Path-Reporting Distance Oracles with Logarithmic Stretch and Linear Size

Shiri Chechik
Tel Aviv University, Israel

Tianyi Zhang
Tel Aviv University, Israel

Abstract

Given an undirected graph $G = (V, E, \mathbf{w})$ on n vertices with positive edge weights, a distance oracle is a space-efficient data structure that answers pairwise distance queries in fast runtime. The quality of a distance oracle is measured by three parameters: space, query time, and stretch. In a landmark paper by [Thorup and Zwick, 2001], they showed that for any integer parameter $k \geq 1$, there exists a distance oracle with size $O(kn^{1+1/k})$, $O(k)$ query time, and $(2k-1)$-stretch error on the approximate distances. After that, there has been a line of subsequent improvements which culminated in the optimal trade-off of $O(n^{1+1/k})$ space, $O(1)$ query time, and $(2k-1)$-stretch [Chechik, 2015].

However, these line of constructions did not require that the distance oracle is capable of printing an actual path besides an approximate distance estimate, and there has been a performance gap between path-reporting distance oracles and ones that are not path-reporting. It is known that the earliest construction by [Thorup and Zwick, 2001] is path-reporting, but the parameters are worse by a factor of k. In a later construction by [Wulff-Nilsen, 2013], the query time was improved from $O(k)$ to $O(\log k)$. Better trade-offs were discovered in [Elkin and Pettie, 2015] where the authors broke the $O(kn^{1+1/k})$ space barrier and achieved $O(n^{1+1/k}\log k)$ space with $O(\log k)$ query time, but their stretch was blown up to a polynomial $O(k^{\log_{4/3} 7})$; they also gave an alternative choice of $O(n^{1+1/k})$ space which is optimal, and $O(k)$-stretch which is also optimal up to a constant factor, but their query time rose exponentially to $O(n^{\epsilon})$. In a recent work [Elkin and Shabat, 2023], the authors obtained significant improvements of $O(n^{1+1/k}\log k)$ space, $O(k)$-stretch, and $O(\log\log k)$ query time, or $O(n^{1+1/k})$ space, $O(k\log k)$-stretch, and $O(\log\log k)$ query time.

All the above constructions of path-reporting distance oracles share a common barrier; that is, they could not achieve optimal space $O(n^{1+1/k})$ and stretch $O(k)$ simultaneously within logarithmic query time; for example, in the natural regime where $k = \lceil \log n \rceil$, previous distance oracles had to pay an extra factor of $\log\log n$ either in the space or stretch. As our result, we bypass this barrier by a new construction of path-reporting distance oracles with $O(n^{1+1/k})$ space and $O(k)$-stretch and $O(\log\log k)$ query time.

2012 ACM Subject Classification Theory of computation → Shortest paths

Keywords and phrases graph algorithms, shortest paths, distance oracles

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.42

Category Track A: Algorithms, Complexity and Games

Funding This publication is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 803118 UncertainENV).

1 Introduction

Given an undirected graph $G = (V, E, \mathbf{w})$ on n vertices with positive edge weights , distance oracles are space-efficient data structures that process distance queries with approximate answers in fast runtime. More specifically, given any pair of vertices $s, t \in V$, the distance oracle needs to return a distance estimate $\mathbf{est}(s,t)$ in the range $[\mathbf{dist}_G(s,t), \beta \cdot \mathbf{dist}_G(s,t)]$,

© Shiri Chechik and Tianyi Zhang;
licensed under Creative Commons License CC-BY 4.0

where β is an upper bound on the stretch error. Intuitively, when the distance oracle has smaller size, the stretch error grows larger, and the general goal is to understand the optimal trade-off between space and stretch with a fast query efficiency.

In a pioneering work by Thorup and Zwick [10], they constructed distance oracles for any integer parameter $k \geq 1$ with $O(kn^{1+1/k})$ space, $(2k-1)$-stretch and $O(k)$ query time. This balance between size and stretch is optimal when $k \leq \frac{\log n}{\log\log n}$ because the famous Erdős girth conjecture implies an $\Omega(n^{1+1/k})$ space lower bound for any distance oracle with $(2k-1)$-stretch, and it was left open in [10] if anything better can be achieved for larger choices of parameter k. In a subsequent work by Mendel and Naor [9], using Ramsey partitions and tree embeddings, the authors diverged from the techniques of [10] and devised a new distance oracle with optimal space $O(n^{1+1/k})$ and $O(1)$ query time, but its stretch becomes $O(k)$ which is worse than the optimal $2k-1$ by a constant factor. Later on, still following the old construction of [10], Wulff-Nilsen [11] improved the query time from $O(k)$ to $O(\log k)$ while retaining $O(kn^{1+1/k})$ space and $(2k-1)$-stretch; in the same paper, he could also obtain constant query time $O(1/\epsilon)$ by allowing a slightly larger stretch of $(2+\epsilon)k$. This line of works culminated in an optimal construction of distance oracles that achieve $O(n^{1+1/k})$ space, $(2k-1)$-stretch and $O(1)$ query time simultaneously [5, 4].

However, these works mentioned above did not require that the distance oracles should be capable of printing an actual path between the queried vertex pair that realizes the approximate distance estimate; distance oracles with this extra function are called *path-reporting* distance oracles. More specifically, in this setting, the query efficiency consists of two parts: the query time which is the amount of time to compute a distance estimate $\mathbf{est}(s,t)$, and after that we need to report a path π between the vertex pair with total weight $\mathbf{w}(\pi) \leq \mathbf{est}(s,t)$ in time $O(|\pi|)$.

So far there has been a performance gap between path-reporting distance oracles and ones that do not support reporting paths. It was known that the earliest construction of Thorup and Zwick [10] is path-reporting, but some later improvements including [9, 11, 4, 5] are not, except one by [11] with $O(kn^{1+1/k})$ space, $(2k-1)$-stretch and $O(\log k)$ query time. In a subsequent work [7], the authors achieved optimal $O(n^{1+1/k})$ space by tolerating a constant blowup in the stretch $O\left(k \cdot (1/\epsilon)^{\log_{4/3} 7}\right)$ and a sublinear query time $O(n^{\epsilon})$, or alternatively, a sub-optimal space complexity of $O(n^{1+1/k}\log k)$ and faster query time $O(\log k)$, but a polynomial blowup in the stretch $O\left(k^{\log_{4/3} 7}\right)$.

In a very recent work [8], the authors devised new constructions of path-reporting distance oracles with multiple choices of trade-offs which are significant improvements over prior works: (1) with space $O\left(n^{1+1/k} \cdot \left\lceil \frac{k \log\log n \cdot \log^{(3)} n}{\log n} \right\rceil\right)$, they could achieve $(4+\epsilon)k$-stretch and $O(\log k)$ query time, and (2) with space $O\left(n^{1+1/k} \cdot \left\lceil \frac{k \log\log n}{\log n} \right\rceil\right)$, they could achieve $(24+\epsilon)k$-stretch and $O(\log\log k)$ time, and (3) with optimal space $O(n^{1+1/k})$ they could achieve $O(k\log k)$-stretch and $O(\log\log k)$ query time; here we have not listed all the trade-offs, but only the representative ones in [8].

One common quantitative barrier of all the above constructions of path-reporting distance oracles is that they could not achieve optimal space $O(n^{1+1/k})$ and linear stretch $O(k)$ simultaneously using a fast query time. For example, when k gets close to $\log n$, the trade-offs in [8] needs to pay $O(\log\log n)$ factor either in the space or in the stretch. As for the results from [7], although could achieve $O(n^{1+1/k})$ space and $O\left(k \cdot (1/\epsilon)^{\log_{4/3} 7}\right)$-stretch at the same time, their query time blows up exponentially to $O(n^{\epsilon})$; besides, their stretch upper bound has a large constant coefficient of $O\left((1/\epsilon)^{\log_{4/3} 7}\right)$ in front of k. This leads to the following natural question even for $k = \lceil \log n \rceil$.

▶ **Question 1.** *How to design a path-reporting distance with $O(n)$ space and $O(\log n)$-stretch that supports $O(\log n)$ query time?*

As our result, we give a positive answer to the above question, which is formalized in the statement below.

▶ **Theorem 2.** *Given an undirected graph $G = (V, E, \mathbf{w})$ with positive edge weights on n vertices, for any integer $k \geq 1$, there is a path-reporting distance oracle with $12k$-stretch, $O(n^{1+1/k})$ space, and $O(\log\log k)$ query time; furthermore, an approximate shortest path π can be retrieved in time $O(\log\log k + |\pi|)$ time.*

1.1 Technical overview

To support the function of path-reporting, prior works adopted different strategies: in [7] the authors were heavily exploiting the power of distance preservers [6], and in [8] the authors were utilizing the connection to hopsets. In our construction, we employed another different strategy of using *tree covers* which is adapted from the notion of cluster covers in the design of the optimal but non-path-reporting distance oracles [5].

The approach of [5]

Similar to [10], let us take a hierarchy of random sets $V = A_0 \supseteq A_1 \supseteq A_2 \supseteq \cdots \supseteq A_{k-1} \supseteq A_k$, where A_{i+1} includes each vertex in A_i with probability $n^{-1/k}$ independently. For any A_i, a *cluster cover* is a collection of (not necessarily disjoint) subsets $\mathcal{C}^i$ with the following properties.

- **Weak radius.** Each vertex $v \in V$ is assigned to a unique cluster $\mathcal{C}^i_{\mathsf{home}}(v) \in \mathcal{C}^i$ which contains v. Plus, $\mathcal{C}^i_{\mathsf{home}}(v)$ has weak radius of roughly $\mathbf{dist}_G(v, A_i)$.
- **Packing.** The total size of clusters in $\mathcal{C}$ is bounded by $O(n^{1+1/k})$.
- **Covering.** For any pair of vertices s, t, if $\mathbf{dist}_G(s,t) \leq \frac{1}{i} \cdot \mathbf{dist}_G(s, A_i)$, then $\mathcal{C}^i_{\mathsf{home}}(s)$ contains both s, t.

