


# Interval Query Problem on Cube-Free Median Graphs

Soh Kumabe 

The University of Tokyo, Japan

---

## Abstract

In this paper, we introduce the *interval query problem* on cube-free median graphs. Let  $G$  be a cube-free median graph and  $\mathcal{S}$  be a commutative semigroup. For each vertex  $v$  in  $G$ , we are given an element  $p(v)$  in  $\mathcal{S}$ . For each query, we are given two vertices  $u, v$  in  $G$  and asked to calculate the sum of  $p(z)$  over all vertices  $z$  belonging to a  $u - v$  shortest path. This is a common generalization of range query problems on trees and grids. In this paper, we provide an algorithm to answer each interval query in  $O(\log^2 n)$  time. The required data structure is constructed in  $O(n \log^3 n)$  time and  $O(n \log^2 n)$  space. To obtain our algorithm, we introduce a new technique, named the *staircases decomposition*, to decompose an interval of cube-free median graphs into simpler substructures.

**2012 ACM Subject Classification** Mathematics of computing → Combinatorial algorithms

**Keywords and phrases** Data Structures, Range Query Problems, Median Graphs

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2021.18

**Related Version** *Full Version*: <https://arxiv.org/abs/2010.05652>

**Acknowledgements** We are grateful to our supervisor Prof. Hiroshi Hirai for supporting our work. He gave us a lot of ideas to improve our paper. In particular, he simplified the proofs and helped us improve the introduction and the overall structure of this paper.

## 1 Introduction

The *range query problem* [18] is one of the most fundamental problems in the literature on data structures, particularly for string algorithms [19]. Let  $f$  be a function defined on arrays. In the range query problem, we are given an array  $P = (p(1), \dots, p(n))$  of  $n$  elements and a *range query* defined by two integers  $i, j$  with  $1 \leq i \leq j \leq n$ . For each query  $(i, j)$ , we are asked to return the value  $f((p(i), \dots, p(j)))$ . The main interest of this problem is the case where  $f$  is defined via a *semigroup operator* [27]. Let  $\mathcal{S}$  be a semigroup with operator  $\oplus$ , and let  $P$  consist of elements in  $\mathcal{S}$ . Then, the function  $f$  is defined as  $f((p(i), \dots, p(j))) = p(i) \oplus \dots \oplus p(j)$ . Typical examples of semigroup operators are sum, max, and min. The fundamental result [27, 28] is that for any constant integer  $k$ , a range query can be answered in  $O(\alpha_k(n))$  time, where  $\alpha_k$  is a slow-growing function related to the inverse of the Ackermann function. The required data structure is constructed in linear time and space. *Range minimum query problem*, i.e.,  $\oplus = \min$ , is one of the well-studied problems in the literature, and it admits a constant-time algorithm with a data structure constructed in linear time and space [1, 4, 5, 18, 20].

This problem is generalized into trees and grids. In these settings, we are given a tree/grid  $G$  and an element  $p(v)$  for each vertex of  $G$ . As a query, given two vertices  $u, v$  in  $G$ , we are asked to calculate the sum<sup>1</sup> of the elements assigned at the vertices on a  $u - v$  shortest path. In particular, we are asked to calculate the sum of the elements on the unique  $u - v$  path for trees and the axis-parallel rectangle with corners  $(u, v)$  on its diagonal for grids. For constant

---

<sup>1</sup> In this paper, for simplicity, we represent the semigroup operation by the terms of summation; that is, we denote  $a \oplus a'$  by the word *sum* of  $a$  and  $a'$  for  $a, a' \in \mathcal{S}$ .



© Soh Kumabe;

licensed under Creative Commons License CC-BY 4.0

32nd International Symposium on Algorithms and Computation (ISAAC 2021).

Editors: Hee-Kap Ahn and Kunihiro Sadakane; Article No. 18; pp. 18:1–18:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

dimensional grids, an almost-constant time algorithm [11] with linear space on semigroup operators and a constant-time algorithm for range minimum query is known [29]. For range query problem on trees, an almost-constant time algorithm [9] with linear space is known on semigroup operators; see [8] for further survey on the problem on trees, particularly for dynamic version.

In this paper, we introduce a common generalization of the two above mentioned cases, named *interval query problem on median graphs*. Let  $G = (V(G), E(G))$  be a connected graph with  $n$  vertices. For two vertices  $u, v \in V(G)$ , let the *interval*  $I[u, v]$  be the set of vertices belonging to a  $u - v$  shortest path, where the length of a path is defined by the number of its edges. The graph  $G$  is called a *median graph* if for all  $u, v, w \in V(G)$ ,  $I[u, v] \cap I[v, w] \cap I[w, u]$  is a singleton [2, 7, 23]. The median graph  $G$  is said to be *cube-free* if  $G$  does not contain a cube as an induced subgraph. Trees and grids are examples of cube-free median graphs. In our problem, we are given a median graph  $G$  and an element  $p(v)$  of a commutative semigroup  $\mathcal{S}$  for each vertex  $v$  of  $G$ . As a query, given two vertices  $u, v$  in  $G$ , we are asked to calculate  $p(I[u, v])$ <sup>2</sup>. The interval query problem on cube-free median graphs is a common generalization of the range query problems on trees and grids.

In this paper, we provide an algorithm to the interval query problem on cube-free median graphs. The main result here is presented as follows:

► **Theorem 1.** *There is an algorithm to answer interval queries on cube-free median graphs in  $O(\log^2 n)$  time. The required data structure is constructed in  $O(n \log^3 n)$  time and  $O(n \log^2 n)$  space, where  $n$  is the number of vertices in a given cube-free median graph.*

The time complexity of answering a query matches the complexity for the two-dimensional *range tree* [21] in the *orthogonal range query problem*, without acceleration via *fractional cascading* [10].

To obtain the algorithm, we introduce a new technique, named the *staircases decomposition*. This technique provides a new method to decompose an interval of cube-free median graphs into a constant number of smaller intervals. Most of the candidates of the smaller intervals, which we refer to as *staircases*, are well-structured, and an efficient algorithm to answer the interval queries can be constructed. The rest are not necessarily staircases; however, each of them are one of the  $O(n \log n)$  candidates, and we can precalculate all the answers of the interval queries on these intervals.

Designing fast algorithms for median graphs is a recently emerging topic. The *distance labeling scheme* [24] is a type of data structure that is defined by the encoder and decoder pair. The encoder receives a graph and assigns a label for each vertex, whereas the decoder receives two labels and computes the distance of the two vertices with these labels. For cube-free median graphs, there is a distance labeling scheme that assigns labels with  $O(\log^3 n)$  bits for each vertex [13]. Very recently, a linear-time algorithm to find the *median* of median graphs was built [6]. This paper continues with this line of research and utilizes some of the techniques presented in these previous studies.

Various applications can be considered in the interval query problem on median graphs. The solution space of a 2-SAT formula forms a median graph, where two solutions are adjacent if one of them can be obtained by negating a set of pairwise dependent variables of the other [3, 22, 26]. For two solutions  $u$  and  $v$ , the interval  $I[u, v]$  corresponds to the set of the solutions  $x$ , such that for each truth variable, if the same truth value is assigned in  $u$  and  $v$ , so does  $x$ . Suppose we can answer the interval queries to calculate sum (resp.

