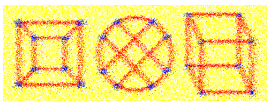

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 30, no. 1, pp. 339–362 (2026)
DOI: 10.7155/jgaa.v30i1.3052

On Shortest Path, BFS- and DFS-Tree Graphs

Prosenjit Bose[1] *Amirali Madani*[1] *Anil Maheshwari*[1] *Bobby Miraftab*[1]

[1]School of Computer Science, Carleton University, Ottawa, Ontario, Canada.

Submitted: June 2025 Accepted: June 2026 Published: August 2026

Article type: Regular Paper Communicated by: Csaba D. Tóth

Abstract. Flip graphs encode the structure of feasible transformations between combinatorial objects, making them a fundamental tool in reconfiguration problems. Tree graphs, which are flip graphs whose nodes represent the spanning trees of a graph, have received significant attention due to their algorithmic and structural properties. In this paper, we introduce new variations of tree graphs by restricting the spanning trees to shortest path trees, breadth-first search (BFS) trees, and depth-first search (DFS) trees. We prove that shortest path tree graphs are hamiltonian. Given any graph G, we present an algorithm that finds a hamiltonian cycle in its corresponding shortest path tree graph. We show that BFS-tree graphs and DFS-tree graphs are not necessarily connected. We establish some necessary conditions for the connectivity of BFS-tree and DFS-tree graphs. We provide an optimal linear-time algorithm for reconfiguration in shortest path tree graphs. Finally, we derive some bounds on the chromatic numbers of these new variations of tree graphs.

1 Introduction

Let $\mathcal{C}$ be a set of combinatorial objects. The *flip graph* of $\mathcal{C}$ is a graph whose nodes correspond to the objects in $\mathcal{C}$, with an edge between two nodes if their corresponding objects can be transformed into each other via a predefined flip operation which is typically a small local change of the object. Flip graphs have numerous applications [5], including graph recoloring, robot motion planning, and data structure reconfiguration, where they model feasible transformations between different states. As a result, they have been extensively studied, with a particular focus on their graph-theoretic properties, such as hamiltonicity [4, 8, 10, 20, 22, 24, 29], Eulerianity [27], connectivity [23, 25, 26], diameter [2, 5, 19], and chromatic number [15, 17]. From an algorithmic perspective, flip graphs are closely related to *reconfiguration problems*. These problems ask whether one can convert an initial feasible configuration into a target feasible configuration through a sequence of steps, where each step adheres to a reconfiguration rule, ensuring that every intermediate configuration remains

Supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

E-mail addresses: jit@scs.carleton.ca (Prosenjit Bose) amiralimadani@cmail.carleton.ca (Amirali Madani) anil@scs.carleton.ca (Anil Maheshwari) bobby.miraftab@gmail.com (Bobby Miraftab)

feasible. A broad range of reconfiguration problems has been studied in the literature for graph-theoretic structures, see [3, 7, 12, 21, 28].

Among the various types of flip graphs, those defined on spanning trees are commonly known as *tree graphs*. The tree graph of a connected graph G, denoted by $\mathsf{T}(G)$, is a graph whose nodes correspond to the spanning trees of G. Two distinct spanning trees, T_1 and T_2, are adjacent in $\mathsf{T}(G)$ if there exist edges $e_1, e_2 \in E(G)$ such that $(T_1 \setminus e_1) + e_2 = T_2$. A well-known result states that for any graph G and any edge $e \in E(\mathsf{T}(G))$, there exists a hamiltonian cycle in $\mathsf{T}(G)$ that contains e [10, 22, 24, 29]. Moreover, similar hamiltonicity results have been shown for restricted variants of tree graphs, where the underlying graph G is restricted to a specific graph class and the allowed edge flips follow additional constraints [4, 8]. In addition to their hamiltonicity, tree graphs have also been studied with respect to their connectivity [26], chromatic number [15], and in the context of reconfiguration problems [7].

Motivated by the study of tree graphs, we introduce new variations. Given a graph G, we define the *shortest path tree graph*, the BFS-*tree graph*, and the DFS-*tree graph*, by restricting the spanning trees to shortest path trees, BFS-trees, and DFS-trees of G, respectively. For a graph G with $r \in V(G)$, a tree T is a BFS-*tree* (respectively, DFS-*tree*) of G rooted at r if it can be obtained by running a breadth-first search (respectively, depth-first search) on G starting at r. For every vertex v in G, let $d_G(r, v)$ denote the length of the shortest path from r to v. A spanning tree T is a *shortest path tree* of G rooted at r if, for every vertex v, we have $d_T(r, v) = d_G(r, v)$. Every BFS-tree of G rooted at r is a shortest path tree, but the converse is not necessarily true. Figure 1 illustrates four shortest path trees of a graph G rooted at $r \in V(G)$: the first two (from left to right) are BFS-trees, while the last two are not.

The main results of this paper are as follows. We first prove that the shortest path tree graph of any connected graph is hamiltonian (Theorem 1) and that one such hamiltonian cycle can be found using our algorithm (Theorem 2). We then present an optimal linear-time algorithm for determining the shortest sequence of edge flips required to transform a shortest path tree T_i into another shortest path tree T_j of a graph G. This transformation guarantees that every intermediate tree is also a valid shortest path tree of G (Theorem 4). We also establish necessary conditions for the connectivity of BFS-tree graphs (Theorem 5) and provide bounds on the chromatic number of shortest path tree graphs (Proposition 1). Finally, we prove a necessary condition for the connectivity of DFS-tree graphs (Theorem 6) and derive an upper bound on their chromatic number (Proposition 3).

The main technical contributions of this paper concern shortest path tree graphs. In particular, Theorem 1 shows that $\mathsf{SPT}(G, r)$ is hamiltonian for every connected graph G, with $|V(G)| = n$, $|E(G)| = m$, and $|V(\mathsf{SPT}(G, r))| = N$. Theorem 3 gives an explicit $\mathcal{O}(n + m + N)$-time algorithm for finding a hamiltonian cycle of $\mathsf{SPT}(G, r)$, and Theorem 4 gives an optimal linear-time algorithm for finding a shortest reconfiguration sequence between two shortest path trees. Our results on BFS-tree graphs and DFS-tree graphs are of a different nature. Theorem 5 and Theorem 6 give necessary conditions for connectivity, and Example 4 shows that the condition in Theorem 6 is not sufficient. Together, these results show a clear difference between shortest path tree graphs and the corresponding BFS-tree and DFS-tree graphs. While shortest path tree graphs always admit strong reconfiguration properties, BFS-tree and DFS-tree graphs can already fail to be connected on simple examples. The chromatic number bounds in Proposition 1 and 3 provide additional structural information about these graph classes.

