Quantum Speedups for Sampling Random Spanning Trees

Simon Apers[1], Minbo Gao[2,3], Zhengfeng Ji[*4], and Chenghua Liu[2,3]

[1] *Université Paris Cité, CNRS, IRIF, Paris, France*
[2] *Institute of Software, Chinese Academy of Sciences, Beijing, China*

[3] *University of Chinese Academy of Sciences, Beijing, China*

[4] *Department of Computer Science and Technology, Tsinghua University, Beijing, China*

September 14, 2025

Abstract

We present a quantum algorithm for sampling random spanning trees from a weighted graph in $\widetilde{O}(\sqrt{mn})$ time, where n and m denote the number of vertices and edges, respectively. Our algorithm has sublinear runtime for dense graphs and achieves a quantum speedup over the best-known classical algorithm, which runs in $\widetilde{O}(m)$ time. The approach carefully combines, on one hand, a classical method based on "large-step" random walks for reduced mixing time and, on the other hand, quantum algorithmic techniques, including quantum graph sparsification and a sampling-without-replacement variant of Hamoudi's multiple-state preparation. We also establish a matching lower bound, proving the optimality of our algorithm up to polylogarithmic factors. These results highlight the potential of quantum computing in accelerating fundamental graph sampling problems.

1 Introduction

Random spanning trees are among the oldest and most extensively studied probabilistic structures in graph theory, with their origins tracing back to Kirchhoff's matrix-tree theorem from the 1840s [Kir47]. This foundational result established a profound connection between spanning tree distributions and matrix determinants. Beyond graph theory, random spanning trees are deeply intertwined with probability theory [LP17b] and exemplify strongly Rayleigh distributions [BBL09]. Intriguingly, sampling random spanning trees has revealed unexpected and remarkable connections to various areas of theoretical computer science. Notably, they have been instrumental in breakthroughs that overcame long-standing approximation barriers for both the symmetric [GSS11] and asymmetric versions of the Traveling Salesman Problem [AGM+10]. Goyal, Rademacher and Vempala [GRV09] demonstrated their utility in constructing cut sparsifiers. Beyond their significance in theoretical computer science, random spanning trees also play a crucial role in machine learning and statistics. They have been applied to a wide range of tasks, including graph prediction [CBGV+13], network activation detection [SSK13], spatial clustering [TAL19], online recommendation systems [HPVP21], and causal structure learning [DYMZ23, RBM23].

---

[*]Correspondence author. Authors are listed in alphabetical order.

Given its significant theoretical importance and broad applications, the problem of sampling random spanning trees has attracted substantial research interest, leading to a series of advancements in performance and algorithmic techniques [Gue83, Bro89, Ald90, Kul90, Wil96, CMN96, KM09, MST14, HX16, DKP+17, DPPR17, Sch18, ALG+21]. These contributions can be broadly categorized into three main approaches:

- Determinant calculation approaches [Gue83, Kul90, CMN96]: These methods build on Kirchhoff's matrix-tree theorem, which states that the total number of spanning trees in a weighted graph is equal to any cofactor of its graph Laplacian. Leveraging this theorem, one can calculate determinants to randomly select an integer between 1 and the total number of spanning trees and efficiently map this integer to a unique tree. This approach was first employed by Guenoche [Gue83] and Kulkarni [Kul90] to develop an $O(mn^3)$ time algorithm, where m and n denote the number of edges and vertices in the graph, respectively. Later, Colbourn, Myrvold, and Neufeld [CMN96] improved it to an $O(n^\omega)$ time algorithm, where $\omega \approx 2.37$ is the matrix multiplication exponent.

- Approaches based on effective resistances [HX16, DKP+17, DPPR17]: The core insights behind there approaches is that the marginal probability of an edge being in a random spanning tree is exactly equal to the product of the weight and the effective resistance of the edge. Harvey and Xu [HX16] proposed a deterministic $O(n^\omega)$ time algorithm that uses conditional effective resistances to decide whether each edge belongs to the tree, iteratively contracting included edges and deleting excluded ones. Building on this, Durfee, Kyng, Peebles, Rao and Sachdeva [DKP+17] leveraged "Schur complements" to efficiently compute conditional effective resistances, resulting in a faster algorithm with a runtime of $\widetilde{O}(n^{4/3}m^{1/2} + n^2)$. In a separate work, Durfee, Peebles, Peng and Rao [DPPR17] introduced determinant-preserving sparsification, which enables the sampling of random spanning trees in $\widetilde{O}(n^2\epsilon^{-2})$ time, from a distribution with a total variation distance of ϵ from the true uniform distribution.

- Random walk based approaches [Bro89, Ald90, Wil96, KM09, MST14, Sch18, ALG+21]: First introduced independently by Broder [Bro89] and Aldous [Ald90], it was shown that a random spanning tree can be sampled by performing a simple random walk on the graph until all nodes are visited, while retaining only the first incoming edge for each vertex. For unweighted graphs, this results in an $O(mn)$ algorithm. The subsequent works [Wil96, KM09, MST14] focused on accelerating these walks using various techniques involving fast Laplacian solving, electrical flows and Schur complements. This line of research culminated in Schild's work [Sch18], which achieved an almost-linear time algorithm with a runtime of $m^{1+o(1)}$. The current state-of-the-art, presented by Anari, Liu, Oveis Gharan, Vinzant and Vuong [ALG+21], introduces a simple yet elegant approach based on "down-up random walks", achieving a near-optimal algorithm with a runtime $O(m \log^2 m)$. Notably, the analysis of the mixing time is both technically intricate and profound.

1.1 Main Result

In this work, we propose a quantum algorithm for approximately sampling random spanning trees, leading to the theorem below. We let $\mathcal{W}_G$ denote the distribution over spanning trees of G, where each tree is sampled with probability proportional to the product of its edge weights.

Theorem 1 (Quantum algorithm for sampling a random spanning tree). *There exists a quantum algorithm* $\mathsf{QRST}(\mathcal{O}_G, \varepsilon)$ *that, given query access* $\mathcal{O}_G$ *to the adjacency list of a connected graph* $G = (V, E, w)$

(with $|V| = n$, $|E| = m$, $w \in \mathbb{R}^E_{\geqslant 0}$), and accuracy parameter ε, with high probability, outputs a spanning tree T of G drawn from a distribution which is ε-close to $\mathcal{W}_G$ in total variation distance. The algorithm makes $\widetilde{O}(\sqrt{mn}\log(1/\varepsilon))$ queries to $\mathcal{O}_G$, and runs in $\widetilde{O}(\sqrt{mn}\log(1/\varepsilon))$ time.

We also prove a matching lower bound, showing that the runtime of our quantum algorithm is optimal up to polylog-factors.

Theorem 2 (Quantum lower bound for sampling a random spanning tree). *Let $\varepsilon < 1/2$ be a constant. For any graph $G = (V, E, w)$ with $|V| = n$, $|E| = m$, $w \in \mathbb{R}^E_{\geqslant 0}$, consider the problem of sampling a random spanning tree from a distribution ε-close to $\mathcal{W}_G$, given adjacency-list access to G. The quantum query complexity of this problem is $\Omega(\sqrt{mn})$.*

1.2 Techniques

Our algorithm is based on the "down-up random walk" approach from [ALG+21]. Their core idea (already present in earlier work [RTF18]) is to consider a random walk over spanning trees. In each step of the random walk, they randomly remove an edge to split it into two components, and then add a new edge sampled from the edges across the two components proportional to the edge weights. Their crucial contribution is to show that this random walk has a mixing time $\widetilde{O}(n)$, which is sublinear with respect to the number of edges m. Since each step of the down-up random walk can be costly ($\widetilde{O}(m)$, naively), their final algorithm actually considers an "up-down random walk", where first an edge is added and then another edge is removed. While this walk has a larger mixing time $\widetilde{O}(m)$, each step can now be implemented in amortized time $\widetilde{O}(1)$ using link-cut trees.

1.2.1 Idea (and Barrier) for Quantum Speedups

To achieve a sublinear-time quantum algorithm, we could follow [ALG+21] and attempt to quantumly accelerate the mixing time of the up-down walk. However, apart from some special cases [AAKV01, MR02, Ric07], no general quantum speedups for the mixing time of classical random walks are known. As a result, any algorithm with a mixing time of $\widetilde{O}(m)$ does not yield a quantum advantage.

A natural alternative is to revisit the down-up random walk, which has a sublinear mixing time of $\widetilde{O}(n)$, and leverage quantum techniques to speed up the implementation of each step. Specifically, after removing an edge, we can sample an edge between the two resulting components in $\widetilde{O}(\sqrt{m})$ time using Grover search [Gro96], compared to the classical $\widetilde{O}(m)$ time. Unfortunately, this still results in an overall complexity of $\widetilde{O}(\sqrt{m}n) \in \widetilde{\Omega}(m)$, offering no speedup over classical algorithms.

