# Quantum Sampling of Random Spanning Trees via Amortized Data Structures

Yassine Hamoudi* Adrian Tanasa† Shrinidhi Teganahally Sridhara‡

LaBRI, Univ. Bordeaux, CNRS UMR 5800, F-33400 Talence, France

Last update: November 5, 2026

**Abstract**

We present a quantum algorithm that generates a uniform superposition over the spanning trees of a graph – also known as a *q-sample* – using only a sublinear number of queries to the graph. This goes beyond previous work that focuses on generating *classical* samples, including a recent quantum algorithm by Apers, Gao, Ji, and Liu [AGJL25]. For an $n$-vertex, $m$-edge graph, our algorithm admits a preprocessing step of $\widetilde{O}(\sqrt{mn} + m^{1-\delta})$, after which each q-sample can be generated in $\widetilde{O}(n^{1+2\delta})$ for any $\delta$. Alternatively, $k$ independent q-samples can be generated at a total cost of $\widetilde{O}(\sqrt{kmn})$. We also prove a matching lower bound up to logarithmic factors, showing that our algorithm is essentially optimal. Further tradeoffs are given in the paper. In comparison, the optimal classical algorithms [ALV22] need an $\widetilde{O}(m)$ preprocessing step, after which each sample costs $\widetilde{O}(n)$ operations. These results yield faster quantum walk-based algorithms for counting spanning trees or finding a marked one.

Our result is obtained via quantum walk sampling over a sequence of slowly-changing Markov chains. Each chain is an *isotropized up-down walk* that mixes rapidly to the spanning tree distribution of the input graph. A key ingredient is an amortized data structure supporting fast implementation of the associated quantum walk operators throughout the sequence. This data structure maintains access to a spectral sparsifier and a leverage score sampler that evolve with the underlying graph.

## 1 Introduction

Sampling and counting discrete structures in graphs is a fundamental task in randomized algorithms, with examples including independent sets, matchings, cuts, paths, and spanning trees. Random spanning trees are among the most extensively studied examples: since Kirchhoff's celebrated matrix-tree theorem [Kir47], they have been closely tied to determinant identities, electrical networks, and graph Laplacians, see for example [DS84; LP17b]. Moreover, random spanning trees form a canonical class of strongly Rayleigh measures, and hence exhibit strong negative-dependence properties [BBL09].

For a weighted connected graph $G(V, E, w)$ with $n$ vertices and $m$ edges, an important algorithmic problem is to sample one spanning tree from the weighted distribution in which each tree $T$ is chosen with probability proportional to $\prod_{e\in T} w(e)$. This task has been studied from several complementary perspectives.

*yassine.hamoudi@labri.fr

†adrian.tanasa@labri.fr

‡shrinidhi.teganahally-sridhara@labri.fr

A first family of algorithms exploits the matrix-tree theorem in a direct manner: Laplacian cofactors give exact counting information that can be used to guide the sampling process. This determinant-based approach led to polynomial-time samplers by Guenoche [Gué83] and Kulkarni [Kul90], and was later refined by Colbourn, Myrvold, and Neufeld to run in $O(n^\omega)$ arithmetic operations [CMN96] where $\omega \approx 2.37$ is the matrix multiplication exponent.

Another standard family of such algorithms gives exact samplers using random walks. The Aldous–Broder algorithm samples a uniform spanning tree from the first-entrance tree of a random walk [Ald90; Bro89], and Wilson's loop-erased random-walk algorithm gave a faster and more practical exact sampler [Wil96]. Subsequent work accelerated this random-walk viewpoint using Laplacian solvers, electrical flows, and Schur complements [KM09; MST15], culminating in Schild's $m^{1+o(1)}$ time algorithm which is an almost-linear-time exact sampler [Sch18].

In parallel, determinant-preserving sparsification algorithms give approximate samplers running in $\widetilde{O}(n^2 \cdot \epsilon^{-2})$ time [DPPR20]. A more recent direction studies approximate sampling through high-dimensional expansion and down-up walks; this viewpoint gives sharp mixing analyses for natural walks on spanning trees and led to near-linear-time approximate samplers, with runtime $O(m \log^2 m)$ in the work of Anari, Liu, Oveis Gharan, Vinzant, and Vuong [ALG+21].

When additional information on the target distribution is available, sampling can be accelerated further. A powerful example is domain sparsification for strongly Rayleigh distributions [AD20; ADVY22; ALV22]: using approximate element marginals, the sampling task is reduced to a sampling task over a much smaller effective domain. For spanning trees, this gives an $\widetilde{O}(n)$-step sampler after $\widetilde{O}(m)$ marginal preprocessing [ALV22].

The same sampling task has also been studied with quantum algorithms. Most recently, Apers, Gao, Ji and Liu [AGJL25] obtained a quantum speedup for the classical approximate sampling task: given quantum query access to the adjacency-list of $G$, their algorithm outputs the classical description of a random spanning tree using $\widetilde{O}(\sqrt{mn})$ queries and time, matching a corresponding quantum query lower bound up to polylogarithmic factors. The output is still a classical spanning tree, as for all the algorithms above, and quantum subroutines are used only to speed up the sampling.

Our paper studies the coherent analogue of this task. Given sparse adjacency-list oracle access to a weighted connected graph $G = (V, E, w)$, we aim to prepare the quantum-sample (q-sample) of its spanning-tree distribution,

$$|\pi\rangle = \sum_T \sqrt{\pi(T)}|T\rangle, \qquad \pi(T) = \frac{\prod_{e\in T} w(e)}{\sum_{T'} \prod_{e\in T'} w(e)},$$

where $|T\rangle$ stores the lexicographically ordered list of canonical representations of the weighted edges of $T$. Query complexity counts only calls to the adjacency-list oracle.

Quantum sampling was put forward by Aharonov and Ta-Shma [AT07] as a state-generation task. A classical sampling algorithm can be used directly to prepare $\sum_r \sqrt{\pi(T(r))}|r\rangle|T(r)\rangle$, but not to obtain the desired state $\sum_T \sqrt{\pi(T)}|T\rangle$, since the random-seed register $|r\rangle$ cannot generally be efficiently erased. They also showed that a generic efficient procedure for turning classical samplers into quantum ones would imply SZK $\subseteq$ BQP, underscoring that q-sampling is likely a difficult task in general. Among the special-purpose quantum algorithms developed for q-sampling, *quantum simulated annealing* has proven particularly successful [SBBK08; WA08; OBD18; HW20; CLL+22; CH23] and is among the primitives that underpin our algorithm.

Beyond state generation, q-samples are a recurring primitive across several quantum algorithms. In *learning theory* [AW17], some classes of functions (such as Parities [BV97] or Disjunctive Normal Forms [BJ99]) can be learned using super-polynomially fewer quantum samples than classical ones. In *Markov Chain Monte Carlo methods*, quantum walk search frameworks typically start from the coherent encoding of the stationary distribution and then amplify or detect marked

states in the chain's state space [MNRS11; AGJK20]; this stationary-q-sample starting point underlies applications such as element distinctness [Amb07] and triangle finding [MSS07]. This has also led to work on the cost of obtaining such starting states, through quantum mixing, controlled quantum walks, analog quantum dynamics, and seed-set methods [CLR20; Ape19; LS23]. Finally, coherent encoding of a system's initial state is essential for *simulating its evolution* under quantum dynamics, such as via Hamiltonian simulation or the quantum adiabatic algorithm.

A broader challenge in quantum algorithms built on quantum walks is the efficient implementation of the quantum walk operator, the quantum analogue of taking a step of the walk. It requires data structures with quantum-specific properties, such as uniqueness and history independence. Examples include the hash table combined with a skip list used in Ambainis's algorithm for element distinctness [Amb07]. Such data structures are very rare and much harder to construct than their classical counterparts, and a further difficulty is porting classical techniques such as amortization to the quantum setting. Implementing the quantum walk operator efficiently is also the main challenge in our setting. We address it with an amortized data structure that is built once, during preprocessing, and reused across many related walks.

Our amortized data structure viewpoint is also related in spirit to recent quantum algorithms that maintain samplers for slowly changing distributions. Of particular relevance is the work of Bouland, Getachew, Jin, Sidford, and Tian [BGJ+23], who apply such ideas to zero-sum games. There, the evolving distributions are Gibbs distributions, whereas in our setting they are leverage-score distributions along a sequence of slowly-varying graphs.

### 1.1 Main result

The main result of this paper is a sublinear-query quantum algorithm for preparing q-samples of the spanning tree distribution. To the best of our knowledge, this is the first work on q-sampling random spanning trees. Beyond preparing a single coherent sample, the algorithm gives a range of preprocessing–sampling tradeoffs and is particularly useful when many q-samples are needed from the same graph. Throughout the paper, query complexity is measured with respect to the sparse adjacency-list oracle $\mathcal{O}_G$, where the neighbor list of each vertex is ordered lexicographically. Below is an informal statement of the theorem.

**Theorem 3.1 (Informal).** *Let $G = (V, E, w)$ be a connected weighted graph with $n$ vertices and $m \geq 100\,n$ edges, and let $\pi$ be its spanning tree distribution.*

*For any $\delta \in [0, 1]$, there is an algorithm that prepares q-samples of $\pi$ that admits a preprocessing step using $\widetilde{O}\left(\sqrt{mn} + m^{1-\delta}\right)$ queries to the adjacency-list oracle $\mathcal{O}_G$, after which each q-sample can be generated using $\widetilde{O}\left(n^{1+2\delta}\right)$ queries to $\mathcal{O}_G$.*

*Alternatively, for any integer $k \geq 1$, there is an algorithm that prepares $k$ independent q-samples of $\pi$ with total query cost $\widetilde{O}(\sqrt{kmn})$, after the initial preprocessing of the same cost.*

We also prove a matching lower bound up to polylogarithmic factors, showing that the query complexity of our quantum algorithm is essentially optimal.

**Theorem 4.2 (Informal).** *Any algorithm preparing $k \geq 1$ independent q-samples must perform at least $\widetilde{\Omega}(\min(\sqrt{kmn}, m))$ queries to $\mathcal{O}_G$.*

In this work, we focus only on adjacency-list query complexity. A loose implementation-level accounting gives an $O(n^2)$ time bound (cf Section 3.5) for both preprocessing and q-sampling, but we do not optimize runtime here. It is natural to expect that the time complexity of our algorithm can be brought closer to its query complexity with suitable quantum analogues of the dynamic tree data structures used in classical spanning-tree algorithms, such as link-cut trees [ALG+21; ALV22; RTF18]. Developing such quantum dynamic-tree data structures is beyond the scope of this work.

## 1.2 Applications

After the preprocessing phase, the spanning-tree q-sample can be used as a primitive for several downstream tasks. We describe two such applications: approximate counting of spanning trees and quantum-walk search for marked spanning trees.

**Counting spanning trees.** For unweighted graphs, the number of spanning trees is the partition function along our cooling schedule, so we can combine q-sampling along the schedule with the quantum partition-function estimator of Cornelissen and Hamoudi [CH23], which adds a factor $\widetilde{O}(n^{1/4})$ over the q-sampling cost. Since the partition function is the weighted spanning-tree sum, and equals the number of spanning trees in the unweighted case, by choosing $\rho = n^{-1/4}$ in Theorem 3.1, we get an algorithm that, for $m \geq n^{5/4}$, approximates the number of spanning trees with $\widetilde{O}\big(\sqrt{m} \cdot n^{5/8}\big)$ queries, including preprocessing.

For comparison, the recent algorithm of Liu, Peng, and Yang [LPY25] estimates the spanning-tree count in time $O(m^{3/2})$. In sparse graphs, this improves over the previously best general bound $\widetilde{O}(m + n^{15/8})$ by Chu, Gao, Peng, Sachdeva, Sawlani and Wang [CGP+20]. Their result is a classical time bound, and since the algorithm reads the input graph, its query complexity is $\Theta(m)$. By contrast, our overall query complexity is $\widetilde{O}(\sqrt{m} \cdot n^{5/8})$ for $m \geq n^{5/4}$, which is $o(m)$ whenever $m = \omega(n^{5/4})$. Thus, in this regime, our quantum counting algorithm is sublinear in the number of edges, including preprocessing.

**Searching for marked spanning trees.** The q-sample also gives the setup state needed in quantum-walk search algorithms. For instance, in the MNRS framework [MNRS11], for a reversible ergodic Markov chain with stationary distribution $\pi$, spectral gap $\gamma$, setup cost $S$, update cost $U$, checking cost $C$, and marked set of stationary measure at least $\alpha$, one can find a marked element with high probability using $\widetilde{O}\left(S + \frac{1}{\sqrt{\alpha}}\left(\frac{1}{\sqrt{\gamma}}U + C\right)\right)$ queries.

In our setting, the setup cost corresponds to the cost of preprocessing and producing one q-sample, the update cost corresponds to the cost of reflection around the q-sample $|\pi\rangle$ which is summarized in Theorem 3.1. Thus our algorithm can be used as a black box inside quantum walk search algorithms for marked spanning trees, with an additional dependence on the marked fraction and the checking cost. An example is electrical distribution network reconfiguration, where one seeks a spanning tree of a power network whose energy loss is small [IKK+26]. Here the marked trees are those whose loss is below a given threshold, and the checking cost is the cost of evaluating the loss of a given tree.

## 1.3 Proof overview

We provide high-level ideas behind the proof of Theorem 3.1. We begin with a simpler approach that illustrates the main tools underlying our algorithm and highlights the key challenges that arise in achieving the improved bounds. We then describe how our techniques address these challenges to obtain the final result.

It is well known that a rapidly mixing reversible Markov chain with stationary distribution $\pi$ can be quantized to obtain a unitary operator $W$, referred to as the quantum walk operator, that enables efficient reflection about the q-sample $|\pi\rangle$. This construction is known as Szegedy quantization [Sze04]. Given the ability to reflect about $|\pi\rangle$, one can start from an arbitrary state $|\psi\rangle$ and use Grover-style amplitude amplification [BHMT02] to prepare $|\pi\rangle$. This requires $O(1/\langle\psi\,|\,\pi\rangle)$ reflections about $|\pi\rangle$. However, the overlap $\langle\psi\,|\,\pi\rangle$ can be exponentially small for an arbitrary initial state $|\psi\rangle$. Thus, having a rapidly mixing Markov chain that allows for efficient sampling from $\pi$ does not, by itself, imply an efficient procedure for preparing the q-sample $|\pi\rangle$.

A standard approach for overcoming this issue is simulated annealing, which considers a sequence of slowly varying Markov chains whose stationary distributions interpolate between an easily preparable initial distribution and the target distribution. The parameters defining this sequence form a cooling schedule, which is chosen so that the q-samples of consecutive distributions have sufficiently large overlap (usually, constant overlap). Consequently, their corresponding quantum walk operators can be used to transform one q-sample into the next, allowing the target q-sample to be prepared efficiently.

We design the following cooling schedule for spanning tree distributions: a sequence of slowly varying graphs that interpolates between a graph whose spanning tree distribution is concentrated on a single spanning tree $S$ and the target graph $G$. The q-sample of the first distribution is the easily preparable state $|S\rangle$, and we prepare $|\pi\rangle$ by moving along the schedule towards $G$. We choose $S$ to be an admissible spanning tree, namely one with non-negligible probability under $\pi$. A maximum weight-product spanning tree has this property, since it has the largest probability under $\pi$, and it can be found using $\widetilde{O}(\sqrt{mn})$ queries [DHHM06].

We parameterize the schedule by an inverse temperature $\beta \geq 0$. For each $\beta$, let $G_\beta$ be the graph obtained from $G$ by scaling the weight of every edge outside $S$ by $e^{-\beta}$ and leaving the edges of $S$ unchanged, that is, $w_\beta(e) = e^{-\beta} w(e)$ for $e \notin S$ and $w_\beta(e) = w(e)$ for $e \in S$. Writing $\pi_\beta$ for the spanning tree distribution of $G_\beta$ and $w(T) := \prod_{e \in T} w(e)$ for the weight of a tree $T$, we have, for every spanning tree $T$,

$$\pi_\beta(T) \propto e^{-\beta |T \setminus S|}\, w(T),$$

where $|T \setminus S|$ is the number of edges of $T$ outside $S$. At $\beta = 0$ we recover the target distribution, $\pi_0 = \pi$. As $\beta$ grows, the distribution increasingly favors $S$, and as $\beta \to \infty$ it collapses onto $S$, so that $\pi_\infty = \mathbb{1}_S$. This is similar in spirit to the schedule considered in [AD20] for estimating marginals of a log-concave distribution. Query access to every $G_\beta$ can be simulated using $\mathcal{O}_G$, the adjacency-list oracle for $G$.

We then show that a short schedule exists: there are inverse temperatures $\beta_l \geq \cdots \geq \beta_0 = 0$ such that (i) $\pi_{\beta_l}$ is concentrated mostly on the admissible tree $S$, (ii) consecutive distributions have constant overlap, $\langle \pi_{\beta_i} \,|\, \pi_{\beta_{i+1}} \rangle = \Omega(1)$, and (iii) the schedule has length $l = \widetilde{O}(\sqrt{n})$ (Proposition 3.21), and it can be found using the adaptive-schedule procedure of Harrow and Wei [HW20] using reflections around $|\pi_\beta\rangle$. Given an efficient way to reflect about each $|\pi_\beta\rangle$, we can compute such a schedule, start from $|S\rangle$ and walk down the schedule from $\beta_l$ to $\beta_0$, converting each q-sample into the next, to end with $|\pi_{\beta_0}\rangle = |\pi\rangle$.

One approach is to obtain these reflections by quantizing the standard up-down walk on spanning trees [CSV23]. From a tree $T$, this walk adds an edge $f$ (the Up step), then removes an edge $e$ from the unique cycle in $T + f$ (the Down step), landing on $T + f - e$. The chain mixes in $\widetilde{O}(m)$ steps [ALGV19], so the corresponding Szegedy walk operator $W_\beta$ can be used to implement a reflection about $|\pi_\beta\rangle$ using $\widetilde{O}(\sqrt{m})$ queries to $\mathcal{O}_G$. Using these reflections, the adaptive-schedule procedure computes the schedule using $\widetilde{O}(\sqrt{mn})$ queries to $\mathcal{O}_G$. Once the schedule is computed, traversing its $l = \widetilde{O}(\sqrt{n})$ steps takes $\widetilde{O}(\sqrt{n}) \cdot \widetilde{O}(\sqrt{m}) = \widetilde{O}(\sqrt{mn})$ queries, so the total cost of preparing the q-sample $|\pi\rangle$ is $\widetilde{O}(\sqrt{mn})$.

This raises two natural questions. Can $k$ copies of $|\pi\rangle$ be prepared with fewer than the naive $\widetilde{O}(k\sqrt{mn})$ queries? And can a one-time preprocessing step make each subsequent q-sample cheaper than $\widetilde{O}(\sqrt{mn})$ queries? Our main result answers both the questions in the affirmative.

**Coherent Isotropization.** Our approach is to quantize a different Markov chain from the standard up-down walk: one in which the transition probabilities of the Up and Down steps are biased by edge marginals. For the spanning tree distribution, the marginal of an edge $e$ is its

leverage score $\ell(e)$:

$$\Pr_{T\sim\pi}(e \in T) = \ell(e) = w(e) \cdot R^G_{\text{eff}}(e),$$

where $R^G_{\text{eff}}(e)$ is the effective resistance of $e$ in $G$. The Up step adds an edge sampled with probability proportional to its marginal, and the Down step removes an edge from the resulting cycle using a second distribution, also biased by marginals. We call this the *marginal-aware walk*, see Markov Chain 2 for a detailed description. Biasing by marginals acts as an implicit isotropic transformation: the walk behaves as if every edge had roughly the same marginal, that is, as if the distribution were in isotropic position. Its transition matrix has spectral gap $\widetilde{\Omega}(1/n)$, so it mixes even faster than the standard up-down walk (Lemma 3.14 and [ALV22]). Consequently, quantizing this chain allows us to implement a reflection about $|\pi_\beta\rangle$ using $\widetilde{O}(\sqrt{n})$ applications of the resulting quantum walk operator $W_\beta$ (see first part of Proposition 3.16). This walk is similar in spirit to the approach of [ALV22], but their walk runs on the spanning trees of a different graph, constructed from the input graph and its marginals, whereas ours runs on the spanning trees of the input graph $G$ itself. This preserves *coherence*: the state space, and hence the encoding of the q-sample, is the same set of spanning trees of $G$ for every $\beta$. The remaining challenge is to implement the quantum walk operator $W_\beta$ efficiently for every $G_\beta$ on the schedule.

**Main technical challenge: amortized q-sampling of leverage scores.** Applying $W_\beta$ requires a unitary that, given a spanning tree $|T\rangle$, prepares a superposition over its neighbors in the marginal-aware walk on $G_\beta$. Following the two steps of the walk, it consists of a Down step and an Up step. The Down step acts only on the registers already holding $T$ and the added edge $f$ (with their weights): it finds the cycle in $T + f$ and samples the removed edge from a distribution depending only on the stored weights, so it needs no queries to $\mathcal{O}_G$ (its time cost is discussed in Section 3.5). The Up step does not depend on $T$. It prepares a second q-sample, this time over the $m$ edges rather than over spanning trees:

$$|\mu_\beta\rangle = \frac{1}{\sqrt{n-1}} \sum_{e\in E} \sqrt{\ell_\beta(e)}|e\rangle.$$

This is the q-sample of the leverage-score distribution of $G_\beta$, where $\ell_\beta(e) = w_\beta(e)R^{G_\beta}_{\text{eff}}(e)$ is the marginal of $e$ under $\pi_\beta$ and the normalization is $n-1$ because leverage scores sum to $n-1$. (The algorithm prepares a variant built from overestimates of the leverage scores, described below.) Thus all queries to $\mathcal{O}_G$ made by $W_\beta$ are spent on this q-sample. Preparing it from scratch for every $G_\beta$ would be too expensive, so the main technical contribution of this work is an *amortized* q-sampling algorithm for leverage scores: after a one-time preprocessing step, the q-sample for every $G_\beta$ on the schedule can be prepared cheaply. It has two ingredients, described next: reusable overestimates of the effective resistances, which bound the leverage scores and drive the rejection sampling, and a reusable set of heavy edges that speeds the rejection sampling up.

**Preprocessing of effective resistances.** The first ingredient is a data structure that, once prepared, provides *effective-resistance overestimates* for any $G_\beta$ without further queries to $\mathcal{O}_G$. These are values $\widetilde{R}^{G_\beta}_{\text{eff}}(e)$ with

$$R^{G_\beta}_{\text{eff}}(e) \le \widetilde{R}^{G_\beta}_{\text{eff}}(e) \le \lambda\, R^{G_\beta}_{\text{eff}}(e),$$

for a constant $\lambda \ge 1$ (we use $\lambda = 2$). The induced marginal overestimates $w_\beta(e)\widetilde{R}^{G_\beta}_{\text{eff}}(e)$ satisfy $\ell_\beta(e) \le w_\beta(e)\widetilde{R}^{G_\beta}_{\text{eff}}(e) \le \lambda\,\ell_\beta(e)$ and sum to at most $\lambda(n-1) = O(n)$, which is what makes rejection sampling from the leverage-score distribution efficient. A spectral sparsifier (a

subgraph with fewer edges that approximates the Laplacian of $G$) suffices to compute such overestimates [SS11]. The direct approach of building a new sparsifier for every $G_\beta$ is too expensive. Instead, we exploit the structure of $G_\beta$: for every $\beta$, its edge set is the same as that of $G$, and only the weights of the edges outside the admissible tree $S$ are rescaled, all by the same factor. We show that a sparsifier for the subgraph of $G$ outside $S$ is enough to obtain a sparsifier for any $G_\beta$ without further queries to $\mathcal{O}_G$ (Lemma 3.5). Such a sparsifier can be computed by the quantum sparsification procedure of Apers and de Wolf [AW20] using $\widetilde{O}(\sqrt{mn})$ queries to $\mathcal{O}_G$. Thus, the same sparsification data, once computed, can be reused for every $G_\beta$ with only lightweight updates (Proposition 3.6). This yields effective-resistance overestimates for every graph $G_\beta$ in the schedule using only $\widetilde{O}(\sqrt{mn})$ queries to $\mathcal{O}_G$, paid once during preprocessing. Amortized over the $\widetilde{O}(\sqrt{n})$ chains on the schedule, this is $\widetilde{O}(\sqrt{m})$ queries per chain instead of $\widetilde{O}(\sqrt{mn})$, and nothing is paid again for each q-sample prepared afterwards.