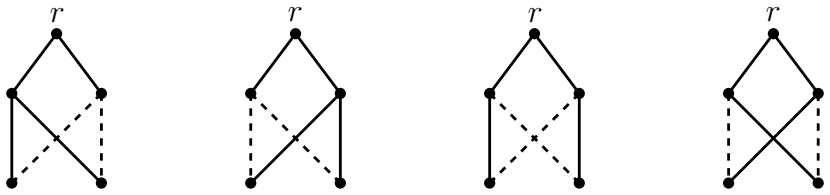


Figure 1: Some examples of shortest path trees, where the dashed edges represent the non-tree edges. Non-tree edges belong to the graph and are not part of the spanning tree.

2 Preliminaries

For graph-theoretic notation and definitions not defined in this paper, we refer the reader to [6]. Unless specified otherwise, G is assumed to be a connected graph with vertex set $V(G)$ and edge set $E(G)$. Throughout this paper, $r \in V(G)$ serves as the root of the shortest path trees, the BFS-trees, and the DFS-trees of G. For an edge $e = uv \in E(G)$, we refer to u and v (with $u, v \in V(G)$) as the *endpoints* of e. For two distinct vertices $u, v \in V(G)$, we define the *distance* between u and v in G, denoted by $d_G(u, v)$, as the minimum number of edges in any path connecting u and v. A path in G whose number of edges realizes $d_G(u, v)$ is called a *shortest path* between u and v, and we arbitrarily denote one such shortest path by $S(G, u, v)$. For a vertex $v \in V(G)$, the *eccentricity* of v is defined as $\epsilon_G(v) \coloneqq \max_{u \in V(G)} d_G(u, v)$. Moreover, $\deg_G(v)$ and $N_G(v)$ denote the degree and the set of neighbours of v in G, respectively (with $|N_G(v)| = \deg_G(v)$). For a graph G, we denote the maximum degree over its vertices by $\Delta(G)$. For two graphs G and H, we write $G \cong H$ if G is isomorphic to H.

Definition 1. A spanning tree T of G is a *shortest path tree* of G rooted at $r \in V(G)$ if for all $u \in V(G)$, we have $d_T(r, u) = d_G(r, u)$.

Notation 1. *Let G be a connected graph with $r \in V(G)$. For a non-negative integer i, the i-th* BFS *layer of G (denoted by $L_i(G, r)$) is the set of vertices at distance i from r in G. The collection of all L_i's is referred to as the* BFS *layering of G.*

When the context is clear, we may sometimes drop r from the above notation and denote the vertices in the i-th BFS layer rooted at r by $L_i(G)$. If two vertices u and v are adjacent in G, then they either belong to the same BFS layer, or they belong to two consecutive BFS layers.

Notation 2. *We define the outgoing and the incoming neighbourhood of a vertex $v \in V(G)$ as follows.*

1. *If $v \in L_i(G, r)$ for some $i \geq 0$, then $N_{out}(v) \coloneqq N_G(v) \cap L_{i+1}(G, r)$.*
2. *If $v \in L_i(G, r)$ for some $i > 0$, then $N_{in}(v) \coloneqq N_G(v) \cap L_{i-1}(G, r)$.*

We generalize Notation 2 to sets of vertices and for a set $S \subseteq V(G)$, we define $N_{\text{in}}(S) \coloneqq \bigcup_{v \in S} N_{\text{in}}(v)$ and $N_{\text{out}}(S) \coloneqq \bigcup_{v \in S} N_{\text{out}}(v)$.

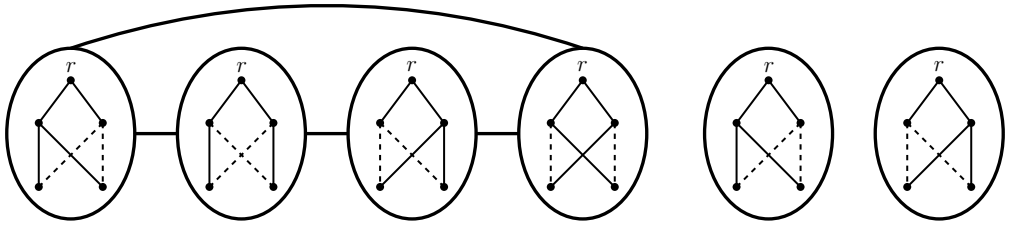
A shortest path tree T of G is a BFS-tree of G if it can be obtained by applying the standard implementation of the breadth-first traversal to G starting at r (see [9, Section 22.2]).

Definition 2. We formally define shortest path tree graphs and BFS-tree graphs as follows.

- The BFS-*tree graph*, denoted by $\mathsf{BFST}(G, r)$, is defined as the graph whose nodes correspond to all BFS-*trees* of G rooted at r. Two BFS trees T_1 and T_2 are adjacent in $\mathsf{BFST}(G, r)$ if there exist edges e_1 and e_2 in $E(G)$ such that $(T_1 \setminus e_1) + e_2 = T_2$. In this case, e_1 and e_2 are called *flippable*.

- The *shortest path tree graph*, denoted by $\mathsf{SPT}(G, r)$, is defined as the graph whose nodes correspond to all *shortest path trees* of G rooted at r. Two shortest path trees T_1 and T_2 are adjacent in $\mathsf{SPT}(G, r)$ if there exist edges e_1 and e_2 in $E(G)$ such that $(T_1 \setminus e_1) + e_2 = T_2$. In this case, e_1 and e_2 are called *flippable*.

Example 1. For the graph G depicted in Figure 1, Figure 2 depicts $\mathsf{SPT}(G, r)$ and $\mathsf{BFST}(G, r)$. As depicted in Figure 2(b), $\mathsf{BFST}(G, r)$ consists of two degree-zero nodes and is therefore disconnected. However, the shortest path tree graph is a hamiltonian cycle of length 4, see Figure 2(a). In this paper, we show that shortest path tree graphs are always hamiltonian and therefore connected (Section 3.1, Theorem 1). Moreover, we show necessary conditions for the connectivity of BFS-tree graphs (Section 3.4, Theorem 5).



(a) $\mathsf{SPT}(G, r)$ (b) $\mathsf{BFST}(G, r)$

Figure 2: An example of the graph G of Figure 1 along with its shortest path tree graph (a) and its BFS-tree graph (b).