<sup>2</sup> For a vertex subset  $X$ , we denote the sum of  $p(z)$  over all  $z \in X$  by  $p(X)$ .

min) in polylogarithmic time with a data structure of subquadratic time and space. Then, if we have the list of all feasible solutions of the given 2-SAT formula, we can calculate the number (resp. minimum weight) of these solutions in polynomial time of the number of variables for each query, without precalculating the answers for all possible queries. Note that, there is a polynomial-delay algorithm to enumerate all solutions to the given 2-SAT formula [16]. Therefore, if the number of the feasible solutions (and thus the number of vertices in the corresponding median graph) is small, we can efficiently list them. In social choice theory, the structure of median graphs naturally arises as a generalization of *single-crossing preferences* [15, 17] and every closed Condorcet domains admits the structure of a median graph [25]. For two preferences  $u$  and  $v$ , the voters with their preferences in interval  $I[u, v]$  prefer candidate  $x$  to candidate  $y$  whenever both  $u$  and  $v$  prefer  $x$  to  $y$ . Therefore, using interval query, we can count the number of voters  $w$  such that for all pairs of candidates, at least one of  $u$  and  $v$  has the same preference order as  $w$  between these candidates. Although these structures are not necessarily cube-free, we hope that our result will be the first and important step toward obtaining fast algorithms for these problems.

## 1.1 Algorithm Overview

Here we give high-level intuition to our algorithm. More detailed outline is given in Section 3.

Let  $G$  be a cube-free median graph. The first idea for our algorithm is to decompose  $G$  recursively. We recursively divide  $V(G)$  into some parts, called *fibers*. Roughly speaking, a fiber is a set of the vertices located on the similar direction from the special vertex  $m$  (see Figure 1). Each fiber induces a cube-free median graph and, if we take  $m$  properly, has at most  $|V(G)|/2$  vertices; there are at most  $O(\log n)$  recursion steps.

Let  $u, v$  be vertices of  $G$ . Consider calculating  $p(I[u, v])$ . If  $u$  and  $v$  are in the same fiber, we calculate it recursively. Otherwise, we can show that  $I[u, v]$  intersects with only a constant number of fibers and the intersections are intervals with one end on the boundary of the fiber (Section 6, see Figure 8). Thus, it is sufficient to construct an algorithm on such intervals.

To do this, we further decompose such an interval into more well-structured intervals, using our main technique named *staircases decomposition* (Section 4, see Figure 4, 5, and 6). Roughly speaking, we decompose the interval into at most two structured substructures names *staircases* (Figure 2) and a special interval of  $O(n)$  candidates. For special intervals  $I$ , we just use the precalculated  $p(V(I))$ . For staircases  $L$ , we construct an algorithm to calculate  $p(V(L))$  in  $O(\log^2 n)$  time (Section 5), using the fact that the boundary of the fiber is actually a tree [13]. We decompose this tree into paths by heavy-light decomposition and build segment trees to answer the queries.

## 2 Basic Tools for Cube-Free Median Graphs and Trees

In this section, we introduce basic facts about cube-free median graphs and trees.

Let  $G$  be a connected, undirected, finite graph. We denote the vertex set of  $G$  by  $V(G)$ . For two vertices  $u$  and  $v$  in  $G$ , we write  $u \sim v$  if  $u$  and  $v$  are adjacent. For two vertices  $u$  and  $v$  of  $G$ , the *distance*  $d(u, v)$  between them is the minimum number of edges on a path connecting  $u$  and  $v$ , and the *interval*  $I[u, v]$  is the set of vertices  $w$  which satisfies  $d(u, v) = d(u, w) + d(w, v)$ . The graph  $G$  is a *median graph* if for any three vertices  $u, v, w$ ,  $I[u, v] \cap I[v, w] \cap I[w, u]$  contains exactly one vertex, called *median* of  $u, v$  and  $w$ . Median graphs are bipartite and do not contain  $K_{2,3}$  as a subgraph. A median graph is *cube-free* if it does not contain a (three-dimensional) cube graph as an induced subgraph. The followings hold.

## 18:4 Interval Query Problem on Cube-Free Median Graphs

► **Lemma 2** ([14]). *Any interval in a cube-free median graph induces an isometric subgraph of a two-dimensional grid.*

► **Lemma 3** ([13]). *Let  $u, v, w_1, w_2$  be four pairwise distinct vertices of a median graph such that  $v \sim w_1, v \sim w_2$  and  $d(u, v) - 1 = d(u, w_1) = d(u, w_2)$ . Then, there is unique vertex  $z$  with  $w_1 \sim z, w_2 \sim z$  and  $d(u, z) = d(u, v) - 2$ .*

From now on, let  $G$  be a cube-free median graph with  $n$  vertices. Let  $X$  be a subset of  $V(G)$ . For vertex  $z \in V(G)$  and  $x \in X$ ,  $x$  is the *gate* of  $z$  in  $X$  if for all  $w \in X$ ,  $x \in I[z, w]$ . The gate of  $z$  in  $X$  is unique (if it exists) because it is the unique vertex in  $X$  that minimizes the distance from  $z$ .  $X$  is *gated* if all vertices  $z \in V(G)$  have a gate in  $X$ . The following equivalence result is known.

► **Lemma 4** ([12, 13]). *Let  $X$  be a vertex subset of the median graph  $G$ . Then, following three conditions are equivalent.*

- (a)  $X$  is gated.
- (b)  $X$  is convex, i.e.,  $I[u, v] \subseteq X$  for all  $u, v \in X$ .
- (c)  $X$  induces a connected subgraph and  $X$  is locally convex, i.e.,  $I[u, v] \subseteq X$  for all  $u, v \in X$  with  $d(u, v) = 2$ .

An induced subgraph of  $G$  is *gated* (resp. *convex*, *locally convex*) if its vertex set is gated (resp. convex, locally convex). The intersection of two convex subsets is convex. Any interval of median graphs are convex.

For a convex subset  $X$  and a vertex  $x \in X$ , the *fiber*  $F_X(x)$  of  $x$  with respect to  $X$  is the set of vertices in  $G$  whose gate in  $X$  is  $x$ . Two fibers  $F_X(x), F_X(y)$  are *neighboring* if there are vertices  $x' \in F_X(x)$  and  $y' \in F_X(y)$  such that  $x' \sim y'$ , which is equivalent to  $x \sim y$  [13]. Fibers for all  $x \in X$  define a partition of  $V(G)$ . For two adjacent vertices  $x, y \in X$ , the *boundary*  $T_X(x, y)$  of  $F_X(x)$  relative to  $F_X(y)$  is the set of the vertices which have a neighbor in  $F_X(y)$ .  $T_X(x, y)$  and  $T_X(y, x)$  are isomorphic. A vertex in  $T_X(x, y)$  has a unique neighbor in  $T_X(y, x)$ , which is the corresponding vertex under that isomorphism. For vertex  $x \in X$ , a *total boundary*  $T_X(x)$  of  $F_X(x)$  is the union of all  $T_X(x, y)$  for  $y \in X$  with  $x \sim y$ . The subgraph  $H$  is *isometric* in  $G$  if for all  $u, v \in V(H)$ , there is a path in  $H$  with length  $d(u, v)$ . A rooted tree has *gated branches* if any of its root-leaf path is convex. The next lemma exploits the structures of the boundaries of fibers of cube-free median graphs.

► **Lemma 5.** *Let  $X$  be a convex vertex subset of cube-free median graph  $G$ . Let  $x, y \in X$  and assume  $x \sim y$ . Then, the followings hold.*

- (i) ([13])  $T_X(x, y)$  induces a tree, which is convex.
- (ii) ([13])  $T_X(x)$  induces a tree with gated branches, which is isometric in  $G$ .