**Faster sampling from the leverage-score distribution.** Given effective-resistance overestimates for every $G_\beta$, it remains to prepare $|\mu_\beta\rangle$ cheaply. A natural way is rejection sampling implemented with amplitude amplification, starting from a proposal state that is cheap to prepare. The uniform proposal $\frac{1}{\sqrt{m}}\sum_e |e\rangle$ costs only $\widetilde{O}(1)$ queries once the vertex degrees and their prefix sums are stored. Given effective-resistance overestimates of $G_\beta$, amplitude amplification then prepares $|\mu_\beta\rangle$ using $\widetilde{O}(\sqrt{m/n})$ queries to $\mathcal{O}_G$.

We improve on this with a *$\rho$-heavy proposal*. For a threshold $\rho \in [n/m, 1]$, let $\textsc{Heavy}_\rho(\beta)$ be the set of edges of $G_\beta$ whose leverage score is at least $\rho$. The proposal gives each edge in $\textsc{Heavy}_\rho(\beta)$ amplitude proportional to $\sqrt{\ell_\beta(e)}$ and every other edge the same amplitude $\sqrt{\rho}$, that is, it is proportional to

$$\sum_{e\in\textsc{Heavy}_\rho(\beta)} \sqrt{\ell_\beta(e)}|e\rangle + \sum_{e\notin\textsc{Heavy}_\rho(\beta)} \sqrt{\rho}|e\rangle.$$

Since leverage scores sum to $n-1$, at most $(n-1)/\rho$ edges are heavy, so $\textsc{Heavy}_\rho(\beta)$ can be stored explicitly. Given this stored set, the proposal can be prepared with $\widetilde{O}(1)$ queries. Using amplitude amplification, this prepares $|\mu_\beta\rangle$ with $\widetilde{O}(\sqrt{m\rho/n})$ queries to $\mathcal{O}_G$, improving on the uniform proposal by a factor $\sqrt{\rho}$.

The catch is that $G_\beta$ varies with $\beta$, so the leverage scores and hence the heavy edges may change along the schedule. We avoid recomputing the heavy edges at every $\beta$ by exploiting the monotonicity of leverage scores along the schedule. Recall that $S$ is the admissible spanning tree. As $\beta$ increases, leverage scores outside $S$ decrease, while those of edges in $S$ increase. Thus, $\textsc{Heavy}_\rho(0)$, the heavy edges at $\beta = 0$, together with $S$, form an envelope $\mathcal{E}_\rho$ containing all heavy edges throughout the schedule. In other words, every edge outside $\mathcal{E}_\rho$ has leverage score below $\rho$ in every $G_\beta$. This envelope has size $O(n/\rho)$ and, together with the effective-resistance data structure, can be computed once during preprocessing with $\widetilde{O}(\sqrt{mn/\rho})$ queries to $\mathcal{O}_G$. Storing these data therefore suffices to prepare the $\rho$-heavy proposal, and hence $|\mu_\beta\rangle$, for any $G_\beta$ using $\widetilde{O}(\sqrt{m\rho/n})$ queries to $\mathcal{O}_G$, with no recomputation as $\beta$ changes. This is exactly the per-application query cost of $W_\beta$ in Proposition 3.16.

**Preparing the q-sample and putting the costs together.** Once these data structures are available, we start from $|S\rangle$ and anneal along the adaptive schedule in reverse order to prepare the target state $|\pi\rangle$. Since the adaptive schedule is constructed so that consecutive q-samples have sufficiently large overlap, we can efficiently transform $|\pi_{\beta_j}\rangle$ to $|\pi_{\beta_{j-1}}\rangle$ using the quantum walk operators at the relevant adaptive temperatures and fixed-point amplification [Gro05] (Lemma A.3). Iterating these transformations prepares the target q-sample $|\pi\rangle$. The preprocessing phase builds the reusable data structures needed throughout the cooling schedule. Choosing

the threshold parameter $\rho$ gives the tradeoff in Theorem 3.1: for any $\delta \in [0,1]$, the preprocessing cost is $\widetilde{O}(\sqrt{mn} + m^{1-\delta})$, and each q-sample after preprocessing costs $\widetilde{O}(n^{1+2\delta})$. When $k$ q-samples are required, we choose the threshold differently, so that both the preprocessing cost and the total q-sampling cost are $\widetilde{O}(\sqrt{kmn})$.

**Lower bound.** We show that $\widetilde{\Omega}(\sqrt{kmn})$ queries are necessary for preparing $k$ independent copies of the q-sample $|\pi\rangle$ (Theorem 4.2). Since measuring a q-sample in the computational basis gives a classical sample, it suffices to lower bound the cost of obtaining $k$ independent classical samples. We reduce from $n$ independent instances of the $k$-threshold problem on $q = \lceil 2m/n \rceil$ bits, where the goal is to decide for each instance whether it contains at least $k$ ones. By the direct product theorem of Ambainis, Špalek and de Wolf [AŠW07], solving all $n$ instances with a one-sided-error algorithm requires $\Omega(n\sqrt{kq}) = \Omega(\sqrt{kmn})$ queries.

We encode the instances in a graph where each instance is a vertex that must attach to the rest of the graph through one of its marked positions. Instance $i$ becomes a vertex $u_i$ joined to a shared set of vertices $v_1, \ldots, v_q$, with the $j$th bit determining the weight of $(u_i, v_j)$, and a root $s$ is joined to every $v_j$ by a very heavy edge. The heavy edges lie in almost every spanning tree, so each $u_i$ is attached by one uniformly random marked position, and a single sample reveals a random marked position in *every* instance at once (Figure 1). Collecting $O(k \log(nk))$ samples reveals $k$ distinct marked positions in every instance with constant probability, which solves all instances with one-sided error. Since only the edge weights depend on the input, each oracle query is simulated by one instance query, so $O(\log(nk))$ rounds of $k$ samples cost $O(Q \log(nk))$ queries, where $Q$ is the cost of $k$ samples. The direct product theorem then gives $Q = \Omega(\sqrt{kmn}/\log(nk))$.

## 2 Preliminaries

We consider undirected weighted graphs $G = (V, E, w)$, where $V$ is the vertex set, $E$ is the edge set, and $w : E \to \mathbb{R}_{\geq 0}$ is the weight function. We set $n := |V|$ and $m := |E|$, and identify $V$ with $[n] := \{1, 2, \ldots, n\}$, so that vertices are totally ordered. Throughout the paper, log denotes the base-2 logarithm and ln the natural logarithm. We use $\widetilde{O}(\cdot)$ to hide polylogarithmic factors. Inside asymptotic notation the base of the logarithm is immaterial; we write ln only where the exact value of a quantity depends on it.

### 2.1 Computational Model

We assume sparse adjacency-list access to $G$ through an oracle $\mathcal{O}_G$ [AW20]. Given a vertex $u$, the oracle returns $\deg(u)$; given additionally an index $i \in [\deg(u)]$, it returns the $i$th neighbor of $u$ and the corresponding edge weight. Neighbors are ordered lexicographically. This sparse-access model is standard in quantum algorithms for linear-system solving and Hamiltonian simulation [HHL09; CKS17]. One has

$$\mathcal{O}_G|u\rangle|0\rangle = |u\rangle|\deg(u)\rangle,$$

and

$$\mathcal{O}_G|u\rangle|i\rangle|0\rangle|0\rangle = |u\rangle|i\rangle|v\rangle|w(u,v)\rangle,$$

where $v$ is the $i$th neighbor of $u$. Thus every undirected edge $\{u, v\}$ has two adjacency-list positions, $(u, i)$ and $(v, j)$, where $v$ is the $i$th neighbor of $u$ and $u$ is the $j$th neighbor of $v$. Throughout, adjacency-list oracle refers to this sparse-access oracle model.

Since adjacency lists are lexicographically sorted, we can also recover the adjacency-list index of an edge using binary search. Given $(u, v)$, we find the index $i$ such that $v$ is the $i$th neighbor

of $u$ using $O(\log n)$ queries to $\mathcal{O}_G$, implementing

$$|u\rangle|v\rangle|0\rangle \mapsto |u\rangle|v\rangle|i\rangle.$$

In particular, given one adjacency-list position $(u, i)$ of an edge, one query returns $v$ and $w(u, v)$, and binary search in the list of $v$ gives its other position $(v, j)$, using $O(\log n)$ queries in total.

Our computational model is a quantum-accessible classical control model similar to [AW20; AGJL25]. For a graph $G$ with $n$ vertices, let $N$ denote the number of spanning trees of $G$. (Note that $N$ is generally exponential in $n$, and can be as large as $n^{n-2}$ for the complete graph.) The algorithm may run quantum subroutines on $O(\text{poly}\log(N))$ qubits, make quantum queries to the input oracle $\mathcal{O}_G$, and access classical data structures constructed during preprocessing through quantum-read/classical-write memory. A QRAM operation is either a classical write to such a data structure or a coherent quantum read, possibly in superposition.

Throughout, query complexity refers only to the number of queries to the input oracle $\mathcal{O}_G$. QRAM operations and ordinary classical computation are not counted as input queries. When discussing time complexity, we count classical gates, quantum gates, input-oracle queries, and QRAM operations. The amount of QRAM used is the size of the data structures explicitly constructed during preprocessing, and will be specified when the relevant preprocessing procedure is invoked.

## 2.2 Spanning Trees

Let $\Delta_G$ be the set of spanning trees of $G$, so that $N = |\Delta_G|$. We call $G$ *connected* if its edges of positive weight connect all of $V$; equivalently, some spanning tree has positive weight.

**Definition 2.1** (Spanning tree distribution)**.** Let $G = (V, E, w)$ be a connected weighted graph. For $T \in \Delta_G$, define $w(T) := \prod_{e\in T} w(e)$. The spanning tree distribution $\pi_G$ is

$$\pi_G(T) = \frac{w(T)}{\sum_{T'\in\Delta_G} w(T')}.$$

The denominator is positive exactly when $G$ is connected, so $\pi_G$ is well defined. We indicate the graph in the notation when it matters (e.g., $\Delta_G$, $\pi_G$) and omit it when it is clear from the context.

A quantum algorithm for sampling from $\pi_G$, with a speedup over classical algorithms, was given in [AGJL25]. We recall this result.

**Theorem 2.2** (Quantum algorithm for classical sampling [AGJL25])**.** *There is a quantum algorithm that, given query access to $\mathcal{O}_G$ for a connected weighted graph $G$, outputs a spanning tree drawn from a distribution $\epsilon$-close to $\pi$ in total variation distance with high probability. The algorithm uses $\widetilde{O}(\sqrt{mn}\log(1/\epsilon))$ queries to $\mathcal{O}_G$ and runs in $\widetilde{O}(\sqrt{mn}\log(1/\epsilon))$ time.*

We emphasize here that our goal is stronger: instead of having as an output a classical tree sampled from $\pi$, we prepare the coherent encoding of the distribution. In our algorithms, the tree register stores not only the edge information but also their weights. We thus use the following canonical weighted encoding:

**Definition 2.3** (Q-sample of $\pi$)**.** Let $\pi$ be the spanning tree distribution of $G = (V, E, w)$. We encode a spanning tree $T$ by listing its $n - 1$ edges in lexicographic order, with each edge stored as an ordered pair of endpoints together with its weight in $G$. Thus we write the spanning tree as,

$$T = \big((u_1, v_1, w(u_1, v_1)), \ldots, (u_{n-1}, v_{n-1}, w(u_{n-1}, v_{n-1}))\big),$$

where $u_i < v_i$ for each $i$ and $(u_1, v_1) <_{\text{lex}} \cdots <_{\text{lex}} (u_{n-1}, v_{n-1})$. The q-sample of $\pi_G$ is the coherent state

$$|\pi_G\rangle = \sum_{T \in \Delta_G} \sqrt{\pi_G(T)}|T\rangle.$$

Here $|T\rangle$ denotes the canonical weighted edge-list encoding above.

**Edge records.** An edge $e = \{u, v\}$ with $u < v$ has two adjacency-list positions $(u, i)$ and $(v, j)$. We store an edge as its *record* $(u, v, w(e), i, j)$, consisting of its endpoints, its weight and both positions. The encoding of $|T\rangle$ above uses only the projection $(u, v, w(e))$ of each record.

**Definition 2.4** (Unique cut and cycle of a spanning tree)**.** For a spanning tree $T$ and an edge $e \in T$, let $\mathrm{Cut}(T - e)$ be the cut created by removing $e$. For an edge $f \notin T$, let $\mathrm{Cycle}(T + f)$ be the unique cycle created by adding $f$ to $T$.

**Definition 2.5** (Graph deletion, scaled addition and rescaling)**.** Let $G = (V, E_G, w_G)$ and $K = (V, E_K, w_K)$ be weighted graphs on the same vertex set. We define

$$G \setminus K := \big(V, E_G \setminus E_K, w_G|_{E_G \setminus E_K}\big),$$

that is, $G \setminus K$ is obtained from $G$ by deleting the edges in $E_G \cap E_K$.
For $c \geq 0$, define $G + cK := (V, E_G \cup E_K, w_{G+cK})$, where

$$w_{G+cK}(e) = \begin{cases} w_G(e), & e \in E_G \setminus E_K, \\ c \cdot w_K(e), & e \in E_K \setminus E_G, \\ w_G(e) + c \cdot w_K(e), & e \in E_G \cap E_K. \end{cases}$$

For $c \geq 0$ and $A \subseteq E$, we define $G_{A,c} := (V, E, w_{G_{A,c}})$, where

$$w_{G_{A,c}}(e) = \begin{cases} w_G(e), & e \notin A, \\ c \cdot w_G(e), & e \in A. \end{cases}$$

## 2.3 Leverage Scores and Spectral Sparsifiers

The following notions apply to any weighted graph, connected or not. We use them for graphs such as the one obtained by deleting a spanning tree from $G$, which may be disconnected.

**Definition 2.6** (Laplacian)**.** Let $G = (V, E, w)$ be a weighted graph, and let $\chi_u \in \mathbb{R}^V$ be the $u$-th standard basis vector. The Laplacian of $G$ is

$$L_G = \sum_{\{u,v\} \in E} w(u, v)\,(\chi_u - \chi_v)(\chi_u - \chi_v)^\top.$$

Equivalently, $(L_G)_{uu} = \sum_{e \in E:\, u \in e} w(e)$, $(L_G)_{uv} = -w(u, v)$ if $\{u, v\} \in E$, and $(L_G)_{uv} = 0$ otherwise.

**Fact 2.7** (Laplacians are PSD)**.** *The Laplacian $L_G$ is positive semidefinite.*

**Definition 2.8** (Spectral sparsifier)**.** Let $G = (V, E, w)$ be a weighted graph. A re-weighted subgraph of $G$ is a graph $\widetilde{G} = (V, \widetilde{E}, \widetilde{w})$ with $\widetilde{E} \subseteq E$ and arbitrary non-negative weights $\widetilde{w}$. For $\epsilon > 0$, it is an $\epsilon$-spectral sparsifier of $G$ if for every $x \in \mathbb{R}^V$,

$$\left| x^\top L_{\widetilde{G}}\, x - x^\top L_G\, x \right| \leq \epsilon \cdot x^\top L_G\, x.$$

The groundbreaking work [SS11] showed that graphs can be sparsified efficiently by sampling $\widetilde{O}(n)$ edges with probability proportional to their leverage scores, the product of an edge's weight and its effective resistance. We recall both notions.

**Definition 2.9** (Effective resistance and leverage score)**.** Let $G = (V, E, w)$ be a weighted graph. The effective resistance of $u, v \in V$ is

$$R^G_{\mathrm{eff}}(u, v) = (\chi_u - \chi_v)^\top L_G^+ (\chi_u - \chi_v) = \left\| (L_G^+)^{1/2} (\chi_u - \chi_v) \right\|^2,$$

where $L_G^+$ is the Moore–Penrose inverse of $L_G$. For an edge $e = \{u, v\} \in E$ we write $R^G_{\mathrm{eff}}(e) := R^G_{\mathrm{eff}}(u, v)$, and the leverage score of $e$ is $\ell_G(e) := w(e)\, R^G_{\mathrm{eff}}(e)$.

The leverage scores are exactly the edge marginals of the spanning tree distribution.

**Lemma 2.10** (Spanning tree marginals [LP17b, Section 4.2])**.** *Let $G = (V, E, w)$ be a connected weighted graph. For every $e \in E$, $\Pr_{T \sim \pi_G}[e \in T] = \ell_G(e)$.*

**Fact 2.11** (Foster's theorem [Fos49])**.** *For every weighted graph $G = (V, E, w)$, we have $\sum_{e \in E} \ell_G(e) = n - c$, where $c$ is the number of connected components of the subgraph of positive-weight edges. In particular, $\sum_{e \in E} \ell_G(e) \le n - 1$, with equality iff $G$ is connected.*

We usually do not have exact leverage scores, so our algorithms use the following approximations.

**Definition 2.12** (Marginal and effective-resistance overestimates)**.** Let $G = (V, E, w)$ be a weighted graph and $\lambda \ge 1$ a constant. A function $\widetilde{R}^G_{\mathrm{eff}} : V \times V \to \mathbb{R}_{\ge 0}$ is a $\lambda$-effective-resistance overestimate if $R^G_{\mathrm{eff}}(u, v) \le \widetilde{R}^G_{\mathrm{eff}}(u, v) \le \lambda \cdot R^G_{\mathrm{eff}}(u, v)$ for all $u, v \in V$. A function $\widetilde{p}_G : E \to \mathbb{R}_{\ge 0}$ is a $\lambda$-marginal overestimate if $\widetilde{p}_G(e) \ge \ell_G(e)$ for every $e \in E$ and $\sum_{e \in E} \widetilde{p}_G(e) \le \lambda(n - 1)$. For connected $G$, $\ell_G(e)$ is the marginal of $e$ under $\pi_G$ (Lemma 2.10), hence the name.

**Lemma 2.13.** *Let $G = (V, E, w)$ be a weighted graph and $\lambda \ge 1$. If $\widetilde{R}^G_{\mathrm{eff}}$ is a $\lambda$-effective-resistance overestimate, then $\widetilde{p}_G(e) := w(e)\, \widetilde{R}^G_{\mathrm{eff}}(e)$ is a $\lambda$-marginal overestimate.*

*Proof.* For every $e \in E$, $\widetilde{p}_G(e) \ge w(e)\, R^G_{\mathrm{eff}}(e) = \ell_G(e)$. Foster's theorem (Fact 2.11) implies that, $\sum_{e \in E} \widetilde{p}_G(e) \le \lambda \sum_{e \in E} \ell_G(e) \le \lambda(n - 1)$. □

The converse does not hold. A $\lambda$-marginal overestimate only needs to be a pointwise upper bound with small total mass, so it can exceed $\ell_G(e)$ by a large factor on individual edges, and it does not give pointwise resistance estimates.

Apers and de Wolf [AW20] give a quantum speedup for sparsification over the classical algorithm of Spielman and Srivastava [SS11].

**Theorem 2.14** (Quantum speedup for sparsification, adapted from [AW20])**.** *There is a quantum algorithm that, given query access to the adjacency-list oracle $\mathcal{O}_G$ of a weighted graph $G = (V, E, w)$ with $n$ vertices and $m$ edges, and parameters $\epsilon \ge \sqrt{n/m}$ and $\eta > 0$, outputs with probability at least $1 - \eta$ an explicit description of an $\epsilon$-spectral sparsifier of $G$ with $\widetilde{O}(n/\epsilon^2)$ edges. It uses $\widetilde{O}(\sqrt{mn} \cdot \log(1/\eta)/\epsilon)$ queries to $\mathcal{O}_G$ and runs in the same time.*

A sparsifier yields effective-resistance overestimates as follows.

**Theorem 2.15** (Effective-resistance overestimates from sparsifiers, adapted from [AW20; SS11])**.** *Let $G = (V, E, w)$ be a weighted graph with $n$ vertices and $m$ edges, let $\epsilon \in (0, 1/3)$, and suppose we are given an $\epsilon$-spectral sparsifier $\widetilde{G}$ of $G$. Then we can prepare an efficiently implementable oracle $O^G_{\mathcal{R}}$ such that, for all $u, v \in V$,*

$$O^G_{\mathcal{R}} |u\rangle |v\rangle |0\rangle = |u\rangle |v\rangle |\widetilde{R}^G_{\mathrm{eff}}(u, v)\rangle,$$

*where $\widetilde{R}^G_{\text{eff}}$ is a $(1+3\epsilon)$-effective-resistance overestimate. Preparing $O^G_{\mathcal{R}}$ takes $\widetilde{O}(n/\epsilon^4)$ time, and afterwards each query to it takes $\widetilde{O}(1/\epsilon^2)$ time and no queries to $\mathcal{O}_G$.*

We apply this theorem with constant $\epsilon$. The oracle $O^G_{\mathcal{R}}$ gives all values $\widetilde{R}^G_{\text{eff}}(u,v)$ without further queries to $\mathcal{O}_G$, computing the marginal overestimates $\widetilde{p}_G(e) = w(e)\,\widetilde{R}^G_{\text{eff}}(e)$, which are $\lambda$-marginal overestimates by Lemma 2.13, still costs one query to $\mathcal{O}_G$ per edge, to read $w(e)$. This is expected, since otherwise one could sometimes identify all edges of $G$ with nonzero marginals using only $\widetilde{O}(\sqrt{mn})$ queries.

## 2.4 Markov Chains and Quantum Walks

For probability distributions $p_1, p_2$ on a finite set $\Delta$, the total variation distance between them is $d_{\text{TV}}(p_1,p_2) := \frac{1}{2}\sum_{x\in\Delta}|p_1(x)-p_2(x)|$. For quantum states, $\|\cdot\|_2$ denotes the Euclidean norm.

**Reversible Markov chains and spectral gaps.** Let $P$ be the transition matrix of a reversible, irreducible, and aperiodic Markov chain on $\Delta$, with stationary distribution $\pi$. We write $D_\pi := \text{diag}(\pi)$ and define the discriminant matrix of $P$ by $D(P) := D_\pi^{1/2} P D_\pi^{-1/2}$. The matrix $D(P)$ is symmetric and has the same eigenvalues as $P$. We write these eigenvalues as $1 = \theta_0, \theta_1, \ldots, \theta_{|\Delta|-1}$, ordered so that $|\theta_1| \geq \cdots \geq |\theta_{|\Delta|-1}|$. The absolute spectral gap is $\gamma_{\text{abs}}(P) := 1 - |\theta_1|$. If $P$ has nonnegative spectrum, equivalently if $P$ is positive semidefinite as an operator on $L^2(\pi)$, then this equals the usual spectral gap $\gamma(P) := 1 - \max_{j\geq 1}\theta_j$.