We also study the properties of the *depth-first search* tree graphs (DFS-*tree graphs*). Along with BFS, DFS is an important graph traversal algorithm, enabling efficient algorithms for many graph problems [16]. Performing a DFS traversal on a graph G generates a spanning tree T of G, referred to as a DFS-tree. Before formally defining DFS-trees, note that every spanning tree T of a graph G gives us a partition of $E(G)$ into two classes, the set of tree edges $E(T)$ and the set of *non-tree edges* $E(G) \setminus E(T)$. In addition, every rooted spanning tree T of G defines a *partial order* $\preceq$ on $V(G)$:

$$x \preceq y \text{ if the unique tree path between } y \text{ to the root of } T \text{ includes } x$$

In order to prove our results for DFS-tree graphs, we use the *comparability* of the endpoints of its non-tree edges. For a spanning tree T of G rooted at $r \in V(G)$, we say that two vertices $u, v \in V(G)$ are *comparable* if $u \preceq v$ or $v \preceq u$. Furthermore, T is a *Trémaux spanning tree* or a *normal spanning tree* [11, 14] of G if the endpoints of every non-tree edge in T are comparable. It is well known in the literature that DFS-*trees* and normal spanning trees are equivalent, and that a depth-first traversal of G starting at $r \in V(G)$ always results in a normal spanning tree of G rooted at r (see [14] or Exercise 29 in Chapter 1 of [13]). By definition, normal spanning trees can be rooted at multiple vertices, as stated below.

Definition 3. Let T be a spanning tree of G with $\{t_1, t_2\} \subseteq V(G)$. We say T is a normal spanning tree or a DFS-tree *simultaneously rooted* at t_1 and t_2 if the endpoints of every non-tree edge of T are comparable with both t_1 and t_2 as the root.

Definition 3 may seem counterintuitive from an algorithmic perspective, as it allows a DFS-tree to be rooted at multiple vertices. Nonetheless, it is well-defined from a graph-theoretic standpoint and will later prove useful in establishing our results. We now formally define DFS-tree graphs as follows.

Definition 4. The DFS-*tree graph*, denoted by $\mathsf{DFST}(G, r)$, is a graph whose nodes correspond to all DFS-trees of G rooted at r. Two DFS-trees T_1 and T_2 are adjacent in $\mathsf{DFST}(G, r)$ if there exist edges $e_1, e_2 \in E(G)$ such that $(T_1 \setminus e_1) + e_2 = T_2$. In this case, the edges e_1 and e_2 are said to be *flippable*.

We return to our running example of Figure 1 to provide an example of a DFS-tree graph.

Example 2. For the graph G depicted in Figure 1, Figure 3 depicts $\mathsf{DFST}(G, r)$. In this example, $\mathsf{DFST}(G, r)$ is disconnected. We show necessary conditions for the connectivity of $\mathsf{DFST}(G, r)$ in Section 4.1 (Theorem 6).

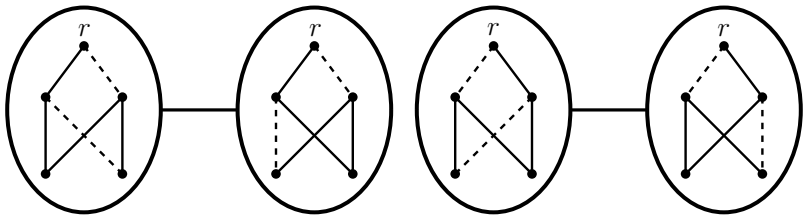


Figure 3: An example of a DFS-tree graph for the graph G of Figure 1.

3 Shortest Path Tree Graphs and BFS-Tree Graphs

3.1 Hamiltonicity of Shortest Path Tree Graphs

In this section, we show that for any connected graph G, $\mathsf{SPT}(G, r)$ is hamiltonian. Let C_m denote a cycle on m vertices and let K_n be the complete graph on n vertices. A graph is Hamiltonian if it contains a cycle that visits every vertex exactly once. For completeness, we also consider K_1 and K_2 to be Hamiltonian. Let G and H be two graphs. The *Cartesian product* of G and H, denoted by $G \square H$, is the graph with vertex set $V(G) \times V(H)$, where two vertices (u_1, v_1) and (u_2, v_2) are adjacent if and only if $u_1 = u_2$ and $v_1 v_2 \in E(H)$, or $v_1 = v_2$ and $u_1 u_2 \in E(G)$. The following result was shown by Foregger [18].

Lemma 1. *[18, Theorem 1] The following statements are true.*

(i) Let $m, n \geq 3$ be two integers. The Cartesian product of two cycles $C_m \square C_n$ is hamiltonian.

(ii) $H_1 \square H_2$ is hamiltonian, where H_1 and H_2 are each isomorphic to K_1 or K_2.

In the following lemma, we describe a procedure to find one such hamiltonian cycle. This procedure will later be used to find a hamiltonian cycle in $\mathsf{SPT}(G, r)$.

Lemma 2. *For any two integers $n, m \geq 2$, there exists a procedure to find a hamiltonian cycle in $H = C_n \square C_m$.*

Proof: We show that there always exists a hamiltonian cycle in H following a specific pattern. H consists of n subgraphs $H_1, \ldots, H_n$ with $H_i \cong C_m$ for all i, such that $V(H_1), \ldots, V(H_n)$ defines a partition of $V(H)$. For each $i \in \{1, \ldots, n\}$, let $v_{i,j}$ denote the j-th vertex on the cycle C_m of H_i. Without loss of generality, suppose that we have $v_{i,j}v_{i+1,j} \in E(H)$ for every $1 \leq i \leq n-1$ and $1 \leq j \leq m$. Construct a cycle in the following way (see Figure 4).

1. Start at $v_{1,1}$, and visit $v_{2,1}, \ldots, v_{n,1}$ in the same order.
2. After arriving at $v_{n,1}$, visit $v_{n,2}, v_{n,3}, \ldots, v_{n,m}$ in the same order. Then, visit $v_{n-1,m}$ and go to Step 3.
3. Suppose we arrive at $v_{i,2}$ or $v_{i,m}$ for some $i \in \{1, \ldots, n-1\}$.
   (i) If we arrive at $v_{i,2}$, then visit $v_{i,3}, \ldots, v_{i,m}$ in the same order. If $i = 1$, then visit $v_{1,1}$ and finish the cycle. Otherwise, visit $v_{i-1,m}$ and go back to Step 3.
   (ii) If we arrive at $v_{i,m}$, then visit $v_{i,m-1}, \ldots, v_{i,2}$ in the same order. If $i = 1$, then visit $v_{1,1}$ and finish the cycle. Otherwise, visit $v_{i-1,2}$ and go back to Step 3.