The idea of using cluster covers to compute approximate distances is to build cluster covers $\mathcal{C}^{(i)}$ for a constant number of indices $i \in \{k/2, k/4, k/8, \ldots, k/128\}$ (assuming k is an integer multiple of 128), and check if $t \in \mathcal{C}^i_{\mathsf{home}}(s)$ for each such i. If we can locate two consecutive indices $i, i/2$ such that $t \in \mathcal{C}^i_{\mathsf{home}}(s)$ and $t \notin \mathcal{C}^{i/2}_{\mathsf{home}}(s)$, then we will estimate $\mathbf{dist}_G(s,t)$ through the weak radius of $\mathcal{C}^i_{\mathsf{home}}(s)$. The reason why this would be a good estimate is because $t \notin \mathcal{C}^{i/2}_{\mathsf{home}}(s)$ and so $\mathbf{dist}_G(s,t) > \frac{2}{i} \cdot \mathbf{dist}_G(s, A_{i/2})$, and therefore $\mathbf{dist}_G(s,t)$ cannot be too much smaller than the weak radius of $\mathcal{C}^i_{\mathsf{home}}(s)$ when $\mathbf{dist}_G(s, A_{i/2})$ and $\mathbf{dist}_G(s, A_i)$ are not too different; if $\mathbf{dist}_G(s, A_{i/2})$ is also way smaller than $\mathbf{dist}_G(s, A_i)$, we will use some other shortcuts between pivots in $A_{i/2}$ to find a better distance estimate.

If we cannot locate any pair of consecutive indices $i, i/2$ such that $t \in \mathcal{C}^i_{\mathsf{home}}(s)$ and $t \notin \mathcal{C}^{i/2}_{\mathsf{home}}(s)$, then this means that both s, t are actually a small cluster $\mathcal{C}^{k/128}_{\mathsf{home}}(s)$. In this case, we should expect that the same distance oracle construction by Thorup and Zwick [10] to give a much better distance estimate with roughly $k/64$-stretch. To take advantage of this slack, we can apply the construction by Mendel and Naor [9] to quickly calculate an estimation.

Tree covers

One of the reasons why the distance oracle of [5] is not path-reporting is that cluster covers only have an upper bound on weak radius rather than strong radius. Therefore, for the cause of reporting paths, we need to ensure that each cluster has a spanning tree whose radius is small; in other words, we are looking for a tree cover $\mathcal{C}^i$ with the following revised properties.

- **Strong radius.** Each vertex $v \in V$ is assigned to a unique cluster $\mathcal{C}^i_{\mathsf{home}}(v) \in \mathcal{C}^i$ which contains v. Plus, $\mathcal{C}^i_{\mathsf{home}}(v)$ has a spanning tree of radius at most $\mathbf{dist}_G(v, A_i)$.

To find such a tree cover, let us start with the basic approach of ball-carving (which is similar to the approach for cluster covers [5]): starting with $W \leftarrow V$, while W is not empty, pick an arbitrary vertex $v \in W$ and grow a tree T in graph $G[W]$ to a radius $r \cdot \rho = \frac{r}{i} \cdot \mathbf{dist}_G(v, A_i)$ such that:

$$\left|\mathbf{Ball}_{G[W]}(v, (r-1)\rho)\right| \geq n^{-1/k} \left|\mathbf{Ball}_{G[W]}(v, r\rho)\right|$$

Then, we add this tree T to $\mathcal{C}^i$ and remove all vertices in $\mathbf{Ball}_{G[W]}(v, (r-1)\rho)$ from W; for those vertices we have just removed, assign their home trees to be T. By doing this we can guarantee the strong radius and the packing property. However, this could damage the covering property. Imagine that for some pair of vertices s, t which are close, some vertices on the shortest path between s, t were removed after adding a spanning tree T to $\mathcal{C}^i$, then the home tree of s might not be able to reach t. To fix this issue, the idea is to assign s to this earlier tree T which might not have removed s at the time it was created.

Graph contraction

Another obstacle of path-reporting is that the construction of Chechik [5] incorporates the construction of Mendel and Naor [9]. To avoid using the distance oracle by Mendel and Naor, we could continue to apply tree covers for smaller indices $i < k/128$, but this would raise the space complexity from $O(n^{1+1/k})$ to $O(n^{1+1/k} \log k)$. Notice that on lower levels, we are obtaining much better stretch than $O(k)$, which gives us some slack when designing our data structures. To take advantage of such slack, the idea is to apply the stretch-friendly partition of a recent work [3] which roughly contracts the graph G into n/τ clusters while blowup the stretch by $O(\tau)$, and then build our tree cover structures on the contracted graph which takes only $O(n^{1+1/k}/\tau)$ space.

2 Preliminaries

Let $\epsilon = 0.01$ be a fixed constant (independent of n, k), and assume $k > 100$. Logarithms are of base 2. For any graph $G = (V, E, \mathbf{w})$ and any vertex subset $S \subseteq V$, let $G[S]$ be the induced subgraph of G on S.

▶ **Definition 3** (bunches and balls). *Let $G = (V, E, \mathbf{w})$ be an undirected weighted graph. For each vertex $v \in V$ and vertex subset $S \subseteq V$, let $\mathbf{piv}_G(v, S) \in S$ be the closest vertex to v, and define the bunch around v with respect to S as:*

$$\mathbf{Bun}_G(v, S) \overset{\text{def}}{=} \{u \mid \mathbf{dist}_G(u, v) < \mathbf{dist}_G(u, S)\}$$

Furthermore, for any $\delta > 0$, define

$$\mathbf{Bun}^{\delta}_G(v, S) \overset{\text{def}}{=} \{u \mid \mathbf{dist}_G(u, v) < \delta \cdot \mathbf{dist}_G(u, S)\}$$

Balls are similar to bunches, except that the radius is specified by a value, rather than a vertex set. More specifically, for any value $\rho \geq 0$, define the ball around v to be:

$$\mathbf{Ball}_G(v, \rho) = \{u \mid \mathbf{dist}_G(u, v) \leq \rho\}$$

2.1 Stretch-friendly partitions

We need the notion of *stretch-friendly partition* from [3].

▶ **Definition 4** ([3]). *Let $G = (V, E, \mathbf{w})$ be an undirected weighted graph, and fix an integer $\tau \geq 1$. A* stretch-friendly τ-partition *of G is a partition $\mathcal{C} \subset 2^V$ of V with the following guarantees for every $U \in \mathcal{C}$.*

(1) *There is a spanning tree $T[U]$ of $G[U]$ rooted at the cluster center vertex $\mathbf{cntr}[U] \in U$, such that for every $v \in U$, the unique tree path between $\mathbf{cntr}[U], v$ has at most τ edges.*

(2) *If $(x, y) \in E$, $x \in U, y \notin U$, then the weight of every edge on the tree path in $T[U]$ between $x, \mathbf{cntr}[U]$ is at most $\mathbf{w}(x, y)$.*

(3) *If $(x, y) \in E$, $x, y \in U$, then the weight of every edge on the tree path in $T[U]$ between x, y is at most $\mathbf{w}(x, y)$.*

For any stretch-friendly partition $\mathcal{C}$, define $G/\mathcal{C}$ to be the *quotient graph* where each cluster in $\mathcal{C}$ is contracted to a single node. Here are some basic properties of stretch-friendly partitions.

▶ **Lemma 5.** *Let $G = (V, E, \mathbf{w})$ be an undirected weighted graph, and let $\mathcal{C}$ be a stretch friendly τ-partition of G. Consider any pair of vertices $s, t \in V$, then we can find a path π between s, t using edges of $G/\mathcal{C}$ and tree edges of $T[C], C \in \mathcal{C}$, such that:*

$$\mathbf{w}(\pi) \leq 4\tau \cdot \mathbf{dist}_G(s, t)$$

Furthermore, given the shortest path π' between $\mathcal{C}(s), \mathcal{C}(t)$ in $G/\mathcal{C}$, the above path π can be computed in time $O(|\pi|)$.

Proof. If $\mathcal{C}(s) = \mathcal{C}(t)$, then we can take the tree path between s, t in $T[\mathcal{C}(s)]$ which has length at most $2\tau \cdot \mathbf{dist}_G(s, t)$. Otherwise, take the shortest path π_0 between $\mathcal{C}(s), \mathcal{C}(t)$ and suppose it visits contracted nodes $\mathcal{C}(s) = U_0, U_1, \ldots, U_l = \mathcal{C}(t)$. Let $(u_i, v_i) \in E$ be the edge connecting $U_i, U_{i+1}, \forall 0 \leq i < l$. By definition of stretch-friendly τ-partitions, we have:

$$\mathbf{dist}_G(s, t) \geq \frac{1}{\tau} \cdot \min\left\{\mathbf{dist}_{T[U_0]}(s, \mathbf{cntr}[U_0]), \mathbf{dist}_{T[U_l]}(t, \mathbf{cntr}[U_l])\right\}$$

$$\mathbf{w}(u_i, v_i) \geq \frac{1}{2\tau} \cdot \left(\mathbf{dist}_{T[U_i]}(u_i, \mathbf{cntr}[U_i]) + \mathbf{dist}_{T[U_{i+1}]}(v_i, \mathbf{cntr}[U_{i+1}])\right)$$

To find a path between s, t, let π be a concatenation of these paths: tree path between s, u_0 in $T[U_0]$, edge (u_0, v_0), tree path between v_0, u_1 in $T[U_1]$, ... , tree path between v_{l-1}, t in $T[U_l]$. Taking a summation of the above inequalities, we have:

$$\mathbf{w}(\pi) \leq 2\tau \cdot \mathbf{dist}_G(s, t) + 2\tau \cdot \mathbf{dist}_{G/\mathcal{C}}\left(\mathcal{C}(s), \mathcal{C}(t)\right) \leq 4\tau \cdot \mathbf{dist}_G(s, t)$$

As for the runtime to compute π, using the tree structure of $T[C]$, it is straightforward to compute π by its definition in time $O(|\pi|)$. ◀

The next statement shows how to construct a hierarchy of stretch-friendly clusters.

▶ **Lemma 6** ([3]). *For any $0 \leq i \leq \lceil \log n \rceil$, there exists a stretch-friendly $(3 \cdot 2^i - 1)$-partition $\mathcal{C}_i$ and each cluster in $\mathcal{C}_i$ has size at least 2^i. Moreover, $\mathcal{C}_i$ is a refinement of $\mathcal{C}_{i+1}$; that is, for each cluster $U \in \mathcal{C}_{i+1}$, U can be partitioned into sub-clusters $U = U_1 \cup U_2 \cup \cdots \cup U_k$, such that $U_j \in \mathcal{C}_i, \forall 1 \leq j \leq k$, and each tree $T[U_j]$ is a sub-tree of $T[U]$.*

2.2 Distance preservers

Throughout this paper, we assume shortest paths are unique by breaking ties using alphabetic ordering. We need a path-reporting distance preserver from [7].