1.2.2 Our Approach

To overcome the above barriers, we use an amortization idea of implementing a "large-step" down-up walk instead of a single-step approach. In a standard down-up walk, a single edge in the spanning tree is changed after each step. In contrast, the large-step down-up walk modifies several edges—approximately $\Theta(n)$—in each step, resulting in a significantly faster mixing time of $\widetilde{O}(1)$. These $\Theta(n)$ edges are sampled from a batch of $\widetilde{O}(n)$ edges, which we select among the m edges in time $\widetilde{O}(\sqrt{mn})$ using quantum search. While the framework appears simple at first glance, implementing each step requires a careful integration of various quantum algorithms. Additionally, analyzing the mixing time relies on deep results from [ALV22]. Below, we outline the main techniques utilized in our approach:

Domain Sparsification Under Isotropy. Domain sparsification [AD20, ADVY22, ALV22] is a framework which aims to reduce the task of sampling from a distribution on $\binom{[m]}{k}$ with $k \ll m$ to sampling from some related distribution on $\binom{[t]}{k}$ for $t \ll m$. For so-called "strongly Rayleigh distributions" [BBL09], the domain size can be reduced to be nearly linear in the input size with $t = \widetilde{O}(k)$ [ALV22]. For random spanning trees, which are a canonical example of a strongly Rayleigh distribution, this reduces the sampling problem from domain $\binom{[m]}{n}$ to $\binom{[t]}{n}$ with $t \in \widetilde{O}(n)$. This provides an opportunity for quantum acceleration, provided that it is easier to sample this subdomain of $\widetilde{O}(n)$ edges.

A key technique enabling efficient sampling of such a subdomain is the "isotropic transformation", which can be interpreted as a discrete analogue of the isotropic position of a continuous distribution [Rud99, LV24]. Originally developed in the context of designing algorithms for sampling from logconcave distributions, the isotropic position uses a linear map to convert a distribution into a standard form, enabling faster and more efficient sampling. In [ALV22], Anari Liu and Vuong describe a discrete isotropic transformation for distributions over $\binom{[m]}{k}$ based on the marginals of individual elements of $[m]$. After this isotropic transformation, domain sparsification reduces to sampling a subdomain uniformly at random from the domain.

For the particular case of sampling random spanning trees, the marginal probability of an edge is given by its effective resistance, and domain sparsification amounts to sampling edges uniformly at random in the isotropic-transformed multigraph (see lemma 18), whose construction is based on the effective resistances of the edges in the original graph.

In our algorithm, rather than explicitly computing this isotropic transformation (whose description size is $\widetilde{O}(m)$ which we cannot tolerate), we utilize a quantum data structure, QResistance, which provides quantum query access to effective resistances, to "implicitly" construct and maintain the isotropic-transformed multigraph. More specifically, with QResistance in place, we can efficiently implement the up step of the walk, which involves sampling a subdomain of $O(n)$ edges uniformly from the implicit isotropic-transformed multigraph, as well as the down step, which requires sampling a random spanning tree from the resulting $\widetilde{O}(n)$-edge subdomain.

Quantum Graph Algorithms and Quantum Multi-Sampling. To achieve the near-optimal quantum speedup, our algorithm incorporates a suite of quantum algorithms as subroutines. During the initialization stage, it utilizes the quantum minimum spanning tree algorithm proposed in [DHHM06] to identify a starting spanning tree, reducing the complexity from $\widetilde{O}(m)$ to $\widetilde{O}(\sqrt{mn})$. For approximating the effective resistance, we adopt the approach in [AdW22] that combines the quantum graph sparsification algorithm with the Spielman-Srivastava toolbox [SS11] to construct an efficient QResistance data structure in $\widetilde{O}(\sqrt{mn})$ time. This data structure provides query access to approximate effective resistances with a cost of $\widetilde{O}(1)$ per query. For the up operation, we develop a variant of quantum search, QIsotropicSample, for sampling multiple edges in the isotropic-transformed graph with up to a quadratic speedup. The key idea behind this algorithm is to leverage a variant of the "preparing many copies of quantum states" technique proposed in [Ham22], utilizing the query access provided by our QResistance. This variant of the Hamoudi's technique can be thought of as a sampling without replacement method, ensuring that the samples are all different. Notably, QIsotropicSample efficiently samples a set of $\widetilde{O}(n)$ edges in $\widetilde{O}(\sqrt{mn})$ time. Leveraging these algorithms, our approach ultimately achieves random spanning tree sampling in $\widetilde{O}(\sqrt{mn})$ time.

1.2.3 Lower Bound

The lower bound follows from a reduction of the problem of finding n marked elements among m elements, whose quantum query complexity is $\Theta(\sqrt{mn})$, to the problem of sampling a uniform spanning tree in a weighted graph. The reduction encodes the search problem into the weights of a fixed graph, in such a way that a uniformly random spanning tree in that graph reveals the marked elements. The argument is similar to the $\Omega(\sqrt{mn})$ lower bound from [DHHM06] on the quantum query complexity of finding a minimum spanning tree.

1.3 Open Questions

Our work raises several interesting questions and future directions, including the following:

- The runtime of our quantum algorithm is tight, up to polylogarithmic factors, for sampling random spanning trees in weighted graphs. Can we potentially improve the runtime for unweighted graphs, for instance, to $\tilde{O}(n)$? This is the case for constructing spanning trees, as was shown in the earlier work [DHHM06]. Note that our $\Omega(\sqrt{mn})$ lower bound applies only to the weighted graph case. We were unable to prove a $\omega(n)$ lower bound for the unweighted case, based on which we conjecture that there exist more efficient algorithms for the unweighted case.

- This work represents the first application of the down-up walk in quantum algorithm design. As demonstrated by a series of breakthroughs in sampling and counting problems for spin systems [AL20, ALG24, CGŠV21, JPV21, AASV21, Liu21, BCC⁺22, ALG22], the down-up walk has proven highly effective in designing classical algorithms for a variety of graph problems. Notable examples include independent set sampling [AL20, ALG24], q-colorings sampling [CGŠV21, JPV21, BCC⁺22], edge-list-colorings sampling [ALG22], planar matching counting and sampling [AASV21], and vertex-list-colorings sampling [Liu21]. Given these successes, it would be intriguing to explore whether quantum speedups can be achieved for these problems.

- Additionally, exploring quantum speedups for determinantal point processes (DPPs) is a natural and meaningful direction. Like random spanning trees, DPPs are strongly Rayleigh distributions, enabling efficient domain sparsification [AD20, ADVY22, ALV22]. They also play a crucial role in applications such as machine learning [KT11, KT⁺12, DKM20], optimization [DBPM20, DKM20, NST22], and randomized linear algebra [DW17, DCMW19, DY24]. Developing quantum algorithms for DPPs could lead to quantum computational advantages in these areas.

2 Preliminaries

For simplicity, we use $[n]$ to represent the set $\{1, 2, \ldots, n\}$ and $[n]_0$ to represent the set $\{0, 1, \ldots, n-1\}$. We consider an undirected weighted graph $G = (V, E, w)$, where V is the vertex set, E is the edge set, and $w : E \mapsto \mathbb{R}_{\geq 0}$ represents the weight function. We use n, m to denote the size of V and E, respectively. We use δ_i to denote the vector whose elements are 0 except for the i-th element, which is 1.

For a distribution defined over all subsets $\mu : 2^{[m]} \to \mathbb{R}_{\geq 0}$ and $S \subseteq [m]$, let μ_S be the restricted distribution of $P \sim \mu$ conditioned on the event $P \subseteq S$.

2.1 Quantum Computational Model

We assume the same quantum computational model as in for instance [DHHM06, AdW22]. This entails a classical control system that (i) can run quantum circuits on $O(\log n)$ qubits, (ii) can make quantum queries to the adjacency list oracle $\mathcal{O}_G$, and (iii) has access to a quantum-read/classical-write RAM of $\tilde{O}(\sqrt{mn})$ bits. The *quantum query complexity* measures the number of quantum queries that an algorithm makes. The *quantum time complexity* measures the number of elementary operations (classical and quantum gates), quantum queries, and single-bit QRAM operations (classical-write or quantum-read). If we only care about the quantum query complexity, then we can drop the QRAM assumption at the expense of a polynomial increase in the time complexity.

2.2 Markov Chains and Mixing time

Let μ, ν be two discrete probability distributions over a finite set Ω. The total variation distance (or TV-distance) between μ and ν is defined as

$$\|\mu - \nu\|_{\mathrm{TV}} := \frac{1}{2} \sum_{x \in \Omega} |\mu(x) - \nu(x)|.$$

The Kullback-Leibler (KL) divergence, also known as relative entropy, between μ and ν is defined as

$$\mathcal{D}_{\mathrm{KL}}(\mu \parallel \nu) := \sum_{x \in \Omega} \mu(x) \log \left(\frac{\mu(x)}{\nu(x)} \right).$$

A well-known result, often summarized as "relative entropy never increases", is stated in the following theorem:

Theorem 3 (Data Processing Inequality). *For any transition probability matrix $P \in \mathbb{R}^{\Omega \times \Omega'}$ and pair of distributions $\mu, \nu : \Omega \to \mathbb{R}_{\geqslant 0}$, it holds that*

$$\mathcal{D}_{\mathrm{KL}}(\nu P \parallel \mu P) \leqslant \mathcal{D}_{\mathrm{KL}}(\nu \parallel \mu).$$

A Markov chain on a finite state space Ω is specified by its transition probability matrix $P \in \mathbb{R}^{\Omega \times \Omega}$. For a comprehensive introduction to the analysis of Markov chains, we refer the reader to [LP17a]. A distribution μ is said to be stationary with respect to P if $\mu P = \mu$. In this case, μ is also called the stationary distribution of the Markov chain. When P is ergodic, the Markov chain has a unique stationary distribution μ, and for any initial probability distribution ν on Ω, $\|\nu P^t - \mu\|_{\mathrm{TV}}$ converges to 0 as $t \to \infty$. To precisely characterize how quickly an ergodic Markov chain converges, one can define the mixing time as below.

