \Delta\}$ is the j-th symbol of $\ell(u)$ (i.e., $(\ell(v))_j = *$ and $\ell(u)_j = i$). (Note that it is impossible that $u_j = *$ due to Properties 7 and 8 of Definition 2.7.)

Next, we prove that the node labeling of T is related to the edge labeling ψ via a useful property.

Lemma 2.18. *Let $v \neq r$ be a reflect node in T and $e = (v, w)$ the edge leading to its parent w. Then $\ell(v) \in \psi(e)$.*

Proof. Set $z := \ell(v)$. Consider the path $r = u_1, u_2, \ldots, u_k = v$ connecting r and v. For each $2 \leq i \leq k$, denote the number of reflect nodes in $\{u_1, \ldots, u_{i-1}\}$ by Λ_i. Observe that the length of z is $\Lambda_k + 1$, by Property 2 of Definition 2.7. Furthermore, for each $2 \leq i \leq \Lambda_k + 1$, set $z^{(i)} := z_{i-1} z_{i-2} \ldots z_0$ to be the final substring of z of length i. We prove by induction that, for each $2 \leq i \leq k$, the string $z^{(\Lambda_i + 1)}$ is contained in $\psi((u_i, u_{i-1}))$.

For the base case $i = 2$, it suffices to observe that $z^{(\Lambda_i+1)} = z^{(2)}$ is contained in $\psi((u_2, u_1)) = \{12, 22, \ldots, \Delta 2\}$ since the 0-th symbol of z is 2 (where we make use of Observation 2.8). For the induction step, assume that the induction hypothesis holds for $i = p$ (where $2 \leq p \leq k-1$). We show that it then also holds for $i = p + 1$.

Consider first the case that u_p is a reflect node. Then $\Lambda_{p+1} = \Lambda_p + 1$, and the string $z^{(\Lambda_{p+1}+1)} = z^{(\Lambda_p+2)}$ is contained in the set $\{jz^{(\Lambda_p+1)} \mid 1 \leq j \leq \Delta\}$. By the induction hypothesis for $i = p$, we know that $z^{(\Lambda_p+1)}$ is contained in $\psi((u_p, u_{p-1}))$. Moreover, by Property 5 of Definition 2.7, we know that $\ell(u_p)$ is not a final substring of z, which implies $z^{(\Lambda_p+1)} \neq \ell(u_p)$. It follows that $z^{(\Lambda_{p+1}+1)}$ is contained in $\{jy \mid 1 \leq j \leq \Delta, y \in \psi((u_p, u_{p-1})), y \neq \ell(u_p)\} = \psi((u_{p+1}, u_p))$, as desired.

Now consider the case that u_p is a split node. Since $p < k$, node u_p is an internal split node. Let j be the index such that the j-th symbol of $\ell(u_p)$ is $*$ (which is unique by Property 7 of Definition 2.7). By Property 8 of Definition 2.7, we know that the j-th symbol of $\ell(u_{p+1})$ is identical to z_j, which, by the definition of ψ, implies that $\psi((u_{p+1}, u_p)) = \{y \mid y \in \psi((u_p, u_{p-1})), y_j = z_j\}$. Since $\Lambda_{p+1} = \Lambda_p$, it follows by the induction hypothesis that $z^{(\Lambda_{p+1}+1)} = z^{(\Lambda_p+1)}$ is contained in $\psi((u_{p+1}, u_p))$, as desired. This concludes the proof by induction.

By setting $i := k$, we obtain that $\ell(v) = z^{(\Lambda_k+1)}$ is contained in $\psi(e) = \psi((u_k, u_{k-1}))$. □

Now, we define the second of the mentioned edge labelings.

Definition 2.19 (edge labeling π)**.** Let T be a Δ-ary construction tree. We define an edge labeling $\pi : E(T) \to 2^{X_\Delta}$ that assigns to each edge of T a finite subset of X_Δ via the following inductive process.

Let e be the edge whose head is the root r of T. Then we set $\pi(e) := \{*1\}$.

Now, let $e = (u, v)$ be any edge of T not connected to the root and assume that the unique edge $e' = (v, w)$ with tail v has already been labeled w.r.t. π. If v is a reflect node, then we set $\pi(e) := \{iz \mid 1 \leq i \leq \Delta, z \in \pi(e'), z \neq \ell(v)\} \cup \{*\ell(v)\}$.[8] If v is a split node, then we set $\pi(e) := \{z \mid z \in \pi(e'), z_j = i\}$ where j is the index such that the j-th symbol of $\ell(v)$ is $*$ (which is unique by Property 7 of Definition 2.7), and $i \in \{1, \ldots, \Delta\}$ is the j-th symbol of $\ell(u)$ (i.e., $(\ell(v))_j = *$ and $\ell(u)_j = i$).

Analogous to the statement Lemma 2.18 makes about ψ, the following lemma shows a relation between π and the node labeling of T.

Lemma 2.20. *Let $v \neq r$ be a split node in T and $e = (v, w)$ the edge leading to its parent w. Then $\ell(v) \in \pi(e)$.*

Proof. Let i be the index such that $v \in \mathcal{S}_i$ and let u be the unique predecessor of v in reflect layer $\mathcal{R}_i$. Set $z := \ell(v)$ and $y := \ell(u)$. By Property 7 of Definition 2.7, we know that $z_i = *$ and $z_j = y_j$, for each $0 \leq j \leq i - 1$. Consider the path $u = u_1, u_2, \ldots, u_k = v$ connecting u and v.

Similar to the proof of Lemma 2.18, for each $1 \leq p \leq k$, denote the number of reflect nodes on the path connecting the root r to the parent of u_p by Λ_p. Observe that the length of z is $\Lambda_k + 1$, by Property 6 of Definition 2.7, and that the length of y is $\Lambda_1 + 1 = i$, by Property 2 of Definition 2.7, which in particular implies that y is a final substring of z. Furthermore, for each $2 \leq p \leq \Lambda_k + 1$, set $z^{(p)} := z_{p-1} z_{p-2} \ldots z_0$ to be the final substring of z of length p. We prove by induction that, for each $2 \leq p \leq k$, the string $z^{(\Lambda_p+1)}$ is contained in $\pi((u_p, u_{p-1}))$.

For the base case $p = 2$, it suffices to observe that the definition of π together with $\text{len}(y) = \Lambda_1 + 1 = \Lambda_2$ implies that $z^{(\Lambda_2+1)} = *y$ is contained in $\pi((u_2, u_1))$. For the induction step, assume that the induction hypothesis holds for $p = q$ (where $2 \leq q \leq k - 1$). We show that it then also holds for $p = q + 1$.

Consider first the case that u_q is a reflect node. By Property 5 of Definition 2.7, we know that $\ell(u_q)$ is not a final substring of y and vice versa, which, by the fact that y is a final substring of z, implies that $\ell(u_q)$ is not a final substring of z (and vice versa). Hence, the fact that $\Lambda_{q+1} = \Lambda_q + 1$ and the induction hypothesis for $p = q$, together with the definition of π, directly imply that $z^{(\Lambda_{q+1}+1)} = z^{(\Lambda_q+2)}$ is contained in $\pi((u_{q+1}, u_q))$.

Now consider the case that u_q is a split node. Since $q < k$, node u_q is an internal split node. Let j be the index such that the j-th symbol of $\ell(u_q)$ is $*$ (which is unique by Property 7 of

[8] In fact, one can show that $z \in \pi(e')$ implies $z \neq \ell(v)$ but at this point it is easier to explicitly add the condition $z \neq \ell(v)$.

Definition 2.7). By Property 8 of Definition 2.7, we know that the j-th symbol of $\ell(u_{q+1})$ is identical to z_j, which, by the definition of π, the induction hypothesis, and the fact that $\Lambda_{q+1} = \Lambda_q$ implies that $z^{(\Lambda_{q+1}+1)} = z^{(\Lambda_q+1)}$ is contained in $\pi((u_{q+1}, u_q))$. This concludes the proof by induction.

By setting $p := k$, we obtain that $\ell(v) = z^{(k+1)}$ is contained in $\pi(e) = \pi((u_k, u_{k-1}))$. □

Next, we examine the edge labels on the edges incident to leaf nodes of T.

Lemma 2.21. *Let v be a leaf node of T and e the unique edge incident to v. Then $\psi(e) = \emptyset$ and $\pi(e) = \{\ell(v)\}$.*

Proof. We start by showing $\psi(e) = \emptyset$. For a contradiction assume that $\psi(e) \neq \emptyset$, and let $z \in X_\Delta^-$ be some string contained in $\psi(e)$. Let k denote the number of reflect layers in T, which, by the definition of ψ (and Observation 2.8) implies that $\text{len}(z) = k + 1$ and that the label of any reflect node of T has length at most k. By Property 5 of Definition 2.7, it follows that there is some reflect node u of T such that $\ell(u)$ is a final substring of z. Let w denote the common ancestor of both u and v that has the largest distance from the root r, which in particular implies that w is a split node or $u = w$.

Consider first the case that $u \neq w$, and let i be the index such that the i-th symbol of $\ell(w)$ is $*$. Then the definition of ψ, together with Properties 7 and 8 of Definition 2.7, implies that the i-th symbol of $\ell(u)$ and the i-th symbol of z (exist and) differ, violating the fact that $\ell(u)$ is a final substring of z.

Now consider the case that $u = w$, which implies that u is an ancestor of v. Observe that the design of ψ ensures that $\psi(e)$ contains only strings whose 0-th symbol is 2, which implies that $u \neq r$ (by the definition of u). Let e' be the edge with tail u, and e'' the edge with head u. By the definition of ψ and Property 2 of Definition 2.7, the length of all strings in $\psi(e')$ is equal and identical to the length of $\ell(u)$. This implies, again due to the definition of ψ, that $\psi(e'')$ does not contain any string of which $\ell(u)$ is a final substring. Now observe that the definition of ψ guarantees that if, for some edge f and some string y, it holds that there is no substring in $\psi(f)$ of which y is a final substring, then this also holds for any edge f' whose head is the tail of f (for the same y). By applying this argument inductively along the path from u to v, the fact that u is an ancestor of v implies that $\ell(u)$ is not a final substring of z, yielding a contradiction and concluding the proof that $\psi(e) = \emptyset$.

Next, we show that $\pi(e) = \{\ell(v)\}$. For a contradiction, assume that $\pi(e) \neq \{\ell(v)\}$. Since $\ell(v) \in \pi(e)$ (by Lemma 2.20), there must exist some string z such that $z \in \pi(e)$ and $z \neq \ell(v)$. As before, let k denote the number of reflect layers in T. Then the design of π (together with Observation 2.8) ensures that $\text{len}(z) = k+1$ and that there exists some index $1 \leq i \leq k$ such that the i-th symbol of z is $*$. Let u be the unique ancestor of v in split layer $\mathcal{S}_i$ and w the unique ancestor of v in reflect layer $\mathcal{R}_i$ (which also implies that w is an ancestor of u). Let e' denote the edge with head w and e'' the edge with tail u. By Definition 2.7 and the definition of π, there is precisely one string $y \in \pi(e')$ satisfying that the i-th symbol of y is $*$. Moreover, by the well-nestedness of T (see Definition 2.3), Definition 2.7, and the definition of π, we obtain that the only string in $\pi(e'')$ whose i-th symbol can possibly be $*$ is $\ell(u)$. This implies that, for each edge e''' with head u, the set $\pi(e''')$ does not contain a string whose i-th symbol is $*$ (again, by the definition of π). Now, the definition of π guarantees (similarly to our considerations for ψ) that this is also true for any edge whose tail is a descendant of u. In particular, we obtain that $\pi(e)$ does not contain z, yielding a contradiction, and concluding the proof. □

Lastly, we prove a technical lemma that relates the labels on an edge whose tail is some split node v to the labels on the edge whose head is the reflect node "corresponding" to v.