▶ **Definition 7** (branching events, [7]). *For a collection of paths Π in an undirected graph G, a pair of paths together with common vertex (π_1, π_2, x) is called a* branching event*, if $x \in V(\pi_1) \cap V(\pi_2)$, and that the edges incident on x of π_1 and π_2 are different.*

▶ **Lemma 8** ([7]). *Given an undirected weighted graph $G = (V, E, \mathbf{w})$ and a collection of paths Π, there is a distance-preserving path-reporting data structure that reports any path $\pi \in \Pi$ in time $O(|\pi|)$. The data structure has size $O(n + |\Pi| + |\mathbf{br}(\Pi)|)$ where $\mathbf{br}(\Pi)$ is the set of branching events of Π.*

As a by-product, though it is not necessary for our main result, we slightly improve the above lemma as following, thus answering a small open question in [7]. This matches the sparsity bound of distance-preservers which do not require the function of path-reporting [6].

▶ **Lemma 9.** *Given an undirected weighted graph $G = (V, E, \mathbf{w})$ and a collection of shortest paths Π, there is a distance-preserving path-reporting data structure that reports any path $\pi \in \Pi$ in time $O(|\pi|)$. The data structure has size $O\left(n + |\Pi| + \sqrt{n|\mathbf{br}(\Pi)|}\right)$ where $\mathbf{br}(\Pi)$ is the set of branching events of Π.*

3 Tree covers

Before describing our path-reporting distance oracles, we will first need a building block which is adapted from [5]. Our stretch guarantee is worse than [5] roughly by a factor of 3/2 because our data structure is path-reporting, while the data structure from [5] could only report distance values, not any real paths in the graph.

▶ **Lemma 10.** *Let $G = (V, E, \mathbf{w})$ be an undirected weighted graph and let $S \subseteq V$ be a vertex subset. Suppose that there exists an integer parameter $r \geq \log^2 k$ such $|\mathbf{Bun}_G(v, S)| \leq O(n^{\frac{r}{k}} \log^2 n)$ for each $v \in V$. Then, there exists a collection of trees $\mathcal{T}$ of G with the following properties.*

- *The size of all trees in $\mathcal{T}$ is bounded by $O(n^{1+1/k})$.*
- *For any pair of vertices u, v, if $\mathbf{dist}_G(u, v) \leq \frac{2}{3(1+\epsilon)^2 r} \cdot \mathbf{dist}_G(u, S)$, then there exists a tree $T \in \mathcal{T}$ such that T contains both u, v, and more importantly:*

$$\mathbf{dist}_T(u, v) \leq \mathbf{dist}_T(u, \mathbf{rt}[T]) + \mathbf{dist}_T(v, \mathbf{rt}[T]) \leq 2(1+\epsilon) \cdot \mathbf{dist}_G(u, S)$$

where $\mathbf{rt}[T]$ is a fixed root of T; plus, such a tree T can be found in constant time.

Proof. For any integer $0 \leq b \leq \lceil \log_{1+\epsilon} 3 \rceil$, consider the sequence of values $(1+\epsilon)^b, 3(1+\epsilon)^b, 9(1+\epsilon)^b, \cdots, 3^i(1+\epsilon)^b, \cdots$. For any $i \geq 0$, let $V_{i,b} \subseteq V$ be the set of vertices v such that $\mathbf{dist}_G(v, S) \in \left[3^i(1+\epsilon)^b, 3^i(1+\epsilon)^{b+1}\right]$; by definition, we have $V = \bigcup_{i,b} V_{i,b}$.

Constructing a tree cover. Fix any offset $0 \leq b \leq \lceil \log_{1+\epsilon} 3 \rceil$, we will build a collection of trees $\mathcal{T}_b$ by a greedy ball-carving scheme on G; in the end, we will take the union of all trees $\mathcal{T} = \bigcup_b \mathcal{T}_b$. Initially, set $W \leftarrow V$, and go over all integers $i = 0, 1, 2, \cdots$. As long as$V_{i,b} \backslash W \neq \emptyset$, enumerate all vertices $v \in V_{i,b}$. Define a distance value $\rho_v = \frac{1}{(1+\epsilon)r} \cdot \mathbf{dist}_G(v, S)$. Therefore, as $r > \log^2 k$, we have

$$\left|\mathbf{Ball}_{G[W]}\left(v, (1+\epsilon)r\rho_v\right)\right| \leq |\mathbf{Bun}_G(v, S)| < O(n^{r/k} \log^2 n) < n^{\left\lfloor \frac{r}{(1-\epsilon)} \right\rfloor \cdot \frac{1}{k}}$$

▷ Claim 11. There must exist an integer $0 \leq r_v < \lfloor (1+\epsilon)r \rfloor$ such that:

$$\left|\mathbf{Ball}_{G[W]}\left(v, (r_v+1)\cdot\rho_v\right)\right| \leq n^{1/k}\left|\mathbf{Ball}_{G[W]}\left(v, r_v\cdot\rho_v\right)\right|$$

Proof of claim. Suppose otherwise, then for any integer $0 \leq l < \lfloor (1+\epsilon)r \rfloor$, we have:

$$\left|\mathbf{Ball}_{G[W]}\left(v, (l+1)\cdot\rho_v\right)\right| > n^{1/k}\left|\mathbf{Ball}_{G[W]}\left(v, l\cdot\rho_v\right)\right|$$

Taking a product of these inequalities over all integers $0 \leq l < \lfloor (1+\epsilon)r \rfloor$, we have:

$$\left|\mathbf{Ball}_{G[W]}\left(v, \lfloor (1+\epsilon)r \rfloor\cdot\rho_v\right)\right| > n^{\left\lfloor \frac{r}{1-\epsilon}\right\rfloor\cdot\frac{1}{k}}$$

Which is a contradiction. ◁

Then, let T be the single-source shortest paths tree rooted at $\mathbf{rt}[T] \leftarrow v$ that spans the vertex set $\mathbf{Ball}_{G[W]}\left(v, (r_v+1)\cdot\rho_v\right)$. All vertices in $\mathbf{Ball}_{G[W]}\left(v, r_v\cdot\rho_v\right) \cap V(T)$ will be called core vertices of T denoted as $\mathbf{core}(T)$, and vertices in $\mathbf{Ball}_{G[W]}\left(v, (r_v+1)\cdot\rho_v\right)$ will be called peripheral vertices of T. After that, add T to $\mathcal{T}_b$, and then remove all the core vertices $\mathbf{Ball}_{G[W]}\left(v, r_v\cdot\rho_v\right)$ from W by updating the vertex set:

$$W \leftarrow W \setminus \mathbf{Ball}_{G[W]}\left(v, r_v\cdot\rho_v\right)$$

After that, move on to the next vertex in $V_{i,b} \setminus W$.

Assigning home trees. To find a tree that meets the requirement in the lemma statement in constant time, we need each vertex to remember a constant number of trees that contain itself, which are called *home trees*. Specifically, for each vertex $v \in V$, we will associate a tree $\mathcal{T}^b_{\mathsf{home}}(v) \in \mathcal{T}_b$ with v which is defined to be the **first tree** T created in $\mathcal{T}_b$ containing v such that:

$$\mathbf{dist}_T(v, \mathbf{core}(T)) \leq \frac{1}{3(1+\epsilon)^2}\rho_v$$

Size of the tree cover. By the algorithm, each time we add a new tree T to $\mathcal{T}_b$, $|T|$ is at most $n^{1/k}|\mathbf{core}(T)|$. As core vertices are removed from W right afterwards, the total size of $\mathcal{T}_b$ is bounded by $O\left(n^{1+1/k}\right)$.

The covering property. Consider any pair of vertices u, v such that $\mathbf{dist}_G(u,v) \leq \frac{2}{3(1+\epsilon)^2 r}\cdot \mathbf{dist}_G(u,S)$. Let us first state an elementary inequality.

▷ Claim 12. $\rho_u \leq (1+\epsilon)\cdot\rho_v$.

Proof of claim. By triangle inequality, we have:

$$\begin{aligned}\mathbf{dist}_G(u,S) &\leq \mathbf{dist}_G(v,S) + \mathbf{dist}_G(u,v)\\ &\leq \mathbf{dist}_G(v,S) + \frac{2}{3(1+\epsilon)^2 r}\cdot\mathbf{dist}_G(u,S)\\ &< \mathbf{dist}_G(v,S) + \frac{\epsilon}{1+\epsilon}\cdot\mathbf{dist}_G(u,S)\end{aligned}$$

The last inequality holds as $r \geq \log^2 k$ and ϵ is a constant. Therefore, by definition of ρ_u, ρ_v, we can conclude the proof. ◁

Let π be the shortest path between u, v in G. Assume $u \in V_{i,b}$ for some $i, b \geq 0$.

▷ Claim 13. Vertices on π cannot be core vertices of trees rooted at any vertices from $V_{j,b}$ for some $j < i$.

Proof of claim. Suppose otherwise that there was a vertex $w \in V_{j,b}$ which grew a tree $T \in \mathcal{T}_b$ whose core includes a vertex $z \in V(\pi)$. Then, by triangle inequality, we would have:

$$\begin{aligned}\mathbf{dist}_G(u,S) &\leq \mathbf{dist}_G(u,z) + \mathbf{dist}_G(z,w) + \mathbf{dist}_G(w,S)\\ &\leq \frac{2}{3(1+\epsilon)r} \cdot 3^i \cdot (1+\epsilon)^{b+1} + 2 \cdot 3^{i-1} \cdot (1+\epsilon)^{b+1}\\ &< \left(\frac{1}{32} + 2\right) \cdot 3^{i-1} \cdot (1+\epsilon)^{b+1}\\ &< 3^i \cdot (1+\epsilon)^b\end{aligned}$$

This contradicts the definition that $u \in V_{i,b}$. ◁

As for the tree covering property, consider the first tree T created whose core intersected with $V(\pi)$, and consider the moment right before T was created. Since $z \in V(\pi)$, it must be:

$$\min\{\mathbf{dist}_{G[W]}(z,u), \mathbf{dist}_{G[W]}(z,v)\} \leq 0.5 \cdot \mathbf{dist}_G(u,v) \leq \frac{\rho_u}{3(1+\epsilon)^2}$$

Consider two possibilities.

