Interval Query Problem on Cube-free Median Graphs

Soh Kumabe[1,a)]

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 stairs decomposition, to decompose an interval of cube-free median graphs into simpler substructures.

Keywords: Median Graphs, Data Structures, Range Query Problems

1. Introduction

The range query problem [1] is one of the most fundamental problems in the literature on data structures, particularly for string algorithms [2]. 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 [3]. 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 [3], [4] is that for any constant integer k, a range query can be answered in $O(\alpha_k(n))$ time, where α_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], [5], [6], [7], [8].

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 [*1] 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 dimensional grids, a constant-time algorithm for range minimum query is known [9]. For range query problem on trees, an almost-constant time algorithm [10] with linear space is known on semigroup operators; see [11] for further survey on the problem on tree, especially 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 [12], [13], [14]. 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 $p(I[u, v])$ [*2]. 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

[1] The University of Tokyo, Bunkyo, Tokyo 113–8654, Japan
[a)] soh_kumabe@mist.i.u-tokyo.ac.jp

[*1] 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 $a, a' \in \mathcal{S}$.

[*2] For a vertex subset X, we denote the sum of $p(z)$ over all $z \in X$ by $p(X)$.

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 [15] in the orthogonal range query problem, without acceleration via fractional cascading [16].

To obtain the algorithm, we introduce a new technique, named the stairs 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 stairs, are well-structured, and an efficient algorithm to answer the interval queries can be constructed. The rest are not necessarily stairs; 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 [17] 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 [18]. Very recently, a linear-time algorithm to find the median of median graphs was built [19]. 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 [20], [21], [22]. 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. 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. In social choice theory, the structure of median graphs naturally arises as a generalization of single-crossing preferences [23], [24] 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.

The rest of this paper is organized as follows. In Section 2, we summarize the basic properties of median graphs and trees. In Section 4, we introduce the stairs decomposition for special intervals. In Section 5, we explain our algorithm for special intervals. We generalize the results to general intervals in Section 6, while some detailed parts of our algorithm is omitted. We omit the construction algorithm of required data structure. Note that, the construction algorithm also accelerate the time complexity of construction algorithm of the distance labeling scheme on cube-free median graphs [18] from $O(n^2 \log n)$ to $O(n \log^2 n)$, together with the algorithm of the linear-time median calculation in [19]. A more detailed outline is provided in Section 3.

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 is 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 cube as an induced subgraph. The followings hold.

Lemma 1 ([26]). Any interval in a cube-free median graph induces an induced subgraph of two-dimensional grid.

Lemma 2 ([18]). 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 exists) because it is the unique vertex in X that minimizes the distance from z. X is gated if all vertices $z \in V(G)$ has a gate in X. The following equivalence result is known.

Lemma 3 ([18], [27]). 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 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 subset is convex. Any interval is convex.

For gated 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 $x \sim y$ [18]. The fibers for all $x \in X$ defines 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 has 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 unique neighbor in $T_X(y, x)$, which is the corresponding vertex under that isomorphism. For vertex $x \in X$, the 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 paths are gated. The next lemma exploits the structures of the boundaries of fibers.

Lemma 4. Let X be a gated vertex subset of cube-free median graph G. Let $x, y \in X$ and assume $x \sim y$. Then, the followings hold.
(i) ([18]) $T_X(x, y)$ induces a tree, which is convex.
(ii) ([18]) $T_X(x)$ induces the tree with gated branches, which is isometric to G.

The following is folklore in the literature of median graphs.

Lemma 5 (folklore). Let X be the convex vertex set of a median graph and let Y be the convex subset of X. For $x \in X$, let $F(x)$ be the fiber of x with respect to X. Then, $\bigcup_{y \in Y} F(x)$ is convex.

Let T be the 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 gated, the imprint is equal to the gate and therefore unique. Even if it is not the case, we can state following.

Lemma 6. Let T be the tree with gated branches. Let $u \in V(G)$. Then, the following statements hold.
(i) ([18]) There are at most two imprints of u in T.
(ii) Assume u has two distinct imprints w^1, w^2 in T. Then, $w^1, w^2 \in I[r, u]$.

Lemma 7. Let T be the tree with gated branches and $w \in V(T)$. Then, the set of vertices with an imprint w in T is gated.

For a vertex $m \in V(G)$, the star $\mathrm{St}(m)$ 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. $\mathrm{St}(m)$ is gated. The vertex $m \in V(G)$ is median of G if it minimizes the sum of distances to all vertices in G. The following holds.