Lemma 2.22. *Let v be a split node of T with split index i and let u be the unique ancestor of v in reflect layer $\mathcal{R}_i$. Let e denote the edge with head u and e' the edge with tail v. Let $w_1, \ldots, w_k$ denote the split nodes on the path from u to v (excluding v, and ordered in increasing distance from the root). For each $1 \leq j \leq k$, let p_j denote the split index of w_j and q_j the element from $\{1, \ldots, \Delta\}$ satisfying that the child of w_j that lies on the path from u to v is the q_j-th child of w_j. Set $Y := \{y_{i+k} \ldots y_0 \mid y_{p_j} = q_j, \text{ for each } 1 \leq j \leq k, \text{ and } y_i \ldots y_0 \in \psi(e) \cup \pi(e)\}$.[9] Then $\psi(e') \cup \pi(e') \subseteq Y$.*

Proof. By the definitions of ψ and π and Lemma 2.20, for any two edges f and f' such that the tail of f is the head of f', it holds that, for any string $z' \in \psi(f') \cup \pi(f')$, there exists some string $z \in \psi(f) \cup \pi(f)$ such that z is a final substring of z'. Applying this argument inductively along the path from u to v, we obtain that for each string $x' \in \psi(e') \cup \pi(e')$, there must be some string $x \in \psi(e) \cup \pi(e)$ such that x is a final substring of x'. Since each string in $\psi(e) \cup \pi(e)$ has length $i+1$ (by the definitions of ψ and π, and Property 2 of Definition 2.7), we obtain that for each string $x' \in \psi(e') \cup \pi(e')$, we have $x'_i \ldots x'_0 \in \psi(e) \cup \pi(e)$. Moreover, by the well-nestedness of T and the definitions of ψ and π, we also obtain for each string $x' \in \psi(e') \cup \pi(e')$ that the length of x' is $i+k+1$, and that $x'_{p_j} = q_j$, for each $1 \leq j \leq k$. It follows that $\psi(e') \cup \pi(e') \subseteq Y$. □

2.4 The Input Tree

In this section, we show how to use the information stored in a construction tree to derive an input tree from which the actual lower bound instances we use in Section 3 can be obtained by minor adaptations (and by combining it with a suitable sequence of queries in the randomized online-LOCAL model).

Let T be a construction tree. We start by showing how to construct from T a new tree G_T (with node labels) that will be the aforementioned input tree. To construct G_T from T, we will make use of two operations—reflecting and splitting. Before defining these two operations, we define the notion of a *marked tree*.

Definition 2.23 (marked tree)**.** Let $\Delta \geq 3$ be some integer. A *marked tree* is a triple (G, ℓ, W) where G is a tree of maximum degree Δ, $\ell : V(G) \to X_\Delta$ is a labeling of the nodes of G with labels from X_Δ, and $W \subseteq V(G)$ is a subset of the nodes of G. We call the nodes in W *marked* and the nodes in $V(G) \setminus W$ unmarked. For simplicity, we may omit ℓ and W and use G instead of (G, ℓ, W) when considering a marked graph.

Moreover, we call a leaf (resp. node) v of G a *2-leaf (resp. 2-node)* if the 0-th symbol of $\ell(v)$ is 2, and a *1-leaf (resp. 1-node)* if the 0-th symbol of $\ell(v)$ is 1. Finally, we call nodes of G that are not leaves *internal nodes*.

Now we are set to define the reflection and split operations.

Definition 2.24 (reflection)**.** Let G be a marked tree and v an unmarked 2-leaf of G. Consider the maximal connected components of the subgraph of G induced by the set of all unmarked nodes. Let H_1 denote the maximal connected component that contains v. In the following, we define a new tree, called $R_v(G)$, that we say is *obtained from G by reflection at v*.

Start by creating $\Delta - 1$ copies $H_2, \ldots, H_\Delta$ of H_1 (which includes copying node labels and whether a node is marked or unmarked). Next, for each $1 \leq i \leq \Delta$ and each $w \in V(H_i)$ that is neither v nor a copy of v, replace the label $\ell(w)$ of w by the string obtained by prepending symbol i to $\ell(w)$. After this, "join" the graphs $G, H_2, \ldots, H_\Delta$ by identifying $v \in V(H_1) \subseteq V(G)$ with its copies in the

[9] Note that Y is well-defined, by the well-nestedness of T, the definitions of ψ and π, and Property 2 of Definition 2.7.

other H_i. (Note that all identified nodes had the same label and were all unmarked, which we will assume to be preserved during the identification.) Next, replace the label $\ell(v)$ of v by the string obtained by prepending symbol $*$ to $\ell(v)$. Finally, change the port numbers at v so that, for each $1 \leq i \leq \Delta$, the port number that v assigns to its incident edge in H_i is i. The obtained tree is the aforementioned tree $R_v(G)$.

When convenient, we will consider $R_v(G)$ as a supergraph of G with partially changed labels in the natural way. Moreover, for each node $w \neq v$ in H_1, we call its copy in H_i the *i-copy of w*, for each $1 \leq i \leq \Delta$.

Definition 2.25 (split)**.** Let G be a marked tree and v an unmarked node of G. We define the tree $S_v(G)$ as the tree obtained from G by marking v and say that $S_v(G)$ is *obtained from G by a split at v*.

When convenient, we will consider $S_v(G)$ as a supergraph of G with a slightly changed set of marked nodes in the natural way.

Note that it directly follows from the definitions of the reflection and split operations that $R_v(G)$ and $S_v(G)$ are marked trees. Given these operations, we are now ready to define the tree G_T.

Definition 2.26 (G_T)**.** Let $\Delta \geq 3$ be an integer and let T be a Δ-ary construction tree. Let $v^{(1)}, v^{(2)}, \ldots, v^{(|V(T)|)}$ be an arbitrary ordering of the nodes in $V(T)$ such that, for any $1 \leq i < j \leq |V(T)|$, either $v^{(j)}$ is a descendant of $v^{(i)}$ or $v^{(j)}$ and $v^{(i)}$ are incomparable. In other words, a node appears in the sequence only after all of its ancestors have appeared. Now, the marked tree G_T is the tree obtained in the following way.

Start with the marked tree $G^{(0)}$ consisting of two unmarked nodes v, w connected by an edge with labels $\ell(v) = 1$ and $\ell(w) = 2$. Then, modify $G^{(0)}$ iteratively by following the "instructions" written in the construction tree T. More precisely, construct a sequence $G^{(0)}, G^{(1)}, \ldots, G^{(|V(T)|)}$ as follows.

For each $1 \leq i \leq |V(T)|$, we define $G^{(i)} := R_{v^{(i)}}(G^{(i-1)})$ if $v^{(i)}$ is a reflect node and $G^{(i)} := S_{v^{(i)}}(G^{(i-1)})$ if $v^{(i)}$ is a split node. In other words, starting with $G^{(0)}$, we process the nodes in the aforementioned node sequence one by one, each time performing a reflect or split operation at the processed node in the currently obtained marked tree, where the choice whether a reflect or split operation is performed is given by the type of the processed node in the construction tree.

Finally, set $G_T := G^{(|V(T)|)}$. We call G_T the *input tree induced by T*.

From the above description, it is not obvious that the tree G_T is well-defined: to this end, we have to show that for each indicated reflection and split operation, the node at which the reflection/split is supposed to happen actually exists and is unique[10] in the currently obtained graph (and is an unmarked 2-leaf in the case of a reflection operation and unmarked in the case of a split operation), and that the resulting graph G_T is independent of the choice of the sequence $v^{(1)}, v^{(2)}, \ldots, v^{(|V(T)|)}$. We will take care of this in the following lemma. (Note that the fact that the nodes at which an operation is to be performed actually exist in the respectively obtained graphs and have the required properties will then also imply that the graphs $G^{(0)}, G^{(1)}, \ldots, G^{(|V(T)|)}$ are all marked trees.)

Lemma 2.27. *The tree G_T is well-defined. In particular, G_T is independent of the choice of $v^{(1)}, v^{(2)}, \ldots, v^{(|V(T)|)}$.*

[10] Here, "existence and uniqueness" is meant in the sense that there is precisely one node in the currently obtained graph whose label is identical to the label that the node at which the reflection/split is supposed to happen has in the construction tree T.

Proof. We start by showing that whenever the described construction of G_T specifies performing a reflection or a split, the node at which the reflection/split is supposed to happen actually exists and is unique in the obtained graph and that the node is an unmarked 2-leaf in the case of a reflection operation and unmarked in the case of a split operation. Our proof proceeds by induction. More precisely, for any fixed sequence $v^{(1)}, v^{(2)}, \ldots, v^{(|V(T)|)}$, we will prove the following statement by induction on i.

For each $1 \leq i \leq |V(T)|$, the following hold:

1. $G^{(i-1)}$ contains precisely one node with label $\ell(v^{(i)})$.
2. If $v^{(i)}$ is a reflect node, then $v^{(i)}$ (i.e., the unique node with label $\ell(v^{(i)})$ in $G^{(i-1)}$) is an unmarked leaf in $G^{(i-1)}$ that is a 1-leaf if $i = 1$ and a 2-leaf if $i > 1$; if $v^{(i)}$ is a split node, then $v^{(i)}$ is unmarked in $G^{(i-1)}$.
3. No two nodes of $G^{(i)}$ have the same label and the set of labels of the nodes of $G^{(i)}$ is independent (as defined in Definition 2.6).
4. Let $\mathcal{C}^{(i)}$ denote the set of maximal connected components in the subgraph of $G^{(i)}$ induced by the unmarked nodes. Let $\mathcal{E}^{(i)}$ denote the set of edges of T with one endpoint in $\{v^{(1)}, \ldots, v^{(i)}\}$ and one endpoint in $\{v^{(i+1)}, \ldots, v^{(|V(T)|)}\}$. Then there is a bijection $\alpha_i : \mathcal{C}^{(i)} \to \mathcal{E}^{(i)}$ such that, for each component $C \in \mathcal{C}^{(i)}$,
   (a) the set of labels of the nodes of C that are 2-leaves of $G^{(i)}$ is precisely $\psi(\alpha_i(C))$, and
   (b) the set of labels of the nodes of C that are not 2-leaves of $G^{(i)}$ is precisely $\pi(\alpha_i(C))$.
5. Let $\mathcal{C}^{(i)}$ be defined as in Property 4. Then, for each component $C \in \mathcal{C}^{(i)}$ and each 2-node u of C, either u is a leaf of $G^{(i)}$ or u has degree Δ in C. Moreover, all neighbors of each 2-node in $G^{(i)}$ are 1-nodes and vice versa.

For the base case, consider $i = 1$. As $G^{(0)}$ contains precisely one node with label 1 and for the node $v^{(1)} = r$ in T we have $\ell(v^{(1)}) = \ell(r) = 1$, Property 1 is satisfied. Since the node with label 1 in T is a reflect node and the node with label 1 is an unmarked 1-leaf in $G^{(0)}$, also Property 2 is satisfied. Moreover, it is straightforward to verify that the tree $G^{(1)}$ obtained from $G^{(0)}$ by reflection at the node with label 1 contains only unmarked nodes and has Δ leaves labeled $12, 22, \ldots, \Delta 2$, as well as one internal node with label $*1$, which implies Properties 3 and 5. Similarly, it is straightforward to verify that the unique edge in T with head r is labeled with the set $\{12, 22, \ldots, \Delta 2\}$ under ψ and with the set $\{*1\}$ under π, which, together with the above, implies Property 4.

For the induction step, assume that the induction hypothesis holds for each $i \leq k$ (where $k \in \{1, \ldots, |V(T)| - 1\}$). We show in the following that it then also holds for $i = k + 1$.

Consider first the case that $v^{(k+1)}$ is a reflect node. By Property 4 of the induction hypothesis and Lemma 2.18, there exists an unmarked 2-leaf in $G^{(k)}$ with label $\ell(v^{(k+1)})$. Now, Property 1 for $i = k + 1$ follows by Property 3 of the induction hypothesis (i.e., for $i = k$). Property 2 for $i = k + 1$ also follows from this discussion.