Definition 4 (Mixing time). *Let P be an ergodic Markov chain on a finite state space Ω and let μ denote its stationary distribution. For any probability distribution ν on Ω and $\varepsilon \in (0, 1)$, we define*

$$t_{\mathrm{mix}}(P, \nu, \varepsilon) := \min \left\{ t \geqslant 0 \mid \|\nu P^t - \mu\|_{\mathrm{TV}} \leqslant \varepsilon \right\}.$$

The following well-known theorem demonstrates the relationship between the modified log-Sobolev constant and mixing times, which bounds the mixing time via modified log-Sobolev inequalities.

Theorem 5 ([BT06], see also corollary 8 in [CGM21]). *Let P denote the transition matrix of an ergodic, reversible Markov chain on a finite state space Ω with the stationary distribution μ. Suppose there exists some $\alpha \in (0, 1]$ such that for any probability distribution ν on Ω, we have*

$$\mathcal{D}_{\mathrm{KL}}(\nu P \parallel \mu P) \leqslant (1 - \alpha)\mathcal{D}_{\mathrm{KL}}(\nu \parallel \mu).$$

Then, for any $\varepsilon \in (0, 1)$,

$$t_{\mathrm{mix}}(P, 1_x, \varepsilon) \leqslant \left\lceil \frac{1}{\alpha} \cdot \left(\log \log \left(\frac{1}{\mu(x)} \right) + \log \left(\frac{1}{2\varepsilon^2} \right) \right) \right\rceil,$$

where 1_x is the point mass distribution supported on x.

2.3 Laplacian and Graph Sparsification

For an undirected weighted graph $G = (V, E, w)$, the weighted degree of a vertex v is defined as $\deg(v) = \sum_{e \in E : v \in e} w_e$, where the sum is taken over all edges e incident to v, and w_e denotes the weight of edge e.

Definition 6 (Laplacian). *The Laplacian of a weighted graph $G = (V, E, w)$ is defined as the matrix $L_G \in \mathbb{R}^{V \times V}$ such that*

$$(L_G)_{ij} = \begin{cases} \deg(i) & \text{if } i = j, \\ -w_{ij} & \text{if } \{i, j\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Equivalently, the Laplacian of graph G is given by $L_G = D_G - A_G$, with A_G the weighted adjacency matrix $(A_G)_{ij} = w_{ij}$ and D the diagonal weighted degree matrix $D_G = \mathrm{diag}\,(\deg(i) : i \in V)$. L_G is a positive semidefinite matrix since the weight function w is nonnegative, and L_G induces the quadratic form

$$x^\top L_G x = \sum_{\{i,j\} \in E} w_{ij} \cdot (x_i - x_j)^2$$

for arbitrary vector $x \in \mathbb{R}^V$. Graph sparsification produces a reweighted graph with fewer edges, known as a graph spectral sparsifier. A graph spectral sparsifier of G is a re-weighted subgraph that closely approximates the quadratic form of the Laplacian for any vector $x \in \mathbb{R}^V$ (see definition 25 for a formal definition).

In the groundbreaking work [SS11], the authors demonstrated that graphs can be efficiently sparsified by sampling $\widetilde{O}(n)$ edges with weights roughly proportional to their effective resistance. Now, we introduce the concept of effective resistance.

Definition 7 (Effective Resistance). *Given a graph $G = (V, E, w)$, the effective resistance R_{ij} of a pair of $i, j \in V$ is defined as*

$$R_{ij} = (\delta_i - \delta_j)^\top L_G^+ (\delta_i - \delta_j) = \|(L_G^+)^{1/2}(\delta_i - \delta_j)\|^2.$$

Here, δ_i is a vector with all elements equal to 0 except for the i-th is 1, and L_G^+ is the Moore-Penrose inverse of the Laplacian matrix L_G.

Definition 8 (Leverage Score and Leverage Score Overestimates). *Given a graph $G = (V, E, w)$, the leverage score ℓ_e of an edge $e \in E$ is defined as the product of its weight and its effective resistance:*

$$\ell_e = w_e R_{ij}, \quad \text{for } e = \{i, j\}.$$

An λ-leverage score overestimates $\widetilde{\ell}$ for G is a vector in $\mathbb{R}^E_{\geqslant 0}$ satisfying $\|\widetilde{\ell}\|_1 \leqslant \lambda$, and $\widetilde{\ell}_e \geqslant \ell_e = w_e R_e$ for all $e \in E$.

Lemma 9 (Foster's theorem [Fos49])**.** *For a weighted graph $G = (V, E, w)$, let ℓ_e denote the leverage score of an edge $e \in E$. Then, it holds that $\sum_{e \in E} \ell_e \leqslant n - 1$.*

Proof. For a edge $\{i, j\}$, recall that $R_{ij} = \left(\delta_i - \delta_j\right)^\top L_G^+ \left(\delta_i - \delta_j\right)$, then

$$\sum_{e \in E} \ell_e = \sum_{\{i,j\} \in E} w_{ij} R_{ij} = \sum_{\{i,j\} \in E} \text{Tr}\left(w_{ij}\left(\delta_i - \delta_j\right)\left(\delta_i - \delta_j\right)^\top L_G^+\right) = \text{Tr}\left(L_G L_G^+\right) \leqslant n - 1,$$

since the rank of L_G is at most $n - 1$. $\qquad\qquad\square$

2.4 Strongly Rayleigh Distributions and Random Spanning Tree

Before defining strongly Rayleigh distributions, we first introduce the stability of generating polynomials, a fundamental property in both mathematics and physics. For a probability distribution $\mu : \binom{[m]}{k} \to \mathbb{R}_{\geqslant 0}$, we define the generating polynomial f_μ of μ as

$$f_\mu(z_1, \ldots, z_m) = \sum_{S \in \binom{[m]}{k}} \mu(S) \cdot \prod_{i \in S} z_i.$$

A nonzero polynomial $f \in \mathbb{C}[z_1, \ldots, z_m]$ is called stable if, $\forall i \in [m]$, the fact that $\text{Im}(z_i) > 0$ implies that $f(z_1, \ldots, z_m) \neq 0$. A stable polynomial with real coefficients is called real stable.

We say μ is a strongly Rayleigh distribution if and only if f_μ is a real stable polynomial. We say μ is k-homogeneous if f_μ is k-homogeneous. A useful fact is that if μ is strongly Rayleigh, then its restricted distribution μ_S for arbitrary $S \subseteq [m]$ is also strongly Rayleigh. For more details, we refer the reader to [BBL09].

For any connected undirected weighted graph $G = (V, E, w)$, let $\mathcal{T}_G \subseteq \binom{[m]}{n-1}$ denote the set of all spanning trees of G. We provide the formal definition of random spanning tree below.

Definition 10 (*w-uniform distribution on trees*)**.** *For any connected weighted undirected graph $G = (V, E, w)$, the distribution $\mathcal{W}_G$ on $\mathcal{T}_G$ is w-uniform if $\mathbb{P}_{X \sim \mathcal{W}_G}[X = T] \propto \prod_{e \in T} w_e$.*

We refer to $\mathcal{W}_G$ as the w-uniform distribution on $\mathcal{T}_G$. A random spanning tree drawn from the distribution $\mathcal{W}_G$ is called w-uniform.

Proposition 11 (Spanning Tree Marginals)**.** *Given a graph $G = (V, E, w)$, the probability that an edge $e = \{i, j\} \in E$ appears in a tree sampled w-uniformly randomly from $\mathcal{T}_G$ is given by its leverage score $\ell_e = w_e R_{ij}$. Namely, $\mathbb{P}_{T \sim \mathcal{W}_G}[e \in T] = \ell_e$.*

Proposition 12 (Spanning Trees are Strongly Rayleigh [BBL09, Theorem 3.6])**.** *For any connected weighted undirected graph $G = (V, E, w)$, the w-uniform distribution $\mathcal{W}_G$ is $(n-1)$-homogenous strongly Rayleigh.*

Theorem 13 (Random Tree Sampling [ALG$^+$21, Theorem 2])**.** *There is an algorithm $\text{RST}(G, \varepsilon)$ that, given a connected weighted graph $G = (V, E, w)$ and $\varepsilon > 0$, outputs a random spanning tree in $O(m \log m \log(m/\varepsilon))$ time. The distribution of output is guaranteed to be ε-close to $\mathcal{W}_G$ in total variation distance.*

2.5 Down-up and Up-down Walks

For a distribution μ on $\binom{[m]}{k}$, we define the complement distribution $\overline{\mu}$ on $\binom{[m]}{m-k}$ by setting $\overline{\mu}(S) \overset{\text{def}}{=} \mu([m] \smallsetminus S)$ for every $S \in \binom{[m]}{m-k}$.

Definition 14 (Down operator). *For $\ell \leqslant k$ define the row-stochastic matrix $D_{k \to \ell} \in \mathbb{R}_{\geqslant 0}^{\binom{[m]}{k} \times \binom{[m]}{\ell}}$ by*

$$D_{k \to \ell}(S, T) = \begin{cases} 0 & \text{if } T \nsubseteq S, \\ \frac{1}{\binom{k}{\ell}} & \text{otherwise.} \end{cases}$$

For a distribution μ on sets of size of k, $\mu D_{k \to \ell}$ will be a distribution on sets of size of ℓ. In particular, $\mu D_{k \to 1}$ will be the marginal distribution of μ: $\mathbb{P}_{S \sim \mu}(i \in S)/k, i \in [m]$.