The following is folklore in a literature of median graphs. A proof is in full version.

► **Lemma 6** (folklore). *Let  $X$  be a convex vertex set of a median graph and let  $Y$  be a convex subset of  $X$ . For  $x \in X$ , let  $F(x)$  be the fiber of  $x$  with respect to  $X$ . Then,  $\bigcup_{y \in Y} F(y)$  is convex.*

Let  $T$  be a tree with gated branches. For a vertex  $v \in V(G)$  and  $w \in T$ ,  $w$  is an *imprint* of  $v$  if  $I[v, w] \cap T = \{w\}$ . If  $T$  is convex, the imprint is equal to the gate and therefore unique. Even if it is not the case, we can state following.

► **Lemma 7.** *Let  $T$  be a tree with gated branches rooted at  $r$ . Let  $u \in V(G)$ . Then, the following statements hold.*

- (i) ([13]) There are at most two imprints of  $u$  in  $T$ .
- (ii) Assume  $u$  has two distinct imprints  $w^1, w^2$  in  $T$ . Then,  $w^1, w^2 \in I[r, u]$ .

**Proof.** We prove (ii). From symmetry, we only prove  $w^1 \in I[r, u]$ . Let  $P_1$  be the root-leaf path of  $T$  that contains  $w^1$ . Then,  $P_1$  is convex and therefore  $d(r, u) = d(r, w^1) + d(w^1, u)$ . ◀

► **Lemma 8.** *Let  $T$  be a tree with gated branches and  $w \in V(T)$ . Then, the set of vertices with an imprint  $w$  in  $T$  is convex.*

**Proof.** Assume the contrary. Then, there are distinct vertices  $z_1, z_2, z_3$  with  $z_1 \sim z_2 \sim z_3$ , such that  $z_1$  and  $z_3$  have an imprint  $w$  but  $z_2$  doesn't. We have  $d(w, z_2) = d(w, z_1) + 1$ ; otherwise,  $d(w, z_2) = d(w, z_1) - 1$  because of bipartiteness of  $G$  holds and in this case,  $I[w, z_2] \subseteq I[w, z_1]$  holds and  $z_2$  has an imprint  $w$ . By the same reason we have  $d(w, z_2) = d(w, z_3) + 1$ . From definition of the imprint, there is a  $z_2 - w$  shortest path that contains a vertex of  $T$  other than  $w$ . Let  $z_4$  be the neighbor of  $z_2$  in this shortest path. Then,  $z_4$  does not have an imprint  $w$  and especially,  $z_1 \neq z_4 \neq z_3$ . Now we have  $d(w, z_4) + 1 = d(w, z_1) + 1 = d(w, z_3) + 1 = d(w, z_2)$  and obtain three squares that all two intersect at an edge from Lemma 3, which contradicts Lemma 2. ◀

For a vertex  $m \in V(G)$ , the *star*  $\text{St}(m)$  of  $m$  is the set of vertices  $x \in V(G)$  such that there is an edge or a square that contains both  $m$  and  $x$ .  $\text{St}(m)$  is convex. The vertex  $m \in V(G)$  is a *median* of  $G$  if it minimizes the sum of distances to all vertices in  $G$ . The following holds.

► **Lemma 9** ([13]). *All the fibers of  $\text{St}(m)$  of a median graph contains at most  $\frac{n}{2}$  vertices.*

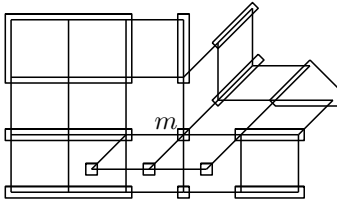
For a rooted tree  $T$  that is rooted at  $r$ , a vertex  $u \in V(T)$  is an *ancestor* of  $v$  and  $v$  is a *descendant* of  $u$  if there is a path from  $u$  to  $v$ , only going toward the leaves. The vertex subset  $X$  is a *column* of  $T$  if for any two vertices  $x, y$  in  $X$ ,  $x$  is either an ancestor or a descendant of  $y$ . The vertex  $t$  is the *lowest common ancestor* [20] of  $u$  and  $v$  if  $t$  is an ancestor of both  $u$  and  $v$  that minimizes the distance between  $u$  and  $t$  (or equivalently,  $v$  and  $t$ ) in  $T$ . There is a data structure that is constructed in linear time and space such that, given two vertices on  $T$ , it returns the lowest common ancestor of them in constant time [5].  $u$  is a *parent* of  $v$  and  $v$  is a *child* of  $u$  if  $u$  is an ancestor of  $v$  and  $u \sim v$ . Let  $X \subseteq V(T)$  and  $u \in V(T)$ . The *nearest ancestor* of  $u$  in  $X$  on  $T$  is the vertex  $v \in X$  such that  $v$  is an ancestor of  $u$  and minimizes  $d(u, v)$ .

Let  $T$  be a rooted tree rooted at  $r$ . For a vertex  $v \in V(T)$ , let  $T_v$  be the subtree of  $T$  rooted at  $v$ . An edge  $(u, v)$  in  $G$  such that  $u$  is the parent of  $v$  is a *heavy-edge* if  $|V(T_u)| \leq 2|V(T_v)|$  and a *light-edge* otherwise. Each vertex has at most one child such that the edge between them is a heavy-edge. The *heavy-path* is a maximal path that only contains heavy-edges. The *heavy-light decomposition* is the decomposition of  $T$  into heavy-paths. Note that, there is at most  $O(\log n)$  light-edges on any root-leaf path on  $T$ .

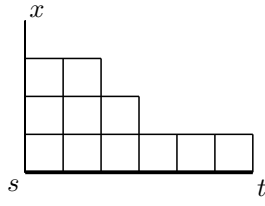
### 3 Outline and Organization

Here we roughly describe our algorithm using the notions in Section 2. Let  $G$  be a cube-free median graph. Let  $m$  be a median of  $G$ ,  $\text{St}(m)$  be the star of  $m$ , and for each  $x \in \text{St}(m)$ , let  $F(x)$  be the fiber of  $x$  in  $\text{St}(m)$  (see Figure 1). Let  $u, v$  be vertices of  $G$ .

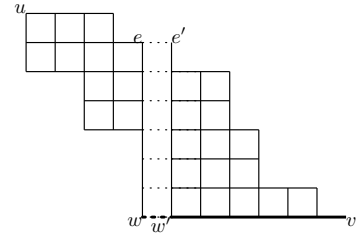
Consider calculating  $p(I[u, v])$ . If  $u$  and  $v$  are in the same fiber  $F(x)$  of  $\text{St}(m)$ , we calculate the answer by using the algorithm on  $F(x)$ , which is recursively defined. Lemma 9 ensures that the recursion depth is at most  $O(\log n)$ . Otherwise, we can show that  $I[u, v]$  intersects with only a constant number of fibers, and for each fiber  $F(x)$  that intersects  $I[u, v]$ ,  $I[u, v] \cap F(x)$  can be represented as  $I[u_x, v_x]$  for some vertices  $u_x, v_x \in F(x)$  such that  $v_x$  is on the total boundary of  $F(x)$ . Thus, it is sufficient to construct an algorithm to answer the query on the interval, such that one of the ends is on the total boundary of  $F(x)$ .



■ **Figure 1** A median graph and its decomposition into fibers of  $\text{St}(m)$ .