For proving Property 3, let W denote the set of nodes in the maximal connected component containing $v^{(k+1)}$ in the subgraph of $G^{(k)}$ induced by the unmarked nodes. Due to the fact that $G^{(k+1)}$ is obtained from $G^{(k)}$ by reflection at $v^{(k+1)}$, we can describe the set of nodes in $G^{(k+1)}$ as follows. For each node in $V(G^{(k)}) \setminus W$ there is one node in $V(G^{(k+1)})$ with precisely the same label. For each node $w \in W \setminus \{v^{(k+1)}\}$, there are Δ nodes in $G^{(k+1)}$ such that $\ell(w)$ is a final substring of the substring of each of the Δ nodes but there are no two of the Δ nodes such that the label of the one is a final substring of the label of the other. For node $v^{(k+1)} \in V(G^k)$, there is precisely

one node in $G^{(k+1)}$ such that $\ell(v^{(k+1)})$ is a final substring of the label of that node. Moreover, the nodes in $V(G^{(k+1)})$ given in this description are all distinct (but cover all of $V(G^{(k+1)})$). From this description and the fact that Property 3 is satisfied for $i = k$, we obtain that there are no two nodes in $V(G^{(k+1)})$ such that the label of the one is a final substring of the label of the other. This implies Property 3 for $i = k + 1$.

For proving Property 4, observe that the reflection at $v^{(k+1)}$ that produces $G^{(k+1)}$ from $G^{(k)}$ induces a natural bijection $\beta : \mathcal{C}^{(k)} \to \mathcal{C}^{(k+1)}$ that simply maps each component $C \in \mathcal{C}^{(k)}$ to the component in $v^{(k+1)}$ containing C as a subgraph. Moreover, let $\gamma : \mathcal{E}^{(k)} \to \mathcal{E}^{(k+1)}$ denote the bijection that maps the edge of T with tail $v^{(k+1)}$ to the edge with head $v^{(k+1)}$ and any other edge in $\mathcal{E}^{(k)}$ to itself. We claim that $\alpha_{k+1} := \gamma \circ \alpha_k \circ \beta^{-1} : \mathcal{C}^{(k+1)} \to \mathcal{E}^{(k+1)}$ is a bijection as desired in Property 4 (where α_k is the bijection guaranteed by Property 4 of the induction hypothesis). For proving this claim, due to the definitions of β and γ (and the induction hypothesis), it suffices to show that, for the component $C \in \mathcal{C}^{(k)}$ containing $v^{(k+1)}$,

1. the set of labels of the nodes of $\beta(C)$ that are 2-leaves of $G^{(k+1)}$ is precisely $\psi(\gamma(\alpha_k(C)))$, and
2. the set of labels of the nodes of $\beta(C)$ that are not 2-leaves of $G^{(k+1)}$ is precisely $\pi(\gamma(\alpha_k(C)))$.

However, the former property follows directly from Property 5 of the induction hypothesis and the definitions of the reflection operation and ψ in Definitions 2.17 and 2.24, and the latter property follows directly from Property 5 of the induction hypothesis and the definitions of the reflection operation and π in Definitions 2.19 and 2.24.

Property 5 follows directly from the definition of the reflection operation in Definition 2.24 and Property 5 of the induction hypothesis.

Now consider the case that $v^{(k+1)}$ is a split node. By Property 4 of the induction hypothesis and Lemma 2.20, there exists an unmarked node in $G^{(k)}$ with label $\ell(v^{(k+1)})$. Now, Property 1 for $i = k + 1$ follows by Property 3 of the induction hypothesis (i.e., for $i = k$). Property 2 for $i = k + 1$ also follows from this discussion. Moreover, Property 3 follows directly from the definition of the split operation in Definition 2.24 and Property 3 of the induction hypothesis.

For proving Property 4, we consider two cases. Consider first the case that $v^{(k+1)}$ is an internal split node, and let C' be the component in $\mathcal{C}^{(k)}$ containing $v^{(k+1)}$. Let p denote the split index of $v^{(k+1)}$ in T, and let q be the index such that $v^{(q)}$ is the unique ancestor of $v^{(k+1)}$ in reflect layer $\mathcal{R}_p$. Observe that there is a node $u \in G^{(q)}$ with label $\ell(u) = *\ell(v^{(q)})$, by Property 2 of the induction hypothesis and the definition of the reflection operation. Let $\hat{C}$ denote the component in $\mathcal{C}^{(q)}$ containing u. Observe that, by Lemma 2.22, Property 4 of the induction hypothesis, and the definition of the reflection operation, there is a sequence $u = w^{(q)}, w^{(q+1)}, \ldots, w^{(k)=v^{(k+1)}}$ of nodes in $G^{(q)}, \ldots, G^{(k)}$, respectively, such that, for each $q \leq d \leq k - 1$, we have that $w^{(d+1)}$ is identical[11] to $w^{(d)}$ or a g-copy[12] of $w^{(d)}$, for some $1 \leq g \leq \Delta$. Moreover, by Lemma 2.22, Property 4 of the induction hypothesis, and the definitions of ψ, π, and the reflection operation, each node of C can be obtained from some node of $\hat{C}$ by a sequence that uses, for each d, precisely the same way to obtain the node in $G^{(d+1)}$ from the node in $G^{(d)}$ (including the choice of the respective g, which are specified by the symbols q_j from Lemma 2.22). It follows that C' is isomorphic to a subgraph of $\hat{C}$ where the isomorphism preserves the 0-th to p-th symbols of the label of each node (where the preservation comes from the fact that the reflection operation does not change already present symbols of a label when copying a node). Hence, by the way in which the reflection operation at $v^{(q)}$ affects the nodes (and in particular their p-th symbols) ending up in $G^{(q)}$ and the fact that the split operation at v^{k+1} turns the image of u under the aforementioned isomorphism into a marked

[11] Recall that we consider $G^{(d+1)}$ as a supergraph of $G^{(d)}$.

[12] See Definition 2.24 for the definition of a g-copy of a node.

node, we obtain that there are precisely Δ components $C'_1, \ldots, C'_\Delta \in \mathcal{C}^{(k+1)}$ that are subgraphs of C' and, for each $1 \leq j \leq \Delta$, the label of each node in C'_j has j as its p-th symbol. Note that the fact that none of these components is empty follows from the fact that $v^{(k+1)}$ has degree Δ in C', which in turn follows from the following argumentation:

From Observation 2.8, Lemma 2.21, and Property 4 of the induction hypothesis, we know that when a 1-node is marked previous to processing $v^{(k+1)}$, then the 1-node did not have any neighbors at that point. Hence, no unmarked 2-leaf of $G^{(k)}$ can contain a $*$ in its label, by the definition of the reflection operation and Property 5 of the induction hypothesis. Since split operations are only performed at nodes with a label that contains a $*$ (Property 7 of Definition 2.7), we obtain that v^{k+1} has degree Δ in C', by Property 5 of the induction hypothesis.

Now we are ready to define the desired bijection α_{k+1}. Let e denote the edge of T with tail $v^{(k+1)}$ and, for each $1 \leq j \leq \Delta$, let e_j denote the edge whose tail is the j-th child of $v^{(k+1)}$. For each component $C \in C^{(k)}$ that does not contain $v^{(k+1)}$ (which implies that we can consider C also as a component in $C^{(k+1)}$ as it does not change during the reflection operation at $v^{(k+1)}$), we set $\alpha_{k+1}(C) := \alpha_k(C)$. Moreover, for each $1 \leq j \leq \Delta$, we set $\alpha_{k+1}(C'_j) = e_j$. Now Property 4 for $i = k+1$ follows directly from the definitions of ψ and π, the fact that the label of each node in C'_j has j as its p-th symbol (as observed above), and Property 4 of the induction hypothesis.

Next, consider the case that $v^{(k+1)}$ is a leaf split node. Then, by Lemma 2.21, and Property 4 of the induction hypothesis, we know that the component in $\mathcal{C}^{(k)}$ containing $v^{(k+1)}$ contains no other node than $v^{(k+1)}$. Hence, setting $\alpha_{k+1}(C) := \alpha_k(C)$ for each component $C \in \mathcal{C}^{(k)}$ that does not contain $v^{(k+1)}$ yields Property 4 for $i = k+1$ (by Property 4 of the induction hypothesis).

Finally, we prove Property 5. The fact that all neighbors of each 2-node in $G^{(k+1)}$ are 1-nodes and vice versa follows directly from Property 5 of the induction hypothesis and the definition of the split operation. Now let u be some 2-node in some component $C' \in \mathcal{C}^{(k+1)}$, and let C be the component in $\mathcal{C}^{(k)}$ containing u. By Property 5 of the induction hypothesis, we know that u is a 2-leaf of $G^{(k)}$ or has degree Δ in C. If u is a 2-leaf of $G^{(k)}$, then it is also a 2-leaf of $G^{(k+1)}$; hence, assume that u has degree Δ in C. Again by Property 5 of the induction hypothesis, we know that all neighbors of u in $G^{(k)}$ are 1-nodes. Thus, by Observation 2.8, Lemma 2.21, and Property 4 of the induction hypothesis, we know that $v_{(k+1)}$ is not a neighbor of u in $G^{(k)}$. It follows that u has degree Δ in C', concluding the proof of Property 5.

This concludes the proof that G_T is well-defined for each *fixed* choice of $v^{(1)}, v^{(2)}, \ldots, v^{(|V(T)|)}$. Next, we show that G_T is independent of the choice.

Let $v^{(1)}, v^{(2)}, \ldots, v^{(|V(T)|)}$ be an arbitrary sequence as described in Definition 2.26, and let $1 \leq j \leq |V(T)|$ be an integer such that $v^{(j)}$ is a split node in this sequence. For simplicity, let w denote the node in $G^{(j-1)}$ with label $\ell(v^{(j)})$. Observe that, by the proof of Property 4 of the statement we just proved via induction (and in particular the inductive choice of the bijections α_i), the following holds: for any two descendants u_1, u_2 of $v^{(j)}$, the two (reflection or split) operations corresponding to u_1 and u_2 will happen in two parts of the respectively obtained graph that can be reached from w (which is a node that exists in any graph $G^{(p)}$ with $p \geq j-1$) via two different edges. In other words, due to the definition of the reflection and split operations (which only affect components of unmarked nodes, whereas w is marked in each graph $G^{(p)}$ with $p \geq j$), it is irrelevant in which order the two aforementioned operations are executed (as none of the operations has any influence on the effects of the other). As this is true for every sequence as described above and every choice of j, we obtain that any choice of $v^{(1)}, v^{(2)}, \ldots, v^{(|V(T)|)}$ will result in the same graph G_T. $\square$

We conclude Section 2.4 by collecting some useful properties of the sequence $G^{(0)}, \ldots, G^{(|V(T)|)}$. First, by Lemmas 2.18 and 2.20, the following observation is implied by the proof of Lemma 2.27.

Observation 2.28. *Let $v^{(1)}, \dots, v^{(|V(T)|)}$ and $G^{(0)}, \dots, G^{(|V(T)|)}$ be as defined in Definition 2.26. Let $2 \leq i \leq |V(T)|$, and let e be the edge with tail $v^{(i)}$. Let C denote the maximal connected component in the subgraph of $G^{(i-1)}$ induced by the unmarked nodes that contains $v^{(i)}$. Then,*

1. *the set of labels of the nodes of C that are 2-leaves of $G^{(i-1)}$ is precisely $\psi(e)$, and*
2. *the set of labels of the nodes of C that are not 2-leaves of $G^{(i-1)}$ is precisely $\pi(e)$.*

Moreover, there is a simple characterization of the property that two nodes are neighbors in one of the trees $G^{(j)}$ used in the definition of G_T, as given in the following lemma.