Definition 15 (Up operator). *For $\ell \leqslant k$ define the row-stochastic matrix $U_{\ell \to k} \in \mathbb{R}_{\geqslant 0}^{\binom{[m]}{\ell} \times \binom{[m]}{k}}$ by*

$$U_{\ell \to k}(T, S) = \begin{cases} 0 & \text{if } T \nsubseteq S, \\ \frac{\mu(S)}{\sum_{S' : T \subseteq S'} \mu(S')} & \text{otherwise.} \end{cases}$$

We consider the following Markov chain M_μ^t defined for any positive integer $t \geqslant k + 1$, with the state space $\text{supp}(\mu)$. Starting from $S_0 \in \text{supp}(\mu)$, one step of the chain M_μ^t is:

1. Sample $T \in \binom{[m] \smallsetminus S_0}{t-k}$ uniformly randomly.
2. Sample $S_1 \sim \mu_{S_0 \cup T}$ and update S_0 to be S_1.

In [ADVY22, ALV22], it was demonstrated that the above constructed Markov chain can be used for sampling, as it possesses several desirable properties.

Proposition 16 (Proposition 25 in [ADVY22], see also Proposition 15 and 16 in [ALV22]). *The complement of S_1 is distributed according to $\overline{\mu}_0 D_{(m-k) \to (m-t)} U_{(m-t) \to (m-k)}$ if we start with $S_0 \sim \mu_0$. Moreover, for any distribution $\mu : \binom{[m]}{k} \to \mathbb{R}_{\geqslant 0}$ that is strongly Rayleigh, the chain M_μ^t for $t \geqslant k + 1$ is irreducible, aperiodic and has stationary distribution μ.*

3 Main Algorithm

Our main quantum algorithm implements a "large-step" up-down walk over the spanning trees of G. Starting from a spanning tree T, the up-step samples $k = 2n$ edges S uniformly at random from $E \backslash T$, and then the down-step samples a uniform spanning tree from the subgraph on $T \cup S$. We can show that, if the graph is nearly-isotropic, meaning that its edges have approximately uniform leverage scores, this walk achieves a mixing time of $\widetilde{O}(1)$ (see proposition 20).

In practice, most input graphs are not naturally isotropic. Isotropy here refers to a balanced structure where every edge contributes roughly equally to the connectivity of the graph. This balanced contribution is captured by the concept of leverage scores, which, in our setting, are given by the product of the effective resistances of the edges and their corresponding weights. The effective resistance of an edge serves as a measure of its "importance" in maintaining connectivity: edges with high effective resistance (or high leverage scores) have an outsized influence on the spanning tree distribution. If these values vary significantly across edges, the uniform sampling in the up-step may become biased, adversely affecting the mixing time of the chain. To address this,

we introduce a preprocessing step that "isotropizes" G by approximately normalizing the effective resistances. Specifically, we construct a quantum data structure QResistance that enables efficient quantum queries to constant-factor approximations of the effective resistances of the edges in G. This builds upon the quantum graph sparsification algorithm from [AdW22], which efficiently estimates effective resistances. This preprocessing step, which we detail in proposition 28, takes $\widetilde{O}(\sqrt{mn})$ time. Once initialized, it allows us to simulate quantum query access to an isotropic-transformed version of G, denoted as G'. With quantum query access to G', we can then implement the up-step efficiently by uniformly sampling k edges, thereby preserving the desired rapid mixing properties of the Markov chain.

As initial state of the random walk, the algorithm obtains a maximum weight-product spanning tree T by running the quantum algorithm QMaxProductTree, a slightly modified version of the algorithm from [DHHM06] (see theorem 30 for details). We label the tree with all its edges having 1 as a label, i.e., $e \to (e, 1)$ (the label is used for identification in the isotropic-transformed graph) and store it in QRAM. Subsequently, the algorithm executes $\widetilde{O}(1)$ rounds of the up-down walk. To implement the up-step, we develop a quantum algorithm QIsotropicSample for uniformly sampling k edges S in the isotropic-transformed graph. The key idea is to combine a routine for "preparing many copies of quantum states" with the leverage score oracle access given by our QResistance algorithm. We detail this algorithm in theorem 24. Notably, the algorithm QIsotropicSample returns a sparsified domain of $k = 2n$ edges in $\widetilde{O}(\sqrt{mn})$ time.

Using oracles to G and $\mathcal{R}$, we then build the subgraph with edges $T \cup S$ using the SubgraphConstruct procedure. This subgraph has $O(n)$ edges, and so we can apply the *classical* spanning tree sampling algorithm RST (theorem 13) to obtain a random spanning tree in the subgraph in $\widetilde{O}(n)$ time. This effectively implements the down-operator.

We summarize the algorithm below.

Algorithm 1 Quantum Algorithm for Random Spanning Tree Sampling, $\mathsf{QRST}(\mathcal{O}_G, \varepsilon)$

Require: Quantum Oracle $\mathcal{O}_G$ to a graph $G = (V, E, w)$ with $|V| = n, |E| = m$ and weights $w : E \to \mathbb{R}_{\geqslant 0}$; accuracy $\varepsilon > 0$.
Ensure: A spanning tree from a distribution which is ε-close to $\mathcal{W}_G$ in total variation distance.
 1: $\mathcal{R} \leftarrow \mathsf{QResistance}(\mathcal{O}_G, \frac{1}{10})$.
 2: $k \leftarrow 2n, M \leftarrow O(\log^3 n \cdot \log(1/\varepsilon))$.
 3: $T_0 \leftarrow \mathsf{QMaxProductTree}(\mathcal{O}_G), T'_0 \leftarrow \mathsf{Label}(T_0)$.
 4: $\mathcal{T} \leftarrow \mathsf{QStoreTree}(T'_0)$.
 5: **for** $t = 1$ to M **do**
 6: $\quad S'_t \leftarrow \mathsf{QIsotropicSample}(\mathcal{O}_G, \mathcal{T}, \mathcal{R}, k)$.
 7: $\quad H'_t \leftarrow \mathsf{SubgraphConstruct}(\mathcal{O}_G, \mathcal{R}, T'_{t-1}, S'_t)$.
 8: $\quad T'_t \leftarrow \mathsf{RST}(H'_t, \varepsilon/(2M)), \mathcal{T} \leftarrow \mathsf{QStoreTree}(T'_t)$.
 9: **end for**
10: Return the spanning tree T'_M without edge label.

More details of the subroutines employed in the algorithm are given below:

- The procedure $\mathsf{Label}(T_0)$ means that we add a specific label 1 to all the edges of T_0.

- The procedure $\mathsf{QStoreTree}(T)$ stores the tree T into a QRAM allowing for coherent quantum queries, given its classical description as the input.

- The procedure SubgraphConstruct($\mathcal{O}_G, \mathcal{R}, T'_{t-1}, S'_t$) constructs the subgraph after the up-step. More precisely, given query access $\mathcal{O}_G$ to $G = (V, E, w)$, query access $\mathcal{R}$ to approximate resistance $\widetilde{R}$, a classical description of a tree T'_{t-1} in G, and a set S'_t of labeled edges, we define the multigraph $G' = (V, E', w')$, where $E' = \{(e, j) \mid e \in E, j \in [q_e]\}$ with $q_e = \left\lceil \frac{m w_e \widetilde{R}_e}{n} \right\rceil$ and $w'_{(e,j)} = w_e / q_e$. We treat T'_{t-1} and S'_t as subsets of edges in G'. The procedure SubgraphConstruct just outputs a classical description of the graph $H'_t = (V, E'_t, w'_t)$, where $E'_t = T'_{t-1} \bigcup S'_t$, and $w'_t(f) = w'(f)$ for $f \in E'_{t-1}$.

3.1 Isotropic Transformation

Inspired by [AD20, ADVY22, ALV22], we consider a process that transforms a graph into a multigraph, ensuring that the resulting random spanning tree distribution is nearly-isotropic.

Definition 17 (Isotropic Transformation of a Graph). *Consider a connected weighted graph $G = (V, E, w)$, with λ-leverage score overestimates $\widetilde{\ell} \in \mathbb{R}^E_{\geq 0}$ for G. The isotropic transformed multigraph G' of G (with respect to $\widetilde{\ell}$) is defined as $G' = (V, E', w')$: Here $E' = \{(e, j) \mid e \in E, j \in [q_e]\}$, where (e, j) connects the same vertices as e, and $q_e = \left\lceil \frac{m \widetilde{\ell}_e}{\lambda} \right\rceil$; $w'_{(e,j)} = w_e / q_e$ for all $e \in E$ and $j \in [q_e]$.*

Lemma 18 (Isotropic Bound). *For a graph $G = (V, E, w)$ with $|V| = n$ and $|E| = m$, let $\widetilde{\ell} \in \mathbb{R}^E_{\geq 0}$ be its λ-leverage score overestimates. Let $G' = (V, E', w')$ denotes the isotropic-transformed graph of G with respect to $\widetilde{\ell}$. Then, it holds that $|E'| \leq 2 |E|$. Furthermore, for all $(e, j) \in E'$, we have*

$$\mathop{\mathbb{P}}_{S \sim \mathcal{W}_{G'}} [(e, j) \in S] \leq \frac{\lambda}{m}.$$

Proof. One has

$$|E'| = \sum_{e \in E} q_e \leq \sum_{e \in E} \left(1 + \frac{m \widetilde{\ell}_e}{\lambda} \right) \leq m + \frac{m \|\widetilde{\ell}\|_1}{\lambda} \leq 2m.$$

For any $e = \{a, b\} \in E$ and $j \in [q_e]$, the marginal

$$\mathop{\mathbb{P}}_{S \sim \mathcal{W}_{G'}} [(e, j) \in S] = w'_{(e,j)} R'_{ab} = w'_{(e,j)} R_{ab} = \frac{w_e R_{ab}}{q_e} \leq \frac{\lambda}{m},$$

where the first equality follows from proposition 11, and the second equality follows from the fact that $R'_{ab} = R_{ab}$. $\qquad\square$

The following theorem demonstrates that the down operator satisfies a significantly stronger entropy contraction inequality when applied to strongly Rayleigh distributions with nearly uniform marginals.