**Labelled Szegedy walks.** We use a labelled-transition formulation of Szegedy walks, in the convention used for quantum simulated annealing [WA08]. Szegedy's original construction gives the spectral mapping theorem for reversible Markov chains [Sze04]; see also [MNRS11]. The standard equivalence between coined and Szegedy walks, including the use of a coin register and flip-flop shift, which maps a transition to the same transition viewed in reverse, is discussed in [Won17a; Won17b; PS17].

Define $\mathcal{K} := \text{span}\{|x\rangle|a\rangle : x \in \Delta,\ a \in C_x \cup \{0\}\}$ as the Hilbert space over which the quantum walk operates. Here, $C_x$ is a finite set of transition labels for each $x \in \Delta$ and 0 is a distinguished blank transition label. In our application, $x$ is a spanning tree and the transition labels will be pairs of edges.

Each label $a \in C_x$ has endpoint $h_x(a) \in \Delta$ and the probability of making a transition from $x$ along the label $a$ is $\Pr[a \mid x]$ . We require $P(x,y) = \sum_{a\in C_x : h_x(a)=y}\Pr[a \mid x]$. We assume that non-self-loop transitions are uniquely labelled: for every $x \neq y$ with $P(x,y) > 0$, there is a unique label $a_{x,y} \in C_x$ with endpoint $y$. Self-loop labels have endpoint $x$ and total probability $P(x,x)$.

We also assume that the labelled transitions admit an involution $g$ such that, if $g(x,a) = (y,a')$, then $y = h_x(a)$ and $h_y(a') = x$. Self-loop labels are fixed by $g$.

Let $\mathcal{U}$ be a transition-preparation unitary satisfying

$$\mathcal{U}|x\rangle|0\rangle = |x\rangle \sum_{a\in C_x} \sqrt{\Pr[a \mid x]}|a\rangle.$$

Let $\mathcal{S}$ be the flip-flop shift, defined by $\mathcal{S}|x\rangle|a\rangle := |g(x,a)\rangle = |h_x(a)\rangle|a'\rangle$. Let $\mathcal{A} := \text{span}\{|x\rangle|0\rangle : x \in \Delta\}$ and $\mathcal{B} := \mathcal{U}^\dagger\mathcal{S}\mathcal{U}\mathcal{A}$ be two subspaces with corresponding orthogonal projectors $\Pi_\mathcal{A}$ and $\Pi_\mathcal{B}$. Let $R_\mathcal{A} := 2\Pi_\mathcal{A} - I$ and $R_\mathcal{B} := 2\Pi_\mathcal{B} - I$; the quantum walk operator is

$$W(P) := R_\mathcal{B} R_\mathcal{A} = \left(\mathcal{U}^\dagger\mathcal{S}\mathcal{U} R_\mathcal{A} \mathcal{U}^\dagger\mathcal{S}\mathcal{U}\right) R_\mathcal{A}.$$

We call $\mathcal{A}+\mathcal{B}$ the busy subspace and denote by $\gamma_{\mathrm{ph}}(W)$ the smallest nonzero angular distance from 1 among the eigenvalues of $W$ on this subspace. For a distribution $\pi$ on $\Delta$, let $|\pi\rangle := \sum_{x\in\Delta}\sqrt{\pi(x)}\,|x\rangle$ denote its q-sample.

**Theorem 2.16** (Spectral mapping for labelled Szegedy walks)**.** *Let $P$ be the transition matrix of a reversible, irreducible, and aperiodic Markov chain on $\Delta$, with $\pi$ as the stationary distribution. Then, on the busy subspace, the unique 1-eigenvector of $W(P)$ is $|\pi\rangle|0\rangle$. Moreover,*

$$\gamma_{\mathrm{ph}}(W(P)) = \Omega(\sqrt{\gamma_{\mathrm{abs}}(P)}).$$

The spectral mapping for coined and Szegedy walks is standard; see, e.g., [Won17b; Won17a]. We restate and prove the result in our labelled-transition setting because our application represents neighbouring spanning trees by transition labels (pairs of edges), rather than by storing the entire neighbouring tree in the second register. This formulation avoids the costly coherent construction of a neighbouring spanning tree. The proof is deferred to Section B.

Throughout the paper, whenever we refer to the quantum walk operator associated with a reversible Markov chain with transition matrix $P$, we mean the labelled Szegedy walk operator $W(P)$ constructed above, for the relevant labelled-transition realization of $P$. We abuse notation and write $|\pi\rangle$ for $|\pi\rangle|0\rangle$.

## 2.5 Up-Down walk over Spanning trees

We recall the up-down walk on spanning trees, which is reversible with respect to the weighted spanning tree distribution and will underlie our sampling procedure.

Given a current tree $T_i$, one step adds a uniformly random non-tree edge and then removes an edge from the resulting cycle according to the target distribution.

**Markov Chain 1** Up-down walk from $T_0$

1: **for** $i = 0, 1, 2, \ldots$ **do**
2: Choose $f \notin T_i$ uniformly at random.
3: Choose $e \in \mathrm{Cycle}(T_i + f)$ with probability $\Pr(e) \propto \pi_G(T_i + f - e)$.
4: Set $T_{i+1} = T_i + f - e$.
5: **end for**

**Fact 2.17** (Up-Down Walk mixes to $\pi_G$, [ALGV19])**.** *The up-down walk is reversible with respect to the weighted spanning tree distribution $\pi_G$, and hence $\pi_G$ is its stationary distribution.*

We say that a spanning tree distribution is in isotropic position when all edge marginals are approximately equal. This discrete notion of isotropic position was introduced by Anari and Dereziński [AD20], in analogy with isotropic position for continuous log-concave distributions.

**Definition 2.18** (Isotropic Distributions)**.** Suppose $G(V, E, w)$ is a connected, weighted graph with the spanning tree distribution $\pi_G$. We say that the distribution $\pi_G$ is isotropic if all the marginals of edges are equal to each other i.e.,

$$\Pr_{T\sim\pi_G}[e \in T] = \frac{(n-1)}{m} \text{ for all } e \in E.$$

Moreover, we call $\pi_G$ nearly isotropic if for all $e \in E$, its marginal satisfies the following,

$$\Pr_{T\sim\pi_G}[e \in T] \leq \frac{c(n-1)}{m} \text{ for some constant } c > 1.$$

The up-down walk has strong mixing guarantees: [ALGV19] proved a general spectral-gap lower bound, and [ALV22] gave an improved log-Sobolev bound when the spanning tree distribution is nearly isotropic. We use the standard fact that the log-Sobolev constant of a reversible Markov chain is bounded by twice its spectral gap [CSV23], so log-Sobolev lower bounds imply spectral-gap lower bounds up to constants.

**Theorem 2.19** (Mixing guarantees for the up-down walk, adapted from [ALGV19; ALV22]). *Let $G = (V, E, w)$ be a connected weighted graph with $n$ vertices, $m$ edges, and non-negative edge weights. Let $\pi_G$ be the spanning tree distribution of $G$, and let $\gamma_{\mathrm{ud}}$ denote the spectral gap of the up-down walk on the set $\Delta_G$ of spanning trees of $G$. Then the up-down walk satisfies $\gamma_{\mathrm{ud}} \geq \Omega(1/m)$. Moreover, if $m \geq 100n$ and $\pi_G$ is nearly-isotropic, then the up-down walk satisfies the improved bound $\gamma_{\mathrm{ud}} \geq \Omega\big(1/n \log^2 n\big)$.*

Anari and Dereziński [AD20] introduced a subdivision procedure that uses marginal overestimates to transform a graph into a multigraph whose spanning tree distribution is nearly isotropic. The procedure replaces each edge by a number of parallel copies proportional to its marginal overestimate and splits the original weight evenly among those copies.

**Definition 2.20** (Isotropic multiplicity). Let $G = (V, E, w)$ be a weighted graph with spanning-tree distribution $\pi_G$, and let $\widetilde{p}_G$ be a $\lambda$-marginal overestimate for $\pi_G$. For each $e \in E$, define the isotropic multiplicity of $e$ induced by $\widetilde{p}_G$ as

$$t_G(e) := \left\lceil \frac{m}{\lambda(n-1)} \widetilde{p}_G(e) \right\rceil .$$

**Proposition 2.21** (Isotropic Transformation; [ALV22]). *Let $G = (V, E, w)$ be a weighted graph with $n$ vertices and $m$ edges, and let $\pi_G$ be its spanning tree distribution. Let $\widetilde{p}_G : E \to \mathbb{R}_{\geq 0}$ be a $\lambda$-marginal overestimate for $\pi_G$ (Definition 2.12). The isotropic transform of $G$ with respect to $\widetilde{p}_G$ is the multigraph $G' = (V, E', w')$ obtained by replacing each edge $e \in E$ by $t_G(e)$ parallel copies $e^{(1)}, \ldots, e^{(t_G(e))}$, each with weight $w(e)/t_G(e)$. Let $\pi_{G'}$ denote the spanning-tree distribution of $G'$.*

*If $T' = \{e_1^{(j_1)}, \ldots, e_{n-1}^{(j_{n-1})}\}$ is a spanning tree in $G'$ whose edges are copies of a spanning tree $T = \{e_1, \ldots, e_{n-1}\}$ in $G$, then $\pi_{G'}(T') = \pi_G(T)/\prod_{e \in T} t_G(e)$. Equivalently, sampling from $\pi_{G'}$ can be done by first sampling $T \sim \pi_G$ and then choosing, independently for each $e \in T$, one of its $t_G(e)$ copies uniformly at random.*

*Moreover, the transformed distribution is nearly isotropic (Definition 2.18) and the number of edges in $G'$ satisfies $|E'| \leq 2m$.*

The transformed multigraph viewpoint is useful for proving near-isotropy, but one need not explicitly construct $G'$ in order to implement the corresponding dynamics. Instead, we will present a way in Section 3.3 to work directly on the original graph and bias the choice of the added edge according to the number of copies it has in $G'$.

## 2.6 Simulated Annealing

We recall the standard simulated-annealing framework involving classical Hamiltonians. Let $\Delta$ be a finite state space, let $\mu : \Delta \to \mathbb{R}_{\geq 0}$ be a base weight function, and let $H : \Delta \to \{0, 1, \ldots, n\}$ be a classical Hamiltonian of degree $n$. We define the partition function at inverse temperature $\beta \geq 0$ as

$$Z(\beta) = \sum_{x \in \Delta} e^{-\beta H(x)} \cdot \sigma(x).$$

Such a partition function corresponds to the normalization factor of the Gibbs distribution at inverse temperature $\beta$ given by $\pi_\beta(x) = \frac{1}{Z(\beta)} e^{-\beta H(x)} \cdot \sigma(x)$. For each inverse temperature $\beta$, we write $|\pi_\beta\rangle := \sum_{x \in \Delta} \sqrt{\pi_\beta(x)} |x\rangle$ for the q-sample associated with the Gibbs distribution $\pi_\beta$.

A cooling schedule is a sequence of inverse temperatures $\beta_0 < \cdots < \beta_l$. We study mainly the following type of cooling schedules.

**Definition 2.22** ($B$-SLOWLY VARYING SCHEDULE)**.** For any partition function $Z(\beta)$ which is log-convex and $B > 1$, a cooling schedule $\beta_0 < \cdots < \beta_l$ is called $B$-slowly varying if

$$\left|\langle \pi_{\beta_i} \,|\, \pi_{\beta_{i+1}} \rangle\right|^2 \geq 1/B.$$

This is a *warm-start* property between $\pi_{\beta_i}$ and $\pi_{\beta_{i+1}}$ and ensures that we can *anneal* i.e., map, $|\pi_{\beta_i}\rangle$ to $|\pi_{\beta_{i+1}}\rangle$ with less effort.

When the partition function is log-convex, the existence of short slowly varying schedules was first shown by Štefankovič, Vempala, and Vigoda [ŠVV09]. We use the following form stated by Harrow and Wei [HW20]. However, it should be noted that the partition functions considered in the above works do not include the additional weight pre-factor $\mu$. Nevertheless, the same results extend to this setting with only minor modifications, since $\sigma(x)$ is independent of $\beta$ and depends only on the state $x$.

**Theorem 2.23** (QUANTUM ALGORITHM FOR COMPUTING A SLOWLY-VARYING SCHEDULE, ADAPTED FROM [HW20])**.** *Let $\{\pi_\beta\}_{\beta\in[0,\beta^\star]}$ be a family of Gibbs distributions with log-convex partition function $Z(\beta)$. There exists an $e^2$-slowly varying schedule $0 = \beta_0 < \cdots < \beta_l = \beta^\star$ of length $l \leq \widetilde{O}\left(\sqrt{\ln \frac{Z(0)}{Z(\beta^\star)}}\right)$.*

*Moreover, if a state $|\widetilde{\pi}_0\rangle$ satisfying $\||\widetilde{\pi}_0\rangle - |\pi_0\rangle\|_2 \leq \epsilon$ can be prepared and, for every $\beta \in [0, \beta^\star]$, there is a reversible Markov chain $\mathcal{M}_\beta$ with stationary distribution $\pi_\beta$ and absolute spectral gap at least $\gamma$, together with an implementable quantum walk operator $W_\beta$, then for any $\eta > 0$, a quantum algorithm constructs such a schedule with probability at least $1 - \eta$ using*

$$\widetilde{O}\left(\frac{l}{\sqrt{\gamma}} \log\left(\frac{1}{\eta}\right) \log\left(\frac{1}{\epsilon}\right)\right)$$

*applications of $W_\beta$ and $W_\beta^{-1}$ at adaptively chosen values of $\beta$.*

# 3 Main Result

In this section, we prove our main result, Theorem 3.1. For any $\rho \in [n/m, 1]$, after a preprocessing step using $\widetilde{O}(\sqrt{mn/\rho})$ queries to the adjacency-list oracle, our algorithm prepares the q-sample $|\pi\rangle$ of the spanning tree distribution $\pi$ using $\widetilde{O}(\sqrt{mn\rho})$ queries.

**Theorem 3.1** (QUANTUM ALGORITHM FOR Q-SAMPLE OF SPANNING TREE DISTRIBUTION)**.** *Let $G = (V, E, w)$ be a connected weighted graph with $n$ vertices and $m \geq 100n$ edges, and let $\pi$ be its spanning tree distribution. Let $\rho \in [n/m, 1]$. For any $\epsilon, \eta > 0$, there is a quantum algorithm which, with probability at least $1 - \eta$, outputs a state $|\widetilde{\pi}\rangle$ satisfying $\||\widetilde{\pi}\rangle - |\pi\rangle\|_2 \leq \epsilon$. The algorithm has the following query costs with respect to the adjacency-list oracle $\mathcal{O}_G$:*

1. *preprocessing cost: $\widetilde{O}\left(\sqrt{mn/\rho} \cdot \log(1/\eta)\right)$ queries;*
2. *reflection cost around $|\pi\rangle$: $\widetilde{O}\left(\sqrt{m\rho}\right)$ queries;*
3. *q-sampling cost: $\widetilde{O}\left(\sqrt{mn\rho} \cdot \log(1/\epsilon)\right)$ queries.*

*In particular, preparing one q-sample uses $\widetilde{O}\left(\sqrt{mn\rho} \cdot \log(1/\epsilon)\right)$ queries to $\mathcal{O}_G$ after the successful completion of the preprocessing.*

Theorem 3.1 exposes a tradeoff, controlled by $\rho$, between the preprocessing cost and the cost of each subsequent q-sample. We instantiate it in two ways, which give the two statements in the informal version of the theorem in the introduction. Throughout, we suppress the dependence on $\epsilon$ and $\eta$ inside $\widetilde{O}(\cdot)$.

**Preparing $k$ q-samples.** Choosing $\rho = 1/k$ for $k \in [1, m/n]$, we obtain that the preprocessing cost is $\widetilde{O}(\sqrt{kmn})$ and the cost of q-sampling is $\widetilde{O}(\sqrt{mn/k})$. Thus for $k$ independent q-samples, we obtain a total query cost of $\widetilde{O}(\sqrt{kmn}) + k \cdot \widetilde{O}(\sqrt{mn/k}) = \widetilde{O}(\sqrt{kmn})$.

**Cheaper q-samples after preprocessing.** Alternatively, for $\delta \in [0, 1]$, we choose the parameter $\rho = \min\{1, n\, m^{2\delta-1}\}$, which lies in $[n/m, 1]$ since $m \geq 100n$ to obtain the following. If $n\, m^{2\delta-1} \leq 1$, the preprocessing cost is $\widetilde{O}(\sqrt{mn/\rho}) = \widetilde{O}(m^{1-\delta})$ and each q-sample costs $\widetilde{O}(\sqrt{mn\rho}) = \widetilde{O}(n\, m^{\delta})$. Otherwise, $\rho = 1$ and both costs are $\widetilde{O}(\sqrt{mn})$, which is at most $\widetilde{O}(n\, m^{\delta})$. In either case, the preprocessing cost is $\widetilde{O}(\sqrt{mn} + m^{1-\delta})$, and since $m \leq n^2$, each q-sample costs $\widetilde{O}(n\, m^{\delta}) \leq \widetilde{O}(n^{1+2\delta})$.

Before proving the theorem, we formalize the cooling schedule described in the introduction.

**Definition 3.2** (Admissible spanning tree, [AD20])**.** For a connected weighted graph $G$, a spanning tree $S \in \Delta_G$ is *admissible* if $\pi_G(S) \geq 1/(2n^{n-2})$.

**Definition 3.3** ($\beta$-scaled graph and spanning tree distribution)**.** Fix an admissible spanning tree $S$. For $\beta \in \mathbb{R}$, let $G_\beta := G_{E\setminus S,\, e^{-\beta}}$ be the graph with weight function

$$w_\beta^S(e) = \begin{cases} e^{-\beta} \cdot w_G(e), & \text{if } e \notin S, \\ w_G(e), & \text{if } e \in S, \end{cases}$$

and let $\pi^S_{G_\beta}$ be its spanning tree distribution. All $G_\beta$ have the same edge set, hence the same set of spanning trees $\Delta_G$, and for every $T \in \Delta_G$,

$$\pi^S_{G_\beta}(T) \propto \prod_{e \in T} w_\beta^S(e) = e^{-\beta |T \setminus S|} \cdot w_G(T).$$

Since $S$ is fixed throughout, we omit it from the notation and index all quantities of $G_\beta^S$ by $\beta$ alone, e.g. $\pi_\beta$, $w_\beta$, $\ell_\beta$ and $R^\beta_{\text{eff}}$. At $\beta = 0$ we have $G_0 = G$, so $\pi_0 = \pi$, and as $\beta \to \infty$ only $S$ survives, so $\pi_\beta \to \mathbb{1}_S$.

We now describe the four intermediate results used in the proof of Theorem 3.1. The first one gives reusable effective-resistance overestimates for any sequence of graphs obtained by rescaling the same set of edges.

**Proposition 3.6** (Reusable Effective-resistance data structure)**.** *Let $G = (V, E, w)$ be a weighted graph with $n$ vertices and $m$ edges, and let $F \subseteq E$ be a fixed subset of edges stored explicitly as edge records. For any $\eta > 0$, there is a quantum algorithm that, with probability at least $1 - \eta$, uses $\widetilde{O}(\sqrt{mn} \cdot \log(1/\eta))$ queries to the adjacency-list oracle $\mathcal{O}_G$ to construct a data structure $\mathcal{H}$ for the family of rescaled graphs $\{G_{E\setminus F,c} : c \geq 0\}$.*

*For any $c \geq 0$, $\mathcal{H}$ can be used to construct an oracle $O_{\mathcal{R}}^{G_{E\setminus F,c}}$ giving access to 2-effective-resistance overestimates in $G_{E\setminus F,c}$. Constructing this oracle, as well as querying it, requires no further queries to $\mathcal{O}_G$.*

The data structure $\mathcal{H}$ is the only place where the preprocessing looks at the whole graph, and it is built once for the entire family $\{G_{E\setminus F,c}\}_{c\geq 0}$. This is what makes a long cooling schedule affordable. All graphs on the schedule differ from $G$ only by rescaling the edges outside a fixed

spanning tree, so their effective resistances can all be overestimated from the same sparsification data. The length of the schedule therefore affects only the time complexity, not the query complexity.

To implement the quantum walk operator $W_\beta$ corresponding to the marginal-aware walk, we need to implement its Up step. This step samples an edge $e$ with probability proportional to the isotropic multiplicity $t_G(e)$, and hence requires preparation of the corresponding quantum state:

**Definition 3.4** (Marginal state)**.** Let $G = (V, E, w)$ and fix a $\lambda$-marginal overestimate $\widetilde{p}_G$, for some constant $\lambda \geq 1$, inducing isotropic multiplicities $t_G(e)$ as in Definition 2.20. The corresponding marginal state is

$$|\mu_G\rangle := \frac{1}{\sqrt{\sum_{e\in E} t_G(e)}} \sum_{e\in E} \sqrt{t_G(e)}\,|e\rangle.$$

**Proposition 3.11** (Reusable marginal data structure)**.** *Consider any weighted graph $G = (V, E, w)$ with $n$ vertices and $m$ edges, and let $F \subseteq E$ be a fixed subset of edges stored explicitly as edge records. For any $\rho \in [n/m, 1]$ and $\eta > 0$, there is a quantum algorithm that constructs a data structure $\mathcal{L}_\rho$ providing coherent query access to a set of edges, stored as edge records, such that every edge $e \in E$ outside this set has marginal less than $\rho$ in every graph in the family $\{G_{E\setminus F,c} : 0 < c \leq 1\}$. The algorithm succeeds with probability at least $1 - \eta$ and uses $\widetilde{O}(\sqrt{mn/\rho} \cdot \log(1/\eta))$ queries to the adjacency-list oracle $\mathcal{O}_G$.*

*Moreover, after constructing $\mathcal{L}_\rho$, for any $0 < c \leq 1$ it can be used to prepare the marginal state $|\mu_{G_{E\setminus F,c}}\rangle$ corresponding to the 2-marginal overestimate $\widetilde{p}_{G_{E\setminus F,c}}(e) := w_{G_{E\setminus F,c}}(e)\widetilde{R}_{\mathrm{eff}}^{G_{E\setminus F,c}}(e)$ (Definition 3.4), using $\widetilde{O}(\sqrt{m\rho/n})$ queries to $\mathcal{O}_G$, given coherent query access to 2-effective-resistance overestimates $\widetilde{R}_{\mathrm{eff}}^{G_{E\setminus F,c}}$.*

We now have, for every temperature on the schedule, coherent access to effective-resistance overestimates and a way to prepare the marginal state $|\mu_G\rangle$ at cost $\widetilde{O}(\sqrt{m\rho/n})$. Crucially, the stored set of heavy edges is *static*. It is computed once and is valid for every graph in the family $\{G_{E\setminus F,c} : 0 < c \leq 1\}$, so it does not need to be recomputed as $\beta$ changes. These two ingredients are exactly what is needed to implement a marginal-aware up-down walk on spanning trees directly on the original graph, without ever explicitly rescaling or isotropizing it. The next proposition records the resulting quantum walk operator.