Lemma 2.29. *Let $v^{(1)}, \dots, v^{(|V(T)|)}$ and $G^{(0)}, \dots, G^{(|V(T)|)}$ be as defined in Definition 2.26. Let $0 \leq i \leq |V(T)|$ be some integer. Let $v \neq w$ be two unmarked nodes of $G^{(i)}$ (which in particular implies that $\mathrm{len}(\ell(v)) = \mathrm{len}(\ell(w))$). Then v and w are neighbors in $G^{(i)}$ if and only if, for each $1 \leq j \leq \mathrm{len}(v) - 1$ satisfying $\ell(v), \ell(w) \in \{1, \dots, \Delta\}$, we have $\ell(v)_j = \ell(w)_j$.*

Proof. We prove the lemma by induction on i. For the base case $i = 0$, the lemma statement follows directly from the definition of $G^{(0)}$. Now assume that the lemma statement holds for $i = k$ (where $0 \leq k \leq |V(T)| - 1$). We show in the following that it then also holds for $i = k + 1$.

If $v^{(k+1)}$ is a split node, then this follows directly from the definition of the split operation (see Definition 2.25). Hence, assume that $v^{(k+1)}$ is a reflect node, and consider two nodes $v \neq w$ of $G^{(k+1)}$. By the definition of the reflection operation (see Definition 2.24), there are two (uniquely defined) unmarked nodes $v' \neq w'$ of $G^{(k)}$ such that $\ell(v')_j = \ell(v)_j$ and $\ell(w')_j = \ell(w)_j$, for each $1 \leq j \leq \mathrm{len}(v) - 2$.

From the definition of the reflection operation, we obtain that v and w are neighbors in $G^{(k+1)}$ if and only if

1. v' and w' are neighbors in $G^{(k)}$ and
2. $\ell(v')_{\mathrm{len}(v)-1} = \ell(w')_{\mathrm{len}(v)-1}$ or $* \in \{\ell(v')_{\mathrm{len}(v)-1}, \ell(w')_{\mathrm{len}(v)-1}\}$

Now the lemma statement for $i = k + 1$ follows from the induction hypothesis. □

2.5 A Relation between the Input Trees Induced by T and $F(T)$

In this section, we will prove a relation between the two input trees G_T and $G_{F(T)}$ that are induced by T and $F(T)$, respectively. This relation is crucial for the proof of our overall lower bound construction. Before stating and proving this relation, we introduce some necessary definition.

Definition 2.30 (distance-D-correct)**.** Let T be a construction tree and D a positive integer. Let $v^{(1)}, v^{(2)}, \dots, v^{(|V(T)|)}$ and $G^{(0)}, G^{(1)}, \dots, G^{(|V(T)|)}$ be as in the definition of G_T in Definition 2.26. We say that T is *distance-D-correct* if, for each $1 \leq i \leq |V(T)|$ such that $v^{(i)}$ is a split node, each 2-leaf in $G^{(i-1)}$ that is in the same component of unmarked nodes as $v^{(i)}$ has distance at least D from $v^{(i)}$ in $G^{(i-1)}$.

We remark that analogously to the argumentation in the proof of Lemma 2.27 why G_T is independent of the choice of the nodes $v^{(1)}, v^{(2)}, \dots, v^{(|V(T)|)}$ we also obtain that whether a construction tree is distance-D-correct is independent of the choice of the $v^{(1)}, v^{(2)}, \dots, v^{(|V(T)|)}$.

Now we are ready to prove the main result of Section 2.5.

Lemma 2.31. *Let D be a positive integer and T a distance-D-correct Δ-ary construction tree. Then $F(T)$ is a distance-$(2D)$-correct Δ-ary construction tree.*

Proof. By Lemma 2.12, $F(T)$ is a Δ-ary construction tree. In the following, we show that it is also distance-$(2D)$-correct.

Let $v^{(1)}, v^{(2)}, \dots, v^{(|V(F(T))|)}$ and $G^{(0)}, G^{(1)}, \dots, G^{(|V(F(T))|)}$ be as in Definition 2.26, just for $F(T)$ instead of T. Recall the definition of the j-th child of a node of a construction tree from Definition 2.7. Let P be the root-leaf path in $F(T)$ that, starting from the root r, always takes the first child when arriving at a split node. We first show "distance-$(2D)$-correctness for any node on P".

Let $v^{(i)}$ be an arbitrary split node on P. By the argumentation in the proof of Lemma 2.27 guaranteeing independence of the choice of the nodes $v^{(1)}, v^{(2)}, \dots, v^{(|V(F(T))|)}$, the maximal connected component of unmarked nodes in $G^{(i-1)}$ containing $v^{(i)}$ is independent of the aforementioned choice and in particular only depends on the nodes (and their labels) in $F(T)$ that lie on the path from r to $v^{(i)}$. Hence, w.l.o.g. assume that i is the number of nodes on the path from r to $v^{(i)}$ and that the nodes on this path are $r = v^{(1)}, v^{(2)}, \dots, v^{(i)}$ (ordered by increasing distance from r).

Recall that V_R denotes the set of reflect nodes of T. Let $Q = (w_1, \dots, w_{2|V_R|})$ be the (chronologically ordered) sequence (with repeating elements) of reflect nodes of T visited by the depth-first search traversal used in the definition of $F(T)$ and recall the definition of a late node (see Definition 2.10). Observe that $v^{(i)} \in L_i^F$. Since $v^{(i)}$ is a split node, we know that w_i is a late node. Let v' denote the reflect node of T such that w_i is the second occurrence of v' in Q, let r' denote the root of T, and let P' denote the path from r' to v'. Moreover, let $r' = v'_1, v'_2, \dots, v'_{|V_R|}$ denote the reflect nodes of T in the order they are visited *for the first time* by the aforementioned depth-first traversal.

Recall the notation for the layers of $F(T)$ from Definition 2.10. Let $\mathcal{K} \subseteq \{1, \dots, |V_R|\}$ be the set of indices k such that $\mathcal{S}_k^F$ has layer index $< i$, and $\mathcal{K}' \subseteq \{1, \dots, |V_R|\}$ the set of indices k' such that $\mathcal{R}_{k'}^F$ has layer index $< i$. By the fact that $F(T)$ is well-nested, we know that $\mathcal{K} \subseteq \mathcal{K}'$. Moreover, by the definition of $F(T)$, a reflect node v'_j lies on P' if and only if $j \in \mathcal{K}' \setminus \mathcal{K}$.

Now consider $G^{(i-1)}$. Let C denote the maximal connected component in the subgraph of $G^{(i-1)}$ induced by the unmarked nodes that contains $v^{(i)}$. Let u be an arbitrary 2-leaf of $G^{(i-1)}$ that is contained in C. Note that if $v^{(i)}$ is a leaf node, then Lemma 2.21 and Observation 2.28 guarantee that no such node u exists and there is nothing to prove. Hence, assume in the following that $v^{(i)}$ is an internal split node.

Let e be the edge of $F(T)$ with tail $v^{(i)}$ (and note that $i \neq 1$ by Observation 2.8). By Observation 2.28, we have $\ell(u) \in \psi(e)$, which, by Lemma 2.20 and Observation 2.28, implies $\ell(u) \neq \ell(v^{(i)})$ and $u \neq v^{(i)}$. By the definitions of ψ and $v^{(i)}$, this implies that $\ell(u)_k = 1$ for each $k \in \mathcal{K}$. Moreover, by the definitions of ψ and $F(T)$, we know that $\mathrm{len}(\ell(u)) = \mathrm{len}(\ell(v^{(i)})) = |\mathcal{K}'| + 1$.

Observe that the definition of ψ in Definition 2.17, together with Lemma 2.21, guarantees that whenever, for some edge e' of a construction tree, a string z is contained in $\psi(e')$, then there is some reflect node w in the subtree of the construction tree hanging from e' such that z is a final substring of $\ell(w)$. Applying this observation to construction tree $F(T)$ (with $e' := e$ and $z := \ell(u)$), we obtain that there is some reflect node w in the subtree of $F(T)$ hanging from $v^{(i)}$ (including $v^{(i)}$) such that $\ell(u)$ is a final substring of $\ell(w)$. Let $1 \leq d \leq 2|V_R|$ be the index such that $w \in L_d^F$. Since $v^{(i)}$ is not a reflect node, we obtain that $d > i$. Recall the definition of Q, and let w' be the node of T such that w_d is the first occurrence of w' in Q. Let x denote the split node of T that is an ancestor of both v' and w' and has the greatest distance from the root r' amongst all nodes with this property. Observe that, by the definition of $F(T)$, node w' is neither an ancestor nor a descendant of v', which implies that $w' \neq x \neq v'$.

Let e_x denote the edge of T with tail x. From the definition of ψ (see Definition 2.17), it follows that there are strings $y_v, y_w \in \psi(e_x)$ such that y_v is a final substring of $\ell(v')$ and y_w a

final substring of $\ell(w')$. Let j be the split index of x and j' the layer index of x in T. Let $u^{(1)}, \ldots, u^{(|V(T)|)}$ be an ordering of the nodes in $V(T)$ as defined in Definition 2.26 but with the additional property that all nodes of P' appear in the ordering before the first node not contained in P'. Let $G_T^{(0)}, G_T^{(1)}, \ldots, G_T^{(|V(T)|)}$ be the graphs obtained from the aforementioned node sequence in the way described in Definition 2.26.

Consider $G_T^{(j'-1)}$. Let $\hat{v}$ be the 2-leaf of $G_T^{(j'-1)}$ with label $\ell(\hat{v}) = y_v$ and $\hat{w}$ the 2-leaf of $G_T^{(j'-1)}$ with label $\ell(\hat{w}) = y_w$. By Observation 2.28, such 2-leaves $\hat{v}$ and $\hat{w}$ indeed exist in $G_T^{(j'-1)}$ and are contained in the same maximal connected component of unmarked nodes of $G_T^{(j'-1)}$ as x. Moreover, by the fact that T is distance-D-correct, we obtain that the distance between x and $\hat{v}$ in $G_T^{(j'-1)}$ is at least D, and that the distance between x and $\hat{w}$ in $G_T^{(j'-1)}$ is at least D.

Observe that the fact that j is the split index of x, together with Property 8 of Definition 2.7 and the fact that $\ell(\hat{v})$ and $\ell(\hat{w})$ are final substrings of $\ell(v')$ and $\ell(w')$ (respectively), implies that $\ell(\hat{v})_j \neq \ell(\hat{w})_j$. By the definitions of ψ and π and Observation 2.28 (applied to $G^{(j')}$), it follows that the neighbor of x on the path between x and $\hat{v}$ is not the same node as the neighbor of x on the path between x and $\hat{w}$. Hence, the distance between $\hat{v}$ and $\hat{w}$ in $G^{(j'-1)}$ is at least $2D$. Let $\hat{v} = x_0, \ldots, x_\alpha = \hat{w}$ denote the nodes on the path from $\hat{v}$ to $\hat{w}$ in $G^{(j'-1)}$ (ordered by increasing distance from $\hat{v}$). From the discussion above, we know that $\alpha \geq 2D$.

Recall the depth-first search traversal used in the definition of $F(T)$ (see Definition 2.10), the definition of the nodes $v'_1, \ldots, v'_{|V_R|}$, and the definition of P. Let $\mathcal{L}' \subseteq \{1, \ldots, |V_R|\}$ denote the set of all indices λ such that the aforementioned depth-first search traversal visits (the reflect node) v'_λ before (the split node) x. Let $\mathcal{L} \subseteq \mathcal{L}'$ denote the set of all indices λ such that v'_λ does not lie on the path from r' to x in T. Moreover, if $v'_{|\mathcal{L}'|}$ lies on the path from r' to x in T, then let v^* denote the reflect node on P (in $F(T)$) with reflect index $|\mathcal{L}'|$; if $v'_{|\mathcal{L}'|}$ does not lie on the path from r' to x in T, then let v^* denote the split node on P (in $F(T)$) with split index $|\mathcal{L}'|$. If v^* is a reflect node, then let e^* be the unique edge with head v^*; if v^* is a split node, then let e^* be the edge between v^* and the first child of v^*. (Note that e^* exists in either case, due to the fact that there are reflect nodes that are descendants of x in T, e.g., v' and w'.)