Theorem 19 ([ALV22, Theorem 28]). *Let $\mu : \binom{[m]}{k} \to \mathbb{R}_{\geq 0}$ be a strongly Rayleigh distribution with marginals $p \in \mathbb{R}^m$ defined by $p_i \stackrel{\text{def}}{=} \mathbb{P}_{S \sim \mu} [i \in S]$. Suppose μ satisfies*

$$p_{\max} \stackrel{\text{def}}{=} \max \{ p_i : i \in [m] \} \leq 1/500.$$

Then for any distribution $\overline{\nu} : \binom{[m]}{m-k} \to \mathbb{R}_{\geq 0}$ and $t \geq k + 1$, we have

$$\mathcal{D}_{\mathrm{KL}}(\overline{\nu} D_{(m-k) \to (m-t)} \parallel \overline{\mu} D_{(m-k) \to (m-t)}) \leq (1 - \kappa) \mathcal{D}_{\mathrm{KL}}(\overline{\nu} \parallel \overline{\mu})$$

with $\kappa = \Theta \left(\frac{t-k}{(m p_{\max} + t) \log^2 m} \right)$.

11

Proposition 20. *Let $G = (V, E, w)$ be a weighted graph with $|V| = n$, $|E| = m$, and $w \in \mathbb{R}_{\geqslant 0}^E$. Suppose G has λ-leverage score overestimates given by $\widetilde{\ell} \in \mathbb{R}_{\geqslant 0}^E$, and assume $m \geqslant 1000n$ and $\lambda \leqslant 2n$. Consider the isotropic transformed multigraph $G' = (V, E', w')$ of G with respect to $\widetilde{\ell}$, as defined in definition 17. Let $t \geqslant n$ be an integer, $m' = |E'|$, and define the transition matrix $P = D_{(m'-n+1)\to(m'-t)}U_{(m'-t)\to(m'-n+1)}$. Then, the mixing time satisfies*

$$t_{\mathrm{mix}}(P, T'_{\max}, \varepsilon) = O(\log^3(n) \log(1/\varepsilon)). \tag{1}$$

Here, $T'_{\max} = \{(e, 1) \mid e \in T_{\max}\} \subseteq E'$, where $T_{\max}$ is the maximum weight-product spanning tree of G.

Proof. For the isotropic-transformed graph G', by lemma 18, we have $m' \leqslant 2m$, and

$$\mathop{\mathbb{P}}_{S \sim \mathcal{W}_{G'}} [(e, j) \in S] \leqslant \frac{\lambda}{m} \leqslant \frac{2n}{m} \leqslant \frac{1}{500}.$$

Let $\mathcal{T}_{G'}$ denote the set of all spanning trees of G', and let $\mu' : \binom{[m']}{n-1} \to \mathbb{R}_{\geqslant 0}$ be the w'-uniform distribution on $\mathcal{T}_{G'}$. Applying theorem 19 with $t = 2n$ and $k = n - 1$, we obtain that, for any distribution $\overline{\nu'} : \binom{[m']}{m'-k} \to \mathbb{R}_{\geqslant 0}$,

$$\mathcal{D}_{\mathrm{KL}}(\overline{\nu'} D_{(m'-n+1)\to(m'-t)} \parallel \overline{\mu'} D_{(m'-n+1)\to(m'-t)}) \leqslant (1-\kappa)\mathcal{D}_{\mathrm{KL}}(\overline{\nu'} \parallel \overline{\mu'}),$$

where $\kappa^{-1} = O(\log^2 m') = O(\log^2 n)$. Thus, by the data processing inequality (theorem 3), we have

$$\mathcal{D}_{\mathrm{KL}}(\overline{\nu'} P \parallel \overline{\mu'} P) \leqslant (1-\kappa)\mathcal{D}_{\mathrm{KL}}(\overline{\nu'} \parallel \overline{\mu'}).$$

Combining the above result with the modified log-Sobolev inequality (theorem 5), we conclude that

$$t_{\mathrm{mix}}(P, T'_{\max}, \varepsilon) \leqslant \left\lceil \frac{1}{\kappa} \cdot \left(\log\log\left(\frac{1}{\mu'(T'_{\max})} \right) + \log\left(\frac{1}{2\varepsilon^2} \right) \right) \right\rceil \tag{2}$$

Now, observe that by the pigeonhole principle we have $\mu(T_{\max}) \geqslant 1/\binom{m}{n-1} = \Omega(1/m^n)$. Consequently, $\mu'(T'_{\max}) \geqslant \mu(T_{\max})/(m')^{n-1} = \Omega(1/(2m)^n)$. Thus, the claim follows as desired. $\square$

As a side remark, we note that by applying proposition 16, we can prove a bound of the mixing time for the complement (up-down walk) in a similar way.

3.2 QIsotropicSample Routine

In this subsection, we provide further details about the QIsotropicSample routine, which is designed for implementing the up-step. The goal of QIsotropicSample is to sample a set of k distinct edges from the isotropic-transformed multigraph uniformly at random. Throughout the discussion, we refer to this type of task as uniformly random k-subset sampling (of a set $[m]$), which means sampling a uniformly random k-size subset of the set $[m]$.

A crucial distinction between uniformly sampling a k-size subset of the set $[m]$ and uniformly and independently sampling k times from $[m]$ lies in the sampling method. The former requires sampling *without replacement*, while the latter involves sampling *with replacement*. Notably, the latter can be addressed directly using the "preparing many copies of a quantum state" algorithm (see theorem 29 for more details). Building on this, we propose a reduction from sampling *with replacement* to sampling *without replacement* in the following, leading to an efficient quantum

implementation for uniformly random k-subset sampling (theorem 23) and the QIsotropicSample routine (theorem 24).

We begin with the following combinatorial proposition, which shows that we can first perform the sampling with replacement, and then discard any repeated elements to accomplish uniformly random k-subset sampling.

Proposition 21. *Let $m \in \mathbb{N}$. Suppose $j_1, j_2, \dots, j_k \in [m]$ are integers which are uniformly and independently sampled from $[m]$. Then, conditioning on the distinctness of these integers, the distribution of $\{j_1, j_2, \dots, j_k\}$ is the same as the distribution of a uniformly random k-subset of $[m]$.*

Proof. Consider any fixed set $S = \{s_1, s_2, \dots, s_k\}$. For the uniformly random k-subset sampling, the probability that S occurs is exactly $\frac{1}{\binom{m}{k}}$.

For $j_1, \dots, j_k$ which are uniformly and independently sampled from $[m]$, the probability that they form a permutation of $\{s_1, \dots, s_k\}$ is $k!/m^k$. The probability that they are pairwise distinct is $m!/(m^k(m-k)!)$. Therefore, conditioning on the pairwise distinctness, the probability that $\{j_1, j_2, \dots, j_k\} = S$ is just

$$\frac{k!/m^k}{m! / \left((m-k)! m^k \right)} = \frac{1}{\binom{m}{k}}. \qquad \square$$

We then consider how many samples are sufficient for k distinct results, which is formalized in the following modified balls-into-bins lemma.

Lemma 22. *Let X be a uniformly random variable taking values in $[m]$. For $k \in [m]$, let R be the set of different results from sampling X for $\widetilde{\Theta}(k)$ times. Then, with high probability, we have $|R| \geqslant k$.*

Proof. Let Y_i denote the number of samples needed for $|R| = i - 1$ to become $|R| = i$. Let $Y = \sum_{i=1}^k Y_i$. When $|R| = i - 1$, the probability that a sample falls in $[m] - R$ is $p_i = (m - i + 1)/m$. By definition, Y_i obeys the geometric distribution, meaning that

$$\Pr[Y_i = k] = (1 - p_i)^{k-1} p_i.$$

Therefore, $\mathbf{E}[Y_i] = 1/p_i = m/(m - i + 1)$. By the linearity of expectations, we have

$$\mathbf{E}[Y] = \sum_{i=1}^k \mathbf{E}[Y_i] = \sum_{i=1}^k \frac{m}{(m - i + 1)} \leqslant \sum_{i=1}^k \frac{k}{(k - i + 1)} \leqslant k \log(k).$$

Thus, by Markov's inequality, with probability at least $2/3$, $Y \leqslant 3k \log(k)$, and the claim follows by amplifying the successful probability using Hoeffding's inequality. $\qquad \square$

By combining the above propositions and theorem 29, we obtain a quantum algorithm for uniformly random k-subset sampling from a set, which is described in the following corollary.