**Proposition 3.16** (Quantum walk operator with large gap)**.** *Let $G = (V, E, w)$ be any weighted graph with $n$ vertices and $m \geq 100n$ edges, and let $\Delta$ denote the set of spanning trees of $G$. There exists a quantum walk operator $W$, associated with a reversible Markov chain on $\Delta$, whose unique 1-eigenvector is $|\pi\rangle$, where $\pi$ is the spanning-tree distribution of $G$, and whose phase gap is at least $\widetilde{\Omega}(1/\sqrt{n})$.*

*Moreover, for any $\rho \in [n/m, 1]$, one application of $W$ and its inverse can be implemented using $\widetilde{O}(\sqrt{m\rho/n})$ queries to $\mathcal{O}_G$, given:*

- *coherent query access to 2-effective-resistance overestimates for all edges of $G$;*
- *the degree of every vertex of $G$; and*
- *a stored set of edges such that every edge outside this set has marginal under $\pi$ less than $\rho$, with each stored edge given as an edge record.*

The quantum walk operators serve both to construct the cooling schedule and to move a q-sample between consecutive temperatures. Their $\widetilde{\Omega}(1/\sqrt{n})$ phase gap, quadratically larger than

the $\widetilde{\Omega}(1/n)$ spectral gap of the underlying chain, reduces each transition to $\widetilde{O}(\sqrt{n})$ walk steps. It remains to bound the number of transitions: following Harrow and Wei [HW20], the schedule can be constructed from the operators $W_\beta$ themselves and has length $\widetilde{O}(\sqrt{n})$. Thus its cost is measured in walk-operator applications, each requiring $\widetilde{O}(\sqrt{m\rho/n})$ queries.

**Proposition 3.21** (Computing a slowly-varying schedule)**.** *Let $G = (V, E, w)$ be a connected weighted graph with $n$ vertices and $m$ edges. Fix an admissible spanning tree $S$ and $\epsilon > 0$, and define $\beta^\star := \ln(16n^{n-2}/\epsilon^2)$. Let $\{G_\beta\}_{\beta\in[0,\beta^\star]}$ be the $\beta$-scaled graphs from Definition 3.3, with spanning tree distributions $\{\pi_\beta\}_{\beta\in[0,\beta^\star]}$.*

*Suppose the admissible spanning tree $S$ is stored explicitly together with its edge weights. Then there is a quantum algorithm that, for any $\eta > 0$, computes an $e^2$-slowly varying schedule $0 = \beta_0 < \cdots < \beta_l = \beta^\star$ of length $l \le \widetilde{O}(\sqrt{n})$ with $\beta_0 = 0$, $\beta_l = \beta^\star$ with probability at least $1 - \eta$. The algorithm uses $\widetilde{O}(n \cdot \log(1/\eta)\log(1/\epsilon))$ applications of the operators $W_\beta$ obtained by applying Proposition 3.16 to $G_\beta$.*

Combining the four ingredients gives Algorithm 1. The preprocessing phase computes an admissible tree and all reusable data, and then finds the schedule. The q-sampling phase starts from the point mass on the tree and anneals down the schedule to $\beta_0 = 0$.

**Algorithm 1** Quantum Sampling of Random Spanning Trees

**Input:** $G = (V, E, w)$ with $n = |V|$ and $m = |E|$, parameter $\rho \in [n/m, 1]$, accuracy $\epsilon$, and failure probability $\eta$.

**Output:** A state $|\widetilde{\pi}\rangle$ satisfying $\||\widetilde{\pi}\rangle - |\pi\rangle\|_2 \le \epsilon$.

Preprocessing

Compute $S$, a maximum weight-product spanning tree of $G$ (Proposition 3.18, [AGJL25]).
Compute $\mathcal{H}$, a reusable effective-resistance data structure for the family $\{G_{E\setminus S, e^{-\beta}} : \beta \ge 0\}$ (Proposition 3.6, [AW20]).
Compute $\mathcal{L}_\rho$, the reusable marginal data structure for the family $\{G_{E\setminus S, e^{-\beta}} : \beta \ge 0\}$ (Proposition 3.11).
Set $\beta^\star \leftarrow \ln(16n^{n-2}/\epsilon^2)$.
Compute an $e^2$-slowly-varying schedule $0 = \beta_0 < \cdots < \beta_l = \beta^\star$ of length $l = \widetilde{O}(\sqrt{n})$ (Proposition 3.21, [HW20]).
**for** $j = l, l-1, \ldots, 0$ **do**
    Construct from $\mathcal{H}$ an oracle $O_{\mathcal{R}}^{G_{\beta_j}}$ giving query access to 2-effective-resistance overestimates in $G_{\beta_j} = G_{E\setminus S, e^{-\beta_j}}$. (Proposition 3.6 and theorem 2.15)
**end for**

Quantum sampling

Prepare $|S\rangle$; by Lemma 3.17, $\||S\rangle - |\pi_{\beta_l}\rangle\|_2 \le \epsilon/2$.
**for** $j = l, l-1, \ldots, 1$ **do**
    Apply $U_j$, which transforms $|\pi_{\beta_j}\rangle$ to $|\pi_{\beta_{j-1}}\rangle$ using Lemma A.3 and Proposition 3.16.
**end for**
Output the resulting state $|\widetilde{\pi}\rangle$.

We are now ready to prove Theorem 3.1 by combining the preceding propositions.

*Proof of Theorem 3.1.* We first analyze the preprocessing phase and then the q-sampling phase.

**Preprocessing: correctness and failure probability.** There are four randomized steps: computing the admissible tree, constructing $\mathcal{H}$, constructing $\mathcal{L}_\rho$, and computing the slowly-varying schedule. We allocate failure probability $\eta/4$ to each. By a union bound, all four

succeed with probability at least $1-\eta$. Below, we condition on the success of the earlier steps when analyzing a later one.

**Preprocessing: cost.** *Admissible tree.* By Proposition 3.18, with probability at least $1-\eta/4$ we obtain an admissible spanning tree $S$, given as $n-1$ adjacency-list positions $(u,i)$, using $\widetilde{O}(\sqrt{mn}\log(1/\eta))$ queries to $\mathcal{O}_G$. We convert each position into an edge record using $O(\log n)$ queries, so $\widetilde{O}(n)$ queries in total, and store $S$ as the lexicographically sorted list of records. The canonical encoding $|S\rangle$ is the projection of these records to $(\min\{u,v\},\max\{u,v\},w(u,v))$ and needs no further queries. We also read the degree of every vertex with $n$ queries. Since rescaling weights does not change the adjacency structure, this degree information is valid for every $G_\beta$. Both costs are $\widetilde{O}(n)\le\widetilde{O}(\sqrt{mn})$ because $m\ge n$.

*Effective resistances.* Apply Proposition 3.6 with $F=S$ and failure probability $\eta/4$. This yields $\mathcal{H}$ for the family $\{G_{E\setminus S,c}: c\ge 0\}$ using $\widetilde{O}(\sqrt{mn}\log(1/\eta))$ queries. For every $\beta\ge 0$, the oracle $O_{\mathcal{R}}^{G_\beta}$ for $G_\beta=G_{E\setminus S,e^{-\beta}}$ can then be built and evaluated with no further queries to $\mathcal{O}_G$.

*Heavy edges.* Apply Proposition 3.11 with $F=S$, threshold $\rho$, and failure probability $\eta/4$. This yields $\mathcal{L}_\rho$ using $\widetilde{O}(\sqrt{mn/\rho}\log(1/\eta))$ queries. It provides a stored set $\mathcal{E}_\rho$ of edges with their weights such that every edge outside $\mathcal{E}_\rho$ has marginal less than $\rho$ in every $G_{E\setminus S,c}$ with $0<c\le 1$. In particular, this holds for every $G_\beta$ with $\beta\ge 0$, since $c=e^{-\beta}\in(0,1]$.

*Slowly-varying schedule.* Set $\beta^\star=\ln(16n^{n-2}/\epsilon^2)$. Apply Proposition 3.21 with failure probability $\eta/4$; $S$ is stored explicitly with its weights, as required. It returns an $e^2$-slowly-varying schedule $0=\beta_0<\cdots<\beta_l=\beta^\star$ of length $l\le\widetilde{O}(\sqrt{n})$, using $\widetilde{O}(n\log(1/\eta)\log(1/\epsilon))$ applications of the operators $W_\beta$ from Proposition 3.16, at points $\beta$ chosen adaptively. We check that each application is available at every such $\beta$. The three hypotheses of Proposition 3.16 for $G_\beta$ are satisfied: (i) $\mathcal{H}$ yields coherent access to 2-effective-resistance overestimates for all edges of $G_\beta$; (ii) the degree table is stored; (iii) $\mathcal{E}_\rho$ is stored with their weights such that every edge outside this set has marginal less than $\rho$. Hence $W_\beta$ and $W_\beta^{-1}$ each cost $\widetilde{O}(\sqrt{m\rho/n})$ queries, and the schedule computation costs

$$\widetilde{O}\big(n\log(1/\eta)\log(1/\epsilon)\big)\cdot\widetilde{O}\big(\sqrt{m\rho/n}\big)=\widetilde{O}\big(\sqrt{mn\rho}\cdot\log(1/\eta)\log(1/\epsilon)\big).$$

Finally, constructing and storing $O_{\mathcal{R}}^{G_{\beta_j}}$ for each $j$ requires no further queries.

*Total.* Summing the four contributions, the preprocessing succeeds with probability at least $1-\eta$ and uses

$$\widetilde{O}\Big(\big(\sqrt{mn}+\sqrt{mn/\rho}+\sqrt{mn\rho}\big)\log(1/\eta)\log(1/\epsilon)\Big)=\widetilde{O}\Big(\sqrt{mn/\rho}\cdot\log(1/\eta)\log(1/\epsilon)\Big)$$

queries, since $\rho\le 1$. The dependence on $\epsilon$ enters only through $\beta^\star$ and is suppressed in $\widetilde{O}(\cdot)$, as in the statement.

**Q-sampling: correctness.** Condition on successful preprocessing. Since $S$ is already stored in canonical form, the basis state $|S\rangle$ is prepared without further queries. By Lemma 3.17 and the choice of $\beta^\star$, we have $\||S\rangle-|\pi_{\beta_l}\rangle\|_2\le\epsilon/2$.

For $j=l,\dots,1$, let $U_j$ be the unitary from Lemma A.3, implemented with the walk operators $W_{\beta_j}$ and $W_{\beta_{j-1}}$ of Proposition 3.16. It satisfies $\big\|U_j|\pi_{\beta_j}\rangle-|\pi_{\beta_{j-1}}\rangle\big\|_2\le\epsilon/(2l)$. Let $U:=U_1U_2\cdots U_l$, so that $U_l$ is applied first. Telescoping gives

$$U|\pi_{\beta_l}\rangle-|\pi_{\beta_0}\rangle=\sum_{j=1}^{l}U_1\cdots U_{j-1}\Big(U_j|\pi_{\beta_j}\rangle-|\pi_{\beta_{j-1}}\rangle\Big).$$

Since each $U_i$ is unitary, this yields

$$\|U|\pi_{\beta_l}\rangle - |\pi_{\beta_0}\rangle\|_2 \le \sum_{j=1}^{l} \left\| U_j|\pi_{\beta_j}\rangle - |\pi_{\beta_{j-1}}\rangle \right\|_2 \le l \cdot \frac{\epsilon}{2l} = \frac{\epsilon}{2}.$$

The algorithm applies $U$ to $|S\rangle$ instead of $|\pi_{\beta_l}\rangle$. Since $\beta_0 = 0$ we have $\pi_{\beta_0} = \pi$, and by unitarity of $U$ and the triangle inequality,

$$\|U|S\rangle - |\pi\rangle\|_2 \le \|U|S\rangle - U|\pi_{\beta_l}\rangle\|_2 + \|U|\pi_{\beta_l}\rangle - |\pi_{\beta_0}\rangle\|_2 \le \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.$$

Hence the output $|\widetilde{\pi}\rangle = U|S\rangle$ satisfies $\||\widetilde{\pi}\rangle - |\pi\rangle\|_2 \le \epsilon$.

**Q-sampling: cost.** Because the schedule is $e^2$-slowly varying, consecutive states have constant overlap, $\langle \pi_{\beta_j} \,|\, \pi_{\beta_{j-1}} \rangle \ge e^{-1}$. By Proposition 3.16, each walk operator has phase gap $\widetilde{\Omega}(1/\sqrt{n})$. Therefore, each $U_j$ (fixed-point search to accuracy $\epsilon/(2l)$) uses $\widetilde{O}(\sqrt{n}\log(l/\epsilon))$ applications of $W_{\beta_j}^{\pm 1}$ and $W_{\beta_{j-1}}^{\pm 1}$. Each application costs $\widetilde{O}(\sqrt{m\rho/n})$ queries. Since $l \le \widetilde{O}(\sqrt{n})$, the total is

$$l \cdot \widetilde{O}(\sqrt{n}\log(l/\epsilon)) \cdot \widetilde{O}(\sqrt{m\rho/n}) = \widetilde{O}\big(\sqrt{n} \cdot \sqrt{n} \cdot \sqrt{m\rho/n}\,\log(1/\epsilon)\big) = \widetilde{O}\big(\sqrt{mn\rho} \cdot \log(1/\epsilon)\big).$$

**Reflection.** A reflection about $|\pi\rangle = |\pi_{\beta_0}\rangle$ is implemented by phase estimation of $W_{\beta_0}$ with precision equal to its phase gap $\widetilde{\Omega}(1/\sqrt{n})$. This takes $\widetilde{O}(\sqrt{n})$ applications of $W_{\beta_0}^{\pm 1}$, each costing $\widetilde{O}(\sqrt{m\rho/n})$ queries, for a total of $\widetilde{O}(\sqrt{m\rho})$ queries.

Combining the three parts proves the theorem. □

### 3.1 Reusable Effective Resistance Data Structures from Sparsifiers

In this section we prove Proposition 3.6. Fix a weighted graph $G = (V, E, w)$ and a set $F \subseteq E$, and let $H$ denote the subgraph of $G$ with edge set $E \setminus F$. By a slight abuse of notation, we also write $F$ for the weighted subgraph of $G$ with edge set $F$. Since $F$ is known, oracle access to $H$ can be simulated using a single query to $\mathcal{O}_G$, by setting the weight of any queried edge in $F$ to zero.

The key observation is that, as $c$ varies, only the edges of $H$ are rescaled. Hence, it suffices to compute a spectral sparsifier $\widetilde{H}$ of $H$ once. For every $c > 0$, the graph $F + c \cdot \widetilde{H}$ is then a spectral sparsifier of $G_{E\setminus F,c}$. Since both $F$ and $\widetilde{H}$ are stored explicitly, this graph can be constructed for any $c \ge 0$ without further queries to $\mathcal{O}_G$. Moreover, oracle access to $F + c \cdot \widetilde{H}$ can be simulated directly from the stored representations of $F$ and $\widetilde{H}$, and therefore also requires no further queries to $\mathcal{O}_G$.

For any $c \ge 0$, we use the stored data to construct an oracle $O_{\mathcal{R}}^{G_{E\setminus F,c}}$ acting as

$$O_{\mathcal{R}}^{G_{E\setminus F,c}}|u\rangle|v\rangle|0\rangle = |u\rangle|v\rangle|\widetilde{R}_{\text{eff}}^{G_{E\setminus F,c}}(u,v)\rangle,$$

where $\widetilde{R}_{\text{eff}}^{G_{E\setminus F,c}}(u,v)$ is a 2-effective-resistance overestimate in $G_{E\setminus F,c}$. Thus, after the one-time preprocessing step, such an oracle can be constructed for every graph in the family $\{G_{E\setminus F,c} : c \ge 0\}$, and subsequently queried, without any further queries to $\mathcal{O}_G$.

The reusable preprocessing procedure is given in Algorithm 2.

**Algorithm 2** REUSABLE EFFECTIVE-RESISTANCE PREPROCESSING

**Input:** $G = (V, E, w)$ with $n$ vertices and $m$ edges, and $F \subseteq E$.
**Output:** A reusable data structure for the family $\{G_{E\setminus F,c} : c \geq 0\}$.

1: Let $H$ be the subgraph of $G$ with edge set $E \setminus F$.
2: Compute a 1/10-spectral sparsifier $\widetilde{H}$ of $H$ using Theorem 2.14.
3: Store $\widetilde{H}$ and $F$.
4: **return** $\mathcal{H} := (\widetilde{H}, F)$.

We first record a property used to analyze Algorithm 2.

**Lemma 3.5** (SPARSIFIER REUSABILITY)**.** *Let $G = (V, E, w)$ be a weighted graph and fix $F \subseteq E$. Let $H$ be the subgraph of $G$ with edge set $E \setminus F$, and suppose that $\widetilde{H}$ is an $\epsilon$-spectral sparsifier of $H$. Then, for every $c \geq 0$,*

$$F + c \cdot \widetilde{H}$$

*is an $\epsilon$-spectral sparsifier of $G_{E\setminus F,c}$.*

*Proof.* Deferred to Section D. □

**Proposition 3.6** (REUSABLE EFFECTIVE-RESISTANCE DATA STRUCTURE)**.** *Let $G = (V, E, w)$ be a weighted graph with $n$ vertices and $m$ edges, and let $F \subseteq E$ be a fixed subset of edges stored explicitly as edge records. For any $\eta > 0$, there is a quantum algorithm that, with probability at least $1 - \eta$, uses $\widetilde{O}(\sqrt{mn} \cdot \log(1/\eta))$ queries to the adjacency-list oracle $\mathcal{O}_G$ to construct a data structure $\mathcal{H}$ for the family of rescaled graphs $\{G_{E\setminus F,c} : c \geq 0\}$.*

*For any $c \geq 0$, $\mathcal{H}$ can be used to construct an oracle $O_{\mathcal{R}}^{G_{E\setminus F,c}}$ giving access to 2-effective-resistance overestimates in $G_{E\setminus F,c}$. Constructing this oracle, as well as querying it, requires no further queries to $\mathcal{O}_G$.*

*Proof.* Run Algorithm 2. Let $H$ be the subgraph of $G$ with edge set $E \setminus F$. Since $F$ is known, oracle access to $H$ can be simulated using a single query to $\mathcal{O}_G$, by setting the weight of any queried edge in $F$ to zero. Hence, by Theorem 2.14, with probability at least $1 - \eta$, we compute a 1/10-spectral sparsifier $\widetilde{H}$ of $H$ using $\widetilde{O}(\sqrt{mn}\log(1/\eta))$ queries to $\mathcal{O}_G$.

Fix any $c > 0$. By Lemma 3.5, $F + c \cdot \widetilde{H}$ is a 1/10-spectral sparsifier of $G_{E\setminus F,c}$. Since $\widetilde{H}$ and $F$ are stored explicitly, $F + c \cdot \widetilde{H}$ can be constructed without further queries to $\mathcal{O}_G$. Moreover, oracle access to $F + c \cdot \widetilde{H}$ can be simulated directly from the stored representations of $F$ and $\widetilde{H}$, and hence also requires no further queries to $\mathcal{O}_G$.

We can therefore apply Theorem 2.15 to the graph $F + c \cdot \widetilde{H}$ with $\epsilon = 1/10$. This gives an oracle providing 2-effective resistance overestimates in $G_{E\setminus F,c}$, using $\widetilde{O}(n)$ time and no further queries to $\mathcal{O}_G$. Thus we can prepare

$$O_{\mathcal{R}}^{G_{E\setminus F,c}}|u\rangle|v\rangle|0\rangle = |u\rangle|v\rangle|\widetilde{R}_{\text{eff}}^{G_{E\setminus F,c}}(u, v)\rangle.$$

For $c = 0$, we have $G_{E\setminus F,0} = F$. Since $F$ is stored explicitly, the corresponding effective-resistance oracle can be constructed directly, without any further queries to $\mathcal{O}_G$.

Therefore, after the initial construction of $\mathcal{H}$, constructing and querying $O_{\mathcal{R}}^{G_{E\setminus F,c}}$ for any $c \geq 0$ requires no further queries to $\mathcal{O}_G$. □

### 3.2 Reusable Marginal Data Structure from Large-Marginal Edges

In this section we prove Proposition 3.11. Although the marginals, or equivalently the leverage scores, vary across the graphs obtained by rescaling the edges outside a fixed set, we show that

all edges that can attain a large marginal can be identified in a single preprocessing step. This allows us to prepare the marginal state efficiently for each graph in the family, a key primitive for implementing the quantum walk operator in Section 3.3.

Throughout this section, fix a connected weighted graph $G = (V, E, w)$ and a set $F \subseteq E$, which we also view as the subgraph $(V, F, w|_F)$. For $c > 0$ write $G_c := G_{E\setminus F,c}$, so $G_1 = G$ and $L_{G_c} = L_F + c\, L_{G\setminus F}$. We write $\ell_c(e)$ for the leverage score (equivalently, the marginal, Lemma 2.10) of $e$ in $G_c$, $\ell := \ell_1$, and $b_e := \chi_u - \chi_v$ for $e = \{u, v\}$.

**Definition 3.7** (Heavy edges)**.** For a connected weighted graph $G$ and $\rho \in (0, 1]$, an edge $e$ of $G$ is *$\rho$-heavy* if its leverage score satisfies $\ell_G(e) \geq \rho$ (equivalently, its marginal under the spanning tree distribution of $G$ is at least $\rho$, by Lemma 2.10). We denote the set of $\rho$-heavy edges by $\textsc{Heavy}_\rho(G)$.

The set $\textsc{Heavy}_\rho(G_c)$ may vary with $c$. The next lemma shows that marginals are monotone in $c$. Consequently, these sets admit a single static envelope: storing $F$ together with the edges that are heavy in $G_1 = G$ gives a set containing $\textsc{Heavy}_\rho(G_c)$ for every $0 < c \leq 1$. We call this set the *heavy-edge envelope*.

**Lemma 3.8** (Marginal monotonicity under rescaling)**.** *For all $0 < c_1 \leq c_2 \leq 1$,*

$$\ell_{c_1}(e) \geq \ell_{c_2}(e) \quad \text{for } e \in F, \qquad \ell_{c_1}(e) \leq \ell_{c_2}(e) \quad \text{for } e \in E \setminus F.$$

*Consequently,* $\textsc{Heavy}_\rho(G_{c_1}) \subseteq F \cup \textsc{Heavy}_\rho(G_{c_2})$ *for every* $\rho \in (0, 1]$.

*Proof.* We have $L_{G_c} = L_F + cL_{G\setminus F}$. Let $0 < c_1 \leq c_2 \leq 1$.

For $e \in E \setminus F$, $\ell_c(e) = c \cdot w(e) \cdot b_e^\top L_{G_c}^+ b_e$. Since $L_F$ is positive semidefinite (Fact 2.7),

$$x^\top (L_F + c_1 L_{G\setminus F})x \geq \frac{c_1}{c_2} x^\top (L_F + c_2 L_{G\setminus F})x.$$

Since $c_1, c_2 > 0$, both $G_{c_1}$ and $G_{c_2}$ have the same edge support as $G$, and are therefore connected. Hence their Laplacians are positive definite on $\mathbb{1}^\perp$, so the preceding quadratic-form inequality may be inverted on $\mathbb{1}^\perp$. Taking $x = b_e$ gives

$$b_e^\top L_{G_{c_1}}^+ b_e \leq \frac{c_2}{c_1} b_e^\top L_{G_{c_2}}^+ b_e.$$

Multiplying by $c_1 w(e)$ yields $\ell_{c_1}(e) \leq \ell_{c_2}(e)$.

For $e \in F$, its weight is unchanged, and $L_F + c_1 L_{G\setminus F} \leq L_F + c_2 L_{G\setminus F}$. Hence, after inversion, $\ell_{c_1}(e) \geq \ell_{c_2}(e)$. Therefore, $\textsc{Heavy}_\rho(G_{c_1}) \subseteq F \cup \textsc{Heavy}_\rho(G_{c_2})$. □

Taking $c_2 = 1$ and using $G_1 = G$, we get $\textsc{Heavy}_\rho(G_c) \subseteq F \cup \textsc{Heavy}_\rho(G)$ for every $0 < c \leq 1$. This envelope is defined through exact leverage scores and is not directly computable, so we replace $\textsc{Heavy}_\rho(G)$ by an estimated heavy set obtained from effective-resistance overestimates, which keeps the containment.