Lemma 8 ([18]). All the fibers of $\mathrm{St}(m)$ 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 is a column of T if for any two vertices x, y in T, x is either an ancestor or a descendant of y. The vertex t is the lowest common ancestor [8] 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 [7]. 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 a subtree of T rooted at v. An edge (u, v) in G such that u is the parent of v is heavy-edge if $|V(T_u)| \leq 2|V(T_v)|$ and light-edge otherwise. Each vertex has at most one child such that the edge between them is a heavy-edge. The heavy-path is the 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

In this section we roughly describe our algorithm. Let m be a median of G and for $x \in \mathrm{St}(m)$, let $F(x)$ be the fiber of x in $\mathrm{St}(m)$. Let u, v be vertices of cube-free median graph G. Consider calculating $p(I[u, v])$. If u and v are in the same fiber $F(x)$ of $\mathrm{St}(m)$, we calculate the answer by using the algorithm on $F(x)$, which is recursively defined. From Lemma 8, we have at most $O(\log n)$ recursion steps. Otherwise, we can show that $I[u, v]$ intersects with only 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 with one end of the interval is on the total boundary of the fiber $F(x)$.

To do this, we introduce a technique to decompose intervals, named the stairs decomposition. Let T be the tree with gated branches and assume $u \in V(G)$ and $v \in V(T)$. We decompose an interval $I[u, v]$ into a disjoint union of an interval I and at most two special structures, named stairs, which we describe later. 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.

Let $P = (s = w_0, \ldots, w_k = t)$ be a column of T. For a vertex x with gate s in P, the interval $I[x, t]$ induces stairs if for all $i = 0, \ldots, k$, the set of vertices in $I[x, t]$ with gate w_i in P induces a path. $P_{s,t}$ is the base of L and the vertex x is the top of L. The base starts at s and ends at t.

Now we just need to construct an algorithm such that, given a base and a top of the stairs L, it calculate the value $p(V(L))$ quickly. Let P be a root-leaf path of T. We can use segment trees to answer the stairs 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 whole paper is organized as follows. In Section 4, we introduce the stairs 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 $\mathrm{St}(m)$. Due to the space constraint, we omit some detailed parts in these sections and algorithm to construct our data structure efficiently.

4. The Stairs Decomposition of the Intervals with One End on the Boundary

In this section, we introduce the stairs decomposition of the interval with one end on the boundary of a fiber.

4.1 The case with One End on a Convex Path

Let P be a convex path. In this section, we investigate the structure of an interval such that one of the endpoints is on 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 stairs (see Figure (a)), if $w \neq v$. We assume $w \neq v$ because otherwise we have no need of decomposition. Let w' be the neighbor of w in P between w and v. We take the embedding of $I[u, v]$ into a two-dimensional grid (see Lemma 1). We naively introduce a xy-coordinate system with $w = (0, 0)$, $w' = (0, 1)$ and $u = (x_u, y_u)$ with $y_u \geq 0$. Now, we can state following.

Lemma 9. If a vertex z on $I[u, v] \cap V(P)$ is not on x-axis, there is no vertex other than z in $I[u, v]$ with gate z in P.

Since such z does not affect the possibility of decomposition (we can just add such vertices at the end of the stairs), we can assume that $v = (0, y_v)$ for $y_v > 0$. Especially, from convexity, we have that all vertices in $I[u, v]$ has non-negative y-coordinate. Now, we have that $I[u, v] \setminus I[u, w]$ is the set of vertices with positive x-coordinate and forms stairs (see figure (a)), 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 stairs. Instead of direct identification, we rather identify the unique neighbor of it in $I[u, w]$, named the entrance e of the stairs: 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 of 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. Due to the space constraint, we omit this algorithm.

4.2 Single Imprint

Let T be a tree with gated branches, rooted at r. Here we give the stairs decomposition of the interval $I[u, v]$, where v is on T. 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. 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 stairs 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 stairs L with base on P and an interval $I[u, w]$. Since u has at most two imprints in T, $I[u, w]$ is one of the $O(n)$ candidates of the intervals. This is the stairs decomposition we obtain here.

To bound the size of the data structure we construct in Section 5, we should ensure that stairs 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 10. 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.