■ **Figure 2** Staircases with top  $x$  with base starts at  $s$  and ends at  $t$ .



■ **Figure 3** Decomposition of  $I[u, v]$  into an interval  $I[u, w]$  and staircases  $I[e', v]$ . The bold line represents  $P$ .

To do this, we introduce a technique to decompose intervals, which we name the *staircases decomposition*. Let  $T$  be a tree with gated branches and assume  $u \in V(G)$  and  $v \in V(T)$ . We partition an interval  $I[u, v]$  into an interval  $I$  and at most two special structures, which we name a *staircases* (Figure 2), which we describe in Section 4. Such a decomposition can be calculated in  $O(\log n)$  time with appropriate preprocessing. Here, we can take  $I$  as one of the  $O(n)$  candidates of intervals. We just precalculate and store the value  $p(I)$  for each candidate, and recall it when we answer the queries.

Now we just need an algorithm to calculate the value  $p(V(L))$  quickly for a staircases  $L$ . Let  $P$  be a root-leaf path of  $T$ . The segment trees can answer the staircases queries whose base is a subpath of  $P$  in  $O(\log n)$  time. To answer the general queries, we use a heavy-light decomposition of  $T$ .

The rest of the paper is organized as follows. In Section 4, we introduce the staircases decomposition of the intervals with one end on the tree with gated branches. In Section 5, we construct an algorithm and a data structure for the interval queries for the same cases. In Section 6, we prove that we can decompose a given interval into constant number of intervals with one of the ends on the total boundaries of the fibers of  $\text{St}(m)$ . This technique can also be applied to the query that asks the median of given three vertices. Some detailed parts in these sections are found in full version. An algorithm to construct our data structure efficiently is given in full version.

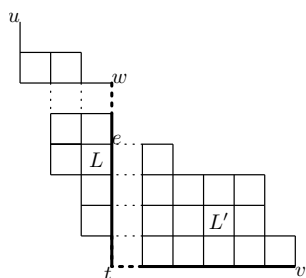
## 4 The Staircases Decomposition of the Intervals with One End on the Boundary

Let  $T$  be a tree with gated branches. In this section, we introduce a technique, *staircases decomposition*, to decompose an interval  $I[u, v]$  such that  $v$  is on  $T$ .

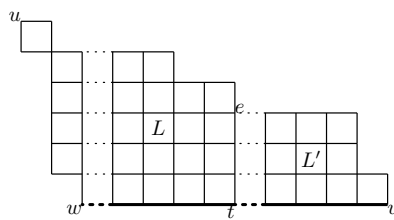
Let  $P = (s = w_0, \dots, w_k = t)$  be a convex path. For a vertex  $x$  with gate  $s$  in  $P$ , the interval  $I[x, t]$  induces *staircases* if for all  $i = 0, \dots, k$ , the set of vertices in  $I[x, t]$  with gate  $w_i$  in  $P$  induces a path.  $P$  is the *base* of  $L$  and the vertex  $x$  is the *top* of  $L$ . The base *starts* at  $s$  and *ends* at  $t$  (see Figure 2). Our staircases decomposition decomposes  $I[u, v]$  into an interval and at most two staircases such that their bases are columns of  $T$ .

### 4.1 The case with One End on a Convex Path

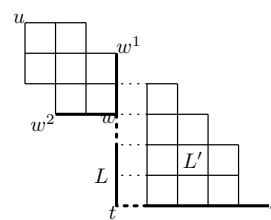
Here we investigate the structure of an interval such that one of the endpoints is on a convex path  $P$ . Consider an interval  $I[u, v]$  such that  $v$  is on  $P$ . Let  $w$  be the gate of  $u$  in  $P$ . The purpose here is to prove that  $I[u, v]$  can be decomposed into the disjoint union of an interval  $I[u, w]$  and a staircases (see Figure 3), if  $w \neq v$ . We assume  $w \neq v$  because otherwise we



■ **Figure 4** Staircases decomposition of  $I[u, v]$  (single imprint, first case).



■ **Figure 5** Staircases decomposition of  $I[u, v]$  (single imprint, second case).



■ **Figure 6** Staircases decomposition of  $I[u, v]$  (double imprints).

have no need of decomposition. Let  $w'$  be the neighbor of  $w$  in  $P$  between  $w$  and  $v$ . We take the isometric embedding of  $I[u, v]$  into a two-dimensional grid (see Lemma 2). We naively introduce a  $xy$ -coordinate system with  $w = (0, 0)$ ,  $w' = (1, 0)$ ,  $u = (x_u, y_u)$  with  $y_u \geq 0$ , and  $v = (x_v, y_v)$  with  $y_v \leq 0$ . Now, we can state the following.

► **Lemma 10.** *If a vertex  $z = (x_z, y_z)$  on  $I[u, v] \cap V(P)$  is not on the  $x$ -axis, there is no vertex other than  $z$  in  $I[u, v]$  with gate  $z$  in  $P$ .*

**Proof.** Assume the contrary and let  $z' = (x_{z'}, y_{z'})$  be a vertex in  $I[u, v]$  with gate  $z$  in  $P$ . Because of the isometricity,  $x_z > 0$  and  $y_z < 0$  holds. Since  $z \in I[w, z']$ , we have  $x_{z'} \geq x_z$ . Since  $z' \in I[u, v]$ ,  $x_v \geq x_{z'}$  holds. Therefore, we can take a vertex  $z''$  in  $P$  with  $x$ -coordinate  $x_{z'}$ , but it means  $z' \in I[w, z'']$  and contradicts to the convexity of  $P$ . ◀

Since such  $z$  does not affect the possibility of decomposition (we can just add such vertices at the end of the staircases), we can assume that  $v = (x_v, 0)$  for  $x_v > 0$ . Moreover, from convexity, we have that all vertices in  $I[u, v]$  have non-negative  $y$ -coordinate. Thus,  $I[u, v] \setminus I[u, w]$  is the set of vertices with positive  $x$ -coordinate and forms staircases (see Figure 3), which is the desired result.

To build an algorithm to calculate  $p(I[u, v])$  as the sum of  $p(I[u, w])$  and  $p(I[u, v] \setminus I[u, w])$ , we should identify the top  $e'$  of the staircases. Instead of direct identification, we rather identify the unique neighbor of it in  $I[u, w]$ , named the *entrance*  $e$  of the staircases: The top  $e'$  can be determined as the neighbor of  $e$  with gate  $w'$  on  $P$ . Here, we have that  $e$  is the gate of  $u$  in the boundary of  $F(w)$  with respect to  $F(w')$ , where  $F(w)$  (resp.  $F(w')$ ) is the fiber of  $w$  (resp.  $w'$ ) with respect to  $P$ . Indeed, this gate should be in  $I[u, w]$  from the definition of the gate and  $e$  is the only candidate for it. We can calculate  $e$  in  $O(\log n)$  time by working on the appropriate data structure on total boundary of the fiber of  $w$  with respect to  $P$ . We discuss this algorithm in full version.

## 4.2 Single Imprint

Here we give the staircases decomposition of the interval  $I[u, v]$ , where  $v$  is on a tree  $T$  with gated branches, rooted at  $r$ . First, we treat the case that there is exactly one imprint  $w$  of  $u$  in  $T$  in  $I[u, v]$ . Let  $t$  be a lowest common ancestor of  $w$  and  $v$  in  $T$ . Note that,  $t$  might coincide with  $w$  or  $v$ . Let  $P$  (resp.  $P'$ ) be the root-leaf path of  $T$  that contains  $w$  (resp.  $v$ ).