**Lemma 3.9** (Heavy set from effective-resistance overestimates)**.** *Let $G = (V, E, w)$ be a connected weighted graph with $n$ vertices and $m$ edges, and let $\rho \in [n/m, 1]$. Suppose we have coherent query access to $\lambda$-effective-resistance overestimates $\widetilde{R}_{\text{eff}}^G(e)$ for all $e \in E$ where $\lambda \geq 1$ is an absolute constant. Define*

$$\widehat{\textsc{Heavy}}_\rho(G) := \{e \in E : w(e) \cdot \widetilde{R}_{\text{eff}}^G(e) \geq \rho\}.$$

*Then* $\textsc{Heavy}_\rho(G) \subseteq \widehat{\textsc{Heavy}}_\rho(G)$ *and* $|\widehat{\textsc{Heavy}}_\rho(G)| \leq \lambda(n-1)/\rho$. *Moreover, for any $\eta > 0$, we can compute $\widehat{\textsc{Heavy}}_\rho(G)$ with probability at least $1 - \eta$ using $\widetilde{O}(\sqrt{mn/\rho} \cdot \log(1/\eta))$ queries to the adjacency-list oracle $\mathcal{O}_G$.*

*Proof.* For any $e \in E$, $\ell(e) = w(e)R^G_{\text{eff}}(e)$. Since $\widetilde{R}^G_{\text{eff}}(e) \geq R^G_{\text{eff}}(e)$, every $e \in \textsc{Heavy}_\rho(G)$ satisfies $w(e)\widetilde{R}^G_{\text{eff}}(e) \geq \ell(e) \geq \rho$, and hence $\textsc{Heavy}_\rho(G) \subseteq \widehat{\textsc{Heavy}}_\rho(G)$.

Moreover,

$$\rho|\widehat{\textsc{Heavy}}_\rho(G)| \leq \sum_{e \in \widehat{\textsc{Heavy}}_\rho(G)} w(e)\widetilde{R}^G_{\text{eff}}(e) \leq \lambda \sum_{e \in E} \ell(e) = \lambda(n-1),$$

where the last equality follows from Fact 2.11.

Finally, we apply repeated Grover search over the $2m$ adjacency-list positions. Each undirected edge $e = \{u, v\}$ appears twice, once as a position $(u, i)$ in the adjacency list of $u$ and once as a position $(v, j)$ in the adjacency list of $v$. For a position $(u, i)$, one query to $\mathcal{O}_G$ returns the corresponding edge $e = \{u, v\}$ and its weight, and we mark the position if $w(e)\widetilde{R}^G_{\text{eff}}(e) \geq \rho$. Hence every edge in $\widehat{\textsc{Heavy}}_\rho(G)$ contributes exactly two marked positions. Lemma A.1 finds all such positions with probability at least $1 - \eta$ using

$$\widetilde{O}\left(\sqrt{2m \cdot \frac{2\lambda(n-1)}{\rho}} \cdot \log(1/\eta)\right) = \widetilde{O}\left(\sqrt{\frac{mn}{\rho}} \cdot \log(1/\eta)\right)$$

queries to the adjacency-list oracle $\mathcal{O}_G$. From each found position $(u, i)$ with $u < v$, where $v$ is the $i$th neighbor of $u$, one query returns $v$ and $w(e)$, and binary search in the sorted adjacency list of $v$ gives the other position $(v, j)$ using $O(\log n)$ queries. This yields the record $(u, v, w(e), i, j)$ of every edge in $\widehat{\textsc{Heavy}}_\rho(G)$. Since $|\widehat{\textsc{Heavy}}_\rho(G)| \leq \lambda(n-1)/\rho$, this costs $\widetilde{O}(n/\rho)$ further queries, which is at most $\widetilde{O}(\sqrt{mn/\rho})$ because $\rho \geq n/m$. □

We now show that one can use this heavy-edge envelope to prepare the marginal-state efficiently.

**Lemma 3.10** (Marginal-state preparation)**.** *Let $G = (V, E, w)$ be a connected weighted graph with $n$ vertices and $m$ edges, and let $\rho \in [n/m, 1]$. There is an algorithm that prepares the marginal state $|\mu_G\rangle$ corresponding to $\widetilde{p}_G(e) := w(e)\widetilde{R}^G_{\text{eff}}(e)$ using $\widetilde{O}(\sqrt{m\rho/n})$ queries to $\mathcal{O}_G$, provided that we are given:*

- *coherent query access to 2-effective-resistance overestimates $\widetilde{R}^G_{\text{eff}}(e)$ for all edges of $G$;*
- *the degree of every vertex of $G$; and*
- *a stored set $\mathcal{E}_\rho \subseteq E$ such that every edge outside $\mathcal{E}_\rho$ has marginal less than $\rho$, with each edge in $\mathcal{E}_\rho$ given as an edge record.*

*Proof.* Throughout the proof, $G$ is fixed, so we write $\widetilde{p}$, $\widetilde{R}_{\text{eff}}$ and $R_{\text{eff}}$ for $\widetilde{p}_G$, $\widetilde{R}^G_{\text{eff}}$ and $R^G_{\text{eff}}$. From the given vertex degrees, we compute the prefix sums $\text{off}_u := \sum_{v<u} \deg(v)$ with respect to the fixed lexicographic order on the vertices. We index the $2m$ adjacency-list positions by assigning $z := \text{off}_u + i$ to $(u, i)$. Conversely, given $z \in [2m]$, $u$ is the unique vertex satisfying $\text{off}_u < z \leq \text{off}_u + \deg(u)$, and $i = z - \text{off}_u$. Thus we can implement

$$|z\rangle \xmapsto{\text{prefix}} |u\rangle|i\rangle \xmapsto{\mathcal{O}_G} |u\rangle|i\rangle|v\rangle|w(u, v)\rangle,$$

where $v$ is the $i$-th neighbor of $u$. For an edge $e = \{u, v\}$, among its two adjacency-list positions $(u, i)$ and $(v, j)$ we use the one whose first vertex is smaller in the fixed lexicographic order.

Consider the 2-marginal overestimate $\widetilde{p}(e) := w(e)\widetilde{R}_{\text{eff}}(e)$ and the corresponding isotropic multiplicities $t(e)$ from Definition 2.20. Writing $M := \sum_{e \in E} t(e)$, the marginal state (cf Definition 3.4) is

$$|\mu_G\rangle = \frac{1}{\sqrt{M}} \sum_{e \in E} \sqrt{t(e)}\,|e\rangle.$$

Let $\ell(e)$ denote the marginal (equivalently, the leverage score, Lemma 2.10) of $e$ under the spanning-tree distribution of $G$. If $e \notin \mathcal{E}_\rho$, then $\ell(e) < \rho$. Since $\widetilde{R}_{\text{eff}}$ is a 2-effective-resistance overestimate (cf Definition 2.12),

$$\widetilde{p}(e) \leq 2w(e)R_{\text{eff}}(e) = 2\ell(e) < 2\rho.$$

Hence $t(e) \leq t_{\text{cap}}$, where $t_{\text{cap}} := \lceil m\rho/(n-1) \rceil$. Define the proposal weights

$$\varphi(e) := \begin{cases} t(e), & e \in \mathcal{E}_\rho, \\ t_{\text{cap}}, & e \notin \mathcal{E}_\rho. \end{cases}$$

Then $\varphi(e) \geq t(e)$ for every $e \in E$.

Since $\lceil x \rceil \leq 1 + x$ and $\widetilde{p}$ is a 2-marginal overestimate, $M \leq m + \frac{m}{2(n-1)} \sum_{e \in E} \widetilde{p}(e) \leq 2m$. Since $\lceil x \rceil \geq x$, $\widetilde{p}(e) \geq \ell(e)$, and $\sum_{e \in E} \ell(e) = n - 1$ by Foster's theorem (Fact 2.11) as $G$ is connected, $M \geq \frac{m}{2(n-1)} \sum_{e \in E} \ell(e) = m/2$. Let $\Phi := \sum_{e \in E} \varphi(e)$. Using $\rho \geq n/m$,

$$\Phi = \sum_{e \in \mathcal{E}_\rho} t(e) + \sum_{e \notin \mathcal{E}_\rho} t_{\text{cap}} \leq \sum_{e \in E} t(e) + m \cdot t_{\text{cap}} \leq 2m + m \left\lceil \frac{m\rho}{n-1} \right\rceil = O\left(\frac{m^2\rho}{n}\right).$$

We next use the proposal weights to prepare a proposal state $|\varphi\rangle$. Construct a binary tree over the $2m$ slots, initially assigning $t_{\text{cap}}$ to every leaf. For each $e \in \mathcal{E}_\rho$, its two adjacency-list positions $(u, i)$ give its two slots $z = \text{off}_u + i$ without queries to $\mathcal{O}_G$. We replace the values at these two leaves by $t(e)$, which is computable from the stored edge weight and $\widetilde{R}_{\text{eff}}(e)$. Thus the leaf corresponding to a slot $z$ has value $\varphi(e(z))$, where $e(z)$ is the edge at the corresponding adjacency-list position. This can be computed without any queries to $\mathcal{O}_G$ since we already have stored the weights of edges in $\mathcal{E}_\rho$ and we have coherent query access to 2-effective-resistance overestimates $\widetilde{R}_{\text{eff}}(e)$. The sum of all leaf values is thus $2\Phi$. Storing the corresponding partial sums at the internal nodes, the recursive state-preparation procedure of Grover and Rudolph [GR02] prepares

$$\frac{1}{\sqrt{2\Phi}} \sum_{z \in [2m]} \sqrt{\varphi(e(z))} |z\rangle$$

without queries to $\mathcal{O}_G$.

Using the stored prefix sums, we first map each slot to its adjacency-list position:

$$\frac{1}{\sqrt{2\Phi}} \sum_{z \in [2m]} \sqrt{\varphi(e(z))} |z\rangle \longmapsto \frac{1}{\sqrt{2\Phi}} \sum_{(u,i)} \sqrt{\varphi(e)} |u, i\rangle,$$

where $e$ is the edge at position $(u, i)$. The slot $z$ is uncomputed using $z = \text{off}_u + i$. We then query $\mathcal{O}_G$ to obtain the neighbor $v$ at position $(u, i)$ and mark the terms with $u < v$. Since every edge occurs at exactly two adjacency-list positions with equal weight and exactly one is marked, the success probability is $1/2$. By a slight abuse of notation, we use $|e\rangle$ to denote the unique marked basis state corresponding to each edge $e$. Hence constant-overhead amplitude amplification [BHMT02] prepares

$$|\varphi\rangle = \frac{1}{\sqrt{\Phi}} \sum_{e \in E} \sqrt{\varphi(e)}\, |e\rangle.$$

Starting from $|\varphi\rangle$, we perform quantum rejection sampling as in [ORR13]. For a basis state $|e\rangle$, one query to $\mathcal{O}_G$ at its corresponding adjacency-list position gives the other endpoint and the edge weight $w(e)$. Together with the given $\widetilde{R}_{\text{eff}}(e)$, this allows us to compute $t(e)$; the proposal

weight $\varphi(e)$ is obtained from the proposal tree without queries to $\mathcal{O}_G$. Since $\varphi(e) \geq t(e)$ for every edge, using $O(1)$ queries we perform

$$|\varphi\rangle \xmapsto{\text{rot.}} \frac{1}{\sqrt{\Phi}} \sum_{e \in E} \sqrt{\varphi(e)}\,|e\rangle \left( \sqrt{\frac{t(e)}{\varphi(e)}}|1\rangle + \sqrt{1 - \frac{t(e)}{\varphi(e)}}|0\rangle \right).$$

The auxiliary information used to implement the rotation is then uncomputed using $\widetilde{O}(1)$ queries to $\mathcal{O}_G$. The component marked by $|1\rangle$ is

$$\frac{1}{\sqrt{\Phi}} \sum_{e \in E} \sqrt{t(e)}\,|e\rangle|1\rangle \propto |\mu_G\rangle,$$

which has squared norm $M/\Phi$. Hence amplitude amplification [BHMT02] gives

$$\frac{1}{\sqrt{\Phi}} \sum_{e \in E} \sqrt{t(e)}\,|e\rangle|1\rangle \xmapsto{\text{AA}} \frac{1}{\sqrt{M}} \sum_{e \in E} \sqrt{t(e)}\,|e\rangle|1\rangle = |\mu_G\rangle|1\rangle$$

using

$$O\left(\sqrt{\frac{\Phi}{M}}\right) = O\left(\sqrt{\frac{m\rho}{n}}\right)$$

iterations. Each iteration uses $O(1)$ queries to $\mathcal{O}_G$, while $\widetilde{R}_{\text{eff}}$ is available coherently by assumption. Hence $|\mu_G\rangle$ is prepared using $\widetilde{O}(\sqrt{m\rho/n})$ queries to $\mathcal{O}_G$. □

**Proposition 3.11** (Reusable marginal data structure)**.** *Consider any weighted graph $G = (V, E, w)$ with $n$ vertices and $m$ edges, and let $F \subseteq E$ be a fixed subset of edges stored explicitly as edge records. For any $\rho \in [n/m, 1]$ and $\eta > 0$, there is a quantum algorithm that constructs a data structure $\mathcal{L}_\rho$ providing coherent query access to a set of edges, stored as edge records, such that every edge $e \in E$ outside this set has marginal less than $\rho$ in every graph in the family $\{G_{E\setminus F,c} : 0 < c \leq 1\}$. The algorithm succeeds with probability at least $1 - \eta$ and uses $\widetilde{O}(\sqrt{mn/\rho} \cdot \log(1/\eta))$ queries to the adjacency-list oracle $\mathcal{O}_G$.*

*Moreover, after constructing $\mathcal{L}_\rho$, for any $0 < c \leq 1$ it can be used to prepare the marginal state $|\mu_{G_{E\setminus F,c}}\rangle$ corresponding to the 2-marginal overestimate $\widetilde{p}_{G_{E\setminus F,c}}(e) := w_{G_{E\setminus F,c}}(e)\widetilde{R}_{\text{eff}}^{G_{E\setminus F,c}}(e)$ (Definition 3.4), using $\widetilde{O}(\sqrt{m\rho/n})$ queries to $\mathcal{O}_G$, given coherent query access to 2-effective-resistance overestimates $\widetilde{R}_{\text{eff}}^{G_{E\setminus F,c}}$.*

*Proof.* Recall $G_c := G_{E\setminus F,c}$. Let $\mathcal{E}_\rho := F \cup \widehat{\textsc{Heavy}}_\rho(G)$. By Lemma 3.8 (with $c_2 = 1$) and $\textsc{Heavy}_\rho(G) \subseteq \widehat{\textsc{Heavy}}_\rho(G)$ (Lemma 3.9), we have $\textsc{Heavy}_\rho(G_c) \subseteq \mathcal{E}_\rho$ for every $0 < c \leq 1$. Hence every edge outside $\mathcal{E}_\rho$ has marginal less than $\rho$ in every $G_c$.

The data structure $\mathcal{L}_\rho$ consists of (i) the set $\mathcal{E}_\rho$, stored as a list of edge records, which has size at most $|F| + 2(n-1)/\rho$ by Lemma 3.9, and (ii) the degree of every vertex. It provides coherent query access to both.

**Cost.** Computing $\widehat{\textsc{Heavy}}_\rho(G)$ uses $\widetilde{O}(\sqrt{mn/\rho}\cdot\log(1/\eta))$ queries by Lemma 3.9, and the Grover search returns the full record of every edge in $\widehat{\textsc{Heavy}}_\rho(G)$. The edges of $F$ are already stored as records. We also read the degree of every vertex with $n$ queries; rescaling does not change the adjacency lists, so these degrees are valid for every $G_c$. This extra cost is $n \leq \widetilde{O}(\sqrt{mn/\rho})$.

**Preparing the marginal state.** For every $0 < c \leq 1$, together with the assumed coherent access to 2-effective-resistance overestimates of $G_c$, the three inputs of Lemma 3.10 are available for $G_c$. Hence $|\mu_{G_c}\rangle$ can be prepared using $\widetilde{O}(\sqrt{m\rho/n})$ queries to $\mathcal{O}_G$. □

### 3.3 Coherent Isotropization

In this section we prove Proposition 3.16. The goal is to implement, for any weighted graph $G = (V, E, w)$, a quantum walk operator $W$ with large phase gap whose unique 1-eigenvector is $|\pi\rangle$, the q-sample of the spanning-tree distribution $\pi$ of $G$. This allows us to implement reflections about $|\pi\rangle$ efficiently.

We quantize the Up-Down walk with transitions biased according to the marginals as described in Markov Chain 2. More specifically, we use the isotropic transformation from Proposition 2.21 only implicitly. Given marginal overestimates, let $t_G(e)$ denote the corresponding isotropic multiplicity of each edge $e$. Following Proposition 2.21, we call a spanning tree $T'$ of $G'$ a copied tree of $T$ if its edges are copies of the edges of $T$, and say that $T'$ projects to $T$. For an original spanning tree $T$, define its copied-tree weight by

$$\nu(T) := \frac{\pi(T)}{\prod_{e\in T} t_G(e)}.$$

Thus every copied tree $T'$ projecting to $T$ satisfies $\pi'(T') = \nu(T)$.

We define an equivalent walk on the original spanning trees. Let

$$M := \sum_{f\in E} t_G(f), \qquad \Psi_T(f) := \sum_{e\in \mathrm{Cycle}(T+f)} \nu(T+f-e).$$

The up-step samples edge $f \in E$ with probability proportional to $t_G(f)$, while the down-step removes an edge $e \in \mathrm{Cycle}(T + f)$ with probability proportional to the copied-tree weights $\nu(T+f-e)$. For simplicity, we take $\mathrm{Cycle}(T+f) = \{f\}$ when $f \in T$, so that the down-step is a self-loop in this case. Working directly on the original spanning trees keeps the state space and its basis encoding independent of the edge copies introduced by the isotropic transformation.

**Markov Chain 2** Marginal-aware up-down walk from $T_0$

**for** $i = 0, 1, 2, \ldots$ **do**
  Choose $f \in E$ with probability $t_G(f)/M$.
  **if** $f \in T_i$ **then**
    Set $T_{i+1} = T_i$.
  **else**
    Choose $e \in \mathrm{Cycle}(T_i + f)$ with probability $\nu(T_i + f - e)/\Psi_{T_i}(f)$.
    Set $T_{i+1} = T_i + f - e$.
  **end if**
**end for**

The down step is exactly the down step of the Up-Down walk on the isotropically transformed graph, expressed on the original edges.

**Proposition 3.12** (Detailed balance of the marginal-aware walk)**.** *The marginal-aware walk is reversible with respect to $\pi$.*

*Proof.* Let $P_{\mathrm{ma}}$ be the transition matrix, and consider neighboring trees $T$ and $S = T + f - e$. Since $T + f = S + e$, we have $\Psi_T(f) = \Psi_S(e)$. Therefore

$$P_{\mathrm{ma}}(T, S) = \frac{t_G(f)}{M}\frac{\nu(S)}{\Psi_T(f)}, \qquad P_{\mathrm{ma}}(S, T) = \frac{t_G(e)}{M}\frac{\nu(T)}{\Psi_T(f)}.$$

Moreover, since $S$ is obtained from $T$ by replacing $e$ with $f$,

$$\frac{\nu(S)}{\nu(T)} = \frac{\pi(S)}{\pi(T)}\frac{t_G(e)}{t_G(f)}.$$

Hence $\pi(T)t_G(f)\nu(S) = \pi(S)t_G(e)\nu(T)$, and dividing by the common factor $M\Psi_T(f)$ gives $\pi(T)P_{\mathrm{ma}}(T,S) = \pi(S)P_{\mathrm{ma}}(S,T)$. The diagonal case is immediate, so detailed balance holds. □

We will also use the following standard fact about the spectra of Markov chains under lumping or projecting based on an equivalence relation defined on the states of the Markov chain.

**Lemma 3.13** (Spectrum of lumped chain; [LP17a, Lemma 12.9]). *Let $\Delta$ be the state space of a Markov chain $\mathcal{M}$ with transition matrix $P$, and let $\sim$ be an equivalence relation on $\Delta$ with equivalence classes $\Delta^\sharp := \{[x] : x \in \Delta\}$. For $x \in \Delta$ and a class $C \in \Delta^\sharp$, write $P(x,[y]) := \sum_{y\in[y]} P(x,y)$, and assume that $P(x,[y]) = P(x',[y])$ for all $x \sim x'$ and all $[y] \in \Delta^\sharp$. Then $P^\sharp([x],[y]) := P(x,[y])$ is well defined, and it is the transition matrix of the lumped chain $\mathcal{M}^\sharp$ on $\Delta^\sharp$. Then:*

1. *If $\phi : \Delta \to \mathbb{R}$ is an eigenfunction of $P$ with eigenvalue $\theta$ that is constant on each equivalence class, then its projection $\phi^\sharp : \Delta^\sharp \to \mathbb{R}$, $\phi^\sharp([x]) := \phi(x)$, is an eigenfunction of $P^\sharp$ with eigenvalue $\theta$.*
2. *Conversely, if $\kappa : \Delta^\sharp \to \mathbb{R}$ is an eigenfunction of $P^\sharp$ with eigenvalue $\theta$, then its lift $\kappa^\flat : \Delta \to \mathbb{R}$, $\kappa^\flat(x) := \kappa([x])$, is an eigenfunction of $P$ with eigenvalue $\theta$.*

**Lemma 3.14** (Spectral gap of the marginal-aware walk). *Let $G = (V,E,w)$ be a connected weighted graph on $n$ vertices and $m$ edges, with $m \geq 100n$. Let $G' = (V,E',w')$ be the isotropically transformed multigraph (cf Proposition 2.21), and suppose that its spanning tree distribution $\pi'$ is nearly isotropic. Then the marginal-aware walk Markov Chain 2 satisfies*

$$\gamma(P_{\mathrm{ma}}) \geq \Omega\left(\frac{1}{n\log^2 n}\right).$$

*Proof.* Let $Q$ be the up-down walk on the isotropically transformed multigraph $G'$. Since $\pi'$ is nearly isotropic, Theorem 2.19 gives $\gamma(Q) \geq \Omega(1/(n\log^2 n))$. Forgetting copy indices projects every spanning tree of $G'$ to a spanning tree of $G$. For $T \in \Delta$, let $[T] := \{T' \in \Delta_{G'} : T' \text{ projects to } T\}$ be the class of its copies. For $\widehat{T} = T + f - e$ and any $T' \in [T]$,

$$Q(T',[\widehat{T}]) = \frac{t_G(f)}{M-(n-1)}\frac{\nu(\widehat{T})}{\Psi_T(f)},$$

which does not depend on the choice of $T' \in [T]$. Hence Lemma 3.13 applies to $Q$ on the state space $\Delta_{G'}$ with classes $[T]$, and yields the lumped chain $Q^\sharp$ on $\Delta$ with $Q^\sharp(T,\widehat{T}) := Q(T',[\widehat{T}])$. By part 2 of the lemma, every eigenvalue of $Q^\sharp$ is an eigenvalue of $Q$, so $\gamma(Q^\sharp) \geq \gamma(Q)$. The stationary distribution of $Q^\sharp$ is $\pi$: for every $T \in \Delta$,

$$\sum_{T'\in[T]} \pi'(T') = |[T]|\cdot\nu(T) = \prod_{e\in T} t_G(e)\cdot\nu(T) = \pi(T).$$

Now compare $Q^\sharp$ with the marginal-aware walk $P_{\mathrm{ma}}$. For $\widehat{T} = T + f - e$,

$$P_{\mathrm{ma}}(T,\widehat{T}) = \frac{t_G(f)}{M}\frac{\nu(\widehat{T})}{\Psi_T(f)} = \zeta\cdot Q^\sharp(T,\widehat{T}), \qquad \zeta := \frac{M-(n-1)}{M}.$$

Since the rows of both matrices sum to 1, this gives $P_{\mathrm{ma}} = \zeta Q^\sharp + (1-\zeta)I$. Every eigenvalue $\theta$ of $Q^\sharp$ is therefore mapped to $1-\zeta+\zeta\theta$ in $P_{\mathrm{ma}}$, so $\gamma(P_{\mathrm{ma}}) = \zeta\cdot\gamma(Q^\sharp)$. Since $M \geq m \geq 100n$, we have $\zeta \geq 99/100$, and hence

$$\gamma(P_{\mathrm{ma}}) \geq \frac{99}{100}\cdot\gamma(Q^\sharp) \geq \Omega\left(\frac{1}{n\log^2 n}\right).$$

□

**Lemma 3.15** (Nonnegative spectrum of the marginal-aware walk)**.** *The marginal-aware walk has nonnegative spectrum. Consequently,* $\gamma_{\mathrm{abs}}(P_{\mathrm{ma}}) = \gamma(P_{\mathrm{ma}})$.