Consider some arbitrary string $y \in \psi(e_x) \cup \pi(e_x)$. Let $\gamma(y)$ be the string obtained by taking the $\mathcal{L}$-padded version of y and then replacing the symbol $\square$ with the symbol 1 (wherever the symbol $\square$ occurs in the $\mathcal{L}$-padded version of y). Then, from Observation 2.11 and the definitions of ψ, π, and $F(T)$, we obtain that $\gamma(y) \in \psi(e^*) \cup \pi(e^*)$. Consider x_β, for some arbitrary index $0 \leq \beta \leq \alpha$. By Observation 2.28, we have $\ell(x_\beta) \in \psi(e_x) \cup \pi(e_x)$. Hence, from the above discussion, we obtain that $\gamma(\ell(x_\beta)) \in \psi(e^*) \cup \pi(e^*)$.

Observe that, by the definitions of $F(T)$, $\mathcal{L}$, and $\mathcal{L}'$, node v^* lies in layer $L^F_{|\mathcal{L}'|+|\mathcal{L}|}$ of $F(T)$. As, by definition, v^* lies on P, this implies that $v^* = v^{(|\mathcal{L}'|+|\mathcal{L}|)}$. Consider $G^{(|\mathcal{L}'|+|\mathcal{L}|)}$. From the above discussion, the definition of the strings $\gamma(\ell(x_\beta))$, and Observation 2.28, it follows that there is a maximal connected component C' of unmarked nodes of $G^{(|\mathcal{L}'|+|\mathcal{L}|)}$ such that, for each $0 \leq \beta \leq \alpha$, there is a node of C' with label $\gamma(\ell(x_\beta))$, and these nodes are pairwise distinct. For each $0 \leq \beta \leq \alpha$, let $\gamma(x_\beta)$ denote the node with label $\gamma(\ell(x_\beta))$. By Lemma 2.29 and the definition of the strings $\gamma(\ell(x_\beta))$, we know that, for each $0 \leq \beta \leq \alpha - 1$, nodes $\gamma(x_\beta)$ and $\gamma(x_{\beta+1})$ are neighbors in C'. It follows that $\gamma(x_0)$ and $\gamma(x_\alpha)$ have distance α in C', and therefore also in $G^{(|\mathcal{L}'|+|\mathcal{L}|)}$.

Recall that $\ell(\hat{v})$ is a final substring of $\ell(v')$ and $\ell(\hat{w})$ a final substring of $\ell(w')$. By Observation 2.11, the definitions of v', w', $\gamma(x_0)$, and $\gamma(x_\alpha)$, and the facts that $x_0 = \hat{v}$, $x_\alpha = \hat{w}$, and $\mathcal{L}' \setminus \mathcal{L} \subseteq \mathcal{K}' \setminus \mathcal{K}$, it follows that, for each $\lambda \in (\mathcal{L}' \setminus \mathcal{L}) \cup \{0\}$, we have $\ell(\gamma(x_0))_\lambda = \ell(v^{(i)})_\lambda$ and $\ell(\gamma(x_\alpha))_\lambda = \ell(w)_\lambda$. Since, $\mathcal{K} \subseteq \mathcal{L}$ and, for each $k \in \mathcal{K}$, we have $\ell(v^{(i)})_k = \ell(w)_k = 1$, we also obtain that, for each $\lambda \in \mathcal{L}$, we have $\ell(\gamma(x_0))_\lambda = \ell(v^{(i)})_\lambda$ and $\ell(\gamma(x_\alpha))_\lambda = \ell(w)_\lambda$. As $\text{len}(\ell(\gamma(x_0))) = \text{len}(\ell(\gamma(x_\alpha))) = |\mathcal{L}'| + 1$,

it follows that $\ell(\gamma(x_0))$ is a final substring of $\ell(v^{(i)})$ and $\ell(\gamma(x_\alpha))$ a final substring of $\ell(w)$. Since $\mathcal{L}' \subseteq \mathcal{K}'$ and, for each $k \in \mathcal{K}'$, we have $\ell(u)_k = \ell(w)_k$, we also obtain that $\ell(\gamma(x_\alpha))$ is a final substring of $\ell(u)$.

Next, observe that, since v' is a descendant of x in T and $v' \neq x$, we have that $v^{(i)}$ is a descendant of $v^* = v^{(|\mathcal{L}'|+|\mathcal{L}|)}$ in $F(T)$ and $v^{(i)} \neq v^{(|\mathcal{L}'|+|\mathcal{L}|)}$. In particular, we obtain that $i > |\mathcal{L}'|+|\mathcal{L}|$. Consider $G^{(i)}$. The above discussion implies that $G^{(i)}$ is obtained from $G^{(|\mathcal{L}'|+|\mathcal{L}|)}$ via the reflection and split operations indicated by the nodes in $F(T)$ on the subpath of P from $v^{(|\mathcal{L}'|+|\mathcal{L}|+1)}$ to $v^{(i)}$.

Let $|\mathcal{L}'| + |\mathcal{L}| \leq \eta \leq i-1$ be some integer, and let $\hat{x}$ and $\overline{x}$ be two nodes of $G^{(\eta+1)}$. Let $\hat{x}'$ and $\overline{x}'$ be the two (uniquely[13] defined) nodes of $G^{(\eta+1)}$ satisfying that $\ell(\hat{x}')$ is a final substring of $\ell(\hat{x})$ and $\ell(\overline{x}')$ a final substring of $\ell(\overline{x})$. Then, the definitions of the reflection and split operations ensure that the distance between $\hat{x}$ and $\overline{x}$ in $G^{(\eta+1)}$ is at least as large as the distance between $\hat{x}'$ and $\overline{x}'$ in $G^{(\eta)}$. Applying this argumentation inductively (and using that $\ell(\gamma(x_0))$ is a final substring of $\ell(v^{(i)})$ and $\ell(\gamma(x_\alpha))$ a final substring of $\ell(u)$), we obtain that the distance between $v^{(i)}$ and u in $G^{(i-1)}$ is at least the distance between $\gamma(x_0)$ and $\gamma(x_\alpha)$ in $G^{(|\mathcal{L}'|+|\mathcal{L}|)}$. As the distance between $\gamma(x_0)$ and $\gamma(x_\alpha)$ in $G^{(|\mathcal{L}'|+|\mathcal{L}|)}$ is at least α (as shown above), and $\alpha > 2D$, it follows that the distance between $v^{(i)}$ and u in $G^{(i-1)}$ is at least $2D$.

Hence, we have shown "distance-$(2D)$-correctness for any node on P" in the sense that we proved that for any arbitrary split node $v^{(i)}$ on P, each 2-leaf in $G^{(i-1)}$ that is in the same component of unmarked nodes as $v^{(i)}$ has distance at least $2D$ from $v^{(i)}$ in $G^{(i-1)}$. Observe that, for each connected component of unmarked nodes, the reflection and split operations (and analogously the labelings produced by ψ and π) are symmetric regarding the symbols from $\{1, \ldots, \Delta\}$ (in each position of a string except the one indicated by index 0). Moreover, as observed in the proof of Lemma 2.27, different connected components of unmarked nodes are independent of each other in the sense that each reflect or split operation affects only the connected component in which it is applied. Hence, by symmetry, the aforementioned distance-$(2D)$-correctness result for the nodes on P extends also to the nodes that do not lie on P (and to arbitrary sequences $v^{(1)}, v^{(2)}, \ldots, v^{(|V(F(T))|)}$ as defined in Definition 2.26). In other words, for each $v^{(1)}, v^{(2)}, \ldots, v^{(|V(F(T))|)}$ and $G^{(0)}, G^{(1)}, \ldots, G^{(|V(F(T))|)}$ (as defined in Definition 2.26) and each $1 \leq i \leq |V(F(T))|$ such that $v^{(i)}$ is a split node of $F(T)$, each 2-leaf in $G^{(i-1)}$ that is in the same component of unmarked nodes as $v^{(i)}$ has distance at least $2D$ from $v^{(i)}$ in $G^{(i-1)}$. Thus, $F(T)$ is distance-$(2D)$-correct. □

Recall the definition of the sequence $T_2, T_3, \ldots$ of construction trees in Definition 2.13. From Lemma 2.31 (and the fact that, as is straightforward to verify, T_2 is distance-2-correct), we obtain the following corollary.

Corollary 2.32. *For each positive integer i, tree T_i is a distance-2^{i-1}-correct Δ-ary construction tree.*

3 The Lower Bound

While Section 2 took care of collecting the main conceptual and technical ingredients for our overall lower bound construction, Section 3 is devoted to conducting the actual lower bound proof. We start by defining the notion of a *canonical sequence*. Similarly to how the actual input tree that we will present to a given online-LOCAL algorithm is obtained by minor adaptations from the tree G_T defined in Section 2.4, the input query sequence that we will use for that input tree will be obtained by minor adaptations from the canonical sequence.

[13] That $\hat{x}'$ and $\overline{x}'$ are uniquely defined follows from the definitions of the reflection and split operations and the fact that the set of labels of the nodes of $G^{(\eta)}$ is independent, as shown in the proof of Lemma 2.27.

Definition 3.1 (canonical sequence)**.** Let T be a Δ-ary construction tree and G_T the input tree induced by T. We call a node v of G_T a *mirror node* if during the process that derives G_T from T (see Definition 2.26), a reflection operation is performed at v.

Consider the uniquely defined sequence $U = (u_1, \ldots, u_{|V_R|})$ of reflect nodes of T with the following properties.

1. Each reflect node of T appears exactly once in the sequence.
2. For any two nodes u_i and u_j in U satisfying $i \neq j$ it holds that
   (a) if u_i has smaller layer index in T than u_j, then $i < j$, and
   (b) if u_i and u_j are in the same reflect layer of T and $\ell(u_i)$ is lexicographically smaller[14] than $\ell(u_j)$, then $i < j$.

Let $W = (w_1, \ldots, w_{|V_R|})$ be the sequence of mirror nodes of G_T satisfying that, for each $1 \leq i \leq |V_R|$, label $\ell(u_i)$ is a final substring of $\ell(w_i)$. We call W the *canoncial sequence for* G_T.

Note that the definition of G_T in Definition 2.26 (in particular the definitions of the reflection and split operations), together with Property 5 of Definition 2.7, ensures that W is uniquely defined (and exists).

Next, we prove a lemma that is crucial for defining the aforementioned "adaptations" that we will use to derive the actual input tree for our lower bound construction from G_T. This lemma ascertains that certain subgraphs of G_T are isomorphic.

Lemma 3.2. *Let T, G_T, U, and W be defined as in Definition 3.1. Consider some arbitrary node $w_i \in W$ with $i \neq |V_R|$ and let D be some positive integer such that T is distance-D-correct. Let W_i be the set of all nodes w of G_T such that*

1. *on the path from w_i to w (including w_i and w) in G_T, there is no node that is contained in $\{w_1, \ldots, w_{i-1}\}$, and*
2. *for some positive integer k, there exists a sequence $(w_i, w_{j_1}, w_{j_2}, \ldots, w_{j_k}, w)$ of (not necessarily distinct) nodes such that*
   (a) for each $1 \leq p \leq k$, we have $1 \leq j_p \leq i$,
   (b) the distance between w_i and w_{j_1} in G_T is at most $2D - 2$,
   (c) for each $1 \leq p \leq k-1$, the distance between w_{j_p} and $w_{j_{p+1}}$ in G_T is at most $2D - 2$, and
   (d) the distance between w_{j_k} and w in G_T is at most $D - 1$.

Let $G[W_i]$ be the subgraph of G_T induced by W_i (which, by definition, is connected). Moreover, for any edge e incident to w_i, let $G_e[W_i]$ be the subgraph of $G[W_i]$ induced by the set of nodes consisting of w_i and all nodes of W_i reachable from w_i via a path whose first edge is e. Then, for any two edges e, e' incident to w_i, the graphs $G_e[W_i]$ and $G_{e'}[W_i]$ are isomorphic and there exists an isomorphism between these two graphs that maps w_i to itself and preserves port numbers except possibly at w_i.