Since  $P'$  is convex, we can decompose  $I[u, v]$  into a staircases  $L'$  with base on  $P'$  and an interval  $I[u, t]$ . Since  $P$  is convex, we can further decompose the interval  $I[u, t]$  into a staircases  $L$  with base on  $P$  and an interval  $I[u, w]$ . Now, for fixed  $T$ ,  $I[u, w]$  is one of the  $O(n)$  candidates of the intervals, because it is specified only by a vertex  $u$  and one of at most two imprints of  $u$  on  $T$ . This is the staircases decomposition we obtain here.

To bound the size of the data structure we construct in Section 5, we should ensure that staircases  $L$  and  $L'$  contains only vertices with an imprint on  $P$  and  $P'$ , respectively. Let  $B_L$  (resp.  $B_{L'}$ ) be the base of  $L$  (resp.  $L'$ ). We prove the following.

► **Lemma 11.** *The following statements hold.*

- (i)  $I[u, v]$  contains no vertices in  $T$  other than the vertices on the  $w - v$  path on  $T$ .
- (ii) For a vertex  $z$  in  $L'$ , the gate of  $z$  in  $P'$  is an imprint of  $z$  in  $T$ .
- (iii) For a vertex  $z$  in  $L$ , the gate of  $z$  in  $P$  is an imprint of  $z$  in  $T$ .

**Proof.** (i) Let  $z \in I[u, v] \cap V(T)$ . Then,  $d(u, v) = d(u, z) + d(z, v)$  holds. Since  $w$  is the unique imprint of  $u$  in  $T$ ,  $d(u, z) = d(u, w) + d(w, z)$  and  $d(u, v) = d(u, w) + d(w, v)$  holds. Therefore  $d(w, v) = d(w, z) + d(z, v)$  and it means  $z$  is on the unique path between  $w$  and  $v$  on  $T$ . (ii) Let  $w'_z$  be the gate of  $z$  in  $P'$ . We prove  $I[z, w'_z] \cap V(T) = \{w'_z\}$ . Assume  $x \in (I[z, w'_z] \cap V(T)) \setminus \{w'_z\}$ . From (i),  $x$  is on  $w - t$  path. From isometricity of  $T$ ,  $t \in I[x, w'_z] \subseteq I[z, w'_z]$  holds and it contradicts the definition of  $w'_z$ . (iii) Similar to (ii). ◀

We should also make algorithms to identify the top of the staircases  $L$  and  $L'$ . The top of  $L$  can be found by applying the discussion in previous subsection by precalculating the entrances for all possible patterns of  $u$  and  $w$ , because the start of the base of  $L$  is uniquely determined as a parent of  $w$ , independent of  $v$ . However, we cannot apply it to find the top of  $L'$ , because the start of the base of  $L'$  is a child of  $t$ , not a parent. Instead, we calculate the top of  $L'$  by case-analysis of the positional relation of the staircases. Intuitively, we divide cases by the angle formed by  $B_L$  and  $B_{L'}$ . We have essentially two cases<sup>3</sup> to tract, which this angle is  $\pi/2$  (Figure 4) or  $\pi$  (Figure 5) (we formally define these cases and prove that they cover all cases in full version). In the case in Figure 4, the entrance  $e$  of  $L'$  can be found on  $B_L$ . In the case in Figure 5,  $e$  can be found on the total boundary of the vertex set with imprint  $t$ . In both case, by appropriate data structure given in full version, we can find the entrance in  $O(\log n)$  time.

### 4.3 Double Imprints

Here we consider the staircases decomposition for the case that there are two imprints  $w^1, w^2$  of  $u$  in  $T$  in  $I[u, v]$ . Let  $w$  be the lowest common ancestor of  $w^1$  and  $w^2$  in  $T$ . From (ii) of Lemma 7 and isometricity of  $T$ ,  $d(u, w^1) + d(w^1, w) = d(u, r) - d(w, r) = d(u, w^2) + d(w^2, w)$  holds and particularly we have  $w^1, w^2 \in I[u, w]$ . From isometricity of  $T$ , we have  $w \in I[w_1, w_2] \subseteq I[u, v]$ . Let  $t$  be the lowest common ancestor of  $w$  and  $v$ . Then, from isometricity of  $T$ , we have  $t \in I[w, v] \subseteq I[u, v]$ . Note that, the lowest common ancestor of  $v$  and  $w^1$  (resp.  $w^2$ ) is also  $t$ , because otherwise we have  $w \notin I[u, v]$ . Let  $P$  (resp.  $P'$ ) be any root-leaf path of  $T$  that contains  $w$  (resp.  $v$ ).

Since  $P'$  is convex, we can decompose  $I[u, v]$  into a staircases  $L'$  with base on  $P'$  and an interval  $I[u, t]$ . Since the subpath of  $P$  between  $r$  and  $w$  is convex, we can further decompose the interval  $I[u, t]$  into a staircases  $L$  with base on  $P$  and an interval  $I[u, w]$  (actually, we can prove that  $L$  is a line). Now, for fixed  $T$ ,  $I[u, w]$  is one of the  $O(n)$  candidates of the intervals, because  $w$  is specified only by a vertex  $u$ , as the lowest common ancestor of two imprints of  $u$  in  $T$ . This is the staircases decomposition we obtain here.

Let  $B_L$  (resp.  $B_{L'}$ ) be the base of  $L$  (resp.  $L'$ ). From the same reason as the case with a single imprint, we prove the following lemma. The proof is similar to the proof of Lemma 11.

<sup>3</sup> To explain all cases by these two, we take  $T$  as the maximal tree with gated branches that contains the fiber we consider, rather than the fiber itself.



► **Lemma 12.** *The following statements hold.*

- (i)  $I[u, v]$  contains no vertices in  $T$  other than vertices in  $w^1 - v$  and  $w^2 - v$  path on  $T$ .
- (ii) For a vertex  $z$  in  $L'$ , the gate of  $z$  in  $P'$  is an imprint of  $z$  in  $T$ .
- (iii) For a vertex  $z$  in  $L$ , the gate of  $z$  in  $P$  is an imprint of  $z$  in  $T$ .

**Proof.** (i) Let  $z \in I[u, v] \cap V(T)$ . Then,  $d(u, v) = d(u, z) + d(z, v)$  holds. Let  $w^i$  be the imprint of  $u$  in  $T$  with  $d(u, z) = d(u, w^i) + d(w^i, z)$ . Then,  $d(u, v) = d(u, w^i) + d(w^i, v)$  holds. Therefore  $d(w^i, v) = d(w^i, z) + d(z, v)$  and it means  $z$  is on the unique path between  $w^i$  and  $v$  on  $T$ . (ii) Let  $w'_z$  be the gate of  $z$  in  $P'$ . We prove  $I[z, w'_z] \cap V(T) = \{w'_z\}$ . Assume  $x \in (I[z, w'_z] \cap V(T)) \setminus \{w'_z\}$ . From (i),  $x$  is on  $t - w^1$  or  $t - w^2$  path. From isometricity of  $T$ ,  $t \in I[x, w'_z] \subseteq I[z, w'_z]$  holds and it contradicts the definition of  $w'_z$ . (iii) Similar to (ii). ◀

We should also provide a way to identify the top of the staircases  $L'$ . We have only one case to tract, shown in Figure 6, which we can find the entrance on  $w^1 - t$  or  $w^2 - t$  path on  $T$  (we formally define the case in full version). We can find it in  $O(\log n)$  time in the algorithm in full version.