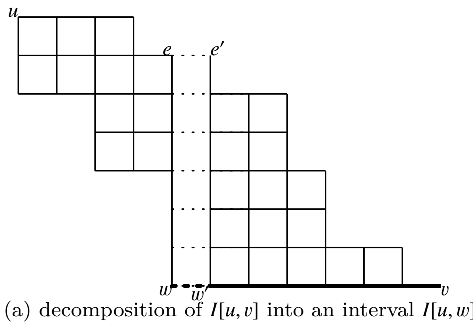
We should also make algorithms to identify the top of the stairs L and L'. The top of L can be found by applying the discussion in previous subsection by precalculating the entrances for all possible patterns of u and w, because the start of the base of L is uniquely determined as a parent of w, independent to 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 stairs. Intuitively, we divide cases by the angle formed by B_L and $B_{L'}$. We have essentially two cases[*3] to tract, which this angle is $\pi/2$ (Figure (b)) or π (Figure (c)) (we omit the formal definition of these cases and proof of the fact that they cover all cases). In the case in Figure (b), the entrance e of L' can be found on B_L. In the case in Figure (c), e can be found on the total boundary of the vertex set with imprint t. In both case, we can find the entrance in $O(\log n)$ time. Due to the space constraint, we omit this algorithm.

4.3 Double Imprints

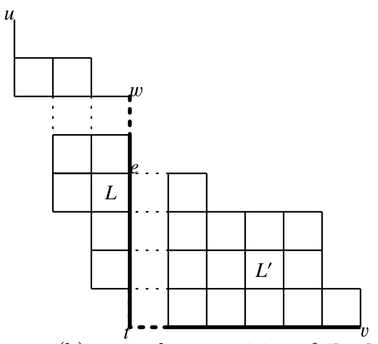
Here we treat the case that there are two imprints w^1, w^2 of u in T in $I[u, v]$. Let w be the lowest common ancestor of w^1 and w^2 in T. From (ii) of Lemma 6 and isometricity of T, $d(u, w^1) + d(w^1, w) = d(u, r) - d(w, r) = d(u, w^2) + d(w^2, w)$ holds and especially we have $w^1, w^2 \in I[u, w]$. Let t be the lowest common ancestor of w and v. From $d(u, w^1) + d(w^1, v) = d(u, w^2) + d(w^2, v)$ and isometricity of T, t is an ancestor of w. 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 stairs

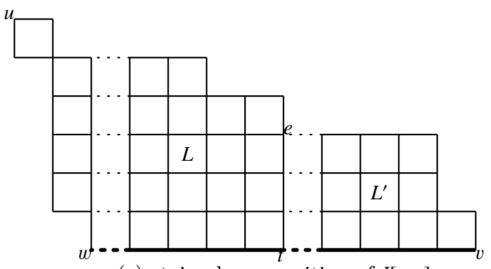
[*3] To explain all cases by these two, we take T as the maximal tree with gated branches that contains the fiber we consider, rather than the fiber itself.



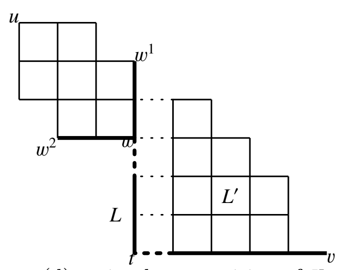
(a) decomposition of $I[u, v]$ into an interval $I[u, w]$ and stairs $I[e', v]$.
The bold line represents P.



(b) stairs decomposition of $I[u, v]$.
(single imprint, first case)



(c) stairs decomposition of $I[u, v]$.
(single imprint, second case)



(d) stairs decomposition of $I[u, v]$.
(double imprints)

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 stairs L with base on P and an interval $I[u, w]$ (actually, we can prove that L is a line). Now, $I[u, w]$ is one of the $O(n)$ candidates of the intervals because w is the lowest common ancestor of two imprints of u in T. This is the stairs 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.

Lemma 11. 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.

We should also make a way to identify the top of the stairs L'. We have only one case to tract, shown in Figure (d), which we can find the entrance on $w^1 - t$ or $w^2 - t$ path on T (we formally define the case in Section ??). We can find it in $O(\log n)$ time. Due to the space constraint, we omit this algorithm.

5. Query Procession 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 Procession for Maximal Stairs with Base on Gated Path

Here we construct an algorithm and a data structure for the stairs whose base is contained in the gated 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 last of P. Let $P = (w_0, \ldots, w_{2^q-1})$. 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., $\bar{P} = (w_{2^q-1}, \ldots, w_0)$. 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 stairs 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 $a \le b$ and the second one treat 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, $w_a \le w_b$ holds.