*Proof.* Let $Q$, $Q^\sharp$ and $\zeta$ be as in the proof of Lemma 3.14. The up-down walk $Q$ on $G'$ has nonnegative spectrum [CSV23]. By part 2 of Lemma 3.13, every eigenvalue of $Q^\sharp$ is an eigenvalue of $Q$, hence nonnegative. Since $P_{\mathrm{ma}} = \zeta Q^\sharp + (1-\zeta)I$, every eigenvalue $\theta$ of $Q^\sharp$ is mapped to $1 - \zeta + \zeta\theta \geq 0$. Thus $P_{\mathrm{ma}}$ has nonnegative spectrum, and its absolute spectral gap equals its ordinary spectral gap. □

**Quantization of the marginal-aware walk.** We now express the marginal-aware walk in the labelled Szegedy framework of Theorem 2.16. For each spanning tree $T$, define the set of transition labels

$$C_T := \{(f, e) : f \notin T,\ e \in \mathrm{Cycle}(T + f)\} \cup \{(f, f) : f \in T\}.$$

For $(f, e) \in C_T$, the probability of making a transition from $T$ along the label $(f, e)$ is

$$\Pr[(f, e) \mid T] := \begin{cases} \dfrac{t_G(f)}{M}, & f \in T,\ e = f, \\ \dfrac{t_G(f)}{M} \dfrac{\nu(T + f - e)}{\Psi_T(f)}, & f \notin T,\ e \in \mathrm{Cycle}(T + f). \end{cases}$$

The endpoint map is $h_T(f, e) := T + f - e$, with $T + f - f := T$. Hence we have

$$P_{\mathrm{ma}}(T, S) = \sum_{(f,e):\, h_T(f,e)=S} \Pr[(f, e) \mid T].$$

For every non-self-loop transition $S = T + f - e$, the exchanged edges are uniquely determined by $f = S \setminus T$ and $e = T \setminus S$. Define the flip-flop shift by

$$g\big(T, (f, e)\big) := \big(T + f - e, (e, f)\big).$$

This is an involution, and every label with $e = f$ is fixed. Therefore the marginal-aware walk satisfies the labelled-transition assumptions of Theorem 2.16.

**Proposition 3.16** (Quantum walk operator with large gap)**.** *Let $G = (V, E, w)$ be any weighted graph with $n$ vertices and $m \geq 100n$ edges, and let $\Delta$ denote the set of spanning trees of $G$. There exists a quantum walk operator $W$, associated with a reversible Markov chain on $\Delta$, whose unique 1-eigenvector is $|\pi\rangle$, where $\pi$ is the spanning-tree distribution of $G$, and whose phase gap is at least $\widetilde{\Omega}(1/\sqrt{n})$.*

*Moreover, for any $\rho \in [n/m, 1]$, one application of $W$ and its inverse can be implemented using $\widetilde{O}(\sqrt{m\rho/n})$ queries to $\mathcal{O}_G$, given:*

- *coherent query access to 2-effective-resistance overestimates for all edges of $G$;*
- *the degree of every vertex of $G$; and*
- *a stored set of edges such that every edge outside this set has marginal under $\pi$ less than $\rho$, with each stored edge given as an edge record.*

*Proof.* Let $\mathcal{M}_{\mathrm{ma}}$ be the marginal-aware walk from Markov Chain 2, and let $P_{\mathrm{ma}}$ be its transition matrix. By Proposition 3.12, $\mathcal{M}_{\mathrm{ma}}$ is reversible with stationary distribution $\pi$. By Lemmas 3.14 and 3.15, we have $\delta_{\mathrm{abs}}(P_{\mathrm{ma}}) \geq \widetilde{\Omega}(1/n)$.

The exchange-pair labels described before give a labelled-transition realization of $\mathcal{M}_{\mathrm{ma}}$. Therefore, Theorem 2.16 applies to the corresponding labelled Szegedy walk operator $W$. On the busy subspace, its unique 1-eigenvector is $|\pi\rangle|0\rangle$. Since the phase gap is $\Omega(\sqrt{\delta_{\mathrm{abs}}(P_{\mathrm{ma}})})$, we obtain

$$\gamma_{\mathrm{ph}}(W) \geq \widetilde{\Omega}(1/\sqrt{n}).$$

It remains to bound the implementation cost of one application of $W$. One application consists of preparing and unpreparing the labelled transition superposition $\mathcal{U}$ and applying the flip-flop shift $\mathcal{S}$ (see Section 2.4). The reflection about the initialized label register and the flip-flop shift themselves use no adjacency-list queries, so it suffices to implement the transition preparation.

Recall that $\mathcal{U}$ acts as

$$\mathcal{U}|T\rangle|0\rangle \mapsto |T\rangle \sum_{(f,e)\in C_T} \sqrt{\Pr[(f,e) \mid T]}|f\rangle|e\rangle.$$

We implement $\mathcal{U}$ in two stages:

$$\begin{aligned}
|T\rangle|0\rangle &\xrightarrow{(1)} |T\rangle|\mu_G\rangle = |T\rangle \frac{1}{\sqrt{M}} \sum_{f\in E} \sqrt{t_G(f)}|f\rangle \\
&\xrightarrow{(2)} |T\rangle \frac{1}{\sqrt{M}} \sum_{f\in E} \sqrt{t_G(f)}|f\rangle \frac{1}{\sqrt{\Psi_T(f)}} \sum_{e\in \mathrm{Cycle}(T+f)} \sqrt{\nu(T+f-e)}|e\rangle \\
&= |T\rangle \sum_{(f,e)\in C_T} \sqrt{\Pr[(f,e) \mid T]}|f\rangle|e\rangle.
\end{aligned}$$

Here, step (1) prepares the up-step, while step (2) prepares the down-step conditioned on $T$ and $f$. We describe the implementation of these two steps separately below.

Step (1) is exactly the preparation of $|\mu_G\rangle$. The assumptions of the proposition satisfy all the requirements of Lemma 3.10, and thus we can perform step (1) using $\widetilde{O}(\sqrt{m\rho/n})$ queries to $\mathcal{O}_G$.

Next, we implement step (2), namely, the down-step. Given the current tree $T$ and the added edge $f$, the unique cycle $\mathrm{Cycle}(T+f)$ can be found from the stored representation of $T$, without any further queries to $\mathcal{O}_G$. This can be done naively by traversing $T$, whose explicit description is already stored in $|T\rangle$.

On this cycle, the down-step probabilities are proportional to $\nu(T+f-e)$, equivalently to $t_G(e)/w(e)$ for $e \in \mathrm{Cycle}(T+f)$. Thus, it suffices to prepare the down-step superposition using weights proportional to $t_G(e)/w(e)$. We construct a binary tree whose leaves correspond to the edges $e \in \mathrm{Cycle}(T+f)$, assigning to the leaf corresponding to $e$ the value $t_G(e)/w(e)$. The value $t_G(e)$ can be computed from the coherent query access to the 2-effective-resistance overestimate for $e$, while $w(e)$ is available from the stored edge weight. Thus, all leaf values can be computed without queries to $\mathcal{O}_G$. Storing the corresponding partial sums at the internal nodes, the recursive state-preparation procedure of Grover and Rudolph [GR02] prepares

$$\frac{1}{\sqrt{\Psi_T(f)}} \sum_{e\in \mathrm{Cycle}(T+f)} \sqrt{\nu(T+f-e)}|e\rangle$$

conditioned on $|T\rangle|f\rangle$, since the leaf weights are proportional to $\nu(T+f-e)$. Hence, step (2) requires no additional queries to $\mathcal{O}_G$.

The remaining part of the implementation of $W$ is the flip-flop shift $\mathcal{S}$, which maps

$$|T\rangle|f\rangle|e\rangle \mapsto |T+f-e\rangle|e\rangle|f\rangle,$$

fixing $|T\rangle|f\rangle|f\rangle$ when $f \in T$. This can be implemented by swapping the two edge registers and updating the tree register, all without any further queries to $\mathcal{O}_G$. Therefore, one application of $W$ can be implemented using $\widetilde{O}(\sqrt{m\rho/n})$ queries to $\mathcal{O}_G$. □

### 3.4 Slowly Varying Schedule for Spanning Trees

In this section we prove Proposition 3.21. We use the conventions of Definition 3.3 and the paragraph following it: $S \in \Delta_G$ is a fixed admissible spanning tree. We approximate $\pi_\infty$ by $\pi_{\beta^\star}$, a finite-$\beta$ distribution, as in Proposition 21 of [AD20].

**Lemma 3.17** (Closeness to point mass). *Let $\beta^\star := \ln\left(16n^{n-2}/\epsilon^2\right)$. Then we have that $\|\pi_{\beta^\star} - \pi_\infty\|_{\mathrm{TV}} \le \epsilon^2/8$. Consequently, $\||\pi_{\beta^\star}\rangle - |\pi_\infty\rangle\|_2 \le \epsilon/2$.*

*Proof.* Since $\pi_\infty = \mathbb{1}_S$, it is enough to bound $\|\pi_{\beta^\star} - \mathbb{1}_S\|_{\mathrm{TV}}$. For a point mass, we have that $\|\pi_{\beta^\star} - \mathbb{1}_S\|_{\mathrm{TV}} = 1 - \pi_{\beta^\star}(S)$. By definition of the $\beta$-scaled distribution,

$$1 - \pi_{\beta^\star}(S) = \frac{\sum_{T \neq S} e^{-\beta^\star |T \setminus S|} w(T)}{w(S) + \sum_{T \neq S} e^{-\beta^\star |T \setminus S|} w(T)} \le \frac{e^{-\beta^\star} \sum_{T \neq S} w(T)}{w(S)} \le \frac{e^{-\beta^\star}}{\pi(S)}.$$

Using admissibility of $S$, namely $\pi(S) \ge 1/2n^{n-2}$, we obtain

$$1 - \pi_{\beta^\star}(S) \le 2n^{n-2} e^{-\beta^\star} = 2n^{n-2} \cdot \frac{\epsilon^2}{16n^{n-2}} = \frac{\epsilon^2}{8}.$$

Finally, for any probability distributions $p_1, p_2$,

$$\||p\rangle_1 - |p\rangle_2\|_2^2 = \sum_x (\sqrt{p_1(x)} - \sqrt{p_2(x)})^2 \le 2 \|p_1 - p_2\|_{\mathrm{TV}}.$$

Therefore, $\||\pi_{\beta^\star}\rangle - |\pi_\infty\rangle\|_2 \le \sqrt{2 \cdot \frac{\epsilon^2}{8}} = \epsilon/2$. □

To analyze the path in $\beta$, we use a cooling schedule from $\beta^\star$ down to 0. Outputting $S$ gives an initial approximation to the q-sample for $\pi_{\beta^\star}$, and the annealing procedure moves from $\beta = \beta^\star$ back to $\beta = 0$, where the target distribution is $\pi$. The tree $S$ is used only to define and initialize this annealing path. It remains to show that such an admissible spanning tree can be found efficiently. The following proposition gives the required preprocessing step.

**Proposition 3.18** (Computing an Admissible Tree). *Consider $G(V, E, w)$, a connected weighted graph. There is a quantum algorithm that, for any $\eta > 0$, outputs an admissible spanning tree $S$ of $G$ with probability $1 - \eta$ using $\widetilde{O}(\sqrt{mn} \cdot \log(1/\eta))$ adjacency-list queries and time.*

*Proof.* Let $S$ be a maximum-weight-product spanning tree of $G$, so that $\prod_{e \in S} w(e) \ge \prod_{e \in T} w(e)$ for every $T \in \Delta$. Hence,

$$\pi(S) = \frac{\prod_{e \in S} w(e)}{\sum_{T \in \Delta} \prod_{e \in T} w(e)} \ge \frac{1}{|\Delta|} \ge \frac{1}{n^{n-2}},$$

where the last inequality follows from the fact that an $n$-vertex graph has at most $n^{n-2}$ spanning trees. Thus, $S$ is admissible. By Theorem A.5, we can find such a tree with probability at least $1 - \eta$ using $\widetilde{O}(\sqrt{mn} \log(1/\eta))$ adjacency-list queries and time. □

The remaining task is to choose a short slowly-varying schedule $0 = \beta_0 < \beta_1 < \cdots < \beta_l = \beta^\star$.

We now instantiate the Gibbs framework of Section 2.6 for spanning trees. Every $G_\beta$ shares the spanning tree set $\Delta$ with $G$. Take the base weight $\sigma(T) = w(T)$ and the Hamiltonian $H(T) := |T \setminus S|$. The partition function is $Z(\beta) := \sum_{T \in \Delta} e^{-\beta H(T)} w(T) = \sum_{T \in \Delta} w_\beta(T)$, where $w_\beta(T) := \prod_{e \in T} w_\beta(e)$, and the Gibbs distribution $\pi_\beta(T) = e^{-\beta H(T)} w(T)/Z(\beta)$ is the $\beta$-scaled distribution of Definition 3.3. In particular, $Z(0) = w(\Delta)$ and $\pi_0 = \pi$.

**Lemma 3.19** (Log-convexity of the partition function). *The partition function $Z_S(\beta)$ is log-convex in $\beta$.*

*Proof.* The proof is deferred to Section C. □

In our application, we start from an admissible tree, corresponding to the large-$\beta$ endpoint $|\pi_\infty\rangle$, and move towards the target graph at smaller $\beta$. Since Theorem 2.23 assumes an efficiently preparable state at $\beta = 0$, we use the following reversed version to construct the schedule starting from the easily preparable large-$\beta$ endpoint. Although the proposition follows from existing results, we state it here for completeness and to keep the argument self-contained.

**Proposition 3.20** (Quantum algorithm for a reversed slowly-varying schedule, adapted from [HW20]). *Let $\{\pi_\beta\}_{\beta\in[0,\beta^\star]}$ be a family of Gibbs distributions with log-convex partition function $Z(\beta)$, and let $0 = \beta_0 < \cdots < \beta_l = \beta^\star$ be an $e^2$-slowly varying schedule of length $l \le \widetilde{O}\left(\sqrt{\ln \frac{Z(0)}{Z(\beta^\star)}}\right)$, which exists by Theorem 2.23.*

*Moreover, if a state $|\widetilde{\pi}_{\beta^\star}\rangle$ satisfying $\||\widetilde{\pi}_{\beta^\star}\rangle - |\pi_{\beta^\star}\rangle\|_2 \le \epsilon$ can be prepared and, for every $\beta \in [0,\beta^\star]$, there is a reversible Markov chain $\mathcal{M}_\beta$ with stationary distribution $\pi_\beta$ and absolute spectral gap at least $\gamma$, together with an implementable quantum walk operator $W_\beta$, then for any $\eta > 0$, a quantum algorithm constructs such a schedule with probability at least $1-\eta$ using $\widetilde{O}\left(\frac{l}{\sqrt{\gamma}} \log\left(\frac{1}{\eta}\right) \log\left(\frac{1}{\epsilon}\right)\right)$ applications of $W_\beta$ and $W_\beta^{-1}$ at adaptively chosen values of $\beta$.*

*Proof.* Let $Z(\beta) = \sum_x \sigma(x) e^{-\beta H(x)}$ be the partition function of the original Gibbs family. To apply Theorem 2.23 in reverse, define the reparameterized partition function $\widehat{Z}(\beta) := Z(\beta^\star - \beta)$ for $\beta \in [0,\beta^\star]$. Equivalently, $\widehat{Z}(\beta) = \sum_x \sigma(x) e^{-\beta^\star H(x)} e^{\beta H(x)}$. Thus, the corresponding Gibbs distribution is $\widehat{\pi}_\beta(x) = \pi_{\beta^\star-\beta}(x)$, so that $\widehat{\pi}_0 = \pi_{\beta^\star}$ and $\widehat{\pi}_{\beta^\star} = \pi_0$. Since $Z(\beta)$ is log-convex, so is $\widehat{Z}(\beta)$, and the schedule length and construction argument applies also to this increasing log-convex partition function [HW20; CH23]. The assumption gives an $\epsilon$-approximate preparation of $|\widehat{\pi}_0\rangle = |\pi_{\beta^\star}\rangle$. Applying Theorem 2.23 to $\widehat{Z}(\beta)$ therefore yields an $e^2$-slowly varying schedule $0 = \widehat{\beta}_0 < \cdots < \widehat{\beta}_l = \beta^\star$ for $\widehat{Z}$. If we set $\beta_i := \beta^\star - \widehat{\beta}_{l-i}$, then $0 = \beta_0 < \cdots < \beta_l = \beta^\star$ and $|\pi_{\beta_i}\rangle = |\widehat{\pi}_{\widehat{\beta}_{l-i}}\rangle$. Since the overlap of two states does not depend on the order in which they are taken, the consecutive overlaps of $\{|\pi_{\beta_i}\rangle\}_i$ are the same as those of $\{|\widehat{\pi}_{\widehat{\beta}_j}\rangle\}_j$, with the order of the pairs reversed. Hence $\{\beta_i\}$ is an $e^2$-slowly-varying schedule for the original family. □

**Proposition 3.21** (Computing a Slowly varying schedule). *Let $G = (V,E,w)$ be a connected weighted graph with $n$ vertices and $m$ edges. Fix an admissible spanning tree $S$ and $\epsilon > 0$, and define $\beta^\star := \ln(16n^{n-2}/\epsilon^2)$. Let $\{G_\beta\}_{\beta\in[0,\beta^\star]}$ be the $\beta$-scaled graphs from Definition 3.3, with spanning tree distributions $\{\pi_\beta\}_{\beta\in[0,\beta^\star]}$.*

*Suppose the admissible spanning tree $S$ is stored explicitly together with its edge weights. Then there is a quantum algorithm that, for any $\eta > 0$, computes an $e^2$-slowly varying schedule $0 = \beta_0 < \cdots < \beta_l = \beta^\star$ of length $l \le \widetilde{O}(\sqrt{n})$ with $\beta_0 = 0$, $\beta_l = \beta^\star$ with probability at least $1-\eta$. The algorithm uses $\widetilde{O}(n \cdot \log(1/\eta)\log(1/\epsilon))$ applications of the operators $W_\beta$ obtained by applying Proposition 3.16 to $G_\beta$.*

*Proof.* By Lemma 3.19, $Z_S(\beta)$ is log-convex in $\beta$. The state $|S\rangle$ can be easily prepared since $S$ is known explicitly, and by Lemma 3.17, $\||\pi_{\beta^\star}\rangle - |\pi_\infty\rangle\|_2 \le \epsilon/2$. Thus, by Proposition 3.20, there exists an $e^2$-slowly varying schedule from 0 to $\beta^\star$ of length $l \le \widetilde{O}(\sqrt{\ln(Z(0)/Z(\beta^\star))}) \le \widetilde{O}(\ln(\sqrt{1/\pi(S)})) \le \widetilde{O}(\sqrt{n})$, where the second inequality follows from $Z(\beta^\star) \ge w(S)$ and the last from the fact that an $n$-vertex graph has at most $n^{n-2}$ spanning trees. By Lemma 3.14, the corresponding Markov chain $\mathcal{M}_\beta$ has spectral gap at least $\widetilde{\Omega}(1/n)$, while Proposition 3.16 gives efficient implementations of the corresponding quantum walk operators $W_\beta$. Hence, the schedule

can be computed using $\widetilde{O}(\sqrt{n}\cdot\sqrt{n}\cdot\log(1/\eta)\log(1/\epsilon)) = \widetilde{O}(n\log(1/\eta)\log(1/\epsilon))$ applications of $W_\beta$ and $W_\beta^{-1}$. □

### 3.5 Time complexity

For completeness, we briefly sketch the time complexity of Algorithm 1, in the model where the algorithm has QRAM access to classical data it has computed and stored, so each coherent lookup into a stored table costs polylog$(n)$ time. Write $\mathcal{T}_W$ for the time to implement one application of the walk operator $W_\beta$ (or its inverse) from Proposition 3.16. In the preprocessing phase, the admissible-tree computation and the reusable effective-resistance data structure $\mathcal{H}$ (sparsification) take time $\widetilde{O}(\sqrt{mn})$, matching their query complexity. The reusable marginal data structure $\mathcal{L}_\rho$ is built by repeated Grover search over the edges, so it also takes time $\widetilde{O}(\sqrt{mn/\rho})$, matching its query complexity. Converting $S$ to canonical form and reading the vertex degrees take $\widetilde{O}(n)$ time. For each temperature $\beta$ at which we need $W_\beta$, we construct the effective-resistance oracle $O_{\mathcal{R}}^{G_\beta}$ from $\mathcal{H}$ in $\widetilde{O}(n)$ time, and each subsequent evaluation of the oracle takes $\widetilde{O}(1)$ time. Finally, computing the slowly-varying schedule uses $\widetilde{O}(n)$ applications of the operators $W_\beta$ (Proposition 3.21), for a time of $\widetilde{O}(n\cdot\mathcal{T}_W)$, plus the time to construct the effective-resistance oracles at the temperatures visited, namely $\widetilde{O}(n\cdot l)$ where $l$ is the length of the slowly-varying schedule which is $\widetilde{O}(\sqrt{n})$. Thus it takes at most $\widetilde{O}(n^{3/2})$.

We next bound $\mathcal{T}_W$. Implementing $W_\beta$ requires a marginal-state preparation for the up step and a cycle-sampling procedure for the down step. Given $\mathcal{L}_\rho$ and the effective-resistance oracle in QRAM, the marginal state $|\mu_{G_\beta}\rangle$ is prepared using $\widetilde{O}(\sqrt{m\rho/n})$ queries (Proposition 3.11). Each query is a stored-table lookup, and the state is loaded with the tree-based Grover–Rudolph method, so each costs $O(\log m)$ time, and the preparation takes $\widetilde{O}(\sqrt{m\rho/n})$ time in total. For the down step, adding the sampled edge to the current tree $T$ creates a unique cycle. We find it by a breadth-first search on $T$ in $O(n)$ time, and then sample an edge of the cycle from the required distribution. This is done by building a binary tree over the cycle edges whose internal nodes store partial sums of the (inverse) weights, which takes $O(n)$ time to build and $O(\log n)$ time to sample from. Hence

$$\mathcal{T}_W = \widetilde{O}\big(\sqrt{m\rho/n}+n\big).$$

Combining these bounds, the preprocessing time is

$$\widetilde{O}\Big(\sqrt{mn/\rho}+n\cdot\mathcal{T}_W+n\cdot l\Big) = \widetilde{O}\Big(\sqrt{mn/\rho}+\sqrt{mn\rho}+n^{3/2}\Big) = \widetilde{O}\Big(\sqrt{mn/\rho}+n^2\Big),$$

where the first equality uses $l\le\widetilde{O}(\sqrt{n})$ and the second uses $\rho\le 1$. The time to prepare one q-sample is $\widetilde{O}(n)$ walk operators (there are $l=\widetilde{O}(\sqrt{n})$ transformations, each using $\widetilde{O}(\sqrt{n})$ walk operators) plus $\widetilde{O}(l\cdot n)=\widetilde{O}(n^{3/2})$ for constructing the effective-resistance oracles along the schedule, that is

$$\widetilde{O}\big(n\cdot\mathcal{T}_W\big) = \widetilde{O}\big(\sqrt{mn\rho}+n^2\big) = \widetilde{O}(n^2),$$

where the last step uses $m\le n^2$, so that $\sqrt{mn\rho}\le n^{3/2}$. Similarly, a reflection about $|\pi\rangle$ takes $\widetilde{O}(\sqrt{n}\cdot\mathcal{T}_W)=\widetilde{O}(\sqrt{m\rho}+n^{3/2})$ time. Note that the time is dominated by the $\widetilde{O}(n)$ cost of the down step in every walk operator, not by the query cost. As a result, the time complexity of q-sampling is $\widetilde{O}(n^2)$ for every $\rho$, and the tradeoff in $\rho$ is visible in the query complexity but not in the time complexity. Matching the query tradeoff in time would require implementing the down step faster than $\widetilde{O}(n)$.