## 5 Query Processing of the Case with One End on the Tree with Gated Branches

In this section, we construct an algorithm and a data structure that answers the queries with one of the endpoints on the tree with gated branches. That part is the core of our algorithm.

### 5.1 Query Processing for Maximal Staircases with Base on Convex Path

Here we construct an algorithm and a data structure for the staircases whose base is contained in a convex path  $P$ . For simplicity, we assume that  $P$  contains  $2^q$  vertices for some integer  $q$ . We do not lose generality by this restriction because we can safely attach dummy vertices at the end of  $P$ . Let  $P = (w_0, \dots, w_{2^q-1})$ . Our data structure uses a segment tree defined on  $P$ . The information of the vertices with base  $w_i$  in  $P$  are stored by linking to  $w_i$ .

It is convenient to consider the *direction* of  $P$ , as if  $P$  is directed from  $w_0$  to  $w_{2^q-1}$ . The *reverse*  $\bar{P}$  of  $P$  is the same path as  $P$  as an undirected path but has different direction, i.e.,  $\bar{P} = (w_{2^q-1}, \dots, w_0)$ . We represent the path between  $w_x$  and  $w_y$  on  $P$  by  $P[x, y]$ .

Let us formally define the queries to answer here. A query is represented by three vertices  $x, w_a, w_b$  such that the gate of  $x$  on  $P$  is  $w_a$ , and asks to answer the value  $p(L(x, w_a, w_b))$ , where  $L(x, w_a, w_b)$  represents the staircases with top  $x$  and base starts at  $w_a$  and ends at  $w_b$ . We construct two data structures, the first one treats the case  $a \leq b$  and the second one treats the case  $a > b$ . The second data structure is just obtained by building the first data structure on the reverse of  $P$ , therefore we can assume that for all queries,  $w_a \leq w_b$  holds.

For  $i = 0, \dots, 2^q - 1$ , let  $F_i$  be the fiber of  $w_i$  with respect to  $P$ . For  $i = 0, \dots, 2^q - 2$  and  $z \in F_i$ , the *successor*  $\text{succ}_P(z)$  of  $z$  is the gate of  $z$  in  $F_{i+1}$  (see Figure 7). Intuitively,  $\text{succ}_P(z)$  represents the next step of  $z$  in the staircases with base in  $P$ ; more precisely, for  $a < i < b$ , if  $F_i \cap V(L(x, w_a, w_b))$  induces  $z - w_i$  path,  $F_{i+1} \cap V(L(x, w_a, w_b))$  induces  $\text{succ}_P(z) - w_{i+1}$  path.

Here we construct a complete binary tree, which is referred to as *segment tree*, to answer the queries. For each  $d = 0, \dots, q$  and for each  $i = 0, 1, \dots, 2^{q-d} - 1$ , we prepare a node that corresponds to  $P[i \times 2^d, (i+1) \times 2^d - 1]$ . For each node  $v$  that corresponds to  $P[l, r]$  and for each  $z \in F_l$ , we store the vertex  $s(z, l, r) = \text{succ}_P^{r-l}(z)$  and the value  $S(z, l, r) = p(L(z, w_l, w_r)) = p(I[\text{succ}_P^0(z), w_l]) \oplus \dots \oplus p(I[\text{succ}_P^{r-l}(z), w_r])$ , where the  $\text{succ}_P^k(z)$  is recursively defined by  $\text{succ}_P^0(z) = z$  and  $\text{succ}_P^{k+1}(z) = \text{succ}_P(\text{succ}_P^k(z))$  for all  $0 \leq k$ .

## 18:10 Interval Query Problem on Cube-Free Median Graphs

The Algorithm 1 calculates  $p(L(x, w_a, w_b))$ . We call the procedure  $\text{StaircasesQuery}_P(0, 2^q - 1, a, b, x)$  to calculate it, and the algorithm returns the pair of the vertex  $\text{succ}_P^{b-a+1}(x)$  and the value  $p(L(x, w_a, w_b))$ . The time complexity is  $O(q) = O(\log n)$ .

■ **Algorithm 1**  $\text{StaircasesQuery}_P(l, r, a, b, x)$ .

---

```

1: if  $[l, r] \subseteq [a, b]$  then
2:   return  $(s(x, l, r), S(x, l, r))$ 
3:  $med \leftarrow \lfloor \frac{l+r}{2} \rfloor$ 
4: if  $b \leq med$  then
5:   return  $\text{StaircasesQuery}_P(l, med, a, b, x)$ 
6: if  $med < a$  then
7:   return  $\text{StaircasesQuery}_P(med + 1, r, a, b, x)$ 
8:  $(x', S_1) \leftarrow \text{StaircasesQuery}_P(l, med, a, b, x)$ 
9:  $(x'', S_2) \leftarrow \text{StaircasesQuery}_P(med + 1, r, a, b, \text{succ}_P(x'))$ 
10: return  $(x'', S_1 \oplus S_2)$ 

```

---

This data structure is constructed as in Algorithm 2. The correctness is clear and the time complexity is  $O(nq) \leq O(n \log n)$ , assuming that we know the vertex  $\text{succ}_P(x)$  and the value  $p(I[x, w_i])$  for all  $i = 0, \dots, 2^q - 1$  and  $x \in F_i$ . The size of the data structure is clearly  $O(nq) \leq O(n \log n)$ . We give algorithms to calculate  $\text{succ}_P(x)$  and  $p(I[x, w_i])$  in full version.

■ **Algorithm 2** Construction of the Data Structure for Staircases with Base on Convex Path.

**Input:** A cube-free median graph  $G$ , a convex path  $P = (w_0, \dots, w_{2^q-1})$

---

```

1: for  $i = 0, \dots, 2^q - 1$  do
2:   for all  $x \in F_i$  do
3:      $s(x, i, i) \leftarrow x$ 
4:      $S(x, i, i) \leftarrow p(L(x, w_i, w_i)) = p(I[x, w_i])$ 
5:   for  $d = q - 1, \dots, 0$  do
6:     for  $i = 0, \dots, 2^{q-d} - 1$  do
7:        $a \leftarrow i \times 2^d, b \leftarrow (i + \frac{1}{2}) \times 2^d, c \leftarrow (i + 1) \times 2^d$ 
8:       for all  $x \in F_i$  do
9:          $s(x, a, c - 1) \leftarrow s(\text{succ}_P(s(x, a, b - 1)), b, c - 1)$ 
10:         $S(x, a, c - 1) \leftarrow S(x, a, b - 1) \oplus S(\text{succ}_P(s(x, a, b - 1)), b, c - 1)$ 

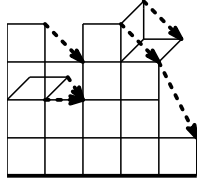
```

---

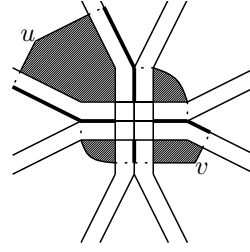
## 5.2 Query Processing for Staircases with Base on the Tree with Gated Branches

Let  $T$  be a tree with gated branches. Here we construct an algorithm and a data structure for the staircases whose base is a column of  $T$ . The simplest idea is to prepare the data structure discussed in the previous subsection for all root-leaf paths on  $T$ , but in this case the total size of the data structure can be as bad as  $O(n^2 \log n)$ . To reduce the size, we instead prepare the above data structure on every heavy-path of heavy-light decomposition of  $T$ .