Theorem 23 (Quantum uniform random k-subset sampling)**.** *There exists a quantum algorithm MultiSample$(\mathcal{O}_w, k)$ that, given query access O_w to a vector $w \in \mathbb{N}^n$ (where $\|w\|_1 = O(\mathrm{poly}(n))$), and $k \in [n]$, with high probability, returns a uniformly random k-size subset S of $S_w = \{(j, \ell) | j \in [n], \ell \in [w_j]\}$. The algorithm uses $\widetilde{O}(\sqrt{nk})$ queries to $\mathcal{O}_w$ and runs in $\widetilde{O}(\sqrt{nk})$ time.*

Proof. In the following, we denote $k' = ck \log k$ where c is a sufficiently large constant.

From theorem 29, we know with high probability, the algorithm MultiPrepare($\mathcal{O}_w, k'$) will return k' copies of the state

$$|w\rangle = \frac{1}{\sqrt{W}} \sum_{j=1}^{n} \sqrt{w_j}|j\rangle,$$

where $W = \sum_{j=1}^{n} w_j$, using $\tilde{O}(\sqrt{nk})$ queries to $\mathcal{O}_w$ and in $\tilde{O}(\sqrt{nk})$ time. We then measure the k' states on the computational basis, obtaining k' samples $j_1, j_2, \ldots j_{k'}$.

Then, for each sample j_i, we query the oracle $\mathcal{O}_w$ with the state $|j_i\rangle|0\rangle$, and measure the state $\mathcal{O}_w|j_i\rangle|0\rangle$ in the computational basis to get the value of w_{j_i}. We then uniformly sample a integer $\ell_i \in [w_{j_i}]$, producing a uniformly random sample (j_i, ℓ_i) from S_w. For each sample j_i, these operations take $\tilde{O}(1)$ time. After the above procedure, we get a sequence $(j_1, \ell_1), (j_2, \ell_2), \ldots, (j_{k'}, \ell_{k'})$.

We then output the first k distinct elements of the above sequence (if there are no such k elements, the algorithm simply outputs fail; by lemma 22, the probability of this situation occurring is small.). By proposition 21, these elements constitute a set of size k which is uniformly randomly drawn from S_w. It is clear that the above algorithm uses $\tilde{O}(\sqrt{nk})$ queries to $\mathcal{O}_w$ in total and runs in $\tilde{O}(\sqrt{nk})$ time. $\square$

As a direct result of theorem 23, given adjacency list access to a graph and query access to its approximate resistance, we can sample k different edges in the isotropic-transformed graph in $\tilde{O}(\sqrt{mk})$ time, yielding an efficient quantum implementation of the up-step.

Theorem 24 (Quantum Multi-Sampling in the Isotropic-Transformed Graph). *There exists a quantum algorithm* QIsotropicSample($\mathcal{O}_G, \mathcal{T}, \mathcal{R}, k$), *that, given query access* $\mathcal{O}_G$ *to a graph* $G = (V, E, W)$ *(where* $|V| = n$ *and* $|E| = m$), *access* $\mathcal{T}$ *to a labeled tree* T' *of the multigraph* $G' = (V, E', w')$, *access* $\mathcal{R}$ *to an instance of* QResistance *that allows quantum query to the approximate effective resistance* $\tilde{R}_e$ *for edge* e, *and integer* $k \leqslant m$, *with high probability, outputs a uniformly random k-size subset S of $E' \backslash T'$, where* $E' = \{(e, j) \mid e \in E, j \in [q_e]\}$, *with* $q_e = \left\lceil \frac{mw_e \tilde{R}_e}{n} \right\rceil$. *The algorithm uses* $\tilde{O}(\sqrt{mk})$ *queries to* $\mathcal{O}_G$ *and* $\mathcal{R}$, *and runs in* $\tilde{O}(\sqrt{mk})$ *time.*

Proof. By definition, we have

$$\mathcal{O}_G|e\rangle|0\rangle = |e\rangle|w_e\rangle, \text{ and } \mathcal{R}|e\rangle|0\rangle = |e\rangle|\tilde{R}_e\rangle.$$

Thus, together with access $\mathcal{T}$ to a labeled tree T', we could implement a unitary U_q satisfying

$$U_q|e\rangle|0\rangle = |e\rangle|q_e\rangle$$

with $q_e = \left\lceil \frac{mw_e \tilde{R}_e}{n} \right\rceil$ if $e \in E' \backslash E'_T$, and $q_e = 0$ if $e \in E'_T$, using $\tilde{O}(1)$ queries and in $\tilde{O}(1)$ time. This is because we can coherently evaluate q_e given $\mathcal{O}_G, \mathcal{R}$ and $\mathcal{T}$.

Then, using theorem 23, with high probability, the algorithm MultiSample(U_q, k) will return a set S of size k which is uniformly drawn from $E' = \{(e, j) \mid e \in E, j \in [q_e]\}$, using $\tilde{O}(\sqrt{mk})$ queries to $\mathcal{O}_G$, and in $\tilde{O}(\sqrt{mk})$ time. $\square$

3.3 Proof of Main Theorem

We can now prove our main theorem.

Proof. Without loss of generality, we assume $n/m = o(1)$. At the initial step, the algorithm executes the subroutine QResistance to obtain query access to $\frac{1}{10}$-overestimates effective resistances of G in $\widetilde{O}(\sqrt{mn})$ time (by proposition 28). It then applies QMaxProductTree (see theorem 30) to find a spanning tree T_0 as the start in $\widetilde{O}(\sqrt{mn})$ time. Each edge $e \in T_0$ is then assigned a label $e^{(1)}$, and the labeled tree is stored in $\widetilde{O}(n)$ time in the procedure Label and QStoreTree.

Subsequently, in each iteration $t \in [M]$, the algorithm first uses the sampling subroutine QIsotropicSample to sample a uniformly random set S'_t of size k from $E' \backslash T'_{t-1}$ in $\widetilde{O}(\sqrt{mk})$ time, where $k = \Theta(n)$. Next, using theorem 13, it samples a labeled random spanning tree in the subgraph H'_t in $\widetilde{O}(n)$ time, since the edge-set of subgraph $T'_{t-1} \bigcup S'_t$ has size at most $n - 1 + k = O(n)$. Overall, the algorithm's time complexity is $\widetilde{O}(\sqrt{mn})$. It remains to prove the correctness of algorithm.

We first observe that $\widetilde{R}_e$ is a $\frac{1}{10}$-overestimate of R_e, i.e., $R_e \leqslant \widetilde{R}_e \leqslant \frac{11}{10} R_e$ for all $e \in E$. Thus, $\widetilde{\ell}_e = w_e \widetilde{R}_e$ provides an $\frac{11}{10} n$-leverage score overestimates, as shown below:

$$\sum_{e \in E} w_e \widetilde{R}_e \leqslant \sum_{e \in E} \frac{11}{10} w_e R_e \leqslant \frac{11}{10} n,$$

where the final inequality follows from Foster's theorem (lemma 9).

If the sampling error for the random spanning tree in each iteration (line 8 of algorithm 1) is zero, then each iteration can be viewed as one step of the chain $M^k_{\mu'}$, where $\mu' : \binom{[m']}{k} \to \mathbb{R}_{\geqslant 0}$ corresponds exactly to the random spanning tree distribution $\mathcal{W}_{G'}$. Here, m' represents the number of edges of the isotropic multigraph G' satisfying $m' \leqslant 2m$. By proposition 20, the (ideal) up-down walk has mixing time $O(\log^3(n) \log(1/\varepsilon))$ (eq. (1)). We can hence pick $M \in O(\log^3(n) \log(1/\varepsilon))$ so as to ensure that the (ideal) up-down walk returns a tree T'_M that is distributed $\varepsilon/2$-close to $\mathcal{W}_{G'}$. Since G' is obtained by equally dividing the weight of each edge in G among the multiedges, mapping the multigraph G' back to G ensures that distribution of the unlabeled tree T_M remains $\varepsilon/2$-close to $\mathcal{W}_G$.

Taking into account the sampling error, note that we chose the error of $\mathsf{RST}(H'_t, \varepsilon/(2M))$ to be $\varepsilon/(2M)$, ensuring that in each iteration, the TV-distance between the actual and ideal case is at most $\varepsilon/(2M)$. After M iterations, the cumulative TV-distance is bounded by $\varepsilon/2$. Since the ideal case returns T_M that is $\frac{\varepsilon}{2}$-close to $\mathcal{W}_G$, actual distribution of T_M will be ε-close to $\mathcal{W}_G$. $\qquad\square$

4 Lower Bound

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

Theorem 2 (Quantum lower bound for sampling a random spanning tree). *Let $\varepsilon < 1/2$ be a constant. For any graph $G = (V, E, w)$ with $|V| = n$, $|E| = m$, $w \in \mathbb{R}^E_{\geqslant 0}$, consider the problem of sampling a random spanning tree from a distribution ε-close to $\mathcal{W}_G$, given adjacency-list access to G. The quantum query complexity of this problem is $\Omega(\sqrt{mn})$.*

Proof. The argument is similar to the $\Omega(\sqrt{mn})$ lower bound from [DHHM06] on the quantum query complexity of finding a minimum spanning tree, and follows from a lower bound on quantum search. Let $m = k(n + 1)$ for some integer k. Consider a binary matrix $M \in \{0, 1\}^{n \times k}$, with the promise that every row of M contains a single 1-entry. The task of finding all n 1-entries corresponds to solving n parallel OR_k instances. By the direct product theorem on the quantum

15

query complexity of the OR_k-function [KŠDW07], the quantum query complexity of this problem is $\Theta(n\sqrt{k}) = \Theta(\sqrt{mn})$.