## 4 Lower bound for $k$ q-samples

Here we prove our lower bound, showing that the algorithm's query complexity is optimal, up to polylogarithmic factors.

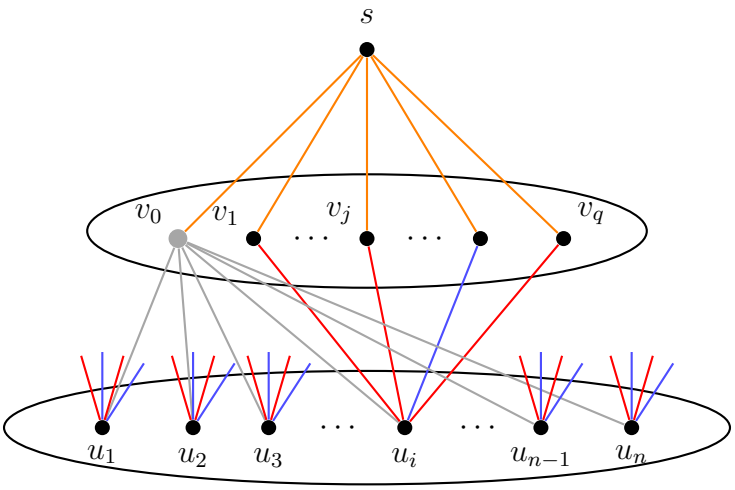



Figure 1: Pictorial representation of the graph construction for the lower bound. The $n$ vertices $u_1, \ldots, u_n$ encode $n$ independent $k$-threshold instances $x^{(1)}, \ldots, x^{(n)} \in \{0,1\}^q$ on $q = \lceil 2m/n \rceil$ bits. For each $i \in [n]$ and $j \in [q]$, the edge $(u_i, v_j)$ has weight $x_j^{(i)}$: red edges have weight 1 and correspond to $x_j^{(i)} = 1$, while blue edges have weight 0 and correspond to $x_j^{(i)} = 0$. The vertex $v_0$ (gray) is joined to every $u_i$ by an edge of weight 1. The orange edges connect $s$ to every $v_j$, $0 \le j \le q$, and all have the fixed weight $\Lambda = 12m$.

We first prove a lower bound for obtaining $k$ independent classical samples from the spanning tree distribution $\pi$.

**Lemma 4.1.** *Let $G = (V, E, w)$ be a weighted graph with $n$ vertices, $m$ edges and $w \in \mathbb{R}^E_{\ge 0}$, let $k \le m/n$ be a positive integer, and let $\epsilon \le \frac{1}{12k}$. Any quantum algorithm that, given access to $\mathcal{O}_G$, the adjacency-list oracle, outputs $k$ independent samples from a distribution $\widetilde{\pi}$ which satisfies $\|\widetilde{\pi} - \pi\|_{\mathrm{TV}} \le \epsilon$, where $\pi$ is the spanning tree distribution of $G$ must make $\Omega(\sqrt{kmn}/\log(nk))$ queries to $\mathcal{O}_G$ in the worst-case.*

*Proof.* Let $q = \lceil 2m/n \rceil$, so $q \ge 2k$ since $k \le m/n$. We reduce from $n$ independent instances $x^{(1)}, \ldots, x^{(n)} \in \{0,1\}^q$ of the $k$-threshold problem, each defined on $q$ bits. In each instance $x^{(i)}$, the goal is to output 1 iff $|x^{(i)}| \ge k$ i.e., given quantum query access to $x^{(i)}$ through the oracle $\mathcal{O}_{x^{(i)}} \colon |j, b\rangle \mapsto |j, b \oplus x_j^{(i)}\rangle$. The graph $G$ (see Figure 1) has vertices $s, v_0, \ldots, v_q, u_1, \ldots, u_n$ and edges $(s, v_j)$ of weight $\Lambda := 12m$ for $0 \le j \le q$, $(u_i, v_0)$ of weight 1 for $i \in [n]$, and $(u_i, v_j)$ of weight $x_j^{(i)}$ for $i \in [n]$, $j \in [q]$.

It has $n + q + 2 = O(n)$ vertices and $q + 1 + n(q+1) = \Theta(m)$ edges. The vertex degrees do not depend on $x$, and each query to $\mathcal{O}_G$ can be simulated with at most one query to some $\mathcal{O}_{x^{(i)}}$ (all other edges are fixed). The dummy vertex $v_0$ keeps $G$ connected even when some $x^{(i)} = 0^q$, so $\pi$ is always defined. Let $J_i = \{0\} \cup \{j \in [q] : x_j^{(i)} = 1\}$, so $|J_i| = |x^{(i)}| + 1$.

We first show that most trees contain all edges joining $s$. Call a spanning tree $T$ *good* if $(s, v_j) \in T$ for all $0 \le j \le q$. We know that the marginal of $(s, v_j)$ equals the leverage score,

$$\Pr_{T \sim \pi}[(s, v_j) \in T] = \ell((s, v_j)) = \Lambda \cdot R^G_{\mathrm{eff}}(s, v_j).$$

For a fixed $v_j$, *short* all its neighbors other than $s$ into $s$, i.e., identify these vertices with $s$. By Rayleigh's Shorting law [DS84], this can only decrease the effective resistance between $s$ and $v_j$. The resulting network consists of the edge $(s, v_j)$, with resistance $1/\Lambda$, in parallel with $d_j \le n$ unit-resistance edges, where $d_j = \left|\{i \in [n] : x_j^{(i)} = 1\}\right|$. Thus $R_{\text{eff}}^G(s, v_j) \ge 1/(n+\Lambda)$, so $\Pr_{T\sim\pi}[(s, v_j) \notin T] \le n/(n+\Lambda) \le n/\Lambda$. A union bound over the $q+1$ edges gives

$$\Pr_{T\sim\pi}[T \text{ not good}] \le \frac{(q+1)n}{\Lambda} \le \frac{4m}{\Lambda} = \frac{1}{3}.$$

Next, we show that a good tree reveals a uniformly random element of $J_i$ for each instance $x^{(i)}$. Recall that a spanning tree $T$ is sampled with probability proportional to its weight, so any tree containing a zero-weight edge has weight zero and occurs with probability zero. Thus, we only need to consider positive-weight edges.

Condition on a good spanning tree $T$. Since all edges $(s, v_j)$ are present, every $v_j$ is already connected to $s$. Hence, each vertex $u_i$ needs exactly one incident edge to be connected to the rest of the tree. It cannot have two such edges, since this would create a cycle through $s$. Therefore, $u_i$ has exactly one incident edge, say $(u_i, v_{j_i(T)})$. For this edge to have positive weight, we must have $j_i(T) \in J_i$.

Conversely, every choice of $j_i \in J_i$ gives a good spanning tree of the same weight. Therefore, conditioned on $T$ being good, $j_i(T)$ is uniformly distributed over $J_i$. If $T$ is good and $j_i(T) \ne 0$, we say that $T$ *reveals* the index $j_i(T)$ for $x^{(i)}$; otherwise $T$ reveals no index for $x^{(i)}$. A revealed index $j$ satisfies $x_j^{(i)} = 1$, since $j_i(T) \in J_i$.

For each instance $i$, let $T_1, T_2, \ldots$ be independent samples from $\widetilde{\pi}$, and let $\mathcal{I}_i(t)$ be the set of indices revealed for $x^{(i)}$ by $T_1, \ldots, T_t$. Then $\mathcal{I}_i(t) \subseteq \mathcal{I}_i(t+1)$ and $|\mathcal{I}_i(t)| \le |x^{(i)}|$.

Suppose $|x^{(i)}| \ge k$. Fix $t$ and an outcome of $T_1, \ldots, T_t$ with $|\mathcal{I}_i(t)| < k$, and let NEW be the event that $T_{t+1}$ reveals an index outside $\mathcal{I}_i(t)$. Since $T_{t+1} \sim \widetilde{\pi}$ is independent of $T_1, \ldots, T_t$, the probability $\Pr[\text{NEW} \mid T_1, \ldots, T_t]$ is the probability that a fresh sample from $\widetilde{\pi}$ reveals an index outside the set $\mathcal{I}_i(t)$. For a tree $T \sim \pi$, $T$ is good with probability at least $2/3$, and conditioned on being good, $j_i(T)$ is uniform over $J_i$, which has $|x^{(i)}| + 1$ elements, exactly $|x^{(i)}| - |\mathcal{I}_i(t)|$ of which are new indices. Hence the probability that $T \sim \pi$ reveals an index outside $\mathcal{I}_i(t)$ is at least

$$\frac{2}{3} \cdot \frac{|x^{(i)}| - |\mathcal{I}_i(t)|}{|x^{(i)}| + 1} = \frac{2}{3}\left(1 - \frac{|\mathcal{I}_i(t)| + 1}{|x^{(i)}| + 1}\right) \ge \frac{2}{3}\left(1 - \frac{|\mathcal{I}_i(t)| + 1}{k+1}\right) = \frac{2}{3} \cdot \frac{k - |\mathcal{I}_i(t)|}{k+1}.$$

Our samples, however, come from $\widetilde{\pi}$, and $\|\widetilde{\pi} - \pi\|_{\text{TV}} \le \epsilon$ changes the probability of any event by at most $\epsilon$, and $\epsilon \le 1/(12k) \le (k - |\mathcal{I}_i(t)|)/(12k)$ and $k + 1 \le 2k$,

$$\Pr[\text{NEW} \mid T_1, \ldots, T_t] \;\ge\; \frac{2(k - |\mathcal{I}_i(t)|)}{3(k+1)} - \frac{k - |\mathcal{I}_i(t)|}{12k} \;\ge\; \frac{k - |\mathcal{I}_i(t)|}{4k}.$$

Let $Y_t := \max\{k - |\mathcal{I}_i(t)|, 0\}$ be the number of indices still missing, so that $|\mathcal{I}_i(t)| < k$ if and only if $Y_t \ge 1$. We claim that $\mathbb{E}[Y_{t+1} \mid T_1, \ldots, T_t] \le \left(1 - \frac{1}{4k}\right) Y_t$. If $Y_t = 0$, then $Y_{t+1} = 0$. Otherwise, since $\mathcal{I}_i(t) \subseteq \mathcal{I}_i(t+1)$, we always have $Y_{t+1} \le Y_t$, and on NEW we have $Y_{t+1} \le Y_t - 1$. Thus $Y_{t+1} \le Y_t - \mathbb{1}_{\text{NEW}}$, and by the bound above $\Pr[\text{NEW} \mid T_1, \ldots, T_t] \ge Y_t/(4k)$, so $\mathbb{E}[Y_{t+1} \mid T_1, \ldots, T_t] \le Y_t - Y_t/(4k) = \left(1 - \frac{1}{4k}\right) Y_t$. Averaging over all outcomes of $T_1, \ldots, T_t$ gives $\mathbb{E}[Y_{t+1}] \le \left(1 - \frac{1}{4k}\right) \mathbb{E}[Y_t]$, and since $Y_0 = k$, induction on $t$ gives $\mathbb{E}[Y_r] \le k\left(1 - \frac{1}{4k}\right)^r$. By Markov's inequality, and choosing $r = k\lceil 8\log(nk)\rceil \ge 8k\log(nk)$,

$$\Pr_{(T_1,\ldots,T_r)\sim\widetilde{\pi}^{\otimes r}}[|\mathcal{I}_i(r)| < k] = \Pr[Y_r \ge 1] \le \mathbb{E}[Y_r] \le k\left(1 - \tfrac{1}{4k}\right)^r \le k\, e^{-r/(4k)} \le k(nk)^{-2}.$$

We now describe an algorithm for the $k$-threshold problem on $n$ independent instances, given quantum query access to $x^{(1)}, \ldots, x^{(n)}$. It runs the assumed sampling algorithm independently

$\lceil 8\log(nk)\rceil$ times, answering each query to $\mathcal{O}_G$ with at most one query to some $\mathcal{O}_{x^{(i)}}$. This gives $r = k\lceil 8\log(nk)\rceil$ independent samples $T_1, \ldots, T_r$ from $\widetilde{\pi}$. From these trees alone, without further queries, it computes $\mathcal{I}_i(r)$ for every $i \in [n]$ and outputs $(b_1, \ldots, b_n)$, where $b_i = 1$ if and only if $|\mathcal{I}_i(r)| \geq k$. If the sampling algorithm makes $Q$ queries to $\mathcal{O}_G$, this algorithm makes $O(Q\log(nk))$ queries.

The algorithm succeeds if $b_i = 1$ exactly when $|x^{(i)}| \geq k$ for all $i \in [n]$ and it has one-sided error if every output $b_i = 1$ is correct, that is, if $b_i = 1$ only when $|x^{(i)}| \geq k$. Our algorithm has one-sided error: if $|x^{(i)}| < k$, then $|\mathcal{I}_i(r)| \leq |x^{(i)}| < k$, since every revealed index $j$ satisfies $x^{(i)}_j = 1$, so $b_i = 0$ always. Hence it can fail only on an instance with $|x^{(i)}| \geq k$, where $b_i = 0$ with probability at most $k(nk)^{-2}$ by the bound above. A union bound over at most $n$ such instances shows that it succeeds with probability at least $1 - nk \cdot (nk)^{-2} = 1 - 1/(nk) \geq 1/2$.

Since $k \leq q/2$, the bounded-error quantum query complexity of the $k$-threshold function on $q$ bits is $\Theta(\sqrt{kq})$ [BBC+01]. By the direct product theorem for one-sided-error algorithms [AŠW07, Theorem 9], any one-sided-error quantum algorithm for $n$ independent instances of the $k$-threshold problem that makes at most $\alpha n\sqrt{kq}$ queries, for a sufficiently small constant $\alpha > 0$, succeeds with probability at most $2^{-\Omega(nk)} < 1/2$. Our algorithm succeeds with probability at least $1/2$, so it makes $\Omega(n\sqrt{kq}) = \Omega(\sqrt{kmn})$ queries. Hence $Q\log(nk) = \Omega(\sqrt{kmn})$, i.e.,

$$Q = \Omega\left(\frac{\sqrt{kmn}}{\log(nk)}\right).$$

□

Since measuring a q-sample yields a classical sample, the bound extends to q-samples.

**Theorem 4.2.** *Let $\epsilon \leq 1/(12k)$, let $n \leq m$ with $n$ sufficiently large, and let $k \leq m/n$. Any algorithm that, given adjacency-list access $\mathcal{O}_G$ to a weighted graph $G = (V, E, w)$ with $n$ vertices, $m$ edges and $w \in \mathbb{R}^E_{\geq 0}$, prepares $k$ independent copies of $|\widetilde{\pi}\rangle$ with $\||\widetilde{\pi}\rangle - |\pi\rangle\|_2 \leq \epsilon$ makes $\Omega(\sqrt{kmn}/\log(nk))$ queries to $\mathcal{O}_G$ in the worst case.*

*Proof.* By the standard relation between total variation distance and $L_2$ norm between quantum states, we have $\||\widetilde{\pi}\rangle - |\pi\rangle\|_{\mathrm{TV}} \leq \||\widetilde{\pi}\rangle - |\pi\rangle\|_2 \leq \epsilon$. Measuring in the computational basis (the basis of spanning trees) yields a classical sample of a spanning tree from the distribution $|\widetilde{\pi}\rangle$ which is within TV distance $\epsilon$ of $\pi$. Measuring each of the $k$ independent copies gives $k$ independent classical samples from it, at no extra query cost, so Lemma 4.1 applies. □

We note that the restriction $k \leq m/n$ is necessary: reading the whole adjacency list takes $O(m)$ queries and afterwards any number of samples (or copies of $|\pi\rangle$) can be produced without further queries, so no bound above $O(m)$ is possible, and $\sqrt{kmn} > m$ exactly when $k > m/n$. For $k > m/n$, an algorithm producing $k$ samples also produces $k' = \lfloor m/n \rfloor$ samples, so Lemma 4.1 with $k'$ gives $\Omega(m/\log m)$. Hence the query complexity is $\widetilde{\Omega}(\min\{\sqrt{kmn}, m\})$ for all $k$.

## Acknowledgements

This work was supported by the Maison du Quantique de Nouvelle-Aquitaine "HybQuant", as part of the HQI initiative and France 2030, under the French National Research Agency (ANR) grant ANR-22-PNCQ-0002. It was also supported by the PEPR integrated project EPiQ under ANR grant ANR-22-PETQ-0007. We acknowledge the use of AI-based tools solely for language editing and proofreading.

## References

[AD20] N. Anari and M. Derezinski. "Isotropy and Log-Concave Polynomials: Accelerated Sampling and High-Precision Counting of Matroid Bases". In: *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 1331–1344 (cit. on pp. 2, 5, 13, 14, 16, 30).

[ADVY22] N. Anari, M. Derezinski, T.-D. Vuong, and E. Yang. "Domain Sparsification of Discrete Distributions Using Entropic Independence". In: *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS)*. 2022, 5:1–5:23 (cit. on p. 2).

[AGJK20] A. Ambainis, A. Gilyén, S. Jeffery, and M. Kokainis. "Quadratic Speedup for Finding Marked Vertices by Quantum Walks". In: *Proceedings of the 52nd Symposium on Theory of Computing (STOC)*. 2020, pp. 412–424 (cit. on p. 3).

[AGJL25] S. Apers, M. Gao, Z. Ji, and C. Liu. "Quantum Speedup for Sampling Random Spanning Trees". In: *Proceedings of the 52nd International Colloquium on Automata, Languages, and Programming (ICALP)*. 2025, 13:1–13:21 (cit. on pp. 1, 2, 9, 18, 40).

[Ald90] D. J. Aldous. "The Random Walk Construction of Uniform Spanning Trees and Uniform Labelled Trees". In: *SIAM Journal on Discrete Mathematics* 3.4 (1990), pp. 450–465 (cit. on p. 2).

[ALG+21] N. Anari, K. Liu, S. O. Gharan, C. Vinzant, and T.-D. Vuong. "Log-Concave Polynomials IV: Approximate Exchange, Tight Mixing Times, and Near-Optimal Sampling of Forests". In: *Proceedings of the 53rd Symposium on Theory of Computing (STOC)*. 2021, pp. 408–420 (cit. on pp. 2, 3).

[ALGV19] N. Anari, K. Liu, S. O. Gharan, and C. Vinzant. "Log-Concave Polynomials II: High-Dimensional Walks and an FPRAS for Counting Bases of a Matroid". In: *Proceedings of the 51st Symposium on Theory of Computing (STOC)*. 2019, pp. 1–12 (cit. on pp. 5, 13, 14).

[ALV22] N. Anari, Y. P. Liu, and T.-D. Vuong. "Optimal Sublinear Sampling of Spanning Trees and Determinantal Point Processes via Average-Case Entropic Independence". In: *Proceedings of the 63rd Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 123–134 (cit. on pp. 1–3, 6, 14).

[Amb07] A. Ambainis. "Quantum Walk Algorithm for Element Distinctness". In: *SIAM Journal on Computing* 37.1 (2007), pp. 210–239 (cit. on p. 3).

[Ape19] S. Apers. "Quantum Walk Sampling by Growing Seed Sets". In: *Proceedings of the 27th European Symposium on Algorithms (ESA)*. 2019, 9:1–9:12 (cit. on p. 3).

[AŠW07] A. Ambainis, R. Špalek, and R. de Wolf. "A New Quantum Lower Bound Method, with Applications to Direct Product Theorems and Time-Space Tradeoffs". In: *Algorithmica* 55 (2007), 422–461 (cit. on pp. 8, 35).

[AT07] D. Aharonov and A. Ta-Shma. "Adiabatic Quantum State Generation". In: *SIAM Journal on Computing* 37.1 (2007), pp. 47–82 (cit. on p. 2).

[AW17] S. Arunachalam and R. de Wolf. "A Survey of Quantum Learning Theory". In: *SIGACT News* 48.2 (2017), pp. 41–67 (cit. on p. 2).

[AW20] S. Apers and R. de Wolf. "Quantum Speedup for Graph Sparsification, Cut Approximation and Laplacian Solving". In: *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 637–648 (cit. on pp. 7–9, 11, 18, 40).

[BBC+01] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. "Quantum lower bounds by polynomials". In: *Journal of the ACM* 48 (2001), 778–797 (cit. on p. 35).

[BBL09] J. Borcea, P. Brändén, and T. Liggett. "Negative Dependence and the Geometry of Polynomials". In: *Journal of the American Mathematical Society* 22.2 (2009), pp. 521–567 (cit. on p. 1).

[BGJ+23] A. Bouland, Y. M. Getachew, Y. Jin, A. Sidford, and K. Tian. "Quantum Speedups for Zero-Sum Games via Improved Dynamic Gibbs Sampling". In: *Proceedings of the 40th International Conference on Machine Learning (ICML)*. 2023, pp. 2932–2952 (cit. on p. 3).

[BHMT02] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. "Quantum Amplitude Amplification and Estimation". In: *Contemporary Mathematics* 305 (2002), pp. 53–74 (cit. on pp. 4, 24, 25).

[BJ99] N. H. Bshouty and J. C. Jackson. "Learning DNF over the Uniform Distribution Using a Quantum Example Oracle". In: *SIAM Journal on Computing* 28.3 (1999), pp. 1136–1153 (cit. on p. 2).

[Bro89] A. Z. Broder. "Generating Random Spanning Trees". In: *Proceedings of the 30th Symposium on Foundations of Computer Science (FOCS)*. 1989, pp. 442–447 (cit. on p. 2).

[BV97] E. Bernstein and U. V. Vazirani. "Quantum Complexity Theory". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1411–1473 (cit. on p. 2).

[CGP+20] T. Chu, Y. Gao, R. Peng, S. Sachdeva, S. Sawlani, and J. Wang. "Graph Sparsification, Spectral Sketches, and Faster Resistance Computation via Short Cycle Decompositions". In: *SIAM Journal on Computing* 52.6 (2020), FOCS18–85–FOCS18–157 (cit. on p. 4).

[CH23] A. Cornelissen and Y. Hamoudi. "A Sublinear-Time Quantum Algorithm for Approximating Partition Functions". In: *Proceedings of the 34th Symposium on Discrete Algorithms (SODA)*. 2023, pp. 1245–1264 (cit. on pp. 2, 4, 31).

[Chi17] A. M. Childs. *Lecture Notes on Quantum Algorithms*. Available at `https://www.cs.umd.edu/~amchilds/qa/`. 2017 (cit. on p. 41).