- $\mathbf{dist}_{G[W]}(z,u) \leq 0.5 \cdot \mathbf{dist}_G(u,v) \leq \frac{\rho_u}{3(1+\epsilon)^2}$.
  In this case, we have $\mathbf{dist}_T(u, \mathbf{core}(T)) \leq \frac{\rho_u}{3(1+\epsilon)^2}$. Hence, by definition of $\mathcal{T}^b_{\mathsf{home}}(u)$, $\mathcal{T}^b_{\mathsf{home}}(u)$ must have been created no later than T. Therefore, when $\mathcal{T}^b_{\mathsf{home}}(u)$ was being created with root $z \in V_{i,b}$ (Claim 13), all vertices on π was still present in W. Therefore, at the moment, we have:

  $$\begin{aligned}\mathbf{dist}_{G[W]}\left(v, \mathbf{core}\left(\mathcal{T}^b_{\mathsf{home}}(u)\right)\right) &\leq \mathbf{dist}_{G[W]}(u,v) + \mathbf{dist}_{\mathcal{T}^b_{\mathsf{home}}(u)}\left(u, \mathbf{core}\left(\mathcal{T}^b_{\mathsf{home}}(u)\right)\right)\\ &\leq \frac{\rho_u}{(1+\epsilon)^2} < \rho_z\end{aligned}$$

  Hence, v was included in $\mathcal{T}^b_{\mathsf{home}}(u)$. As for their distance in the tree, by our algorithm, the radius of $\mathcal{T}^b_{\mathsf{home}}(u)$ is at most $3^i \cdot (1+\epsilon)^{b+1} \leq (1+\epsilon)\mathbf{dist}_G(u,S)$. Hence, we have:

  $$\mathbf{dist}_{\mathcal{T}^b_{\mathsf{home}}(u)}(u,v) \leq 2(1+\epsilon) \cdot \mathbf{dist}_G(u,S)$$

- $\mathbf{dist}_{G[W]}(z,v) \leq 0.5 \cdot \mathbf{dist}_G(u,v) \leq \frac{1}{3(1+\epsilon)^2}\rho_u$.
  In this case, we have $\mathbf{dist}_T(v, \mathbf{core}(T)) \leq \frac{1}{3(1+\epsilon)^2}\rho_u \leq \frac{1}{3(1+\epsilon)}\rho_v$. Hence, by definition of $\mathcal{T}^b_{\mathsf{home}}(v)$, $\mathcal{T}^b_{\mathsf{home}}(v)$ must have been created no later than T. Therefore, when $\mathcal{T}^b_{\mathsf{home}}(v)$ was being created with root $z \in V_{i,b}$ (Claim 13), all vertices on π was still present in W. Therefore, at the moment, we have:

  $$\begin{aligned}\mathbf{dist}_{G[W]}\left(u, \mathbf{core}\left(\mathcal{T}^b_{\mathsf{home}}(v)\right)\right) &\leq \mathbf{dist}_{G[W]}(u,v) + \mathbf{dist}_{\mathcal{T}^b_{\mathsf{home}}(v)}\left(v, \mathbf{core}\left(\mathcal{T}^b_{\mathsf{home}}(v)\right)\right)\\ &\leq \frac{1}{1+\epsilon}\rho_v \leq \rho_z\end{aligned}$$

  Hence, u was included in $\mathcal{T}^b_{\mathsf{home}}(v)$. As for their distance in the tree, by our algorithm, the radius of $\mathcal{T}^b_{\mathsf{home}}(v)$ is at most $3^i \cdot (1+\epsilon)^{b+1} \leq (1+\epsilon)\mathbf{dist}_G(u,S)$. Hence, we have:

  $$\mathbf{dist}_{\mathcal{T}^b_{\mathsf{home}}(u)}(u,v) \leq 2(1+\epsilon) \cdot \mathbf{dist}_G(u,S)$$ ◀

Given such a collection of trees $\mathcal{T}$ satisfying the conditions of Lemma 10, let us state a subroutine $\mathbf{TreeCover}(s,t,\mathcal{T})$ with constant runtime that checks if the tree cover data structure $\mathcal{T}$ can provide a distance estimation for $\mathbf{dist}_G(s,t)$ as long with the a path.

Algorithm 1 **TreeCover**$(s, t, \mathcal{T})$.

```
1 Assume $s \in V_{i,a}, t \in V_{j,b}$;
2 for $v \in \{s, t\}, c \in \{a, b\}$ do
3     if $\mathcal{T}^c_{\mathsf{home}}(v)$ contains both $\{s, t\}$ then
4         return $\mathbf{dist}_{\mathcal{T}^c_{\mathsf{home}}(v)}(s, t)$, along with the tree path in $\mathcal{T}^c_{\mathsf{home}}(v)$ between $s, t$ if
           required;
5 return $\perp$;
```

4 Path-reporting distance oracles

4.1 Data structures

Define $\alpha = 3/4 + \epsilon$. Apply Lemma 6 on G to create a sequence of stretch-friendly partitions $\{\mathcal{C}_i\}_{0 \leq i \leq \lceil \log n \rceil}$. Build a level ancestor data structure with $O(n)$ space so that for any $u \in V$ and any index $0 \leq i \leq \lceil \log n \rceil$, we can access the cluster $\mathcal{C}_i(u)$ in constant time [2].

Take a hierarchy of vertex sets $V = A_0 \supseteq A_1 \supseteq A_2 \supseteq \cdots \supseteq A_k$, where A_{i+1} includes each vertex in A_i independently with probability $n^{-1/k}$, $\forall 0 \leq i < k$. Therefore, with high probability, size of A_i is at most $O(n^{1-\frac{i}{k}} \log n)$.

Define a sequence of integers $h_0, h_1, h_2, \ldots, h_\iota$ satisfying $h_0 = k$, $h_1 = \lceil \alpha k \rceil$, and for $l \geq 1$ define $h_l = \max\left\{\lceil \alpha^l k \rceil, 2\lceil \log^2 k \rceil\right\}$, the sequence stops until it reaches $h_\iota = 2\lceil \log^2 k \rceil$.

Data structures for high levels

Let $\kappa = 50$ be an integer threshold. For every integer $0 \leq l \leq \iota$, we will build some data structures separately. When $l > \kappa$, let $\mathcal{C}^{(l)} \in \{\mathcal{C}_i\}_{0 \leq i \leq \lceil \log n \rceil}$ which is a stretch-friendly τ_l-partition where $\tau_l \in [\alpha^{-l/5}, 2\alpha^{-l/5}]$; we can make sure that the sequence $\{\tau_l\}_{\kappa \leq l \leq \iota}$ is non-decreasing. Plus, if $0 \leq l \leq \kappa$, then set $\mathcal{C}^{(l)}$ to be singletons (that is, $\tau_l = 1$). For each vertex $v \in V$, let $\mathcal{C}^{(l)}(v) \in \mathcal{C}^{(l)}$ be the unique cluster that contains v. Let $G/\mathcal{C}^{(l)}$ be the quotient graph of G where each cluster in $\mathcal{C}^{(l)}$ is contracted to a single node, and the edges of $G/\mathcal{C}^{(l)}$ are edges in G between different clusters in $\mathcal{C}^{(l)}$.

For the rest, we will build some graph data structures. For each integer $1 \leq l \leq \iota$, let B_{h_l} be the set of contracted nodes in $G/\mathcal{C}^{(l)}$ containing at least one vertex from the random set A_{h_l}, and let $C_{h_{l-1}}$ be the set of contracted nodes in $G/\mathcal{C}^{(l)}$ containing at least one vertex from the random set $A_{h_{l-1}}$. Build the following data structures.

(i) For each integer $1 \leq l \leq \iota$, apply Lemma 10 on the quotient graph $G/\mathcal{C}^{(l)}$ to build a tree cover $\mathcal{T}_l$ with respect to the terminal set B_{h_l} by setting the parameter $r = h_l$; we will prove that this parameter r satisfies the requirement of Lemma 10 with high probability over the random choices of $A_1, \ldots, A_k$.

For the special case when $l = 0$, simply set $\mathcal{T}_0$ to be single-source shortest paths trees rooted at each vertex in A_k in G, and each vertex in $u \in V$ has a pointer to the vertex in A_k that is the closest one to u. It is straightforward to see that $\mathcal{T}_0$ is also a valid tree cover for A_k using the definition of Lemma 10 (by setting $r \leftarrow k, S \leftarrow A_k$), and **TreeCover**$(s, t, \mathcal{T}_0)$ can always return a nonempty value whether or not the inequality $\mathbf{dist}_G(u, v) \leq \frac{2(1-\epsilon)^2}{3k} \cdot \mathbf{dist}_G(u, A_k)$ is satisfied.

(ii) For each integer $0 \leq l \leq \iota$ and for each contracted node x in $G/\mathcal{C}^{(l)}$, store a shortest path from x to its nearest node in B_{h_l} denoted as $\mathbf{piv}_{G/\mathcal{C}^{(l)}}(x, B_{h_l})$.

(iii) For each integer $1 \leq l \leq \iota$, we will build a set of paths Π_l in $G/\mathcal{C}^{(l)}$ in the following manner.
For each pair of nodes $x \in B_{h_l}$ and $y \in B_{h_l} \cap \mathbf{Bun}^{1/2}_{G/\mathcal{C}^{(l)}}\left(x, C_{h_{l-1}}\right)$, let $\pi_{x,y}$ the shortest path between x, y in graph $G/\mathcal{C}^{(l)}$, and add $\pi_{x,y}$ to Π_l. After building path set Π_l, apply Lemma 8 to build a path-reporting distance preserver data structure in the quotient graph $G/\mathcal{C}^{(l)}$ with respect to Π_l.

Data structures for low levels

Let $\mathcal{C}^{(\iota)}$ be a stretch-friendly $\Theta(\log^2 k)$-partition of G, and let $G/\mathcal{C}^{(\iota)}$ be the quotient graph. For each $0 \leq i \leq 2\left\lceil \log^2 k \right\rceil$, let B_i be the set of nodes in $G/\mathcal{C}^{(\iota)}$ containing at least one vertex in A_i. Store the following data structures for each $0 \leq i \leq 2\left\lceil \log^2 k \right\rceil$.