For a vertex  $w \in V(T)$ , let  $F(w)$  be the set of vertices with an imprint  $w$ . For an edge  $(w, w')$  of  $T$  and a vertex  $z \in F(w)$ , we denote  $\text{succ}_{w, w'}(z)$  by the gate of  $z$  in  $F(w')$ . For the staircases  $L$  whose base starts at  $w_1$  and ends at  $w_2$  such that  $w_1, w, w', w_2$  are located on some column of  $T$  in this order, if  $F(w) \cap V(L)$  induces  $z - w$  path,  $F(w') \cap V(L)$  is  $\text{succ}_{w, w'}(z) - w'$  path.



■ **Figure 7** The arrows go from  $v$  to  $\text{succ}_P(v)$ . The bold line represents  $P$ .



■ **Figure 8** Decomposition of  $I[u, v]$ .

Let  $P$  be a heavy-path of  $T$ . Let  $V_P$  be the set of vertices that has an imprint in  $P$ . We build a data structure discussed in the previous subsection on the graph induced by  $V_P$  together with the convex path  $P$ ; Lemma 11 and Lemma 12 ensures that, for any staircases  $L$  we want to treat, all the vertices in  $L$  has an imprint in the base of  $L$ . We can calculate the answer for the queries by Algorithm 3, where the vertices in a heavy-path is represented as  $P = (w_{P,0}, \dots, w_{P,w^q_P})$ .

■ **Algorithm 3** StaircasesQuery( $u, w, v$ ).

**Input:**  $w, v \in V(T)$ ,  $u \in V(G)$  such that  $w$  and  $v$  are on the same column of  $T$  and  $u \in F(w)$

- 1: Let  $Q$  be the  $w - v$  path on  $T$  and  $P_1, \dots, P_k$  be the list of heavy-paths that contains vertices in  $Q$ , in the same order appearing in  $Q$
- 2: Let  $P_1 \cap Q = (w = w_{P_1, s_1}, \dots, w_{P_1, t_1})$ ,  $P_2 \cap Q = (w_{P_2, s_2}, \dots, w_{P_2, t_2})$ ,  $\dots$ ,  $P_k \cap Q = (w_{P_k, s_k}, \dots, w_{P_k, t_k} = v)$
- 3:  $(x, S) \leftarrow \text{StaircasesQuery}_{P_1}(0, 2^{q_{P_1}} - 1, s_1, t_1, u)$
- 4: **for**  $i = 2, \dots, k$  **do**
- 5:    $(x', S') \leftarrow \text{StaircasesQuery}_{P_i}(0, 2^{q_{P_i}} - 1, s_i, t_i, \text{succ}_{w_{P_{i-1}, t_{i-1}}, w_{P_i, s_i}}(x))$
- 6:    $x \leftarrow x', S \leftarrow S \oplus S'$
- 7: **return**  $(x, S)$

The correctness of the algorithm is clear. The size of the data structure is bounded by  $O(n \log n)$ , because the size of the data structure on a heavy-path  $P$  is bounded by  $O(|V_P| \log |V_P|)$  and each vertex is in  $V_P$  for at most two heavy-paths  $P$ . We should make an algorithm to calculate the successor efficiently. We describe an algorithm that works in  $O(\log n)$  time in full version.

Now, the time complexity of Algorithm 3 is  $O(\log^2 n)$  because  $k$  in the algorithm is at most  $O(\log n)$ .

### 5.3 Putting them Together

Here we summarize our work on the interval query problem with one end on the tree with gated branches. In Section 4, for the fixed tree  $T$  with gated branches, we have seen that any interval with one end on  $T$  can be decomposed to at most two staircases (say,  $L$  and  $L'$ , for instance we allow any of them to be empty) and a special interval  $I$  that is one of  $O(n)$  candidates. As we roughly described in Section 4, such decomposition can be calculated in  $O(\log n)$  time (See full version for details).

## 18:12 Interval Query Problem on Cube-Free Median Graphs

Now we consider calculating the answer as  $p(L) + p(L') + p(I)$ .  $p(L)$  and  $p(L')$  can be calculated in  $O(\log^2 n)$  time by above algorithm. Furthermore,  $p(I)$  is precalculated in the construction of our data structure and we can take this value in constant time. Therefore we can answer the interval query in the case with one end on the tree with gated branches in  $O(\log^2 n)$  time. We summarize our algorithm in full version.

Here we describe how  $p(I)$  can be precalculated. Recall that, we construct our data structure recursively on each fibers. Therefore, after constructing the smaller data structure on each fiber, we can calculate the value  $p(I)$  in  $O(\log^2 n)$  time by using an interval query on them to complete construction. This is the bottleneck part of our construction algorithm, along with  $O(\log n)$  recursion steps. Note that, this procedure can be implemented during preprocessing because there are only  $O(n)$  candidates of  $I$ . When answering to the queries, we do not need to use the smaller data structure; we have only to refer these precalculated values.

### 6 Decomposing Intervals into intervals with One End on the Boundary

In this section, we consider decomposing an interval with both ends in different fibers into smaller intervals with one end on boundaries (see Figure 8). Specifically, we bound the number of such fibers by 9. Let  $m$  be the median of  $G$ . For  $x \in \text{St}(m)$ , let  $F(x)$  be the fiber of  $x$  with respect to  $\text{St}(m)$ . For  $v \in V(G)$ , let  $r(v)$  be the vertex in  $\text{St}(m)$  that is nearest from  $v$ . From definition of fibers,  $v \in F(r(v))$  holds.

First, we prove that the intersection of an interval and a fiber is indeed an interval. The following lemma holds.

► **Lemma 13.** *Let  $u, v$  be vertices and let  $x \in \text{St}(m)$ . Let  $g_u, g_v$  be the gate of  $u, v$  in  $F(x)$ , respectively. Then,  $I[u, v] \cap F(x)$  coincides with  $I[g_u, g_v]$  if it is nonempty.*

**Proof.** Assume  $z \in I[u, v] \cap F(x)$ . From the definition of the gate, there is a  $u - z$  (resp.  $v - z$ ) shortest path that passes through  $g_u$  (resp.  $g_v$ ). Therefore there is a  $u - v$  shortest path that passes through  $u, g_u, z, g_v, v$  in this order, which means  $z \in I[g_u, g_v]$ . Converse direction is clear from  $I[g_u, g_v] \subseteq I[u, v]$ , which is from the definition of the gate. ◀

Note that, unless  $r(u) = r(v)$ , one of the gates of  $u$  or  $v$  in  $F(x)$  is on the total boundary of  $F(x)$ . Therefore, to obtain the desired structural result, we just need to bound the number of fibers with non-empty intersection with  $I[u, v]$ . We use the following lemma from [13].

► **Lemma 14 ([13]).** *Let  $u, v$  be vertices with  $r(u) \neq r(v)$ . Then, one of the  $m \in I[u, v]$ ,  $r(u) \sim r(v)$ , or  $d(m, r(u)) = d(m, r(v)) = d(r(u), r(v)) = 2$  holds.*

Assume  $m \in I[u, v]$ . Then,  $I[u, v] \cap F(x) \neq \emptyset$  means  $x \in I[u, v]$ . Therefore the number of such fibers  $F(x)$  is same as the number of vertices in  $I[u, v] \cap \text{St}(m)$ . Now, from the fact that  $I[u, v]$  has a grid structure (see Lemma 2) and  $\text{St}(m)$  consists of the vertices in an edge or a square that contains  $m$ , we have that  $|I[u, v] \cap \text{St}(m)| \leq 9$ .