We now show that the described procedure generates a hamiltonian cycle. Observe that this procedure visits each vertex exactly once, because vertices $v_{i,1}$ for $i \in \{1, \ldots, n\}$ are all visited once in Step 1, and never visited again in Step 2 or Step 3. For any other vertex $v_{i,j}$ with $i \in \{1, \ldots, n\}$ and $j \in \{2, \ldots, m\}$, we visit it exactly once either through Step 2, Step 3(i), or Step 3(ii) in the descending order of i. Finally, we start the cycle at $v_{1,1}$ (Step 1), and also end it at $v_{1,1}$ either through $v_{1,m}$ (Step 3(i)) or $v_{1,2}$ (Step 3(ii)). Therefore, this procedure finds a hamiltonian cycle of $C_n \square C_m$. □

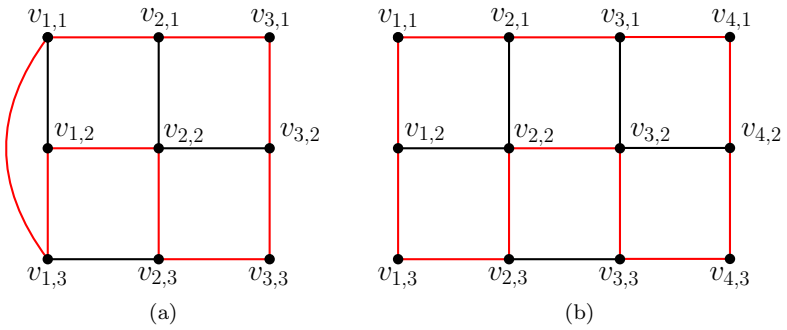


Figure 4: An example of the procedure of Lemma 2 for $C_3 \square C_3$ (a) and $C_4 \square C_3$. The edges of the hamiltonian cycle are depicted in red. For convenience, some edges of the graphs have been omitted from the figure.

We now show our first result.

Theorem 1. $\mathsf{SPT}(G, r)$ *is hamiltonian.*

Proof: For convenience, we denote K_1 and K_2 by C_1 and C_2, respectively. We prove this theorem by induction on $|V(G)|$.
Base Step: When $|V(G)| = 1$, $\mathsf{SPT}(G, r)$ is a single node and thus hamiltonian.
Induction Step: Suppose that Theorem 1 holds for any graph with fewer than $n > 1$ vertices and G is an n-vertex connected graph with $r \in V(G)$. Let $v \in V(G)$ be a vertex in the last BFS layer (recall Notation 1).
Let T be any shortest path tree of G rooted at r. Since v is in the last BFS layer of G, it must be a leaf of T. Therefore, $T \setminus v$ is connected, implying that $G \setminus v$ is also connected. Moreover, by the inductive hypothesis, $\mathsf{SPT}(G \setminus v, r)$ is hamiltonian. Let N denote the number of shortest path trees of $G \setminus v$ rooted at r. Let $d = |N_{\text{in}}(v)|$ (see Notation 2) and $e_j = v_j v$ for all $j \in \{1, \ldots, d\}$ (with $N_{\text{in}}(v) = \{v_1, \ldots, v_d\}$).

Observation 1. *Let $\{T_1, \ldots, T_N\} = V(\mathsf{SPT}(G \setminus v, r))$ be the shortest path trees of $G \setminus v$ rooted at r. Then, the shortest path trees of G rooted at r correspond to the set*

$$\{T_i \cup e_j \mid T_i \in V(\mathsf{SPT}(G \setminus v, r)) \text{ and } 1 \leq j \leq d\}.$$

We partition the shortest path trees of G into d sets as follows. For every $1 \leq j \leq d$, define S_j as the set of all shortest path trees of G (rooted at r) that contain the edge e_j as a tree edge. By Observation 1, we have $\cup_{j=1}^{d} S_j = V(\mathsf{SPT}(G, r))$ and the sets $S_1, \ldots, S_d$ define a partition of $V(\mathsf{SPT}(G, r))$. For each S_j, we have $S_j = \{T_1 \cup e_j, \ldots, T_N \cup e_j\}$, where $T_1, \ldots, T_N$ are the shortest path trees of $G \setminus v$ rooted at r.

We introduce some notation: let H be any graph with $X \subseteq V(H)$, then $H[X]$ denotes the subgraph of H induced by X. We show the following claim.

Claim 1. *$\mathsf{SPT}(G, r)$ has $K_d \square C_N$ as a spanning subgraph.*

Proof: We first show that for any set S_j in the defined partition, we have $\mathsf{SPT}(G, r)[S_j] \cong \mathsf{SPT}(G \setminus v, r)$. Let $S_j = \{T_1 \cup e_j, \ldots, T_N \cup e_j\}$. If two trees T_i and T_ℓ are adjacent in $\mathsf{SPT}(G \setminus v, r)$, then $T_\ell \cup e_j$ and $T_i \cup e_j$ are adjacent in $\mathsf{SPT}(G, r)$. Conversely, if $T_\ell \cup e_j$ and $T_i \cup e_j$ are adjacent in $\mathsf{SPT}(G, r)$, then we must have $|E(T_i) \setminus E(T_\ell)| = 1$ and T_i and T_ℓ are adjacent in $\mathsf{SPT}(G \setminus v, r)$.

Partition the shortest path trees of $\mathsf{SPT}(G, r)$ into d sets $S_1, \ldots, S_d$ as stated in Observation 1. From the first part of this proof, we know that for every j, $\mathsf{SPT}(G, r)[S_j] \cong \mathsf{SPT}(G \setminus v, r)$. By the inductive hypothesis, $\mathsf{SPT}(G, r)[S_j]$ has C_N as a spanning subgraph for every j.

Let S_k and S_j be any two of these sets (with $S_k \neq S_j$). For any $T_\ell \in V(\mathsf{SPT}(G \setminus v, r))$, we have $T_\ell \cup e_k \in S_k$ and $T_\ell \cup e_j \in S_j$. Furthermore, $T_\ell \cup e_k$ and $T_\ell \cup e_j$ are adjacent in $\mathsf{SPT}(G, r)$ as they differ by exactly one edge. Since this analysis applies to any S_k, S_j, and $T_\ell \in V(\mathsf{SPT}(G \setminus v, r))$, we can see that $\mathsf{SPT}(G, r)$ has $K_d \square C_N$ as a spanning subgraph because for each j, $\mathsf{SPT}(G, r)[S_j]$ has C_N as a spanning subgraph (see Figure 5 for a depiction). □

By Claim 1, $SPT(G, r)$ has $K_d \square C_N$ as a spanning subgraph. If $d \geq 3$, then K_d contains a spanning cycle C_d, and hence $K_d \square C_N$ contains $C_d \square C_N$ as a spanning subgraph. By Lemma 1(i), $C_d \square C_N$ is hamiltonian, and therefore $SPT(G, r)$ is hamiltonian. If $d \in \{1, 2\}$, then $K_d \cong C_d$, and thus $K_d \square C_N \cong C_d \square C_N$. In this case, the result follows from Lemma 1(ii) (and the convention that $C_1 \cong K_1$ and $C_2 \cong K_2$). This completes the proof.