(i) For each node x in graph $G/\mathcal{C}^{(\iota)}$, store the shortest paths to all nodes $y \in B_i$ in $G/\mathcal{C}^{(\iota)}$ such that $\mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(x, y) < \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1})$, as well as the shortest path from x to $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(x, B_i)$. We assume shortest paths are unique by breaking ties alphabetically.

(ii) For each node x in graph $G/\mathcal{C}^{(\iota)}$ and each even index $0 \leq i < 2\left\lceil \log^2 k \right\rceil$, store the difference:

$$\Delta(x, i) \overset{\text{def}}{=} \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+2}) - \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(x, B_i)$$

Then, for each such x, store a range minimum query data structure for all entries $\Delta(x, i), i = 0, 2, \ldots, 2\left\lceil \log^2 k \right\rceil - 2$ with linear space and constant query time [1].

4.2 Query algorithm

Given any query $(s, t) \in V^2$, the query algorithm consists of two phases, one for high levels (in range $\left[2\left\lceil \log^2 k \right\rceil, k\right]$), and one for low levels (in range $\left[0, 2\left\lceil \log^2 k \right\rceil\right]$).

High-level phase

By the design of the data structure, $\mathbf{TreeCover}(s, t, \mathcal{T}_0)$ always returns nonempty value because each tree in $\mathcal{T}_0$ is a spanning tree of G. For each $1 \leq l \leq \iota$, define $u_l = \mathcal{C}^{(l)}(s), v_l = \mathcal{C}^{(l)}(t)$. Next, run the subroutine $\mathbf{TreeCover}(u_\iota, v_\iota, \mathcal{T}_\iota)$; if it successfully returns a nonempty value, then move on to the next phase. Otherwise, the algorithm maintains two indices $l_1 \leftarrow 0, l_2 \leftarrow \iota$ and performs the following binary search procedure; the goal of the binary search procedure is to find a consecutive pair of indices $l-1, l$ such that $\mathbf{TreeCover}(u_l, u_l, \mathcal{T}_l)$ returns an empty value, while $\mathbf{TreeCover}(u_{l-1}, v_{l-1}, \mathcal{T}_{l-1})$ returns successfully a nonempty value.

(1) If $l_1 < l_2 - 1$, define $l_3 = \lfloor (l_1 + l_2)/2 \rfloor$, and run the subroutine $\mathbf{TreeCover}(u_{l_3}, v_{l_3}, \mathcal{T}_{l_3})$. If the subroutine returns an empty value, then assign $l_2 \leftarrow l_3$; otherwise, assign $l_1 \leftarrow l_3$. Then, continue with the new pair (l_1, l_2).

(2) Now suppose $l_1 + 1 = l_2 = l$ and $\mathbf{TreeCover}(u_{l-1}, v_{l-1}, \mathcal{T}_{l-1})$ returns a nonempty value, but the procedure $\mathbf{TreeCover}(u_l, v_l, \mathcal{T}_l)$ returns null. Then, query part (ii) of the data structure in the storage to find the vertices $x = \mathbf{piv}_{G/\mathcal{C}^{(l)}}(u_l, B_{h_l})$ and $y = \mathbf{piv}_{G/\mathcal{C}^{(l)}}(v_l, B_{h_l})$, and check two possibilities below.

(a) The shortest path between nodes x, y in $G/\mathcal{C}^{(l)}$ is preserved in the distance-preserver; that is, there is a path between x, y in Π_l
In this case, query the shortest path π_1' from u_l to x using part (ii), and the shortest path π_2' from x to y using the path-reporting distance preserver in part (iii), and the shortest path π_3' from y to v_l using part (ii). Then, define the concatenation $\pi' = \pi_1' \circ \pi_2' \circ \pi_3'$ and then recover a path π between s, t in G by unpacking the contracted nodes in $\mathcal{C}^{(l)}$ using π' by Lemma 5 with runtime $O(|\pi|)$.

(b) The shortest path between nodes x, y in $G/\mathcal{C}^{(l)}$ is not preserved in the distance-preserver.
In this case, as **TreeCover** $(u_{l-1}, v_{l-1}, \mathcal{T}_{l-1})$ returned a nonempty value, we can use the tree in $\mathcal{T}_{l-1}$ that covers both u_{l-1}, v_{l-1} to get a path π' between them. Then, by Lemma 5 we can obtain a path between s, t in G which unpacks the contracted nodes in $\mathcal{C}^{(l-1)}$.

Low-level phase

This part is pretty much the same as the query algorithm from [11]. The difference is that we are making the queries in the contracted graph $G/\mathcal{C}^{(\iota)}$, and total number of levels is at most $2\lceil\log^2 k\rceil$ rather than k. For convenience, rename the variables by $u = \mathcal{C}^{(\iota)}(s)$ and $v = \mathcal{C}^{(\iota)}(t)$. We first check if $u = v$; if it is the case, we can directly retrieve a path using the spanning trees of the stretch-friendly τ_ι-partition $\mathcal{C}^{(\iota)}$ which has stretch error at most $2\tau_\iota < k$. For the rest, let us assume that $u \neq v$.

▶ **Definition 14** ([11])**.** *For a pair of contracted nodes u, v in $G/\mathcal{C}^{(\iota)}$, and even index $j \in [0, 2\log^2 k]$ is called* (u,v)-terminal *if (1) $j = 2\lceil\log^2 k\rceil$ or (2) $j < 2\lceil\log^2 k\rceil$ and one of the following conditions holds:*

$$\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(u, B_j) \in \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(v, B_{j+1})$$

$$\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(v, B_{j+1}) \in \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(u, B_{j+2})$$

Next, we will perform a binary search on the interval $\left[0, 2\lceil\log^2 k\rceil\right]$ as did in [11]. Initially, set $i_1 \leftarrow 0, i_2 \leftarrow 2\lceil\log^2 k\rceil$. In each iteration we perform the following steps while guaranteeing that i_2 is always (u,v)-terminal.

- If $i_1 = i_2$, since i_2 is (u,v)-terminal, we can consider three different possibilities.
  - $i_2 = 2\lceil\log^2 k\rceil$.
    Since **TreeCover**$(u, v, \mathcal{T}_\iota)$ returns successfully, we can use the tree in $\mathcal{T}_\iota$ that covers both u, v to get a path π' between them. Then, by Lemma 5 we can obtain a path between s, t in G which unpacks the contracted nodes in $\mathcal{C}^{(\iota)}$.
  - $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(u, B_{i_2}) \in \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(v, B_{i_2+1})$.
    According to part (i) of our low-level data structures, we have stored the shortest path π_1 from v to $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(u, B_{i_2})$ in $G/\mathcal{C}^{(\iota)}$, as well as the shortest path π_2 from u to $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(u, B_{i_2})$. Then, we can retrieve a path between s, t by applying Lemma 5 on $\pi_1 \circ \pi_2$.
  - $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(v, B_{i_2+1}) \in \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(u, B_{i_2+2})$.
    Similarly, according to part (i) of our low-level data structures, we have stored the shortest path π_1 from u to $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(v, B_{i_2+1})$ in $G/\mathcal{C}^{(\iota)}$, as well as the shortest path π_2 from v to $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(u, B_{i_2+2})$. Then, we can retrieve a path between s, t by applying Lemma 5 on $\pi_1 \circ \pi_2$.
- Otherwise, let j be the middle even index among $i_1, i_1+2, \cdots, i_2-2$, and let i_3 be the even index in $i_1, i_1+2, \cdots, j$ maximizing $\Delta(u, i_3)$ by querying part (ii) of our low-level data structure. If i_3 is not (u,v)-terminal, then recurse on the index pair $(j+2, i_2)$; otherwise, recurse on (i_1, i_3).

4.3 Space analysis

Size of high-level data structures

Let us analyze the size of our data structures part by part for each index $0 \leq l < \iota$.

▷ Claim 15. Let $N = O(n \cdot \alpha^{l/5})$ be the number of contracted nodes in $G/\mathcal{C}^{(l)}$. With high probability over the randomness of A_{h_l}, for any contracted node $u \in G/\mathcal{C}^{(l)}$, we have the following bounds.

- $\left|\mathbf{Bun}_{G/\mathcal{C}^{(l)}}\left(u, B_{h_l}\right)\right| \leq O\left(N^{\frac{h_l}{k}} \log^2 N\right)$.
- $\left|B_{h_l} \cap \mathbf{Bun}_{G/\mathcal{C}^{(l)}}\left(u, C_{h_{l-1}}\right)\right| \leq O\left(n^{\frac{h_{l-1}-h_l}{k}} \log n\right)$.

Proof. Let S be the set of nodes in $G/\mathcal{C}^{(l)}$ which are the nearest $10n^{\frac{h_l}{k}} \log n$ ones to u. Then, since A_{h_l} samples each vertices in V independently with probability $n^{-\frac{h_l}{k}}$, with high probability, A_{h_l} contains at least one vertex contracted in some nodes in S. Therefore, in this case we have $\left|\mathbf{Bun}_{G/\mathcal{C}^{(l)}}\left(u, B_{h_l}\right)\right| \leq O\left(n^{\frac{h_l}{k}} \log n\right) \leq O\left(N^{\frac{h_l}{k}} \log^2 N\right)$.

As for the second inequality bound, let $u_1, u_2, \ldots, u_j \in B_{h_l}$ be the nearest nodes to u in $G/\mathcal{C}^{(l)}$, for some $j = O\left(n^{\frac{h_{l-1}-h_l}{k}} \log n\right)$. Then, since each u_i belongs to $C_{h_{l-1}}$ with probability at least $n^{-\frac{h_{l-1}-h_l}{k}}$, at least one node $u_i, 1 \leq i \leq j$ should belong to $C_{h_{l-1}}$ with high probability. Therefore, the size of $\left|B_{h_l} \cap \mathbf{Bun}_{G/\mathcal{C}^{(l)}}\left(u, C_{h_{l-1}}\right)\right|$ should be $\leq O\left(n^{\frac{h_{l-1}-h_l}{k}} \log n\right)$. ◁

Using Lemma 10 and Claim 15, we know that the total size of the tree cover data structure for the quotient graph $G/\mathcal{C}^{(l)}$ is bounded by $O\left(n^{1+\frac{1}{k}} \cdot \alpha^{l/5}\right)$. Then, taking a summation over all indices $0 \leq l < \iota$, the size bound becomes $O\left(n^{1+\frac{1}{k}}\right)$. Similarly, we can also bound the total size of shortest paths from all the nodes x to $\mathbf{piv}_{G/\mathcal{C}^{(l)}}\left(x, B_{h_l}\right)$.