Proof. Let $v^{(1)}, v^{(2)}, \ldots, v^{(|V(T)|)}$ and $G^{(0)}, G^{(1)}, \ldots, G^{(|V(T)|)}$ be as defined in Definition 2.26 with the additional property[15] that, for any two reflect nodes $v^{(q)}, v^{(q')}$ with $q < q'$, node $v^{(q')}$ appears

[14] That we order nodes in the same layer of T lexicographically in U is an arbitrary choice; any fixed ordering of the nodes of a layer of T works for our purposes.

[15] The existence of such node and graph sequences with this additional property directly follows from the definition of U.

before $v^{(q)}$ in U. Let i' be the index such that $v^{(i')} = u_i$. Consider $G^{(i')}$. Let $W'_{i'}$ denote the set of nodes of $G^{(i')}$ that lie in the same maximal connected component of unmarked nodes as $v^{(i')}$. Due to the fact that $G_T = G^{(|V(T)|)}$ is a supergraph[16] of $G^{(i')}$, we can consider $W'_{i'}$ to be a subset of the nodes of G_T.

Let $G[W'_{i'}]$ be the subgraph of G_T induced by $W'_{i'}$. Moreover, for any edge e incident to $v^{(i')}$, let $G_e[W'_{i'}]$ be the subgraph of $G[W'_{i'}]$ induced by the set of nodes consisting of $v^{(i')}$ and all nodes of $G[W'_{i'}]$ reachable from $v^{(i')}$ via a path whose first edge is e. Now, it directly follows from the definition of the reflection operation and the fact that $G^{(i')}$ is the graph obtained from $G^{(i'-1)}$ by reflection at $v^{(i')}$ that, for any two edges e, e' incident to $v^{(i')}$, the graphs $G_e[W'_{i'}]$ and $G_{e'}[W'_{i'}]$ are isomorphic and there exists an isomorphism between these two graphs that maps $v^{(i')}$ to itself and preserves port numbers except possibly at $v^{(i')}$.

Observe that, by the definition of G_T (in particular by the label manipulation happening during a reflection operation), the fact that $v^{(i')} = u_i$ in T implies that $w_i = v^{(i')}$ in G_T. Moreover, observe that the construction of G_T, together with the definition of the canonical sequence W, ensures that, for any node $w' \in V(G_T) \setminus V(G^{(i')})$, the path from w_i to w' contains a node from $\{w_1, \dots, w_{i-1}\}$ or a node that is marked in $G^{(i')}$. Observe that, due to the fact that T is distance-D-correct, the distance in G_T between any node in $\{w_1, \dots, w_i\}$ and any node that is marked in $G^{(i')}$ is at least D. Hence, by the definition of W_i, we obtain $W_i \subseteq W'_{i'}$.

Observe that, due to Property 5 of Definition 2.7 and the definitions of W and G_T, a node from $W'_{i'}$ is contained in $\{w_1, \dots, w_{i-1}\}$ if and only if it is also a 2-leaf in $G^{(i')}$. Now, analogously to above, the fact that $G^{(i')}$ is obtained from $G^{(i'-1)}$ by reflection at $v^{(i')}$ implies that, for any two edges e, e' incident to $w_i = v^{(i')}$, the graphs $G_e[W_i]$ and $G_{e'}[W_i]$ are isomorphic and there exists an isomorphism between these two graphs that maps w_i to itself and preserves port numbers except possibly at w_i. □

As mentioned before, the lower bound instance that we present to a given algorithm depends on the respective algorithm. In Lemma 3.4, informally speaking, we show that if an algorithm would solve sinkless orientation w.h.p. with too small complexity, then there is an input instance on which the algorithm fails with a relatively large probability. This lemma constitutes the main ingredient for our lower bound proof for sinkless orientation presented in Theorem 1.2. Before proving Lemma 3.4, we introduce the key notion of the *smallest frequent edge* incident to some node v, which is a certain edge that the considered algorithm orients away from v with a large probability, relatively speaking.

Definition 3.3 (smallest frequent edge)**.** Let $\mathcal{A}$ be a sinkless orientation algorithm in the randomized online-LOCAL model that, when presented a node, always (i.e., for any random bits it may use) chooses at least one incident edge to be outgoing.[17] Consider an arbitrary sequence of nodes $(u_1, u_2, \dots)$ that is presented to $\mathcal{A}$ and let $i \geq 2$ be some integer such that u_i is contained in this sequence. Let $e_1, \dots, e_{\deg(u_i)}$ be the edges incident to u_i corresponding to port numbers $1, \dots, \deg(u_i)$, respectively. Consider the point of the execution of $\mathcal{A}$ when $\mathcal{A}$ has finished deciding on the outputs of the edges incident to $u_1, \dots, u_{i-1}$ and is presented with u_i. Consider any fixed set of possible behaviors $\mathcal{A}$ might have displayed (i.e., any fixed set of overall outputs it might have produced) and let $\mathcal{E}$ be the probabilistic event that $\mathcal{A}$ showed one of these behaviors in the execution so far. Assume that $\mathcal{E}$ occurs with nonzero probability. Let e_q be the edge with minimal index q such that $\mathcal{A}$ chooses e_q to be oriented away from u_i with probability at least $1/\Delta$, conditioned on $\mathcal{E}$. Then, we call e_q the *smallest frequent edge incident to u_i conditioned on $\mathcal{E}$*. For the special case of

[16] Recall that we consider the graph obtained by a reflection or split operation as a supergraph of the graph before the operation, see Definitions 2.24 and 2.25.

[17] Note that since randomized online-LOCAL algorithms may fail with a small probability, such an algorithm may in principle decide to choose all incident edges to be incoming with small probability.

$i = 1$, we call the edge e_q with minimal index q such that $\mathcal{A}$ chooses e_q to be oriented away from u_i with probability at least $1/\Delta$ (without any conditioning) the *smallest frequent edge incident to* u_i.

Lemma 3.4. *Let D and $\Delta \geq 3$ be positive integers. Let T be a distance-D-correct Δ-ary construction tree and let V_R denote the set of reflect nodes of T. Let $\mathcal{A}$ be an algorithm for sinkless orientation in the randomized online-LOCAL model with complexity $D-1$. Then for each $n \geq \Delta \cdot |V(G_T)|$ there exists an n-node tree $\mathcal{T}$ (together with a suitable query sequence) which $\mathcal{A}$ solves incorrectly with probability at least $1/\Delta^{|V_R|}$.*

Proof. We can assume w.l.o.g. that whenever a node is being presented to it, Algorithm $\mathcal{A}$ (which has to output an orientation for each incident edge) will orient at least one edge incident away from the node.[18] Let n be some integer satisfying $n \geq \Delta \cdot |V(G_T)|$. We first show how to construct a tree $\mathcal{T}$ as mentioned in the lemma and then show why $\mathcal{A}$ solves $\mathcal{T}$ incorrectly with probability at least $1/\Delta^{|V_R|}$. To prove the latter part, we will show how to construct (together with $\mathcal{T}$) a sequence $\mathcal{W}$ of nodes of $\mathcal{T}$ such that when $\mathcal{A}$ is presented with the nodes of $\mathcal{W}$ (in the order given by $\mathcal{W}$) on $\mathcal{T}$, the decisions made by $\mathcal{A}$ will lead to an incorrect output with probability at least $1/\Delta^{|V_R|}$.

On a high level, $\mathcal{T}$ is obtained from G_T by (a minor initial step for technical reasons and) a number of operations that adapt the structure of G_T to the design of Algorithm $\mathcal{A}$. Essentially, we aim at tweaking G_T so that on the obtained tree, for each decision that $\mathcal{A}$ takes when presented with $\mathcal{W}$, the choice of $\mathcal{A}$ is "reasonably bad" with large probability (relatively speaking). Sequence $\mathcal{W}$ is obtained by tweaking the canoncial sequence for G_T in a manner aligning with the tweaking of G_T.

Let $W = (w_1, \ldots, w_{|V_R|})$ be the canonical sequence for G_T. We construct $\mathcal{T}$ from G_T via a sequence of graphs $G_T = G_{-1}, G_0, \ldots, G_{|V_R|-1} = \mathcal{T}$. At the same time, we construct $\mathcal{W}$ from W via a sequence of nodes sequences $W = W^{(0)}, \ldots, W^{(|V_R|-1)} = \mathcal{W}$, where, for each $1 \leq g \leq |V_R| - 1$, we denote the elements of $W^{(g)}$ by $W_1^{(g)}, \ldots, W_{|V_R|}^{(g)}$ (given in the order in which they appear in $W^{(g)}$). G_0 is simply the graph obtained from G_T by attaching to each node v of G_T precisely $\Delta - \deg(v)$ (additional) leaf nodes and then attaching a path of suitable length to one of the new leaf nodes to increase the number of nodes to precisely n (which is possible due to the requirement that $n \geq \Delta \cdot |V(G_T)|$). Hence, every node of G_0 that is not one of the added nodes has degree Δ in G_0. The port numbers for the new edges are assigned in an arbitrary correct manner that does not change already fixed port numbers.

For the construction of $G_1, \ldots, G_{|V_R|-1}$, we emphasize that while the nodes of G_T have been assigned labels (and nodes in the graphs $G_0, \ldots, G_{|V_R|-1}$ could be considered to inherit node labels), these labels are only for the purpose of being able to address the nodes (in particular for defining the canonical sequence for G_T); the labels are not available to Algorithm $\mathcal{A}$. When asked to provide some output at a node, as following from the description of the randomized online-LOCAL model, Algorithm $\mathcal{A}$ (besides remembering its previous decisions) has access only to the structure and port numbers of the parts of the input graph that are at distance at most the complexity of $\mathcal{A}$ from some node that has been already given to $\mathcal{A}$ (including the node for which $\mathcal{A}$ has to make the current decision).

Let $1 \leq i \leq |V_R| - 1$. Assume that $G_{-1}, G_0, \ldots, G_{i-1}$ and $W^{(0)}, \ldots, W^{(i-1)}$ have already been defined. We define G_i and $W^{(i)}$ as follows.

[18] If $\mathcal{A}$ does not satisfy this property, simply replace $\mathcal{A}$ by an algorithm $\mathcal{A}'$ that behaves like $\mathcal{A}$ except that when $\mathcal{A}$ would orient all edges towards a presented node (for the first time), $\mathcal{A}'$ chooses one incident edge to be outgoing (and after that behaves in an arbitrary manner that guarantees the aforementioned property). It follows that, on any instance, the failure probability of $\mathcal{A}'$ is at least as small as the failure probability of $\mathcal{A}$, which implies that it suffices to prove the lemma statement for $\mathcal{A}'$.

Let $e_1, \dots, e_\Delta$ be the edges incident to $w_i^{(i-1)}$ corresponding to port numbers $1, \dots, \Delta$, respectively. Consider the execution of $\mathcal{A}$ on $G^{(i-1)}$ with sequence $W^{(i-1)}$. Let $\mathcal{E}_1$ be the probabilistic event that $\mathcal{A}$, when being presented with $w_1^{(i-1)}$, chooses the smallest frequent edge incident to $w_1^{(i-1)}$ to be outgoing. For each $2 \leq d \leq i-1$ let $\mathcal{E}_d$ be the probabilistic event[19] that

1. when being presented with $w_1^{(i-1)}$, Algorithm $\mathcal{A}$ chooses the smallest frequent edge incident to $w_1^{(i-1)}$ to be outgoing, and
2. for each $2 \leq d' \leq d$, Algorithm $\mathcal{A}$, when presented with $w_{d'}^{(i-1)}$, chooses the smallest frequent edge incident to $w_{d'}^{(i-1)}$ conditioned on $\mathcal{E}_{d'-1}$ to be outgoing.

Let e_q be the smallest frequent edge incident to $w_i^{(i-1)}$ conditioned on $\mathcal{E}_{i-1}$. Let G' denote the (connected) subgraph of G consisting of all nodes that

1. have not been presented to $\mathcal{A}$ so far, and
2. can be reached from $w_i^{(i-1)}$ via a path that uses only nodes that have not been presented to $\mathcal{A}$ so far (excluding $w_i^{(i-1)}$ itself) and uses only edges that $\mathcal{A}$ has already seen[20] (after $w_i^{(i-1)}$ has been presented to $\mathcal{A}$).