□

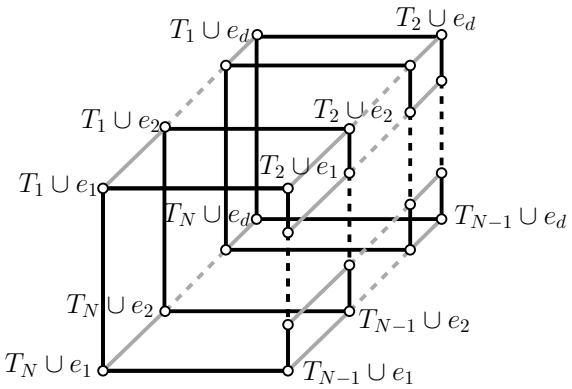


Figure 5: $\mathsf{SPT}(G, r)$ has $K_d \square C_N$ as a spanning subgraph. Each S_j has C_N as a spanning subgraph. The edges within each S_j have been depicted in black, and the edges between two distinct sets S_i and S_j have been depicted in gray. For readability, only a $P_d \square C_N$ was drawn in this figure, where P_d is a path on d vertices.

3.2 An Algorithm for Finding a Hamiltonian Cycle in $\mathsf{SPT}(G, r)$

Given a connected graph G, in this section we present an $\mathcal{O}(m + n + N)$-time algorithm to find a hamiltonian cycle in $\mathsf{SPT}(G, r)$, where $n = |V(G)|$, $m = |E(G)|$, and $N = |V(\mathsf{SPT}(G, r))|$. For convenience, we assume $C_1 \cong K_1$ and $C_2 \cong K_2$ in the remainder of this section.

For the sake of simplicity, we describe our algorithm in two parts. The rest of this section is organized as follows. Section 3.2.1 describes an $\mathcal{O}(nN)$-space data structure $\mathcal{D}$ for storing the shortest path trees of G. Using the procedure of Lemma 2 and the data structure of Section 3.2.1, we present an $\mathcal{O}(m + nN)$-time algorithm in Section 3.2.2 and prove its correctness. Later in Section 3.2.3, we briefly explain that the data structure $\mathcal{D}$ can be compressed into another data structure $\mathcal{D}'$ of size $\mathcal{O}(n + N + M)$, using which the time complexity of our algorithm can be improved to $\mathcal{O}(n + N + m)$ (Theorem 3).

3.2.1 A Data Structure for Storing the Shortest Path Trees

We describe a data structure motivated by the induction step in the proof of Theorem 1. Recall that for any vertex v in the last BFS layer of G and any shortest path tree T of G, $T \setminus v$ is a shortest path tree of $G \setminus v$. Consider any breadth-first search traversal of G, and let f be a function $f : V(G) \to \{1, \ldots, n\}$, where $n = |V(G)|$. For each $u \in V(G)$, u is the $f(u)$-th vertex visited during this traversal. This labeling by f defines a total order on $V(G)$ such that $f(v_1) < \cdots < f(v_n)$ with $V(G) = \{v_1, \ldots v_n\}$. In the remainder of this section, we assume $V(G) = \{v_1, \ldots, v_n\}$ with $v_1 = r$ and $f(v_1) < \cdots < f(v_n)$. These correspond to the order in which the vertices are first visited in a breadth-first search traversal. In any such ordering, all vertices in an earlier BFS layer receive smaller numbers than all vertices in a later BFS layer, while the order of the vertices within the same layer may be arbitrary.

We now describe our data structure, denoted by $\mathcal{D}$. $\mathcal{D}$ is a tree with n layers whose root and leaves are in layers 1 and n, respectively. Moreover, the nodes and the edges of $\mathcal{D}$ are defined as follows.

(i) For each $1 \leq \ell \leq n$, define $\mathcal{V}_\ell = \{v_1, \ldots, v_\ell\}$.

(ii) The nodes in each layer $1 \leq \ell \leq n$ of $\mathcal{D}$ correspond to the shortest path trees of $G[\mathcal{V}_\ell]$.

(iii) Let $2 \leq \ell \leq n$ and let u be a node in layer ℓ of $\mathcal{D}$ corresponding to a shortest path tree T of $G[\mathcal{V}_\ell]$. The parent of u in layer $\ell - 1$ of $\mathcal{D}$ is a node corresponding to shortest path tree $T \setminus v_\ell$ of $G[\mathcal{V}_{\ell-1}]$.

Example 3. For the graph G depicted in Figure 6(a) on five vertices with $f(v_1) < f(v_2) < f(v_3) < f(v_4) < f(v_5)$, where $f(v_i) = i$ and $v_1 = r$, the corresponding data structure $\mathcal{D}$ is depicted in Figure 6(b). $\mathcal{D}$ has five layers. In the first layer lies the root of $\mathcal{D}$, which is the unique shortest path tree T_1 of $G[\{v_1\}]$ with $T_1 \cong K_1$ and $T_1 = v_1 = r$. The nodes in the fifth layer of $\mathcal{D}$ correspond to the shortest path trees of G. Notice how each node in any layer $1 \leq \ell \leq 4$ has exactly $|N_{\text{in}}(v_{\ell+1})|$ children.

The following observation follows from the definition of $\mathcal{D}$.

Observation 2. *Let G be any n-vertex graph, and let $\mathcal{D}$ be as described. Then, any node u in layer ℓ of $\mathcal{D}$ has exactly $|N_{in}(v_{\ell+1})|$ children for $1 \leq \ell \leq n-1$.*

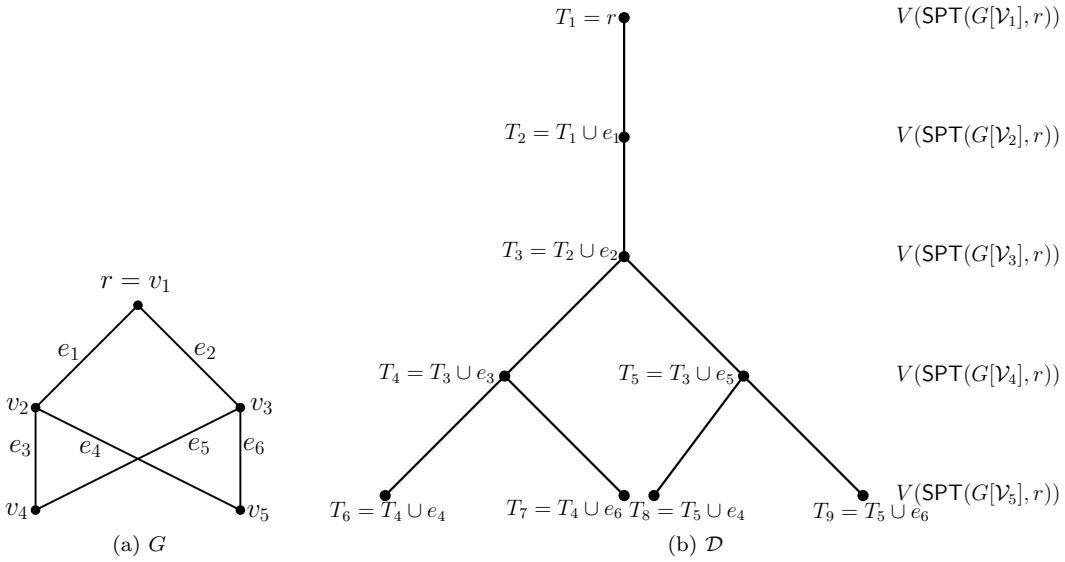


Figure 6: An example of a graph G (a) and its corresponding data structure $\mathcal{D}$ (b).