To bound the total size of our path-reporting distance preservers, we need to bound the total number of branching events of Π_l.

▷ Claim 16. For each $0 \leq l \leq \iota$, the expectation of $|\mathbf{br}(\Pi_l)| + |\Pi_l|$ over the randomness of $A_1, \ldots, A_k$ is bounded by $O(n/\log n)$.

Proof. Let $(\pi_1, \pi_2, *) \in \mathbf{br}(\Pi_l)$ be any branching event, and assume π_1, π_2 are shortest paths in $G/\mathcal{C}^{(l)}$ between nodes u_1, v_1 and u_2, v_2 such that $v_b \in \mathbf{Bun}^{1/2}_{G/\mathcal{C}^{(l)}}\left(u_b, C_{h_{l-1}}\right), \forall b \in \{1, 2\}$. Without loss of generality, assume that $\mathbf{w}(\pi_1) \geq \mathbf{w}(\pi_2)$. Then, we have:

$$\mathbf{dist}_{G/\mathcal{C}^{(l)}}(u_1, x) \leq \mathbf{w}(\pi_1) + \mathbf{w}(\pi_2) < \mathbf{dist}_{G/\mathcal{C}^{(l)}}\left(u_1, C_{h_{l-1}}\right), \forall x \in \{v_1, u_2, v_2\}$$

Therefore, $v_1, u_2, v_2 \in B_{h_l} \cap \mathbf{Bun}_{G/\mathcal{C}^{(l)}}\left(u_1, C_{h_{l-1}}\right)$. Therefore, using Claim 15, with high probability, $|\mathbf{br}(\Pi_l)|$ is bounded by the following up to a constant factor (recall that $\alpha = 3/4 + \epsilon$):

$$\begin{aligned}
\sum_{u \in B_{h_l}} \left|B_{h_l} \cap \mathbf{Bun}_{G/\mathcal{C}^{(l)}}\left(u, C_{h_{l-1}}\right)\right|^3 &\leq |B_{h_l}| \cdot O\left(n^{\frac{3h_{l-1}-3h_l}{k}} \log^3 n\right) \\
&\leq |A_{h_l}| \cdot O\left(n^{\frac{3h_{l-1}-3h_l}{k}} \log^3 n\right) \\
&\leq O\left(n^{1-\frac{h_l}{k}+\frac{3h_{l-1}-3h_l}{k}} \log^4 n\right) \\
&< O\left(n^{1-\frac{4\lceil \alpha^l k\rceil - 3\lceil \alpha^{l-1} k\rceil}{k}} \log^4 n\right) \\
&\leq O\left(n^{1-\frac{3\epsilon \cdot \alpha^{l-1} k - 3}{k}} \log^4 n\right) \\
&< O\left(n^{1-\epsilon \cdot \log^2 k} \log^4 n\right) \\
&< n/\log n
\end{aligned}$$

As for the size of Π_l, by definition, it is bounded by $\sum_{u\in B_{h_l}} \left|B_{h_l} \cap \mathbf{Bun}^{1/2}_{G/\mathcal{C}^{(l)}}(u, C_{h_{l-1}})\right|$ which is also at most $O(n/\log n)$ according to the above calculation. ◁

Applying Definition 7 and Claim 16, we can bound the expected size of part-(iii) of our data structures by $O(n)$.

Size of low-level data structures

It is clear that part (ii) only takes space $O(n\log^2 k/\tau_\iota) \leq O(n)$. As for part (i), for each node x, we have stored the shortest path from x to every node $y \in B_i \cap \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1})$ and $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(x, B_i)$. It is clear that the total size of the shortest paths to pivots is at most $O(n/\tau_\iota)$, so it is at most $O(n)$ by taking a summation over all indices $0 \leq i \leq 2\lceil\log^2 k\rceil$. Next, let us focus on shortest paths from x to nodes in $B_i \cap \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1})$.

▷ Claim 17. For each node x in $G/\mathcal{C}^{(\iota)}$, the expected size of $B_i \cap \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1})$ is at most $n^{1/k}$ over the randomness of $A_1, A_2, \ldots, A_k$.

Proof. Fix any node x, order all nodes in $G/\mathcal{C}^{(\iota)}$ in an increasing order of distances as $y_0, y_1, \ldots, y_l, \ldots$. For each y_j, let $|y_j|$ be the number of vertices in V contracted within. Conditioning on $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1})$ being y_l, the expected total number vertices contracted in nodes from $B_i \cap \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1})$ is equal to $n^{-i/k}\sum_{j=0}^{l-1}|y_j|$. Therefore, the overall expected total number vertices contracted in nodes from $B_i \cap \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1})$ would be:

$$\begin{aligned}
&n^{-i/k}\sum_{l\geq 1}\sum_{j=0}^{l-1}|y_j|\cdot \Pr\left[\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1}) = y_l\right]\\
&= n^{-i/k}\sum_{l\geq 1}\sum_{j=0}^{l-1}|y_j|\cdot\left(1-n^{-(i+1)/k}\right)^{\sum_{j=0}^{l-1}|y_j|}\cdot\left(1-\left(1-n^{-(i+1)/k}\right)^{|y_l|}\right)\\
&\leq n^{-i/k}\cdot n^{(i+1)/k} = n^{1/k}
\end{aligned}$$

The inequality holds as the summation is maximized when $|y_j| = 1, \forall j \geq 0$. Therefore, the expected size of $B_i \cap \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1})$ is also at most $n^{1/k}$. ◁

To bound the total size of shortest paths from x to $B_i \cap \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1})$, we need the one more statement below.

▷ Claim 18. Fix any node x, and for any $y \in B_i \cap \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1})$ and node z on the shortest path from x to y in graph $G/\mathcal{C}^{(\iota)}$, we have $y \in \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(z, B_{i+1})$.

Proof. By triangle inequality, we have:

$$\begin{aligned}
\mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(z, y) &= \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(x, y) - \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(x, z)\\
&< \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1}) - \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(x, z)\\
&< \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(z, B_{i+1})
\end{aligned}$$

So by definition of bunches, we have $y \in \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(z, B_{i+1})$. ◁

By Claim 18, if the data structure needs to store the shortest path $\gamma_{x,y}$ from x to $y \in B_i \cap \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1})$, then it also needs to store each shortest path $\gamma_{z,y}$ for any $z \in \gamma_{x,y}$. Therefore, to store all the path $\{\gamma_{x,y}\}$ in a space-efficient manner, we only need to store the first edge of $\gamma_{x,y}$ (which is incident on x) for all x in $G/\mathcal{C}^{(\iota)}$ and $y \in B_i \cap \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(x, B_{i+1})$; this also allows us to retrieve the path $\gamma_{x,y}$ in $|\gamma_{x,y}|$ time. Then, by Claim 17, the total size of this shortest path data structure over all node x in $G/\mathcal{C}^{(\iota)}$ would be bounded by $O(n^{1+1/k}/\tau_\iota)$, which is $O(n^{1+1/k})$ summing over all $0 \leq i \leq 2\lceil\log^2 k\rceil$.

4.4 Stretch analysis

Let us analyze the stretch of the high-level phase and the low-level phase separately.

High-level phase

By the algorithm description, suppose that the subroutine $\textbf{TreeCover}\,(u_\iota, v_\iota, \mathcal{T}_\iota)$ does not return any nonempty value. Then, by the binary search procedure, in the end we will find an index $1 \le l \le \iota$ such that $\textbf{TreeCover}\,(u_l, v_l, \mathcal{T}_l)$ returns null and $\textbf{TreeCover}\,(u_{l-1}, v_{l-1}, \mathcal{T}_{l-1})$ returns a nonempty value. Define $x = \textbf{piv}_{G/\mathcal{C}^{(l)}}\,(u_l, B_{h_l})$ and $y = \textbf{piv}_{G/\mathcal{C}^{(l)}}\,(v_l, B_{h_l})$.

▷ Claim 19. If $1 \le l \le \kappa$, then:

$$\textbf{dist}_G(x,y) \le \big(3(1+\epsilon)^2 h_l + 1\big) \cdot \textbf{dist}_G(s,t)$$

If $\kappa < l \le \iota$, then:

$$\textbf{dist}_{G/\mathcal{C}^{(l)}}(x,y) \le \big(3(1+\epsilon)^2 h_l + 1\big) \cdot \textbf{dist}_{G/\mathcal{C}^{(l)}}(u_l,v_l)$$

Proof. For the first inequality, since $\textbf{TreeCover}\,(s,t,\mathcal{T}_l)$ does not return anything, by the guarantee of Lemma 10, we know that:

$$\textbf{dist}_G(s,t) > \frac{2}{3(1+\epsilon)^2 h_l} \cdot \max\left\{\textbf{dist}_G\left(s, A_{h_l}\right), \textbf{dist}_G\left(t, A_{h_l}\right)\right\}$$

Thus, by triangle inequality, we have:

$$\textbf{dist}_G(x,y) \le \textbf{dist}_G(x,s) + \textbf{dist}_G(s,t) + \textbf{dist}_G(t,y) \le \big(3(1+\epsilon)^2 h_l + 1\big) \cdot \textbf{dist}_G(s,t)$$

Similarly, for the second inequality, since $\textbf{TreeCover}\,(u_l, v_l, \mathcal{T}_l)$ does not return anything, by the guarantee of Lemma 10, we know that:

$$\textbf{dist}_{G/\mathcal{C}^{(l)}}(u_l,v_l) > \frac{2}{3(1+\epsilon)^2 h_l} \cdot \max\{\textbf{dist}_{G/\mathcal{C}^{(l)}}\left(u_l, B_{h_l}\right), \textbf{dist}_{G/\mathcal{C}^{(l)}}\left(v_l, B_{h_l}\right)\}$$

Thus, by triangle inequality, we have:

$$\begin{aligned}\textbf{dist}_{G/\mathcal{C}^{(l)}}(x,y) &\le \textbf{dist}_{G/\mathcal{C}^{(l)}}(x,u_l) + \textbf{dist}_{G/\mathcal{C}^{(l)}}(u_l,v_l) + \textbf{dist}_{G/\mathcal{C}^{(l)}}(v_l,y)\\ &\le \big(3(1+\epsilon)^2 h_l + 1\big) \cdot \textbf{dist}_{G/\mathcal{C}^{(l)}}(u_l,v_l)\end{aligned}$$ ◁

Next, let us study two possibilities. First, assume that the shortest path between nodes x, y in $G/\mathcal{C}^{(l)}$ belongs to Π_l.