For each $1 \leq p \leq \Delta$, let $G'[p]$ denote the (connected) subgraph of G' induced by those nodes that can be reached from $w_i^{(i-1)}$ via the edge incident to $w_i^{(i-1)}$ that has been assigned port number p by $w_i^{(i-1)}$ (excluding $w_i^{(i-1)}$ itself). Let ξ be an isomorphism from G' to itself such that

1. every node in $\{w_i^{(i-1)}\} \cup \bigcup_{1 \leq p \leq \Delta, 1 \neq p \neq q} V(G'[p])$ is mapped to itself by ξ,
2. every node in $V(G'[1])$ is mapped to some node in $V(G'[q])$ by ξ and vice versa,
3. $\xi(\xi(u)) = u$ for each node $u \in V(G'[1]) \cup V(G'[q])$, and
4. ξ preserves port numbers except possibly regarding the two port numbers assigned by $w_i^{(i-1)}$ to e_1 and e_q.

Now, G_i is defined as the graph obtained from G_{i-1} by the following operation. For each edge e of G_{i-1} between some node $u \in V(G'[1]) \cup V(G'[q])$ and some node $u' \in V(G) \setminus V(G')$, replace $e = \{u, u'\}$ by a new edge $e' = \{\xi(u), u'\}$ (where the port number assigned to e' by u' is identical to the port number assigned to e by u' and the port number assigned to e' by $\xi(u)$ is identical to the port number assigned to e by u). Similarly, $W^{(i)}$ is obtained from $W^{(i-1)}$ by the following operations. For each node $w_g^{(i)}$ that is contained in G', set $w_g^{(i)} := w_g^{(i-1)}$ for each $1 \leq g \leq i$ and $w_g^{(i)} := \xi(w_g^{(i-1)})$ for each $i+1 \leq g \leq |V_R|$. For each node $w_g^{(i)}$ that is not contained in G', set $w_g^{(i)} := w_g^{(i-1)}$.

In the following we argue that the defined graphs and node sequences are well-defined. In particular, we show that an isomorphism as described above indeed exists.

Consider for a moment the scenario that for each node that $\mathcal{A}$ is presented with, $\mathcal{A}$ decides to orient the edge that is connected to the considered node via port 1 outwards with probability at least $1/\Delta$. In this case, the desired isomorphism trivially exists and we obtain $\mathcal{T} = G_{|V_R|-1} = \dots = G_0$

[19] Note that the recursive definition guarantees that we condition only on events that occur with nonzero probability, which implies the applicability of Definition 3.3.

[20] See Definition 1.4 for when an online-LOCAL algorithm has seen some edge.

and $\mathcal{W} = W^{(|V_R|-1)} = \cdots = W^{(0)} = W$. Moreover, two further properties hold: First, the fact that T is distance-D-correct, together with the definition of W (and the fact that $\mathcal{A}$ has complexity $D-1$), implies that, after being presented all nodes in $\mathcal{W} = W$ on $\mathcal{T} = G_0$, Algorithm $\mathcal{A}$ has seen only nodes that belong to G_T, all of which have degree Δ in $\mathcal{T}$. Second, the definition[21] of G_T implies that, whenever a node $w_i \in \mathcal{W}$ is presented to $\mathcal{A}$, all nodes of $\mathcal{W}$ that are presented later to $\mathcal{A}$ (i.e., that have index greater than i in $\mathcal{W} = W$) lie in the part of $\mathcal{T}$ that can be reached from w_i via the smallest frequent edge incident to w_i (with the usual conditioning), i.e., via the edge at port 1 in the currently considered case.

Now consider the general case that does not restrict the decisions of $\mathcal{A}$ to always orient the incident edge at port 1 outwards. From the definitions of the G_i and $W^{(i)}$ and Lemma 3.2, it follows inductively that the aforementioned two properties (where the restricted $\mathcal{W} = W$, $\mathcal{T} = G_0$ etc. are replaced by the $\mathcal{W}$, $\mathcal{T}$ etc. respectively induced by the considered $\mathcal{A}$) also hold in the general case, and that the desired isomorphism exists. (Note that Property 2 of Lemma 3.2 essentially provides a reformulation of "the view of an online-LOCAL algorithm with complexity $D-1$" that does not need to refer to such an algorithm.)

Next, we prove that when $\mathcal{A}$ is executed on $\mathcal{T}$ with[22] node sequence $\mathcal{W}$, the failure probability of $\mathcal{A}$ is at least $1/\Delta^{|V_R|}$. To this end, consider the events $\mathcal{E}_d$ discussed above, but defined (analogously) for $\mathcal{W} = W^{(|V_R|-1)}$ instead of for $W^{(i-1)}$ (with $1 \leq d \leq |V_R|-1$). By the above discussion, it follows that

1. the probability of $\mathcal{E}_d$ to occur is at least $1/\Delta^d$, for each $1 \leq d \leq |V_R|-1$, and
2. when $\mathcal{E}_{|V_R|-1}$ occurs, then, for each $1 \leq d' \leq |V_R|-1$, the first edge on the path from $w_{d'}^{(|V_R|-1)}$ to $w_{|V_R|}^{(|V_R|-1)}$ in $\mathcal{T}$ is oriented towards $w_{|V_R|}^{(|V_R|-1)}$.

In particular, the probability that, for each $1 \leq d' \leq |V_R|-1$, the first edge on the path from $w_{d'}^{(|V_R|-1)}$ to $w_{|V_R|}^{(|V_R|-1)}$ in $\mathcal{T}$ is oriented towards $w_{|V_R|}^{(|V_R|-1)}$ is at least $1/\Delta^{|V_R|-1}$.

Now consider the last node $w_{|V_R|}$ in the canonical sequence W for G_T. Observe that Property 5 of Definition 2.7 (together with the definitions of W and the reflection operation) implies that all neighbors of $w_{|V_R|}$ in G_T (and therefore also in G_0) appear in W (and necessarily before $w_{|V_R|}$). Observe further that the defined "isomorphic operations" performed to obtain $\mathcal{W} = W^{|V_R|-1}$ from W and $\mathcal{T}$ from G_0 preserve this property: for each $1 \leq g \leq |V_R|-1$, all neighbors of $w_{|V_R|}^{(g)}$ in G_g appear in $W^{(g)}$ (before $w_{|V_R|}^{(g)}$). In particular, all neighbors of $w_{|V_R|}^{(|V_R|-1)}$ in $G_{|V_R|-1} = \mathcal{T}$ appear in $W^{(|V_R|-1)}$ (before $w_{|V_R|}^{(|V_R|-1)}$).

Hence, the probability that $\mathcal{A}$ orients all edges incident to $w_{|V_R|}^{(|V_R|-1)}$ towards $w_{|V_R|}^{(|V_R|-1)}$ is at least $1/\Delta^{|V_R|-1}$. This implies that $\mathcal{A}$ solves $\mathcal{T}$ incorrectly with probability at least $1/\Delta^{|V_R|}$. Since the operations by which $\mathcal{T}$ is obtained from G_0 do not change the number of nodes (and $|V(G_0)| = n$), the lemma statement follows. □

[21] In particular, it is crucial that in the definition of the reflection operation (see Definition 2.24), the original graph H_1 is the one whose nodes can be reached *via port* 1 from the node at which the reflection takes place.

[22] Technically, in the randomized online-LOCAL model an algorithm is presented with *all* nodes (in some order) but if the algorithm outputs an incorrect partial output (that cannot be completed to a correct global output) with some probability already for a sequence of nodes that only contains a subset of the nodes of the input graph, then, by a simple completion argument, there must also exist a full sequence (containing all nodes) for which the algorithm fails with at least the same probability. As our goal is to prove a lower bound, we can therefore also consider such partial sequences as valid input query sequences.

Now we have all the ingredients to prove our lower bound for sinkless orientation in the randomized online-LOCAL model (which implies all the results stated in Corollary 1.3, including the same lower bound for the complexity of the distributed LLL in quantum-LOCAL).

Theorem 1.2. *The complexity of sinkless orientation in the randomized online-LOCAL model is in* $2^{\Omega(\log^* n)}$. *Moreover, this lower bound holds already on trees with maximum degree* 4 *in which no node has degree* 3.

Proof. Set $\Delta := 4$ and recall the compact power tower notation introduced in Definition 2.14. Let $n > 4^{P_4(2,5)}$ be some arbitrary integer. Let i be the largest integer such that $n > 4^{P_4(i,5)}$.

Consider the 4-ary construction tree T_i. By Corollary 2.32, T_i is distance-2^{i-1}-correct. Moreover, by Lemma 2.15, the number of nodes of T_i (and therefore also the number of reflect nodes of T_i) is at most $P_4(i, 5)$.

Observe that Observation 2.28, together with Lemma 2.21, implies that every node of G_{T_i} is marked. By the construction of G_{T_i}, it follows that for each node v of G_{T_i}, there exists a unique node of T_i that corresponds to the operation of marking v (during the process of operations used to construct G_{T_i}), and all of these unique nodes are distinct. Hence, $|V(G_{T_i})| \leq |V(T_i)|$. This in particular also implies that $n > 4^{|V(T_i)|} \geq 4^{|V(G_{T_i})|} \geq 4 \cdot |V(G_{T_i})| = \Delta \cdot |V(G_{T_i})|$.

Now let $\mathcal{A}$ be an arbitrary algorithm that solves sinkless orientation on n-node graphs in the randomized online-LOCAL model with probability at least $1 - 1/n$. By Lemma 3.4 (and the above discussion), the complexity of $\mathcal{A}$ cannot be smaller than 2^{i-1} as otherwise, on some n-node tree $\mathcal{T}$, Algorithm $\mathcal{A}$ would produce an incorrect output with probability at least $1/4^{P_4(i,5)} > 1/n$.

Observe that the definition of i (together with the fact that $2^{2^x} > 4^x$ for any sufficiently large real number x) implies that $i \in \Theta(\log^* n)$. We obtain that $\mathcal{A}$ has complexity in $2^{\Omega(\log^* n)}$, as desired. Moreover, by the construction of $\mathcal{T}$ in the proof of Lemma 3.4, we also obtain that the lower bound holds already on trees with maximum degree 4 in which no node has degree 3. □

Acknowledgements

We would like to thank Jukka Suomela for helpful discussions regarding the online-LOCAL model and proposing the studied problem. We would also like to thank Francesco d'Amore for notifying us about the fact that our approach for lifting the deterministic to the randomized online-LOCAL lower bound can be replaced by a simple black-box application of [1, Lemma 7.6, arxiv version]

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. This work is supported by ERC grant OLA-TOPSENS (grant agreement number 101162747) under the Horizon Europe funding programme.

References

[1] Amirreza Akbari, Xavier Coiteux-Roy, Francesco D'Amore, François Le Gall, Henrik Lievonen, Darya Melnyk, Augusto Modanese, Shreyas Pai, Marc-Olivier Renou, Václav Rozhon, and Jukka Suomela. Online locality meets distributed quantum computing. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, 2025. arxiv version at doi.org/10.48550/arXiv.2403.01903.

[2] Amirreza Akbari, Xavier Coiteux-Roy, Francesco d'Amore, François Le Gall, Henrik Lievonen, Darya Melnyk, Augusto Modanese, Shreyas Pai, Marc-Olivier Renou, Václav Rozhoň, and

Jukka Suomela. Online locality meets distributed quantum computing. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, STOC '25, page 1295–1306, New York, NY, USA, 2025. Association for Computing Machinery.

[3] Amirreza Akbari, Navid Eslami, Henrik Lievonen, Darya Melnyk, Joona Särkijärvi, and Jukka Suomela. Locality in online, dynamic, sequential, and distributed graph algorithms. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 10:1–10:20, 2023.

[4] Noga Alon. A parallel algorithmic version of the local lemma. *Random Structures & Algorithms*, 2(4):367–378, 1991.