In the next lemma, we describe how to construct $\mathcal{D}$.

Lemma 3. *Let $n = |V(G)|$, $m = |E(G)|$, and $N = |V(\mathsf{SPT}(G, r))|$. $\mathcal{D}$ can be constructed in $\mathcal{O}(m + nN)$ time and it can be stored in $\mathcal{O}(nN)$ space.*

Proof: By Observation 2, we can deduce that $\mathcal{D}$ can have many nodes with exactly one child, possibly resulting in long paths. Therefore, $\mathcal{D}$ has at most $\mathcal{O}(nN)$ nodes. We use a breadth-first search of G in $\mathcal{O}(n + m)$ time to calculate $f(v)$ for all $v \in V(G)$. Moreover, we can also compute $L_i(G)$ for all i, and $N_{\text{in}}(v)$ for all v in $\mathcal{O}(n+m)$ time. We can store $N_{\text{in}}(v)$ for all v in an adjacency list that requires $\mathcal{O}(n + m)$ space, and for each vertex v, we store $|N_{\text{in}}(v)|$ for an additional $\mathcal{O}(1)$

space per vertex. Finally, we can find the ordering $v_1, \ldots, v_n$ with $f(v_1) < \cdots < f(v_n)$ without having to sort the vertices. To do so, every time a new vertex is visited in the breadth-first traversal, we store it in an array of size n such that v_ℓ can be retrieved in $\mathcal{O}(1)$ time for all $\ell \in \{1, \ldots, n\}$. $\mathcal{D}$ can then be constructed in a top-down manner. For the first layer, we store $\mathcal{V}_1 = \{v_1\}$ and add a root to $\mathcal{D}$. This root corresponds to the unique shortest path tree of $G[\mathcal{V}_1]$.

To construct $\mathcal{D}$ in a top-down manner after designating a root, let u be any node in layer $1 \leq \ell \leq n-1$ of $\mathcal{D}$, and let $\{v_{\ell+1}\} = \mathcal{V}_{\ell+1} \setminus \mathcal{V}_\ell$ (recall that $v_{\ell+1}$ can be retrieved in $\mathcal{O}(1)$ time). By Observation 2, u must have d children in $\mathcal{D}$, where $d = |N_{\text{in}}(v_{\ell+1})|$. Let $N_{\text{in}}(v_{\ell+1}) = \{v_1, \ldots, v_d\}$ and $e_j = v_j v_{\ell+1} \in E(G)$ for all $j \in \{1, \ldots, d\}$. Recall that u must correspond to a shortest path tree T of $G[\mathcal{V}_\ell]$. We assign d children to u in $\mathcal{D}$ corresponding to shortest path trees $T \cup e_1, \ldots, T \cup e_d$ of $G[\mathcal{V}_{\ell+1}]$. This step enumerates all shortest path trees of $G[\mathcal{V}_{\ell+1}]$. For each newly added node $T \cup e_j$ in layer $\ell + 1$, we store e_j for a constant amount of space per node of $\mathcal{D}$. Given any node u' in layer $\ell + 1$ of $\mathcal{D}$, its corresponding edge e_j can be retrieved in $\mathcal{O}(1)$ time (see Figure 6 for an example of these edges). □

3.2.2 An Algorithm

We now present an algorithm for finding a hamiltonian cycle in $\mathsf{SPT}(G, r)$. The algorithm (SPTG_HAMIL) is described in Algorithm 1. Instead of explicitly listing all shortest path trees on the hamiltonian cycle, SPTG_HAMIL outputs only the first tree on the cycle and a sequence of N edge flips that transform each tree into the next. See Line 3 of SPTG_HAMIL for details.

SPTG_HAMIL (in Line 4) first constructs $\mathcal{D}$ using the procedure of Lemma 3. Then, through the loop in Line 6 to Line 9, SPTG_HAMIL constructs a hamiltonian cycle of $\mathsf{SPT}(G[\mathcal{V}_\ell], r)$ for each layer $\ell \in \{1, \ldots, n\}$ of $\mathcal{D}$. The hamiltonian cycle of $\mathsf{SPT}(G[\mathcal{V}_\ell], r)$ is constructed using the procedure of Lemma 2 based on the hamiltonian cycle of $\mathsf{SPT}(G[\mathcal{V}_{\ell-1}], r)$, with the base case being $\mathsf{SPT}(G[\mathcal{V}_1], r)$ (which is isomorphic to K_1).

Algorithm 1

1: **procedure** SPTG_HAMIL
2: **Input:** A graph G
3: **Output:** A hamiltonian cycle in $\mathsf{SPT}(G, r)$ in the following format: A shortest path tree T_1 of G and N tuples $(e_1, e'_1), \ldots, (e_N, e'_N)$ such that for all j, $\{e_j, e'_j\} \subseteq E(G)$. The hamiltonian cycle is $T_1, \ldots, T_{N+1}$, where $T_1 = T_{N+1}$ and $T_j = (T_{j-1} \setminus e_{j-1}) + e'_{j-1}$ for all $j > 1$.
4: Construct $\mathcal{D}$ (Lemma 3).
5: Store the root of $\mathcal{D}$ as the hamiltonian cycle for $\mathsf{SPT}(G[\mathcal{V}_1], r)$ (note that $\mathsf{SPT}(G[\mathcal{V}_1], r) \cong K_1$).
6: **for** each $\ell \in \{2, \ldots, n\}$ **do**
7: Find and store a hamiltonian cycle of $\mathsf{SPT}(G[\mathcal{V}_\ell], r)$ based on the hamiltonian cycle of $\mathsf{SPT}(G[\mathcal{V}_{\ell-1}], r)$ (see Lemma 2)
8: For each tree on the hamiltonian cycle of $\mathsf{SPT}(G[\mathcal{V}_\ell], r)$, store its corresponding node in layer ℓ of $\mathcal{D}$. For every two consecutive trees on this cycle, store the corresponding edge flip (e, e') (see Line 3).
9: **end for**
10: For the hamiltonian cycle of $\mathsf{SPT}(G[\mathcal{V}_n], r) = \mathsf{SPT}(G, r)$, find the corresponding shortest path tree of the first node on the cycle (Claim 2). Return the first shortest path tree, along with all other edge flips.
11: **end procedure**

We first prove the correctness of SPTG_HAMIL through the following lemma.