For $i = 0, \ldots, 2^q - 1$, let F_i be the fiber of w_i with respect to P. For $i = 0, \ldots, 2^q-2$ and $z \in F_i$, the successor $\mathrm{succ}_P(z)$ of z is the gate of z in F_{i+1}. It is important that, 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 $\mathrm{succ}_P(z) - w_{i+1}$ 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 $s(z, l, r) = \mathrm{succ}_P^{r-l}(z)$ and the value $S(z, l, r) = p(L(z, w_l, w_r)) = p(I[\mathrm{succ}_P^0(z), w_l]) \oplus \dots \oplus p(I[\mathrm{succ}_P^{r-l}(z), w_r])$, where the $\mathrm{succ}_P^k(z)$ is recursively defined by $\mathrm{succ}_P^0(z) = z$ and $\mathrm{succ}_P^{k+1}(z) = \mathrm{succ}_P(\mathrm{succ}_P^k(z))$ for all $0 \le k$.

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

Algorithm 1 $\mathrm{StairsQuery}_P(l, r, a, b, x)$

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

This data structure is constructed as Algorithm 2. The correctness is clear and the time complexity is $O(nq) \le O(n \log n)$, assuming that we know the vertex $\mathrm{succ}_P(x)$ 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 $O(nq) \le O(n \log n)$. Due to the space constraint, we omit algorithms to calculate such $\mathrm{succ}_P(x)$ and $p(I[x, w_i])$.

5.2 Query Procession for Stairs with Base on the Tree with Gated Branches

Let T be the tree with gated branches. Here we construct an algorithm and a data structure for the stairs 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 worse 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 $\mathrm{succ}_{w,w'}(z)$ by the gate of z in $F(w')$. For the stairs 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 $\mathrm{succ}_{w,w'}(z) - w'$ path.

Algorithm 2 Construction of the Data Structure for Stairs with Base on Gated Path

```
Input: A cube-free median graph $G$, a gated path $P = (w_0, \dots, w_{2^q-1})$
  for $i = 0, \dots, 2^q - 1$ do
    for all $x \in F_i$ do
      $s(x, i, i) \leftarrow x$
      $S(x, i, i) \leftarrow p(L(x, w_i, w_i)) = p(I[x, w_i])$
    end for
  end for
  for $d = q - 1, \dots, 0$ do
    for $i = 0, \dots, 2^{q-d} - 1$ do
      $a \leftarrow i \times 2^d, b \leftarrow (i + \frac{1}{2}) \times 2^d, c \leftarrow (i + 1) \times 2^d$
      for all $x \in F_i$ do
        $s(x, a, c - 1) \leftarrow s(\mathrm{succ}_P(s, a, b - 1), b, c - 1)$
        $S(x, a, c - 1) \leftarrow S(x, a, b - 1) \oplus S(\mathrm{succ}_P(s(x, a, b - 1)), b, c - 1)$
      end for
    end for
  end for
```

Algorithm 3 $\mathrm{StairsQuery}(u, w, v)$

```
Input: $w, v \in V(T)$, $u \in V(G)$ such that $w$ and $v$ are on the same
  column of $T$ and $u \in F(w)$
  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
  $Q$
  Let $P_1 \cap Q = (w = w_{P_1,s_1}, \dots, w_{P_1,t_1}), P_2 \cap Q = (w_{P_2,s_2}, \dots, w_{P_2,t_2}), \dots, P_k \cap$
  $Q = (w_{P_k,s_k}, \dots, w_{P_k,t_k} = v)$
  $(x, S) \leftarrow \mathrm{StairsQuery}_{P_1}(0, 2^{q_{P_1}} - 1, s_1, t_1, u)$
  for $i = 2, \dots, k$ do
    $(x', S') \leftarrow \mathrm{StairsQuery}_{P_1}(0, 2^{q_{P_i}} - 1, s_i, t_i, \mathrm{succ}_{w_{P_{i-1},t_{i-1}}, w_{P_i,s_i}}(x))$
    $x \leftarrow x', S \leftarrow S \oplus S'$
  end for
  return $(x, S)$
```

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 gated path P; Lemma 10 and Lemma 11 ensures that, for any stairs 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 $P = (w_{P,0}, \ldots, w_{P,w^{q_P}})$.