Now we construct a weighted graph G_M from M, in such a way that solving the aforementioned search problem on M reduces to sampling a random spanning tree from G_M. The vertices of G_M are $\{s, \ell_1, \ldots, \ell_k, r_1, \ldots, r_n\}$. Its edges are all (s, ℓ_i) with edge weight 1, and all (ℓ_i, r_j) with edge weight M_{ij}. Notably, there are $n + k + 1$ vertices and $n + k$ weight-1 edges. The union of these edges forms the unique spanning tree T with nonzero weight $\Pi_{e \in T} w_e = 1$. As a consequence, sampling a random spanning tree in G_M always returns T. Since T identifies all weight-1 edges in G_M (and hence all 1-entries in M), the quantum query complexity is $\Omega(\sqrt{mn})$[1]. $\qquad\square$

Acknowledgement

References

[AAKV01]   Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani. Quantum walks on graphs. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 50–59, 2001.

[AASV21]   Yeganeh Alimohammadi, Nima Anari, Kirankumar Shiragur, and Thuy-Duong Vuong. Fractionally log-concave and sector-stable polynomials: counting planar matchings and more. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 433–446, 2021.

[AD20]   Nima Anari and Michał Dereziński. Isotropy and log-concave polynomials: Accelerated sampling and high-precision counting of matroid bases. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1331–1344. IEEE, 2020.

[ADVY22]   Nima Anari, Michał Dereziński, Thuy-Duong Vuong, and Elizabeth Yang. Domain Sparsification of Discrete Distributions Using Entropic Independence. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:23, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

---

[1]A more refined argument shows that the lower bound even holds when all the edge weights are nonzero. The argument works by replacing $w_{ij} \in \{0, 1\}$ by $w_{ij} \in \{1/n^4, 1\}$, and noting that a random spanning tree still returns T with constant probability.

[AdW22]   Simon Apers and Ronald de Wolf. Quantum speedup for graph sparsification, cut approximation, and laplacian solving. *SIAM Journal on Computing*, 51(6):1703–1742, 2022.

[AGM$^+$10] Arash Asadpour, Michel X. Goemans, Aleksander Mądry, Shayan Oveis Gharan, and Amin Saberi. An $O(\log n \log \log n)$-approximation algorithm for the asymmetric traveling salesman problem. In *Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 379–389, 2010.

[AL20]   Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1198–1211, 2020.

[Ald90]   David J Aldous. The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM Journal on Discrete Mathematics*, 3(4):450–465, 1990.

[ALG$^+$21] Nima Anari, Kuikui Liu, Shayan Oveis Gharan, Cynthia Vinzant, and Thuy-Duong Vuong. Log-concave polynomials iv: approximate exchange, tight mixing times, and near-optimal sampling of forests. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 408–420, 2021.

[ALG22]   Dorna Abdolazimi, Kuikui Liu, and Shayan Oveis Gharan. A matrix trickle-down theorem on simplicial complexes and applications to sampling colorings. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 161–172. IEEE, 2022.

[ALG24]   Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. *SIAM Journal on Computing*, 53(6):FOCS20–1–FOCS20–37, 2024.

[ALV22]   Nima Anari, Yang P. Liu, and Thuy-Duong Vuong. Optimal sublinear sampling of spanning trees and determinantal point processes via average-case entropic independence. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 123–134, 2022.

[BBL09]   Julius Borcea, Petter Brändén, and Thomas Liggett. Negative dependence and the geometry of polynomials. *Journal of the American Mathematical Society*, 22(2):521–567, 2009.

[BCC$^+$22] Antonio Blanca, Pietro Caputo, Zongchen Chen, Daniel Parisi, Daniel Štefankovič, and Eric Vigoda. On mixing of markov chains: Coupling, spectral independence, and entropy factorization. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3670–3692, 2022.

[Bro89]   Andrei Z Broder. Generating random spanning trees. In *FOCS*, volume 89, pages 442–447, 1989.

[BT06]   Sergey G Bobkov and Prasad Tetali. Modified logarithmic Sobolev inequalities in discrete settings. *Journal of Theoretical Probability*, 19:289–336, 2006.

[CBGV$^+$13] Nicolo Cesa-Bianchi, Claudio Gentile, Fabio Vitale, Giovanni Zappella, et al. Random spanning trees and the prediction of weighted graphs. *Journal of Machine Learning Research*, 14(1):1251–1284, 2013.

[CGM21]   Mary Cryan, Heng Guo, and Giorgos Mousa. Modified log-Sobolev inequalities for strongly log-concave distributions. *The Annals of Probability*, 49(1):506–525, 2021.

[CGŠV21]   Zongchen Chen, Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Rapid mixing for colorings via spectral independence. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1548–1557. SIAM, 2021.

[CMN96]   Charles J Colbourn, Wendy J Myrvold, and Eugene Neufeld. Two algorithms for unranking arborescences. *Journal of Algorithms*, 20(2):268–281, 1996.

[DBPM20]   Michal Derezinski, Burak Bartan, Mert Pilanci, and Michael W Mahoney. Debiasing distributed second order optimization with surrogate sketching and scaled regularization. *Advances in Neural Information Processing Systems*, 33:6684–6695, 2020.

[DCMW19]   Michał Dereziński, Kenneth L Clarkson, Michael W Mahoney, and Manfred K Warmuth. Minimax experimental design: Bridging the gap between statistical and worst-case approaches to least squares regression. In *Conference on Learning Theory*, pages 1050–1069. PMLR, 2019.

[DHHM06]   Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006.

[DKM20]   Michal Derezinski, Rajiv Khanna, and Michael W Mahoney. Improved guarantees and a multiple-descent curve for column subset selection and the nystrom method. *Advances in Neural Information Processing Systems*, 33:4953–4964, 2020.

[DKP$^+$17]   David Durfee, Rasmus Kyng, John Peebles, Anup B Rao, and Sushant Sachdeva. Sampling random spanning trees faster than matrix multiplication. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 730–742, 2017.

[DPPR17]   David Durfee, John Peebles, Richard Peng, and Anup B Rao. Determinant-preserving sparsification of SDDM matrices with applications to counting and sampling spanning trees. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 926–937. IEEE, 2017.

[DW17]   Michal Derezinski and Manfred KK Warmuth. Unbiased estimates for linear regression via volume sampling. *Advances in Neural Information Processing Systems*, 30, 2017.

[DY24]   Michał Dereziński and Jiaming Yang. Solving dense linear systems faster than via preconditioning. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1118–1129, 2024.

[DYMZ23]   Leo L Duan, Zeyu Yuwen, George Michailidis, and Zhengwu Zhang. Low tree-rank bayesian vector autoregression models. *Journal of Machine Learning Research*, 24(286):1–35, 2023.

[Fos49]   Ronald M Foster. The average impedance of an electrical network. *Contributions to Applied Mechanics (Reissner Anniversary Volume)*, 333, 1949.

[Gro96]   Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[GRV09]   Navin Goyal, Luis Rademacher, and Santosh Vempala. Expanders via random spanning trees. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 576–585. SIAM, 2009.

[GSS11]   Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. A randomized rounding approach to the traveling salesman problem. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 550–559. IEEE, 2011.

[Gue83]   Alain Guenoche. Random spanning tree. *Journal of Algorithms*, 4(3):214–220, 1983.

[Ham22]   Yassine Hamoudi. Preparing many copies of a quantum state in the black-box model. *Physical Review A*, 105:062440, Jun 2022.

[HPVP21]   Mark Herbster, Stephen Pasteris, Fabio Vitale, and Massimiliano Pontil. A gang of adversarial bandits. *Advances in Neural Information Processing Systems*, 34:2265–2279, 2021.

[HX16]   Nicholas JA Harvey and Keyulu Xu. Generating random spanning trees via fast matrix multiplication. In *LATIN 2016: Theoretical Informatics: 12th Latin American Symposium, Ensenada, Mexico, April 11-15, 2016, Proceedings 12*, pages 522–535. Springer, 2016.

[JPV21]   Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. Spectral independence, coupling with the stationary distribution, and the spectral gap of the glauber dynamics. *arXiv preprint arXiv:2105.01201*, 2021.

[Kir47]   Gustav Kirchhoff. Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik*, 148(12):497–508, 1847.

[KM09]   Jonathan A Kelner and Aleksander Madry. Faster generation of random spanning trees. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 13–21. IEEE, 2009.

[KŠDW07]   Hartmut Klauck, Robert Špalek, and Ronald De Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. *SIAM Journal on Computing*, 36(5):1472–1493, 2007.

[KT11]   Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1193–1200, 2011.

[KT$^+$12] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[Kul90]   Vidyadhar G. Kulkarni. Generating random combinatorial objects. *Journal of Algorithms*, 11(2):185–207, 1990.

[Liu21]   Kuikui Liu. From Coupling to Spectral Independence and Blackbox Comparison with the Down-Up Walk. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:21, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[LP17a]   David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

[LP17b]   Russell Lyons and Yuval Peres. *Probability on trees and networks*, volume 42. Cambridge University Press, 2017.

[LV24]   Yin Tat Lee and Santosh S Vempala. Eldan's stochastic localization and the KLS conjecture: Isoperimetry, concentration and mixing. *Annals of Mathematics*, 199(3):1043–1092, 2024.