Lemma 4. *For any layer $\ell \in \{1, \ldots, n\}$ of $\mathcal{D}$, SPTG_HAMIL finds a hamiltonian cycle of $\mathsf{SPT}(G[\mathcal{V}_\ell], r)$. Moreover, for any two consecutive nodes on this cycle, SPTG_HAMIL finds the corresponding edge flip (e, e') with $\{e, e'\} \subseteq E(G)$.*

Proof: We prove this claim by induction on ℓ.

Base Step: When $\ell = 1$, the root of $\mathcal{D}$ corresponds to the unique shortest path tree of $G[\mathcal{V}_1]$ and the claim trivially holds.
Induction Step: Let $2 \leq \ell \leq n$ and let $T_1, \ldots, T_{N'}$ be a hamiltonian cycle of $\mathsf{SPT}(G[\mathcal{V}_{\ell-1}], r)$. By the inductive hypothesis, the algorithm has correctly found the corresponding node in layer $\ell - 1$ to each tree T_i on this cycle. Moreover, for any two consecutive nodes of this hamiltonian cycle, the algorithm has correctly computed the corresponding edge flip (e, e') with $\{e, e'\} \subseteq E(G)$.

For each shortest path tree T_i of $G[\mathcal{V}_{\ell-1}]$ (with $i \in \{1, \ldots, N'\}$), let u_i denote its corresponding node in layer $\ell - 1$ of $\mathcal{D}$. Furthermore, let $u_{i,j}$ denote the node in layer ℓ corresponding to the shortest path tree $T_i \cup e_j$ of $G[\mathcal{V}_\ell]$ for $j \in \{1, \ldots, d\}$, where $d = |N_{\text{in}}(v_\ell)|$ with $\{v_\ell\} = \mathcal{V}_\ell \setminus \mathcal{V}_{\ell-1}$.

Since $T_1, \ldots, T_{N'}$ is a hamiltonian cycle in $\mathsf{SPT}(G[\mathcal{V}_{\ell-1}], r)$, trees $T_1 \cup e_j, \ldots, T_{N'} \cup e_j$ (corresponding to nodes $u_{1,j}, \ldots u_{N',j}$ of $\mathcal{D}$) form a cycle in $\mathsf{SPT}(G[\mathcal{V}_\ell], r)$. We can now construct the hamiltonian cycle of $\mathsf{SPT}(G[\mathcal{V}_\ell], r)$ using the procedure of Lemma 2. When constructing the cycle, if we arrive at $u_{i,j}$ from $u_{i,j'}$ $(j \neq j')$, we designate $(e_j, e_{j'})$ as the corresponding edge flip on the cycle. Otherwise, if we arrive at $u_{i,j}$ from $u_{i',j}$ $(i \neq i')$, we designate the edge flip between u_i and $u_{i'}$ as the edge flip tuple between $u_{i,j}$ from $u_{i',j}$. By the inductive hypothesis, the edge flip between u_i and $u_{i'}$ has been correctly computed in the hamiltonian cycle of $\mathsf{SPT}(G[\mathcal{V}_{\ell-1}], r)$. □

An example of SPTG_HAMIL applied to the graph G of Figure 6 is depicted in Figure 7. In Figure 7, the details related to SPTG_HAMIL are highlighted in red. The hamiltonian cycle for each depth is computed in a top-down manner starting from the root. In each layer, each node is annotated with the order at which it appears on the hamiltonian cycle of its respective layer. Each edge of the hamiltonian cycle is also annotated with its corresponding edge flip (e, e').

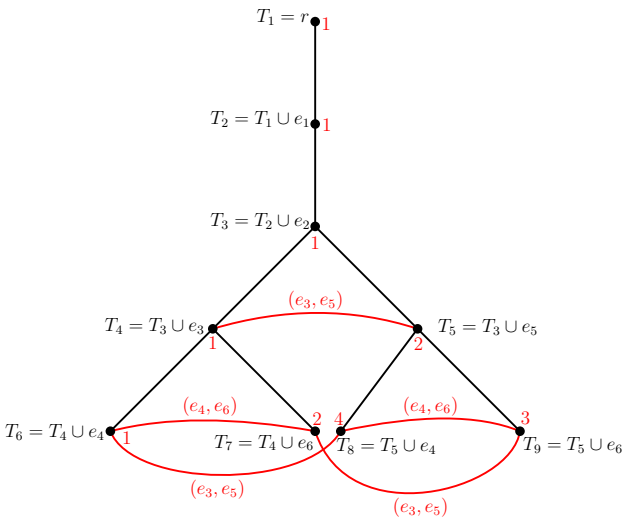


Figure 7: An example of SPTG_HAMIL when applied to the graph of Figure 6.

We present our main result of this section.

Theorem 2. *SPTG_HAMIL finds a hamiltonian cycle of* $\mathsf{SPT}(G, r)$ *in* $\mathcal{O}(m + nN)$ *time, where* $n = |V(G)|$, $m = |E(G)|$, *and* $N = |V(\mathsf{SPT}(G, r))|$.

Proof: The correctness of SPTG_HAMIL is due to Lemma 4 for $\ell = n$, thus we prove the time complexity.

Claim 2. *Let $n = |V(G)|$ and u be any node in the last layer of $\mathcal{D}$. The shortest path tree T of G corresponding to u can be retrieved in $\mathcal{O}(n)$ time.*

Proof: We traverse $\mathcal{D}$ starting from u in the last layer and go upwards towards the root. For any layer $2 \leq \ell \leq n$, v_ℓ can be retrieved in constant time. Moreover, for each node $T \cup e_j$ in layer ℓ of $\mathcal{D}$, we retrieve e_j in $\mathcal{O}(1)$ time (refer to the proof of Lemma 3 for details). Edge $e_j \in E(G)$ determines the BFS ancestor of v_ℓ. By traversing all layers of $\mathcal{D}$, we can determine the BFS ancestor of any vertex $v \in V(G)$. By determining the BFS ancestor of every vertex $v \in V(G)$, we can construct the shortest path tree of G corresponding to u in $\mathcal{O}(n)$ time, since $\mathcal{D}$ has exactly n layers. □

Line 4 of SPTG_HAMIL can be implemented to run in $\mathcal{O}(m+nN)$ time by Lemma 3. Moreover, finding the hamiltonian cycle of $\mathsf{SPT}(G[\mathcal{V}_1], r)$ can be done in $\mathcal{O}(1)$ time (Line 5). We now show that each iteration of the loop in Line 6 to Line 9 of SPTG_HAMIL runs in $\mathcal{O}(|V(\mathsf{SPT}(G[\mathcal{V}_\ell], r)|)$ time, implying that in total, the loop itself runs in $\mathcal{O}(nN)$ time by Lemma 3. Observe that we can modify $\mathcal{D}$ without any space overhead so that given the node u_i for any $i \in \{1, \ldots, N'\}$, any of its children $u_{i,j}$ in $\mathcal{D}$ can be accessed in constant time for all $j \in \{1, \ldots, d\}$. For each non-leaf node of $\mathcal{D}$, we can store an array pointing to each of its children, and the aforementioned constant-time look-ups are supported.