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 can construct an $O(\log n)$-time algorithm. Due to the space constraint, we omit this algorithm.

Now, the time complexity of Algorithm 3 is $O(\log^2 n)$ because k in the algorithm is at most $O(\log n)$. We summarize our work to answer the interval queries in Algorithm 4, which works in $O(\log^2 n)$ time.

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

One of the remaining problems is decomposing an interval with both ends in different fibers into smaller intervals with one end on boundaries. Let m be the median of G. For $x \in \mathrm{St}(m)$, let $F(x)$ be the fiber of x with respect to $\mathrm{St}(m)$. For $v \in V(G)$, let $r(v)$ be the vertex in $\mathrm{St}(m)$ 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 12. Let u, v be vertices and let $x \in \mathrm{St}(m)$. 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.

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 following lemma in [18].

Lemma 13 ([18]). 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 $I[u, v] \cap \mathrm{St}(m)$. Now, from the fact that $I[u, v]$ has a grid structure (see Lemma 1) and $\mathrm{St}(m)$ consists of the vertices in an edge or a square that contains m, we have that $|I[u, v] \cap \mathrm{St}(m)| \leq 9$.

If $r(u) \sim r(v)$, from Lemma 5, 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 5, 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 any 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. Due to the space constraint, we omit this algorithm.

References

[1] 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, pp. 135–143, 1984.

[2] Dan Gusfield. Algorithms on stings, trees, and sequences: Computer science and computational biology. Acm Sigact News, Vol. 28, No. 4, pp. 41–60, 1997.

[3] Andrew C Yao. Space-time tradeoff for answering range queries. In Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, pp. 128–136, 1982.

[4] Andrew C Yao. On the complexity of maintaining partial sums. SIAM Journal on Computing, Vol. 14, No. 2, pp. 277–288, 1985.

[5] 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, pp. 258–264, 2002.

[6] Michael A Bender and Martin Farach-Colton. The LCA problem revisited. In Latin American Symposium on Theoretical Informatics, pp. 88–94, 2000.

[7] 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, Vol. 57, No. 2, pp. 75–94, 2005.

[8] Dov Harel and Robert E Tarjan. Fast algorithms for finding nearest common ancestors. SIAM Journal on Computing, Vol. 13, No. 2, pp. 338–355, 1984.

[9] 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, pp. 150–160, 2010.

[10] Bernard Chazelle. Computing on a free tree via complexity-preserving mappings. Algorithmica, Vol. 2, No. 1-4, pp. 337–361, 1987.

[11] Gerth Stølting Brodal, Pooya Davoodi, and S Srinivasa Rao. Path minima queries in dynamic weighted trees. In Workshop on Algorithms and Data Structures, pp. 290–301. Springer, 2011.

[12] SP Avann. Metric ternary distributive semi-lattices. Proceedings of the American Mathematical Society, Vol. 12, No. 3, pp. 407–414, 1961.

[13] Garrett Birkhoff and Stephen A Kiss. A ternary operation in distributive lattices. Bulletin of the American Mathematical Society, Vol. 53, No. 8, pp. 749–752, 1947.

[14] Ladislav Nebeskỳ. Median graphs. Commentationes Mathematicae Universitatis Carolinae, Vol. 12, No. 2, pp. 317–325, 1971.

[15] George S Lueker. A data structure for orthogonal range queries. In Proceedings of the nineteenth Annual Symposium on Foundations of Computer Science, pp. 28–34, 1978.

[16] Bernard Chazelle and Leonidas J Guibas. Fractional cascading: I. A data structuring technique. Algorithmica, Vol. 1, No. 1-4, pp. 133–162, 1986.

[17] David Peleg. Proximity-preserving labeling schemes. Journal of Graph Theory, Vol. 33, No. 3, pp. 167–176, 2000.

[18] 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, pp. 15:1–15:14, 2019.

[19] Laurine Bénéteau, 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, p. to appear, 2020.

[20] Hans-Jurgen Bandelt and Victor Chepoi. Metric graph theory and geometry: a survey. Contemporary Mathematics, Vol. 453, pp. 49–86, 2008.