If  $r(u) \sim r(v)$ , from Lemma 6, we have  $I[u, v] \subseteq F(r(u)) \cup F(r(v))$ . If  $d(r(u), r(v)) = 2$ , let  $w$  be the unique common neighbor of  $r(u)$  and  $r(v)$ . Then, from Lemma 6, we have  $I[u, v] \subseteq F(r(u)) \cup F(w) \cup F(r(v))$ . Therefore, in all cases, the number of fibers with nonempty intersection with  $I[u, v]$  is bounded by 9.

In all of these cases, we can list the fibers  $F(x)$  with nonempty intersection with the given interval  $I[u, v]$ ; it is the set of the fibers of the vertices in  $I[r(u), r(v)]$  because  $\bigcup_{x \in I[r(u), r(v)]} F(x)$  is convex, and, we can list them efficiently by using the list of all squares in  $G$ . Now it is sufficient to give a way to calculate the gate of  $u$  and  $v$  in each of these fibers for our algorithm. We give the algorithm in full version.

Above technique can also be applied for the following query. We are given three vertices  $v_1, v_2, v_3$  in a cube-free median graph  $G$  and asked to answer the median  $v$  of these three vertices. Let  $x$  be the median of  $r(v_1)$ ,  $r(v_2)$  and  $r(v_3)$ .  $x$  can be calculated in  $O(\log n)$  time because each of  $I[r(v_1), r(v_2)]$ ,  $I[r(v_2), r(v_3)]$  and  $I[r(v_3), r(v_1)]$  contains at most 9 vertices and  $x$  is the unique vertex in the intersection of these intervals. Now, we can state that  $v \in F(x)$ , because  $F(x)$  is the only fiber that can intersect all of  $I[v_1, v_2]$ ,  $I[v_2, v_3]$  and  $I[v_3, v_1]$ .

Let  $g_{v_1}$  (resp.  $g_{v_2}, g_{v_3}$ ) be the gate of  $v_1$  (resp.  $v_2, v_3$ ) in  $F(x)$ , which can be calculated in  $O(\log n)$  time. Then, from Lemma 13,  $v$  coincides with the median of  $g_{v_1}$ ,  $g_{v_2}$  and  $g_{v_3}$ . Therefore we can reduce the median query on the original graph into the median query on the fiber  $F(x)$  in  $O(\log n)$  time. By recursively working on the fiber, we can calculate  $v$  after  $O(\log n)$  recursion steps. Therefore the query can be answered in  $O(\log^2 n)$  time in total. The data structure required here is constructed in  $O(n \log^2 n)$  time just by taking the necessary parts of the algorithm in full version.

---

## References

- 1 Stephen Alstrup, Cyril Gavoille, Haim Kaplan, and Theis Rauhe. Nearest common ancestors: a survey and a new distributed algorithm. In *Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 258–264, 2002.
- 2 S P Avann. Metric ternary distributive semi-lattices. *Proceedings of the American Mathematical Society*, 12(3):407–414, 1961.
- 3 Hans-Jurgen Bandelt and Victor Chepoi. Metric graph theory and geometry: a survey. *Contemporary Mathematics*, 453:49–86, 2008.
- 4 Michael A Bender and Martin Farach-Colton. The LCA problem revisited. In *Latin American Symposium on Theoretical Informatics*, pages 88–94, 2000.
- 5 Michael A Bender, Martín Farach-Colton, Giridhar Pemmasani, Steven Skiena, and Pavel Sumazin. Lowest common ancestors in trees and directed acyclic graphs. *Journal of Algorithms*, 57(2):75–94, 2005.
- 6 Laurine Bénêteau, Jérémie Chalopin, Victor Chepoi, and Yann Vaxès. Medians in median graphs and their cube complexes in linear time. In *Proceedings of the Forty-Seventh International Colloquium on Automata, Languages, and Programming*, page to appear, 2020.
- 7 Garrett Birkhoff and Stephen A Kiss. A ternary operation in distributive lattices. *Bulletin of the American Mathematical Society*, 53(8):749–752, 1947.
- 8 Gerth Stølting Brodal, Pooya Davoodi, and S Srinivasa Rao. Path minima queries in dynamic weighted trees. In *Workshop on Algorithms and Data Structures*, pages 290–301. Springer, 2011.
- 9 Bernard Chazelle. Computing on a free tree via complexity-preserving mappings. *Algorithmica*, 2(1-4):337–361, 1987.
- 10 Bernard Chazelle and Leonidas J Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(1-4):133–162, 1986.
- 11 Bernard Chazelle and Burton Rosenberg. Computing partial sums in multidimensional arrays. In *Proceedings of the fifth annual symposium on Computational geometry*, pages 131–139, 1989.
- 12 Victor Chepoi. Classification of graphs by means of metric triangles. *Metody Diskret. Analiz*, 49:75–93, 1989.
- 13 Victor Chepoi, Arnaud Labourel, and Sébastien Ratel. Distance labeling schemes for cube-free median graphs. In *44th International Symposium on Mathematical Foundations of Computer Science*, pages 15:1–15:14, 2019.
- 14 Victor Chepoi and Daniela Maftuleac. Shortest path problem in rectangular complexes of global nonpositive curvature. *Computational Geometry*, 46(1):51–64, 2013.

- 15 Adam Clearwater, Clemens Puppe, and Arkadii Slinko. Generalizing the single-crossing property on lines and trees to intermediate preferences on median graphs. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
- 16 Nadia Creignou and J-J Hébrard. On generating all solutions of generalized satisfiability problems. *RAIRO-Theoretical Informatics and Applications*, 31(6):499–511, 1997.
- 17 Gabrielle Demange. Majority relation and median representative ordering. *SERIEs: Journal of the Spanish Economic Association*, 3(1):95–109, 2012.
- 18 Harold N Gabow, Jon Louis Bentley, and Robert E Tarjan. Scaling and related techniques for geometry problems. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 135–143, 1984.
- 19 Dan Gusfield. Algorithms on stings, trees, and sequences: Computer science and computational biology. *Acm Sigact News*, 28(4):41–60, 1997.
- 20 Dov Harel and Robert E Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, 1984.
- 21 George S Lueker. A data structure for orthogonal range queries. In *Proceedings of the nineteenth Annual Symposium on Foundations of Computer Science*, pages 28–34, 1978.
- 22 Henry Martyn Mulder and Alexander Schrijver. Median graphs and helly hypergraphs. *Discrete Mathematics*, 25(1):41–50, 1979.
- 23 Ladislav Nebeský. Median graphs. *Commentationes Mathematicae Universitatis Carolinae*, 12(2):317–325, 1971.
- 24 David Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33(3):167–176, 2000.
- 25 Clemens Puppe and Arkadii Slinko. Condorcet domains, median graphs and the single-crossing property. *Economic Theory*, 67(1):285–318, 2019.
- 26 Thomas J Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.
- 27 Andrew C Yao. Space-time tradeoff for answering range queries. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 128–136, 1982.
- 28 Andrew C Yao. On the complexity of maintaining partial sums. *SIAM Journal on Computing*, 14(2):277–288, 1985.
- 29 Hao Yuan and Mikhail J Atallah. Data structures for range minimum queries in multidimensional arrays. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 150–160, 2010.