[MR02]   Cristopher Moore and Alexander Russell. Quantum walks on the hypercube. In *Randomization and Approximation Techniques in Computer Science: 6th International Workshop, RANDOM 2002 Cambridge, MA, USA, September 13–15, 2002 Proceedings 5*, pages 164–178. Springer, 2002.

[MST14]   Aleksander Madry, Damian Straszak, and Jakub Tarnawski. Fast generation of random spanning trees and the effective resistance metric. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 2019–2036. SIAM, 2014.

[NST22]   Aleksandar Nikolov, Mohit Singh, and Uthaipon Tantipongpipat. Proportional volume sampling and approximation algorithms for a-optimal design. *Mathematics of Operations Research*, 47(2):847–877, 2022.

[RBM23]   Robin Richter, Shankar Bhamidi, and Sach Mukherjee. Improved baselines for causal structure learning on interventional data. *Statistics and Computing*, 33(5):93, 2023.

[Ric07]   Peter C Richter. Quantum speedup of classical mixing processes. *Physical Review A—Atomic, Molecular, and Optical Physics*, 76(4):042306, 2007.

[RTF18]   Luís MS Russo, Andreia Sofia Teixeira, and Alexandre P Francisco. Linking and cutting spanning trees. *Algorithms*, 11(4):53, 2018.

[Rud99]   Mark Rudelson. Random vectors in the isotropic position. *Journal of Functional Analysis*, 164(1):60–72, 1999.

[Sch18]   Aaron Schild. An almost-linear time algorithm for uniform random spanning tree generation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 214–227, 2018.

[SS11]   Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[SSK13]   James Sharpnack, Aarti Singh, and Akshay Krishnamurthy. Detecting activations over graphs using spanning tree wavelet bases. In *Artificial intelligence and statistics*, pages 536–544. PMLR, 2013.

[TAL19]   Leonardo V Teixeira, Renato M Assunção, and Rosangela H Loschi. Bayesian space-time partitioning by sampling and pruning spanning trees. *Journal of Machine Learning Research*, 20(85):1–35, 2019.

[Wil96]   David Bruce Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 296–303, 1996.

A Useful Theorems

In this part, we revisit some useful quantum algorithms.

We first recall the following theorems about quantum speedups for graph sparsifications and approximating the effective resistances. While we state these theorems for a general choice of ε, we will use these routines only with $\varepsilon = \frac{1}{10}$ in our algorithm.

Definition 25 (Graph Spectral Sparsifier). *Let $G = (V, E, w)$ be a weighted graph, and let $\widetilde{G} = (V, \widetilde{E}, \widetilde{w})$ be a reweighted subgraph of G, where $\widetilde{w} : \widetilde{E} \to \mathbb{R}_{\geqslant 0}$ and $\widetilde{E} = \{e \in E : \widetilde{w}_e > 0\} \subseteq E$. For any $\varepsilon > 0$, $\widetilde{G}$ is an ε-spectral sparsifier of G if for any vector $x \in \mathbb{R}^V$, the following holds:*

$$\left| x^\top L_{\widetilde{G}} x - x^\top L_G x \right| \leqslant \varepsilon \cdot x^\top L_G x.$$

Theorem 26 (Quantum Speedups for Graph Sparsification, [AdW22, Theorem 1.1]). *There exists a quantum algorithm* GraphSparsify$(\mathcal{O}_G, \varepsilon)$ *that, given query access $\mathcal{O}_G$ to a weighted graph $G = (V, E, w)$ and a parameter $\varepsilon \geqslant \sqrt{n/m}$, with high probability, outputs the explicit description of an ε-spectral sparsifier of G with $\widetilde{O}(n/\varepsilon^2)$ edges. The algorithm uses $\widetilde{O}(\sqrt{mn}/\varepsilon)$ queries to $\mathcal{O}_G$, and runs in time $\widetilde{O}(\sqrt{mn}/\varepsilon)$.*

Theorem 27 (Approximating Effective Resistances, [SS11, Theorem 2]). *There exists a classical algorithm* ApproxResistance(G, ε) *that, on input graph $G = (V, E, w)$ and accuracy parameter $\varepsilon > 0$, with high probability, outputs a matrix Z_G of size $p \times n$ with $p = \lceil 24 \log n/\varepsilon^2 \rceil$ satisfying*

$$(1 - \varepsilon) R_{ab} \leqslant \|Z_G(\delta_a - \delta_b)\|^2 \leqslant (1 + \varepsilon) R_{ab}$$

for every pair of $a, b \in V$, where R_{ab} is the effective resistance between a and b. The algorithm runs in $\widetilde{O}(m/\varepsilon^2)$ time.

By first applying the quantum graph sparsification algorithm, and then computing the approximate effective resistance on the sparsified graph, we can obtain a quantum speedup for approximating effective resistance overestimates. This is characterized by the following theorem.

Proposition 28 (Quantum Algorithms for the Effective Resistance Overestimates, Adapted from [AdW22, Claim 7.9]). *Assume quantum query acces $\mathcal{O}_G$ to $G = (V, E, w)$. There is a quantum data structure* QResistance, *that takes in the accuracy parameter $\varepsilon \in (0, 1/3)$, and supports the following operations:*

- *Initialization* QResistanceInit(G, ε): *outputs an instance $\mathcal{R}$ of* QResistance. *This operation uses $\widetilde{O}(\sqrt{mn}/\varepsilon)$ queries to $\mathcal{O}_G$, and runs in $\widetilde{O}(\sqrt{mn}/\varepsilon + n/\varepsilon^4)$ time.*

- *Query $\mathcal{R}$.Query: outputs a unitary satisfying*

$$\mathcal{R}.\text{Query}|a\rangle|b\rangle|0\rangle = |a\rangle|b\rangle|\widetilde{R}_{ab}\rangle$$

with $R_{ab} \leqslant \widetilde{R}_{ab} \leqslant (1 + \varepsilon) R_{ab}$ for every pair of vertices $a, b \in V$, where R_{ab} is the effective resistance between vertices a and b in graph G. The operation runs in $\widetilde{O}(1/\varepsilon^2)$ time.

Proof. Originally the quantum algorithm in [AdW22, Claim 7.9] provides query access to the estimate R'_{ab} of the effective resistance satisfying $(1 - \varepsilon') R_{ab} \leqslant R'_{ab} \leqslant (1 + \varepsilon') R_{ab}$. Therefore, for our purpose, it suffices to take $\varepsilon' = \varepsilon/3$ and $\widetilde{R}_{ab} = R'_{ab}/(1 - \varepsilon')$ in their algorithm. $\qquad\square$

The following theorem allows us to obtain a quantum speedup for the task of preparing multiple copies of a state.

Theorem 29 (Preparing many copies of a quantum state, [Ham22, Theorem 1])**.** *There exists a quantum algorithm* $\mathsf{MultiPrepare}(\mathcal{O}_w, k)$ *that, given oracle access* $\mathcal{O}_w$ *to a vector* $w \in \mathbb{R}^n_{\geq 0}$ *(0-indexed), and* $k \in [n]$*, with high probability, outputs* k *copies of the state* $|w\rangle$*, where* $|w\rangle = \frac{1}{\sqrt{W}} \sum_{i \in [n]_0} \sqrt{w_i} |i\rangle$ *with* $W = \sum_{i \in [n]_0} w_i$*. The algorithm uses* $\widetilde{O}(\sqrt{nk})$ *queries to* $\mathcal{O}_w$*, and runs in* $\widetilde{O}(\sqrt{nk})$ *time.*

We further recall the quantum algorithm for finding a minimum spanning tree proposed in [DHHM06].

Theorem 30 (Quantum algorithm for finding a minimum spanning tree, Theorem 4.2 in [DHHM06])**.** *There exists a quantum algorithm* $\mathsf{QMinTree}(\mathcal{O}_G)$ *that, given query access* $\mathcal{O}_G$ *to an undirected weighted graph* $G = (V, E, w)$ *with* $|V| = n$ *and* $|E| = m$*, with high probability, outputs a minimum spanning tree of* G*. The algorithm uses* $O(\sqrt{mn})$ *queries to* $\mathcal{O}_G$ *and runs in* $\widetilde{O}(\sqrt{mn})$ *time.*

In our paper, we need to find a spanning tree with maximal weight product in an undirected weighted graph. This could be done by directly applying above algorithm to the same graph with slight modifications of weight. We give more details in the following.

Corollary 31 (Quantum algorithm for finding a maximum spanning tree, Adapted from Theorem 4.2 in [DHHM06])**.** *There exists a quantum algorithm* $\mathsf{QMaxProductTree}(\mathcal{O}_G)$ *that, given adjacency query access* $\mathcal{O}_G$ *to an undirected weighted graph* $G = (V, E, w)$ *with* $|V| = n$ *and* $|E| = m$*, and weight* w *bounded by some constant* W*, with high probability, outputs a maximum weight-product spanning tree of* G*. The algorithm uses* $O(\sqrt{mn})$ *queries to* $\mathcal{O}_G$ *and runs in* $\widetilde{O}(\sqrt{mn})$ *time.*

Proof. Note that 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 $w'_e = \log(W/w_e)$ for all $e \in E$. Given query access to G, a query access to G' can be constructed in $\widetilde{O}(1)$ time. Therefore, by applying theorem 30, we know $\mathsf{QMinTree}(\mathcal{O}_{G'})$ is what we need. $\square$