- $1 \le l \le \kappa$.
  In this case, using Claim 19, the path reported by our distance oracle has length at most:

$$\textbf{dist}_G(s,x) + \textbf{dist}_G(x,y) + \textbf{dist}_G(y,t) \le \big(6(1+\epsilon)^2 h_l + 1\big) \cdot \textbf{dist}_G(s,t) \le 5k \cdot \textbf{dist}_G(s,t)$$

- $\kappa < l \le \iota$.
  In this case, let π' be the concatenation of shortest paths between u_l, x, and x, y, and y, v_l in graph $G/\mathcal{C}^{(l)}$. Using Claim 19 and triangle inequality, we have:

$$\begin{aligned}\mathbf{w}(\pi') &= \textbf{dist}_{G/\mathcal{C}^{(l)}}(u_l,x) + \textbf{dist}_{G/\mathcal{C}^{(l)}}(x,y) + \textbf{dist}_{G/\mathcal{C}^{(l)}}(y,v_l)\\ &\le \big(6(1+\epsilon)^2 h_l + 1\big) \cdot \textbf{dist}_{G/\mathcal{C}^{(l)}}(u_l,v_l)\end{aligned}$$

Applying Lemma 5 on π' and then we can obtain a path between s, t with length at most:

$$\begin{aligned} 4\tau_l \cdot \left(6(1+\epsilon)^2 h_l + 1\right) \cdot \mathbf{dist}_{G/\mathcal{C}^{(l)}}(u_l, v_l) &\leq 4\tau_l \cdot \left(6(1+\epsilon)^2 h_l + 1\right) \cdot \mathbf{dist}_G(s,t) \\ &\leq 8\alpha^{-l/5} \cdot \left(6(1+\epsilon)^2 \lceil \alpha^l k \rceil + 1\right) \cdot \mathbf{dist}_G(s,t) \\ &< 50\alpha^{4l/5} k \cdot \mathbf{dist}_G(s,t) \\ &< 10k \cdot \mathbf{dist}_G(s,t) \end{aligned}$$

Next, consider the case where the shortest path between nodes x, y in $G/\mathcal{C}^{(l)}$ does not belong to Π_l. Consider two possibilities.

- $1 \leq l \leq \kappa$.
  In this case, using Claim 19, we know $\mathbf{dist}_G(x,y) \leq \left(3(1+\epsilon)^2 h_l + 1\right) \cdot \mathbf{dist}_G(s,t)$. Also, in the proof of Claim 19, we have also shown:

$$\mathbf{dist}_G(s,t) > \frac{2}{3(1+\epsilon)^2 h_l} \cdot \max\left\{\mathbf{dist}_G(s,x), \mathbf{dist}_G(t,y)\right\}$$

  Now, since $y \notin \mathbf{Bun}_G^{1/2}\left(x, A_{h_{l-1}}\right)$ and $x \notin \mathbf{Bun}_G^{1/2}\left(y, A_{h_{l-1}}\right)$, we know that:

$$\max\left\{\mathbf{dist}_G\left(x, A_{h_{l-1}}\right), \mathbf{dist}_G\left(y, A_{h_{l-1}}\right)\right\} \leq 2\mathbf{dist}_G(x,y)$$

  Therefore, we have:

$$\begin{aligned} &\max\left\{\mathbf{dist}_G\left(s, A_{h_{l-1}}\right), \mathbf{dist}_G\left(t, A_{h_{l-1}}\right)\right\} \\ &\leq \max\left\{\mathbf{dist}_G(s,x) + \mathbf{dist}_G\left(x, A_{h_{l-1}}\right), \mathbf{dist}_G(t,y) + \mathbf{dist}_G\left(y, A_{h_{l-1}}\right)\right\} \\ &\leq \left(\frac{15(1+\epsilon)^2 h_l}{2} + 2\right) \cdot \mathbf{dist}_G(s,t) \end{aligned}$$

  As **TreeCover** $(s, t, \mathcal{T}_l)$ successfully returns a nonempty value, by Lemma 10, the path retrieved between s, t has length at most ($k > 100, \epsilon < 0.1$):

$$\begin{aligned} &\left(\frac{15(1+\epsilon)h_l}{(1-\epsilon)} + 4(1+\epsilon)\right) \cdot \mathbf{dist}_G(s,t) \\ &< \left(\frac{15(1+\epsilon)\lceil (0.75+\epsilon)k \rceil}{(1-\epsilon)} + 4(1+\epsilon)\right) \cdot \mathbf{dist}_G(s,t) \\ &< 12k \cdot \mathbf{dist}_G(s,t) \end{aligned}$$

- $l > \kappa$.
  In this case, using Claim 19, we know that $\mathbf{dist}_{G/\mathcal{C}^{(l)}}(x,y) \leq \left(3(1+\epsilon)^2 h_l + 1\right) \cdot \mathbf{dist}_{G/\mathcal{C}^{(l)}}(u_l, v_l)$. Also, in the proof of Claim 19, we have also show:

$$\mathbf{dist}_{G/\mathcal{C}^{(l)}}(u_l, v_l) > \frac{2}{3(1+\epsilon)^2 h_l} \cdot \max\{\mathbf{dist}_{G/\mathcal{C}^{(l)}}(u_l, x), \mathbf{dist}_{G/\mathcal{C}^{(l)}}(v_l, y)\}$$

  Now, since $x \notin \mathbf{Bun}_{G/\mathcal{C}^{(l)}}(y, C_{h_{l-1}})$ and $y \notin \mathbf{Bun}_{G/\mathcal{C}^{(l)}}(x, C_{h_{l-1}})$, we know that:

$$\max\left\{\mathbf{dist}_{G/\mathcal{C}^{(l)}}\left(x, C_{h_{l-1}}\right), \mathbf{dist}_{G/\mathcal{C}^{(l)}}\left(y, C_{h_{l-1}}\right)\right\} \leq 2\mathbf{dist}_{G/\mathcal{C}^{(l)}}(x,y)$$

  Therefore, we have:

$$\begin{aligned} &\max\left\{\mathbf{dist}_{G/\mathcal{C}^{(l)}}\left(u_l, C_{h_{l-1}}\right), \mathbf{dist}_{G/\mathcal{C}^{(l)}}\left(v_l, C_{h_{l-1}}\right)\right\} \\ &\leq \max\left\{\mathbf{dist}_{G/\mathcal{C}^{(l)}}(u_l, x) + \mathbf{dist}_{G/\mathcal{C}^{(l)}}\left(x, C_{h_{l-1}}\right), \mathbf{dist}_{G/\mathcal{C}^{(l)}}(v_l, y) + \mathbf{dist}_{G/\mathcal{C}^{(l)}}\left(y, C_{h_{l-1}}\right)\right\} \\ &\leq \left(\frac{15(1+\epsilon)^2 h_l}{2} + 2\right) \cdot \mathbf{dist}_{G/\mathcal{C}^{(l)}}(u_l, v_l) \end{aligned}$$

Take an arbitrary vertex $z \in V$ contracted in the node $\mathbf{piv}_{G/\mathcal{C}^{(l)}}(u_l, C_{h_{l-1}})$. Then, applying Lemma 5, we can bound $\mathbf{dist}_G(s, A_{h_{l-1}})$ as:

$$\begin{aligned}\mathbf{dist}_G(s, A_{h_{l-1}}) &\leq \mathbf{dist}_G(s, z) \leq 4\tau_l \cdot \left(\frac{15(1+\epsilon)^2 h_l}{2} + 2\right) \cdot \mathbf{dist}_{G/\mathcal{C}^{(l)}}(u_l, v_l) \\ &< 8\alpha^{-l/5} \cdot 8\alpha^l k \cdot \mathbf{dist}_G(s,t) = 64\alpha^{4l/5} k \cdot \mathbf{dist}_G(s,t)\end{aligned}$$

Similarly we can prove that $\mathbf{dist}_G(t, A_{h_{l-1}}) < 64\alpha^{4l/5} k \cdot \mathbf{dist}_G(s,t)$.
As $\mathbf{TreeCover}\,(u_{l-1}, v_{l-1}, \mathcal{T}_{l-1})$ successfully returns a nonempty value, by Lemma 10, the path retrieved between u_{l-1}, v_{l-1} in $G/\mathcal{C}^{(l-1)}$ has length at most ($k > 100, \epsilon < 0.1$):

$$\begin{aligned}&2(1+\epsilon)\max\left\{\mathbf{dist}_{G/\mathcal{C}^{(l-1)}}\left(u_{l-1}, B_{h_{l-1}}\right), \mathbf{dist}_{G/\mathcal{C}^{(l-1)}}\left(v_{l-1}, B_{h_{l-1}}\right)\right\} \\ &\leq 2(1+\epsilon)\cdot\max\left\{\mathbf{dist}_G(s, A_{h_{l-1}}), \mathbf{dist}_G(t, A_{h_{l-1}})\right\} \\ &< 130 \cdot \alpha^{4l/5} k \cdot \mathbf{dist}_G(s,t)\end{aligned}$$

Then, applying Lemma 5 again, we can unpack this path and retrieve a path between s, t with weight at most (note that $l > \kappa = 50$ and $\alpha = \frac{3}{4} + \epsilon$):

$$4\tau_l \cdot 130 \cdot \alpha^{4l/5} k \cdot \mathbf{dist}_G(s,t) < 1040 \cdot \alpha^{3l/5} k \cdot \mathbf{dist}_G(s,t) < 10k \cdot \mathbf{dist}_G(s,t)$$

Low-level phase

Next, let us turn to the stretch if the query procedure is in the low-level phase. By the algorithm description, we should assume that the subroutine $\mathbf{TreeCover}\,(u_\iota, v_\iota, \mathcal{T}_\iota)$ returns a nonempty value.

▷ Claim 20. During the binary search procedure on the index pair (i_1, i_2), it always holds that $\mathbf{dist}_{G/\mathcal{C}^{(\iota)}}\,(u, B_{i_1}) \leq i_1 \cdot \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}\,(u, v)$. Plus, i_2 is always (u, v)-terminal.