Algorithm 4 The Algorithm to Interval Query with One End on T

```
Input: $u, v \in V(G)$
if $d(u,v) = d(u,w) + d(w,v)$ holds for exactly one imprint $w$ of $u$ then
    Let $t$ be the lowest common ancestor of $w$ and $v$
    Let $P = (r = w_0, \ldots, w_k)$ and $P' = (r = w'_0, \ldots, w'_k)$, where $w_a = w'_a = t$ and $w_b = w$ be the root-leaf path that contains $w$ and $v$, respectively
    if $w \neq t$ then
        (Consider the decomposition of $I[u,v]$ into $I[u,w]$, $L = L(s, w_{b-1}, t)$ and $L' = L(s', w'_{a+1}, v)$)
        Find the entrance $e$ of $L$
        $(z, S) \leftarrow \mathsf{StairsQuery}(s, w_{b-1}, t)$, where $s$ is the neighbor of $e$ in $F(w_{b-1})$
        if $v = t$ then
            return $p(I[u,w]) \oplus S$
        end if
        if $w_{a+1}$ has a neighbor in $F(w'_{a+1})$ then
            Find the entrance $e'$ of $L'$ using appropriate data structure (omitted)
        else
            Find the entrance $e'$ of $L'$ using using appropriate data structure (omitted)
        end if
        $(z', S') \leftarrow \mathsf{StairsQuery}(s', w'_{a+1}, v)$, where $s'$ is the neighbor of $e'$ in $F(w'_{a+1})$
        return $p(I[u,w]) \oplus S \oplus S'$
    else
        if $v = t$ then
            return $p(I[u,w])$
        end if
        (Consider the decomposition of $I[u,v]$ into $I[u,w]$ and $L' = L(s', w'_{a+1}, v)$)
        Find the entrance $e'$ of $L'$
        $(z', S') \leftarrow \mathsf{StairsQuery}(s', w'_{a+1}, v)$, where $s'$ is the neighbor of $e'$ in $F(w'_{a+1})$
        return $p(I[u,w]) \oplus S'$
    end if
else
    Let $w^1, w^2$ be the imprints of $u$
    Let $w$ be the lowest common ancestor of $w^1$ and $w^2$
    Let $t$ be the lowest common ancestor of $w$ and $v$
    Let $P_1 = (r = w_{1,0}, \ldots, w_{1,k_1})$, $P_2 = (r = w_{2,0}, \ldots, w_{2,k_2})$ and $P' = (r = w'_0, \ldots, w'_{k'})$, where $w_{1,a} = w_{2,a} = w'_a = t$, $w_{1,b} = w_{2,b} = w$ be the root-leaf path that contains $w^1$, $w^2$ and $v$, respectively
    (Consider the decomposition of $I[u,v]$ into $I[u,w]$, the path $(w_{1,a}, \ldots, w_{1,b-1})$ and $L' = L(s', w'_{a+1}, v)$)
    Find the entrance $e'$ of $L'$ using using appropriate data structure (omitted)
    $(z', S') \leftarrow \mathsf{StairsQuery}(s', w'_{a+1}, v)$, where $s'$ is the neighbor of $e'$ in $F(w'_{a+1})$
    if $w = t$ then
        return $p(I[u,w]) \oplus S'$
    end if
    $(z, S) \leftarrow \mathsf{StairsQuery}(w_{1,b-1}, w_{1,b-1}, w_{1,a})$
    return $p(I[u,w]) \oplus S \oplus S'$
end if
```

[21] Henry Martyn Mulder and Alexander Schrijver. Median graphs and helly hypergraphs. Discrete Mathematics, Vol. 25, No. 1, pp. 41–50, 1979.

[22] Thomas J Schaefer. The complexity of satisfiability problems. In Proceedings of the Tenth Annual ACM Symposium on Theory of Computing, pp. 216–226, 1978.

[23] 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.

[24] Gabrielle Demange. Majority relation and median representative ordering. SERIEs: Journal of the Spanish Economic Association, Vol. 3, No. 1, pp. 95–109, 2012.

[25] Clemens Puppe and Arkadii Slinko. Condorcet domains, median graphs and the single-crossing property. Economic Theory, Vol. 67, No. 1, pp. 285–318, 2019.

[26] Victor Chepoi and Daniela Maftuleac. Shortest path problem in rectangular complexes of global nonpositive curvature. Computational Geometry, Vol. 46, No. 1, pp. 51–64, 2013.

[27] Victor Chepoi. Classification of graphs by means of metric triangles. Metody Diskret. Analiz, Vol. 49, pp. 75–93, 1989.