[5] Heger Arfaoui and Pierre Fraigniaud. What can be computed without communications? *ACM SIGACT News*, 45(3):82–104, 2014.

[6] Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. The distributed complexity of locally checkable problems on paths is decidable. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 262–271, 2019.

[7] Alkida Balliu, Sebastian Brandt, Xavier Coiteux-Roy, Francesco d'Amore, Massimo Equi, François Le Gall, Henrik Lievonen, Augusto Modanese, Dennis Olivetti, Marc-Olivier Renou, Jukka Suomela, Lucas Tendick, and Isadora Veeren. Distributed quantum advantage for local problems. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, STOC '25, page 451–462, New York, NY, USA, 2025. Association for Computing Machinery.

[8] Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Classification of distributed binary labeling problems. In *34th International Symposium on Distributed Computing, DISC 2020*, volume 179 of *LIPIcs*, pages 17:1–17:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[9] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. *Journal of the ACM (JACM)*, 68(5):1–30, 2021.

[10] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed Δ-coloring plays hide-and-seek. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 464–477, 2022.

[11] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed maximal matching and maximal independent set on hypergraphs. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2632–2676. SIAM, 2023.

[12] Alkida Balliu, Sebastian Brandt, and Dennis Olivetti. Distributed lower bounds for ruling sets. *SIAM Journal on Computing*, 51(1):70–115, 2022.

[13] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, Jan Studenỳ, Jukka Suomela, and Aleksandr Tereshchenko. Locally checkable problems in rooted trees. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 263–272, 2021.

[14] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. Almost global problems in the local model. *Distributed Computing*, 34(4):259–281, 2021.

[15] Alkida Balliu, Juho Hirvonen, Janne H Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. New classes of distributed time complexity. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1307–1318, 2018.

[16] József Beck. An algorithmic approach to the Lovász local lemma. *Random Structures and Algorithms*, 2(4):343–365, 1991.

[17] Sebastian Brandt. An automatic speedup theorem for distributed problems. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 379–388, 2019.

[18] Sebastian Brandt, Yi-Jun Chang, Jan Grebík, Christoph Grunau, Václav Rozhon, and Zoltán Vidnyánszky. Local problems on trees from the perspectives of distributed algorithms, finitary factors, and descriptive combinatorics. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022*, volume 215 of *LIPIcs*, pages 29:1–29:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[19] Sebastian Brandt, Yi-Jun Chang, Jan Grebík, Christoph Grunau, Václav Rozhoň, and Zoltán Vidnyánszky. On homomorphism graphs. In *Forum of Mathematics, Pi*, volume 12, page e10. Cambridge University Press, 2024.

[20] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed lovász local lemma. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 479–488, 2016.

[21] Sebastian Brandt, Christoph Grunau, and Václav Rozhoň. Generalizing the sharp threshold phenomenon for the distributed complexity of the lovász local lemma. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 329–338, 2020.

[22] Sebastian Brandt, Juho Hirvonen, Janne H Korhonen, Tuomo Lempiäinen, Patric RJ Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemysław Uznański. Lcl problems on grids. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 101–110, 2017.

[23] Sebastian Brandt, Yannic Maus, and Jara Uitto. A sharp threshold phenomenon for the distributed complexity of the lovász local lemma. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 389–398, 2019.

[24] Sebastian Brandt and Dennis Olivetti. Truly tight-in-δ bounds for bipartite maximal matching and variants. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 69–78, 2020.

[25] Keren Censor-Hillel, Orr Fischer, François Le Gall, Dean Leitersdorf, and Rotem Oshman. Quantum distributed algorithms for detection of cliques. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022*, volume 215 of *LIPIcs*, pages 35:1–35:25, 2022.

[26] Karthekeyan Chandrasekaran, Navin Goyal, and Bernhard Haeupler. Deterministic algorithms for the Lovász local lemma. *SIAM Journal on Computing*, 42(6):2132–2155, 2013.

[27] Yi-Jun Chang. The complexity landscape of distributed locally checkable problems on trees. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020*, volume 179 of *LIPIcs*, pages 18:1–18:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[28] Yi-Jun Chang. The distributed complexity of locally checkable labeling problems beyond paths and trees. In *15th Innovations in Theoretical Computer Science Conference, ITCS*, volume 287 of *LIPIcs*, pages 26:1–26:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[29] Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. Distributed edge coloring and a special case of the constructive lovász local lemma. *ACM Transactions on Algorithms (TALG)*, 16(1):1–51, 2019.

[30] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the local model. *SIAM Journal on Computing*, 48(1):122–143, 2019.

[31] Yi-Jun Chang, Gopinath Mishra, Hung Thuan Nguyen, Mingyang Yang, and Yu-Cheng Yeh. A tight lower bound for 3-coloring grids in the online-local model. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, pages 106–116, 2024.

[32] Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the local model. *SIAM Journal on Computing*, 48(1):33–69, 2019.

[33] Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed algorithms for the lovász local lemma and graph coloring. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 134–143, 2014.

[34] Xavier Coiteux-Roy, Francesco d'Amore, Rishikesh Gajjala, Fabian Kuhn, François Le Gall, Henrik Lievonen, Augusto Modanese, Marc-Olivier Renou, Gustav Schmid, and Jukka Suomela. No distributed quantum advantage for approximate graph coloring. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1901–1910, 2024.

[35] Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986.

[36] Artur Czumaj and Christian Scheideler. A new algorithmic approach to the general Lovász local lemma with applications to scheduling and satisfiability problems. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 38–47, 2000.

[37] Francesco d'Amore. On the limits of distributed quantum computing. *Bulletin of EATCS*, 145(2), 2025.

[38] Peter Davies. Improved distributed algorithms for the lovász local lemma and edge coloring. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4273–4295, 2023.

[39] Peter Davies-Peck. On the locality of the Lovász local lemma. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, 2025. arxiv version at doi.org/10.48550/arXiv:2502.11690.

[40] Anubhav Dhar, Eli Kujawa, Henrik Lievonen, Augusto Modanese, Mikail Muftuoglu, Jan Studený, and Jukka Suomela. Local problems in trees across a wide range of distributed models. In *28th International Conference on Principles of Distributed Systems, OPODIS 2024, December 11-13, 2024, Lucca, Italy*, volume 324 of *LIPIcs*, pages 27:1–27:17, 2024.

[41] Fabien Dufoulon, Frédéric Magniez, and Gopal Pandurangan. Quantum communication advantage for leader election and agreement. *arXiv preprint arXiv:2502.07416*, 2025.

[42] Michael Elkin, Hartmut Klauck, Danupon Nanongkai, and Gopal Pandurangan. Can quantum communication speed up distributed computation? In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 166–175, 2014.

[43] Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and finite sets*, 2(2):609–627, 1975.

[44] Manuela Fischer and Mohsen Ghaffari. Sublogarithmic distributed algorithms for lovász local lemma, and the complexity hierarchy. In Andréa W. Richa, editor, *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, volume 91 of *LIPIcs*, pages 18:1–18:16, 2017.

[45] Pierre Fraigniaud, Maël Luce, Frédéric Magniez, and Ioan Todinca. Even-cycle detection in the randomized and quantum congest model. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, pages 209–219, 2024.

[46] François Le Gall, Harumichi Nishimura, and Ansis Rosmanis. Quantum advantage for the LOCAL model in distributed computing. In *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPIcs*, pages 49:1–49:14, 2019.

[47] François Le Gall and Ansis Rosmanis. Non-trivial lower bound for 3-coloring the ring in the quantum local model. *arXiv preprint arXiv:2212.02768*, 2022.

[48] Cyril Gavoille, Ralf Klasing, Adrian Kosowski, Łukasz Kuszner, and Alfredo Navarra. On the complexity of distributed graph coloring with local minimality constraints. *Networks: An International Journal*, 54(1):12–19, 2009.

[49] Cyril Gavoille, Adrian Kosowski, and Marcin Markiewicz. What can be observed locally? Round-based models for quantum distributed computing. In *International Symposium on Distributed Computing*, pages 243–257. Springer, 2009.

[50] Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 270–277. SIAM, 2016.

[51] Mohsen Ghaffari, David G Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 662–673. IEEE, 2018.

[52] Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 784–797, 2017.

[53] Mohsen Ghaffari and Hsin-Hao Su. Distributed degree splitting, edge coloring, and orientations. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2505–2523. SIAM, 2017.

[54] Christoph Grunau, Václav Rozhoň, and Sebastian Brandt. The landscape of distributed complexities on trees and beyond. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, pages 37–47, 2022.

[55] Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the Lovász local lemma. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 397–406, 2010.

[56] Magnús M Halldórsson, Yannic Maus, and Saku Peltonen. Distributed Lovász local lemma under bandwidth limitations. *arXiv preprint arXiv:2405.07353*, 2024.

[57] Atsuya Hasegawa, Srijita Kundu, and Harumichi Nishimura. On the power of quantum distributed proofs. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, pages 220–230, 2024.

[58] Alexander E Holroyd. Symmetrization for finitely dependent colouring. *Electronic Communications in Probability*, 29:1–12, 2024.

[59] Alexander E Holroyd, Tom Hutchcroft, and Avi Levy. Finitely dependent cycle coloring. *Electronic Communications in Probability*, 23:1–12, 2018.

[60] Alexander E Holroyd and Thomas M Liggett. Finitely dependent coloring. In *Forum of Mathematics, Pi*, volume 4, page e9. Cambridge University Press, 2016.

[61] Alexander E Holroyd, Oded Schramm, and David B Wilson. Finitary coloring. *The Annals of Probability*, pages 2867–2898, 2017.

[62] Taisuke Izumi, François Le Gall, and Frédéric Magniez. Quantum distributed algorithm for triangle finding in the CONGEST model. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 23:1–23:13, 2020.

[63] Taisuke Izumi and François Le Gall. Quantum distributed algorithm for the all-pairs shortest path problem in the congest-clique model. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 84–93, 2019.

[64] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *Journal of the ACM (JACM)*, 63(2):1–44, 2016.

[65] François Le Gall and Frédéric Magniez. Sublinear-time quantum computation of the diameter in congest networks. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 337–346, 2018.

[66] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on computing*, 21(1):193–201, 1992.

[67] Frédéric Magniez and Ashwin Nayak. Quantum distributed complexity of set disjointness on a line. *ACM Transactions on Computation Theory (TOCT)*, 14(1):1–22, 2022.

[68] Andrew S Marks. A determinacy approach to borel combinatorics. *Journal of the American Mathematical Society*, 29(2):579–600, 2016.

[69] Yannic Maus and Jara Uitto. Efficient CONGEST algorithms for the Lovász local lemma. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing, DISC 2021*, volume 209 of *LIPIcs*, pages 31:1–31:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[70] Michael Molloy and Bruce Reed. Further algorithmic aspects of the local lemma. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 524–529, 1998.

[71] Robin A Moser. Derandomizing the Lovász local lemma more effectively. *arXiv preprint arXiv:0807.2120*, 2008.

[72] Robin A. Moser. A constructive proof of the Lovász local lemma. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 343–350, 2009.

[73] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *Journal of the ACM*, 57(2):11:1–11:15, 2010.

[74] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000.

[75] Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 350–363, 2020.

[76] James B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985.

[77] Joel Spencer. Asymptotic lower bounds for ramsey functions. *Discrete Mathematics*, 20:69–76, 1977.

[78] Aravind Srinivasan. Improved algorithmic versions of the Lovász local lemma. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 611–620, 2008.

[79] Jukka Suomela. Open problem session, Dagstuhl Seminar 24471 "Graph Algorithms: Distributed Meets Dynamic", 2024.

[80] Joran van Apeldoorn and Tijn de Vos. A framework for distributed quantum queries in the congest model. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, pages 109–119, 2022.

[81] Changsheng Wang, Xudong Wu, and Penghui Yao. Complexity of eccentricities and all-pairs shortest paths in the quantum congest model. In *Spin*, volume 11(03). World Scientific, 2021.

[82] Xudong Wu and Penghui Yao. Quantum complexity of weighted diameter and radius in congest networks. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, pages 120–130, 2022.