Proof. The second half of the statement is straightforward. So, let us prove the first half by an induction. At the beginning of the algorithm, $i_1 = 0$, and thus $\mathbf{dist}_{G/\mathcal{C}^{(\iota)}}\,(u, B_{i_1}) = 0$. In each iteration, if i_3 is (u, v)-terminal, then i_1 does not change, so the induction holds. Otherwise, since $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}\,(u, B_{i_3}) \notin \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}\,(v, B_{i_3+1})$ and $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}\,(v, B_{i_3+1}) \notin \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}\,(u, B_{i_3+2})$, we know that by definition of bunches:

$$\mathbf{dist}_{G/\mathcal{C}^{(\iota)}}\,(v, B_{i_3+1}) \leq \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(u, v) + \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}\,(u, B_{i_3})$$

$$\mathbf{dist}_{G/\mathcal{C}^{(\iota)}}\,(u, B_{i_3+2}) \leq \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(u, v) + \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}\,(v, B_{i_3+1})$$

Hence, $\Delta(u, i_3) \leq 2\mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(u, v)$. Since $\Delta(u, i_3)$ is the maximum among $\Delta(u, i), i_1 \leq i \leq j-2$, we know that $\mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(u, B_j) \leq j \cdot \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(u, v)$. ◁

In the end, when the algorithm terminates, we have $i_1 = i_2 = i$. By the above claim, we know that $\mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(u, B_i) \leq i \cdot \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(u, v)$, and either $i = 2\lceil \log^2 k \rceil$ or i is (u, v)-terminal. Consider two possibilities.

- $i = 2\lceil \log^2 k \rceil$.
  In this case, as $\mathbf{TreeCover}(u, v, \mathcal{T}_\iota)$ successfully returned a nonempty value, by Lemma 10, the path length between u, v in $G/\mathcal{C}^{(\iota)}$ reported by our distance oracle is at most:

$$2(1+\epsilon) \cdot \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(u, B_i) < 5\log^2 k \cdot \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(u, v)$$

  If we unpack the clusters in $\mathcal{C}^{(\iota)}$, according to Lemma 5, this path has length at most:

$$20\tau_\iota \cdot \log^2 k \cdot \mathbf{dist}_G(u, v) < 40k^{1/5}\log^2 k \cdot \mathbf{dist}_G(u, v) < 10k \cdot \mathbf{dist}_G(u, v)$$

- $i < 2\lceil \log^2 k \rceil$.
  In this case, as i is (u,v)-terminal, we know that either $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(u, B_i) \in \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(v, B_{i+1})$ or $\mathbf{piv}_{G/\mathcal{C}^{(\iota)}}(v, B_{i+1}) \in \mathbf{Bun}_{G/\mathcal{C}^{(\iota)}}(u, B_{i+1})$. Therefore, our data structure can report a path in $G/\mathcal{C}^{(\iota)}$ of length at most $(2i+1)\cdot \mathbf{dist}_{G/\mathcal{C}^{(\iota)}}(u,v)$ between u, v. If we unpack the clusters in $\mathcal{C}^{(\iota)}$, according to Lemma 5, this path has length at most:

$$4\tau_\iota \cdot (2i+1)\cdot \mathbf{dist}_G(u,v) < 10k \cdot \mathbf{dist}_G(u,v)$$

References

1 Michael A Bender and Martin Farach-Colton. The LCA problem revisited. In *LATIN 2000: Theoretical Informatics: 4th Latin American Symposium, Punta del Este, Uruguay, April 10-14, 2000 Proceedings 4*, pages 88–94. Springer, 2000.

2 Michael A Bender and Martın Farach-Colton. The level ancestor problem simplified. *Theoretical Computer Science*, 321(1):5–12, 2004.

3 Marcel Bezdrighin, Michael Elkin, Mohsen Ghaffari, Christoph Grunau, Bernhard Haeupler, Saeed Ilchi, and Václav Rozhoň. Deterministic distributed sparse and ultra-sparse spanners and connectivity certificates. In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 1–10, 2022.

4 Shiri Chechik. Approximate distance oracles with constant query time. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 654–663, 2014.

5 Shiri Chechik. Approximate distance oracles with improved bounds. In *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, pages 1–10, 2015.

6 Don Coppersmith and Michael Elkin. Sparse sourcewise and pairwise distance preservers. *SIAM Journal on Discrete Mathematics*, 20(2):463–501, 2006.

7 Michael Elkin and Seth Pettie. A linear-size logarithmic stretch path-reporting distance oracle for general graphs. *ACM Transactions on Algorithms (TALG)*, 12(4):1–31, 2016.

8 Michael Elkin and Idan Shabat. Path-Reporting Distance Oracles with Logarithmic Stretch and Size $O(n \log\log n)$. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2278–2311. IEEE, 2023.

9 Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. *Journal of the European Mathematical Society*, 9(2):253–275, 2007.

10 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005.

11 Christian Wulff-Nilsen. Approximate distance oracles with improved query time. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 539–549. SIAM, 2013.

A Missing proofs

▶ **Lemma 21** (restate Lemma 9). *Given an undirected weighted graph $G = (V, E, \mathbf{w})$ and a collection of shortest paths Π, there is a distance-preserving path-reporting data structure that reports any path $\pi \in \Pi$ in time $|\pi|$. The data structure has size $O\left(n + |\Pi| + \sqrt{n|\mathbf{br}(\Pi)|}\right)$ where $\mathbf{br}(\Pi)$ is the set of branching events of Π.*

Proof. First, for any path $\pi \in \Pi$, store both of endpoints in V as well as its first and last edges. Since all paths in Π are shortest paths, there are no two paths sharing the same pair of endpoints. For any path $\pi \in \Pi$, we can identify π with its starting and ending vertex which is denoted by $\mathsf{ID}(\pi)$. This part takes space $O(n + |\Pi|)$.

We will build a routing table data structure $\mathcal{D}_u$ at any vertex $u \in V$. That is, for any path $\pi \in \Pi$ that passes through u through a pair of edges $(v, u), (u, w)$, given the ID of π, the routing data structure $\mathcal{D}_u$ is able to answer the edges $(v, u), (u, w)$ in constant time. Once this is goal is fulfilled, as we have also stored starting and ending edges of each path, our path-reporting distance preserver is complete. For the rest, let us focus on the design of $\mathcal{D}_u$.

For any vertex $u \in V$, let Π_u be the set of paths in Π that pass through u, and for any unordered pairs of different neighbors v, w of u in G, let $\Pi_u^{\{v,w\}}$ be the set of paths that use edges $(v, u), (u, w)$ to go through u, and so $|\Pi_u| = \sum_{v \neq w} \left|\Pi_u^{\{v,w\}}\right|$, and also by definition of branching events, the number of branchings at x is equal to:

$$|\mathbf{br}(\Pi_u)| = \sum_{\{v,w\} \neq \{v',w'\}} \left|\Pi_u^{\{v,w\}}\right| \cdot \left|\Pi_u^{\{v',w'\}}\right| = \frac{1}{2}\left(|\Pi_u|^2 - \sum_{v \neq w} \left|\Pi_u^{\{v,w\}}\right|^2\right)$$

Consider two different cases.

- For any vertex pair v, w, we have $\left|\Pi_u^{\{v,w\}}\right| \leq \frac{1}{2} \cdot |\Pi_u|$.
  In this case, for each path $\pi \in \Pi_u$ using edges $(v, u), (u, w)$, store a triple $(\mathsf{ID}(\pi), v, w)$; all such triples will be stored in a hash table that supports constant-time queries using path IDs.
  As for the space of $\mathcal{D}_u$, on the one hand, it is $O(|\Pi_u|)$. On the other hand, under this condition we have:

$$|\mathbf{br}(\Pi_u)| = \frac{1}{2}\left(|\Pi_u|^2 - \sum_{v \neq w} \left|\Pi_u^{\{v,w\}}\right|^2\right) \geq \frac{1}{2}\left(|\Pi_u|^2 - \frac{1}{2}|\Pi_u|^2\right) = \frac{1}{4}|\Pi_u|^2$$

  Therefore, $|\mathcal{D}_u| = O\left(\sqrt{|\mathbf{br}(\Pi_u)|}\right)$.
- There exists a vertex pair v_*, w_* such that $\left|\Pi_u^{\{v_*,w_*\}}\right| > \frac{1}{2} \cdot |\Pi_u|$.
  In this case, for each path $\pi \in \Pi_u$ using edges $(v, u), (u, w)$ such that $\{v, w\} \neq \{v_*, w_*\}$, store a triple $(\mathsf{ID}(\pi), v, w)$; all these triples will be stored in a hash table that supports constant-time queries using path IDs. For those paths π which pass through u using edges $(v_*, u), (u, w_*)$, we can query this hash table with $\mathsf{ID}(\pi)$ which returns nothing, and then answer the query with $\{(v_*, u), (u, w_*)\}$.
  As for the size of $\mathcal{D}_u$, on the one hand, its space is $O\left(\Pi_u \setminus \Pi_u^{\{v_*,w_*\}}\right)$. On the other hand, we have:

$$\begin{aligned}|\mathbf{br}(\Pi_u)| &= \frac{1}{2}\left(|\Pi_u|^2 - \sum_{v \neq w} \left|\Pi_u^{\{v,w\}}\right|^2\right) \geq \frac{1}{2} \cdot \left(|\Pi_u|^2 - \left|\Pi_u^{\{v_*,w_*\}}\right|^2 - \left|\Pi_u \setminus \Pi_u^{\{v_*,w_*\}}\right|^2\right) \\ &\geq \left|\Pi_u^{\{v_*,w_*\}}\right| \cdot \left|\Pi_u \setminus \Pi_u^{\{v_*,w_*\}}\right| \geq \left|\Pi_u \setminus \Pi_u^{\{v_*,w_*\}}\right|^2\end{aligned}$$

  Therefore, we also have $|\mathcal{D}_u| = O\left(\sqrt{\mathbf{br}(\Pi_u)}\right)$

In either case, we are able to show $|\mathcal{D}_u| = O\left(\sqrt{\mathbf{br}(\Pi_u)}\right)$, and so the total size can be bounded by:

$$\sum_{u \in V} |\mathcal{D}_u| \leq \sqrt{n \sum_{u \in V} |\mathcal{D}_u|^2} = O\left(\sqrt{n|\mathbf{br}(\Pi)|}\right)$$ ◀