[CKS17] A. M. Childs, R. Kothari, and R. D. Somma. "Quantum Algorithm for Systems of Linear Equations with Exponentially Improved Dependence on Precision". In: *SIAM Journal on Computing* 46.6 (2017), pp. 1920–1950 (cit. on p. 8).

[CLL+22] A. M. Childs, T. Li, J.-P. Liu, C. Wang, and R. Zhang. "Quantum Algorithms for Sampling Log-concave Distributions and Estimating Normalizing Constants". In: *Proceedings of the Advances in Neural Information Processing Systems*. 2022 (cit. on p. 2).

[CLR20] S. Chakraborty, K. Luh, and J. Roland. "Analog Quantum Algorithms for the Mixing of Markov Chains". In: *Physical Review A* 102 (2020), p. 022423 (cit. on p. 3).

[CMN96] C. J. Colbourn, W. J. Myrvold, and E. Neufeld. "Two Algorithms for Unranking Arborescences". In: *Journal of Algorithms* 20.2 (1996), pp. 268–281 (cit. on p. 2).

[CSV23] Z. Chen, D. Stefankovic, and E. Vigoda. *Spectral Independence and Local-to-Global Techniques for Optimal Mixing of Markov Chains*. arXiv:2307.13826 [cs.DM]. 2023 (cit. on pp. 5, 14, 28).

[DH96] C. Dürr and P. Høyer. *A Quantum Algorithm for Finding the Minimum*. arXiv:9607014 [quant-ph]. 1996 (cit. on p. 40).

[DHHM06] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla. "Quantum Query Complexity of Some Graph Problems". In: *SIAM Journal on Computing* 35.6 (2006), pp. 1310–1328 (cit. on pp. 5, 40).

[DPPR20] D. Durfee, J. Peebles, R. Peng, and A. B. Rao. "Determinant-Preserving Sparsification of SDDM Matrices". In: *SIAM Journal on Computing* 49.4 (2020), FOCS17–350–FOCS17–408 (cit. on p. 2).

[DS84] P. G. Doyle and J. L. Snell. *Random Walks and Electric Networks*. Vol. 22. Carus Mathematical Monographs. American Mathematical Society, 1984 (cit. on pp. 1, 34).

[Fos49] R. M. Foster. "The Average Impedance of an Electrical Network". In: *Contributions to Applied Mechanics (Reissner Anniversary Volume)* 333 (1949), pp. 333–340 (cit. on p. 11).

[GR02] L. K. Grover and T. Rudolph. *Creating Superpositions that Correspond to Efficiently Integrable Probability Distributions*. arXiv:0208112 [quant-ph]. 2002 (cit. on pp. 24, 29).

[Gro05] L. K. Grover. "Fixed-Point Quantum Search". In: *Physical Review Letters* 95 (2005), p. 150501 (cit. on pp. 7, 40).

[Gro97] L. K. Grover. "Quantum Mechanics Helps in Searching for a Needle in a Haystack". In: *Physical Review Letters* 79.2 (1997), pp. 325–328 (cit. on p. 40).

[Gué83] A. Guénoche. "Random Spanning Tree". In: *Journal of Algorithms* 4.3 (1983), pp. 214–220 (cit. on p. 2).

[HHL09] A. W. Harrow, A. Hassidim, and S. Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Physical Review Letters* 103.15 (2009), p. 150502 (cit. on p. 8).

[HW20] A. W. Harrow and A. Y. Wei. "Adaptive Quantum Simulated Annealing for Bayesian Inference and Estimating Partition Functions". In: *Proceedings of the 31st Symposium on Discrete Algorithms (SODA)*. 2020, pp. 193–212 (cit. on pp. 2, 5, 15, 18, 31).

[IKK+26] Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. "Loss Minimization for Electrical Flows over Spanning Trees on Grids". In: *Theoretical Computer Science* 1083 (2026), p. 116132 (cit. on p. 4).

[Kir47] G. Kirchhoff. "Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird". In: *Annalen der Physik* 148.12 (1847), pp. 497–508 (cit. on p. 1).

[KM09] J. A. Kelner and A. Mądry. "Faster Generation of Random Spanning Trees". In: *Proceedings of the 50th Symposium on Foundations of Computer Science (FOCS)*. 2009, pp. 13–21 (cit. on p. 2).

[Kul90] V. G. Kulkarni. "Generating Random Combinatorial Objects". In: *Journal of Algorithms* 11.2 (1990), pp. 185–207 (cit. on p. 2).

[LP17a] D. Levin and Y. Peres. *Markov Chains and Mixing Times*. 2nd. American Mathematical Society, 2017 (cit. on p. 27).

[LP17b] R. Lyons and Y. Peres. *Probability on Trees and Networks*. Cambridge University Press, 2017 (cit. on pp. 1, 11).

[LPY25] Y. P. Liu, R. Peng, and J. Yang. *Approximate Spanning Tree Counting from Uncorrelated Edge Sets*. arXiv:2505.14666 [cs.DS]. 2025 (cit. on p. 4).

[LS23] X. Li and Y. Shang. "Faster Quantum Sampling of Markov Chains in Nonregular Graphs with Fewer Qubits". In: *Physical Review A* 107 (2023), p. 022432 (cit. on p. 3).

[MNRS11] F. Magniez, A. Nayak, J. Roland, and M. Santha. "Search via Quantum Walk". In: *SIAM Journal on Computing* 40.1 (2011), pp. 142–164 (cit. on pp. 3, 4, 12).

[MSS07] F. Magniez, M. Santha, and M. Szegedy. "Quantum Algorithms for the Triangle Problem". In: *SIAM Journal on Computing* 37.2 (2007), pp. 413–424 (cit. on p. 3).

[MST15] A. Mądry, D. Straszak, and J. Tarnawski. "Fast Generation of Random Spanning Trees and the Effective Resistance Metric". In: *Proceedings of the 26th Symposium on Discrete Algorithms (SODA)*. 2015, pp. 2019–2036 (cit. on p. 2).

[OBD18] D. Orsucci, H. J. Briegel, and V. Dunjko. "Faster Quantum Mixing for Slowly Evolving Sequences of Markov Chains". In: *Quantum* 2 (2018), p. 105 (cit. on p. 2).

[ORR13] M. Ozols, M. Roetteler, and J. Roland. "Quantum Rejection Sampling". In: *ACM Transactions on Computation Theory* 5.3 (2013), 11:1–11:33 (cit. on p. 24).

[PS17] R. Portugal and E. Segawa. "Connecting Coined Quantum Walks with Szegedy's Model". In: *Interdisciplinary Information Sciences* 23.1 (2017), pp. 119–125 (cit. on p. 12).

[RTF18] L. M. S. Russo, A. S. Teixeira, and A. P. Francisco. "Linking and Cutting Spanning Trees". In: *Algorithms* 11.4 (2018), p. 53 (cit. on p. 3).

[SBBK08] R. D. Somma, S. Boixo, H. Barnum, and E. Knill. "Quantum Simulations of Classical Annealing Processes". In: *Physical Review Letters* 101 (2008), p. 130504 (cit. on p. 2).

[Sch18] A. Schild. "An Almost-Linear Time Algorithm for Uniform Random Spanning Tree Generation". In: *Proceedings of the 50th Symposium on Theory of Computing (STOC)*. 2018, pp. 214–227 (cit. on p. 2).

[SS11] D. A. Spielman and N. Srivastava. "Graph Sparsification by Effective Resistances". In: *SIAM Journal on Computing* 40.6 (2011), pp. 1913–1926 (cit. on pp. 7, 11).

[ŠVV09] D. Štefankovič, S. S. Vempala, and E. Vigoda. "Adaptive Simulated Annealing: A Near-Optimal Connection between Sampling and Counting". In: *Journal of the ACM* 56.3 (2009), pp. 1–36 (cit. on p. 15).

[Sze04] M. Szegedy. "Quantum Speed-Up of Markov Chain Based Algorithms". In: *Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS)*. 2004, pp. 32–41 (cit. on pp. 4, 12).

[WA08] P. Wocjan and A. Abeyesinghe. "Speedup via Quantum Sampling". In: *Physical Review A* 78.4 (2008), p. 042336 (cit. on pp. 2, 12, 40).

[Wil96] D. B. Wilson. "Generating Random Spanning Trees more Quickly than the Cover Time". In: *Proceedings of the 28th Symposium on Theory of Computing (STOC)*. 1996, pp. 296–303 (cit. on p. 2).

[Won17a] T. G. Wong. "Coined quantum walks on weighted graphs". In: *Journal of Physics A: Mathematical and Theoretical* 50 (2017), p. 475301 (cit. on pp. 12, 13).

[Won17b] T. G. Wong. "Equivalence of Szegedy's and Coined Quantum Walks". In: *Quantum Information Processing* 16.9 (2017) (cit. on pp. 12, 13).

# A Useful quantum sub-routines

**Lemma A.1** (Repeated Grover search, [AW20; Gro97]). *Let $f : [N] \to \{0,1\}$ be a function that marks a set of elements $S = \{i \in [N] \mid f(i) = 1\}$. Then for any $\eta > 0$, there is a quantum algorithm that finds $S$ with probability at least $1 - \eta$ in $\widetilde{O}(\sqrt{N \cdot |S|})$ elementary operations and queries to $f$, and uses $O(\log N)$ qubits and a QRAM of $\widetilde{O}(|S|)$ bits.*

We use Grover's $\frac{\pi}{3}$-amplitude amplification (fixed point search, [Gro05]) as in [WA08] to move between states in the cooling schedule.

**Lemma A.2** (Fixed-point amplitude amplification; adapted from [WA08, Corollary 1]). *Let $|\psi\rangle$ and $|\phi\rangle$ be arbitrary quantum states in $\mathbb{C}^d$ with $|\langle\psi \,|\, \phi\rangle| \geq B$. Denote by $\Pi_\psi$ the projection on the subspace spanned by $|\psi\rangle$ and by $\Pi_\psi^\perp$ the projection onto the orthogonal subspace. Let $\omega = e^{\frac{\pi}{3}i}$. Define the unitaries $R_0 = \omega\Pi_\psi + \Pi_\psi^\perp$ and $R_1 = \omega\Pi_\phi + \Pi_\phi^\perp$. Given the state $|\psi\rangle$, we can prepare a state $|\widetilde{\phi}\rangle$ such that $\left\||\widetilde{\phi}\rangle - |\phi\rangle\right\|_2 \leq \epsilon_1$, for any $\epsilon_1 > 0$, by invoking the unitaries from $R_0$ and $R_1$ no more than*

$$L = \frac{12 \log(2/\epsilon_1)}{\log(1/(1-B))}$$

*times.*

**Lemma A.3** (Q-sample transformation from quantum walks; adapted from [WA08, Corollary 1, Corollary 2]). *Let $W_\psi$ and $W_\phi$ be quantum walk operators whose unique 1-eigenvectors are $|\psi\rangle$ and $|\phi\rangle$, respectively, and suppose their phase gaps are at least $\gamma_{\text{ph}}$. Assume $|\langle\psi \,|\, \phi\rangle| \geq B$. Then, for any $\epsilon > 0$, given $|\psi\rangle$, there is a quantum circuit that outputs a state $|\widetilde{\phi}\rangle$ satisfying $\left\||\widetilde{\phi}\rangle - |\phi\rangle\right\|_2 \leq \epsilon$ using*

$$\widetilde{O}\left(\frac{1}{\gamma_{\text{ph}}} \cdot \frac{\log(1/\epsilon)}{\log(1/(1-B))}\right)$$

*controlled applications of $W_\psi, W_\phi$ and their inverses.*

**Theorem A.4** (Quantum algorithm for finding a minimum spanning tree, Theorem 4.2 in [DHHM06]). *There exists a quantum algorithm that, for any $\eta > 0$, given adjacency query access $\mathcal{O}_G$ to an undirected weighted graph $G = (V, E, w)$ with $n$ vertices and $m$ edges, with probability at least $1 - \eta$, outputs a minimum spanning tree of $G$. The algorithm uses $O(\sqrt{mn}\log(1/\eta))$ queries to $\mathcal{O}_G$ and runs in $\widetilde{O}(\sqrt{mn})$ time.*

**Theorem A.5** (Quantum Maximum-Weight-Product Spanning Tree; adapted from [DHHM06; AGJL25]). *There exists a quantum algorithm that, for any $\eta > 0$, given adjacency query access $\mathcal{O}_G$ to an undirected weighted graph $G = (V, E, w)$ with $n$ vertices and $m$ edges, outputs a maximum-weight-product spanning tree of $G$ with probability at least $1 - \eta$. The algorithm uses $O(\sqrt{mn}\log(1/\eta))$ queries to $\mathcal{O}_G$ and runs in $\widetilde{O}(\sqrt{mn})$ time.*

*Proof.* The proof is almost the same as in [AGJL25, Corollary 31]. A maximum weight-product spanning tree of $G(V, E, w)$ corresponds to a minimum spanning tree of the graph $G'(V, E, w')$, with $w'(e) = \log(\frac{w_{\max}}{w(e)})$ for all $e \in E$ where $w_{\max} = \max_{e \in E} w(e)$. Given query access to $G$, one can compute $w_{\max}$ using $O(\sqrt{m})$ queries to $\mathcal{O}_G$ [DH96]. Moreover, a query access to $G'$ can be constructed in $\widetilde{O}(1)$ time. Therefore, by applying Theorem A.4 on $G'$, we get the claimed result. □

# B Coined quantum walk

We restate and prove Theorem 2.16. We use the notation and labelled-transition setup from Section 2.4.

**Theorem 2.16** (Spectral mapping for labelled Szegedy walks)**.** *Let $P$ be the transition matrix of a reversible, irreducible, and aperiodic Markov chain on $\mathcal{X}$, with $\pi$ as the stationary distribution. Then, on the busy subspace, the unique 1-eigenvector of $W(P)$ is $|\pi\rangle|0\rangle$. Moreover,*

$$\gamma_{\mathrm{ph}}(W(P)) = \Omega(\sqrt{\gamma_{\mathrm{abs}}(P)}).$$

*Proof.* The proof follows the standard spectral analysis of Szegedy walks; see, e.g., [Chi17, Lecture 18]. We recall the notation from the main text. Let $\mathcal{A} = \mathrm{span}\{|x\rangle|0\rangle : x \in \Delta\}$, $\mathcal{B} = \mathcal{U}^\dagger \mathcal{S} U \mathcal{A}$ and let $R_{\mathcal{A}}$ and $R_{\mathcal{B}}$ denote the corresponding reflections, so that $W(P) = R_{\mathcal{B}} R_{\mathcal{A}}$. We identify $\mathcal{A}$ with $\mathbb{C}^\Delta$ via $|x\rangle|0\rangle \leftrightarrow |x\rangle$.

For $x, y \in \Delta$, the labelled-transition construction gives

$$\langle y|\langle 0|\mathcal{U}^\dagger \mathcal{S} \mathcal{U}|x\rangle|0\rangle = \sum_{a \in C_x : h_x(a) = y} \sqrt{p_x(a) p_y(a')}.$$

For $x \neq y$, the unique-label assumption and reversibility therefore give

$$\langle y|\langle 0|\mathcal{U}^\dagger \mathcal{S} \mathcal{U}|x\rangle|0\rangle = \sqrt{P(x,y)P(y,x)} = \sqrt{\frac{\pi(x)}{\pi(y)}} P(x,y).$$

For $x = y$, the self-loop labels contribute $P(x,x)$. Hence, using the discriminant matrix $D(P) := D_\pi^{1/2} P D_\pi^{-1/2}$, $\Pi_{\mathcal{A}} \mathcal{U}^\dagger \mathcal{S} \mathcal{U} \Pi_{\mathcal{A}} = D(P)$.

Let $|v_\theta\rangle \in \mathcal{A}$ be an eigenvector of $D(P)$ with eigenvalue $\theta$. Since $\Pi_{\mathcal{B}} = \mathcal{U}^\dagger \mathcal{S} \mathcal{U} \Pi_{\mathcal{A}} \mathcal{U}^\dagger \mathcal{S} \mathcal{U}$, we obtain $\Pi_{\mathcal{B}}|v_\theta\rangle = \theta \mathcal{U}^\dagger \mathcal{S} \mathcal{U}|v_\theta\rangle$. Writing $|\widetilde{v}_\theta\rangle := R_{\mathcal{B}}|v_\theta\rangle$ gives

$$|\widetilde{v}_\theta\rangle = 2\theta \mathcal{U}^\dagger \mathcal{S} \mathcal{U}|v_\theta\rangle - |v_\theta\rangle, \qquad \Pi_{\mathcal{A}}|\widetilde{v}_\theta\rangle = (2\theta^2 - 1)|v_\theta\rangle.$$

Consequently, $\mathrm{Span}\{|v_\theta\rangle, |\widetilde{v}_\theta\rangle\}$ is invariant under $W(P)$, and in this basis

$$W(P)\big|_{\mathrm{Span}\{v_\theta, \widetilde{v}_\theta\}} = \begin{pmatrix} 0 & -1 \\ 1 & 2(2\theta^2 - 1) \end{pmatrix}.$$

Its eigenvalues satisfy $z^2 - 2(2\theta^2 - 1)z + 1 = 0$. Since this depends on $\theta$ only through $\theta^2$, set $\xi := \arccos|\theta|$, so that $z = e^{\pm 2i\xi}$. For the stationary eigenvector $\pi$, reversibility implies that

$$\mathcal{U}|\pi\rangle|0\rangle = \sum_{x \in \Delta} \sum_{a \in C_x} \sqrt{\pi(x) p_x(a)} |x\rangle|a\rangle$$

is invariant under the flip-flop shift $\mathcal{S}$. Hence $\mathcal{U}^\dagger \mathcal{S} \mathcal{U}|\pi\rangle|0\rangle = |\pi\rangle|0\rangle$, so $|\pi\rangle|0\rangle \in \mathcal{A} \cap \mathcal{B}$ and is a 1-eigenvector of $W(P)$. Since $P$ is irreducible and aperiodic, 1 is a simple eigenvalue of $D(P)$ and all other eigenvalues satisfy $|\theta| < 1$. Thus this is the unique 1-eigenvector on the busy subspace.

Finally, let $\theta \neq 1$ be an eigenvalue of $D(P)$, and let $\xi$ be as above. The eigenvalues of $D(P)$ and $P$ coincide, so $\cos\xi = |\theta| \leq 1 - \gamma_{\mathrm{abs}}(P)$. Since $1 - \cos\xi \leq \xi^2/2$, this gives $\xi \geq \sqrt{2\gamma_{\mathrm{abs}}(P)}$, and hence the eigenphase $2\xi$ of $W(P)$ is at least $2\sqrt{2\gamma_{\mathrm{abs}}(P)}$. Therefore $\gamma_{\mathrm{ph}}(W(P)) = \Omega(\sqrt{\gamma_{\mathrm{abs}}(P)})$. □

## C Cooling schedule

**Lemma 3.19** (Log-convexity of the partition function)**.** *Fix an admissible spanning tree $S$, and define $H(T) := |T \setminus S|$ and $Z(\beta) := \sum_{T\in\Delta} e^{-\beta H(T)} w(T)$. Then $Z(\beta)$ is log-convex in $\beta$.*

*Proof.* Fix $\beta_1, \beta_2 \in \mathbb{R}$ and $\alpha \in [0,1]$. Using $H(T) = |T \setminus S|$, we have

$$Z(\alpha\beta_1 + (1-\alpha)\beta_2) = \sum_{T\in\Delta_G} \left(e^{-\beta_1 H(T)} w(T)\right)^{\alpha} \left(e^{-\beta_2 H(T)} w(T)\right)^{1-\alpha}.$$

By Hölder's inequality with exponents $1/\alpha$ and $1/(1-\alpha)$,

$$Z(\alpha\beta_1 + (1-\alpha)\beta_2) \le \left(\sum_{T\in\Delta} e^{-\beta_1 H(T)} w(T)\right)^{\alpha} \left(\sum_{T\in\Delta} e^{-\beta_2 H(T)} w(T)\right)^{1-\alpha} = Z_S(\beta_1)^{\alpha} Z(\beta_2)^{1-\alpha}.$$

Taking natural logarithms gives $\ln Z(\alpha\beta_1 + (1-\alpha)\beta_2) \le \alpha \ln Z(\beta_1) + (1-\alpha)\ln Z(\beta_2)$, so $\ln Z$ is convex. Hence $Z$ is log-convex. □

## D Spectral sparsifiers

**Proposition D.1** (Sum of Laplacians)**.** *Let $G_1 = (V, E_1, w_1)$ and $G_2 = (V, E_2, w_2)$ be two weighted graphs on a common vertex set $V$. Then, for any $c \ge 0$,*

$$L(G_1 + c \cdot G_2) = L(G_1) + c \cdot L(G_2).$$

*Proof.* Recall that $L(G) = \sum_{e\in E} w(e) b_e b_e^{\top}$. By Definition 2.5, the weight of every edge $e$ in $G_1 + c \cdot G_2$ is $w_1(e) + c\, w_2(e)$, taking the weight to be zero when $e$ is absent from the corresponding graph. Therefore, $L(G_1 + c \cdot G_2) = \sum_e \big(w_1(e) + c\, w_2(e)\big) b_e b_e^{\top} = L(G_1) + c \cdot L(G_2)$. □

**Lemma D.2** (Sparsifier under disjoint edge addition)**.** *Let $G_1 = (V, E_1, w_1)$ and $G_2 = (V, E_2, w_2)$ be weighted graphs on the same vertex set, with $E_1 \cap E_2 = \emptyset$. If $\widetilde{H}$ is an $\epsilon$-spectral sparsifier of $G_1$, then $\widetilde{H} + G_2$ is an $\epsilon$-spectral sparsifier of $G_1 + G_2$.*

*Proof.* Since $\widetilde{H}$ is an $\epsilon$-spectral sparsifier of $G_1$, for every $x \in \mathbb{R}^{|V|}$,

$$(1-\epsilon) x^{\top} L_{G_1} x \le x^{\top} L_{\widetilde{H}} x \le (1+\epsilon) x^{\top} L_{G_1} x.$$

Since $L_{G_2} \succeq 0$,

$$(1-\epsilon) x^{\top} L_{G_2} x \le x^{\top} L_{G_2} x \le (1+\epsilon) x^{\top} L_{G_2} x.$$

Adding the two inequalities gives

$$(1-\epsilon) x^{\top} (L_{G_1} + L_{G_2}) x \le x^{\top} (L_{\widetilde{H}} + L_{G_2}) x \le (1+\epsilon) x^{\top} (L_{G_1} + L_{G_2}) x.$$

By Definition 2.5 and proposition D.1, $L_{G_1+G_2} = L_{G_1} + L_{G_2}$ and $L_{\widetilde{H}+G_2} = L_{\widetilde{H}} + L_{G_2}$. Hence $\widetilde{H} + G_2$ is an $\epsilon$-spectral sparsifier of $G_1 + G_2$. □

**Lemma 3.5** (Sparsifier reusability)**.** *Let $G = (V, E, w)$ be a weighted graph and fix $F \subseteq E$. Let $H$ be the subgraph of $G$ with edge set $E \setminus F$, and suppose that $\widetilde{H}$ is an $\epsilon$-spectral sparsifier of $H$. Then, for every $c \ge 0$, $F + c \cdot \widetilde{H}$ is an $\epsilon$-spectral sparsifier of $G_{E\setminus F, c}$.*

*Proof.* Since $\widetilde{H}$ is an $\epsilon$-spectral sparsifier of $H$, scaling all edge weights by any $c \ge 0$ shows that $c \cdot \widetilde{H}$ is an $\epsilon$-spectral sparsifier of $c \cdot H$. Moreover, by Definition 2.5, $G_{E\setminus F, c} = F + c \cdot H$. Applying Lemma D.2 to $c \cdot H$ and $F$ gives the claim. □