Therefore, we deduce that applying the procedure of Lemma 2 to each layer ℓ can be done in $\mathcal{O}(|V(\mathrm{SPT}(G[V_\ell], r))|)$ time in total, because each child of a non-leaf node u_i of $\mathcal{D}$ can be accessed in constant time. Furthermore, each edge flip on the hamiltonian cycle of layer $\ell - 1$ can be retrieved in constant time, refer to the induction step in the proof of Lemma 4 for more details. Finally, we can retrieve the shortest path tree corresponding to the first node on this cycle (Line 10) in $\mathcal{O}(n)$ time (Claim 2) and report it along with all edge flips between consecutive trees on the cycle. This concludes the proof of Theorem 2. □

3.2.3 A Linear-Time Algorithm

We now modify data structure $\mathcal{D}$ into another data structure $\mathcal{D}'$ with space complexity $\mathcal{O}(n+m+N)$. This allows us to improve the time complexity of the algorithm in Theorem 2 to $\mathcal{O}(n+m+N)$.

Data structure $\mathcal{D}'$ is a variation of $\mathcal{D}$ in which all non-leaf nodes are forced to have at least two children (given that $|V(\mathcal{D})| > 1$). To begin, we present the following key lemma, the correctness of which is due to Claim 1.

Lemma 5. *If $|N_{in}(v_n)| = 1$, then $\mathsf{SPT}(G \setminus v_n, r) \cong \mathsf{SPT}(G, r)$.*

By Lemma 5, we can improve the "peeling" process in the construction $\mathcal{D}$. Precisely, we remove the long paths in $\mathcal{D}$ by only considering layers ℓ with $|N_{\mathrm{in}}(v_{\ell+1})| > 1$.

Before giving the formal definition of $\mathcal{D}'$, we briefly explain the idea behind it. The data structure $\mathcal{D}$ may be viewed as a layered decision tree for shortest path trees, where each layer corresponds to adding one vertex in breadth-first search order. Thus, if we compare two consecutive layers of $\mathcal{D}$, the set of newly added vertices always has size one. A branching occurs only when the new vertex has more than one possible parent in the previous BFS layer. If $|N_{\mathrm{in}}(v_i)| = 1$, then there is no choice when v_i is added, since it has a unique possible parent in the previous BFS layer. Thus, in $\mathcal{D}$, the transition from the layer for $G[\{v_1, \ldots, v_{i-1}\}]$ to the layer for $G[\{v_1, \ldots, v_i\}]$ does not branch, and each node produces exactly one child. Consequently, any maximal consecutive block of such vertices produces a path of internal degree one in $\mathcal{D}$. In $\mathcal{D}'$, we compress each such

block into a single step. Here, the subscript ℓ denotes the depth in $\mathcal{D}'$. Accordingly, $\mathcal{V}'_\ell$ is the set of vertices present up to depth ℓ, and $\Delta_\ell = \mathcal{V}'_\ell \setminus \mathcal{V}'_{\ell-1}$ is the set of vertices added from depth $\ell-1$ to depth ℓ. Unlike in $\mathcal{D}$, the set Δ_ℓ may contain more than one vertex. Its first vertex is the one at which the new compressed step begins, while the remaining vertices, if any, are exactly the consecutive vertices whose addition is forced, that is, the vertices v with $|N_{\text{in}}(v)| = 1$.

More formally, $\mathcal{D}'$ has the following properties (in the remainder of this section, we assume that G has at least two shortest path trees rooted at r).

(i) $\mathcal{D}'$ is of depth β, where the root belongs to depth 0 and the leaves are at depth $\beta - 1$.

(ii) The nodes at each depth $0 \le \ell \le \beta - 1$ of $\mathcal{D}'$ correspond to the shortest path trees of $G[\mathcal{V}'_\ell]$, where each $\mathcal{V}'_\ell$ is defined as follows.

  (a) $\mathcal{V}'_0 = \{v_1\} \cup (\bigcup_{i=2}^{j} v_i)$ where j is the greatest integer such that $|N_{\text{in}}(v_i)| = 1$ for all $2 \le i \le j$.

  (b) $\mathcal{V}'_{\beta-1} = V(G)$.

  (c) For $0 \le \ell \le \beta - 2$, let $\mathcal{V}'_\ell = \{v_1, \ldots, v_k\}$ with $v_1 = r$ and $f(v_1) < \cdots < f(v_k)$. Define $\mathcal{V}'_{\ell+1} = \mathcal{V}'_\ell \cup \{v_{k+1}\} \cup \bigcup_{i=k+2}^{j} v_i$, such that j is the greatest integer for which $|N_{\text{in}}(v_i)| = 1$ for all $(k+2) \le i \le j$. If no such j exists and $|N_{\text{in}}(v_{k+2})| \ge 2$, then set $\mathcal{V}'_{\ell+1} = \mathcal{V}'_\ell \cup \{v_{k+1}\}$.

(iii) For $1 \le \ell \le \beta - 1$, let $\Delta_\ell = \mathcal{V}'_\ell \setminus \mathcal{V}'_{\ell-1}$. Let u be any node at depth ℓ of $\mathcal{D}'$ that corresponds to a shortest path tree T of $G[\mathcal{V}'_\ell]$. The parent of u in $\mathcal{D}'$ corresponds to a shortest path tree $T \setminus \Delta_\ell$ of $G[\mathcal{V}'_{\ell-1}]$.

An example of $\mathcal{D}'$ for the same graph G (Figure 6(a)) is depicted in Figure 8.

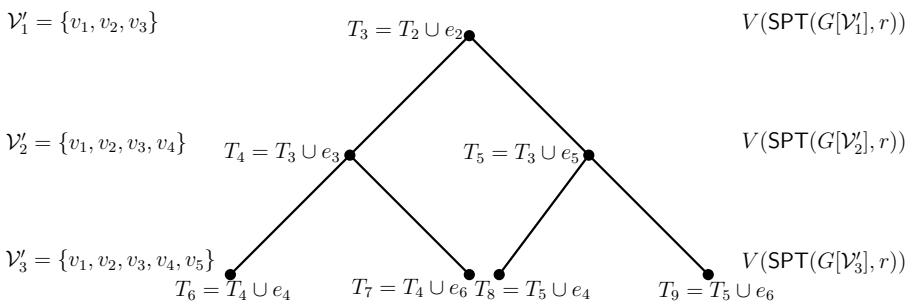


Figure 8: An example of $\mathcal{D}'$ for the same graph G depicted in Figure 6(a).

We have the following lemma.

Lemma 6. *Let $n = |V(G)|$, $m = |E(G)|$, and $N = |V(\mathsf{SPT}(G, r))|$. $\mathcal{D}'$ can be constructed in $\mathcal{O}(m+n+N)$ time and it can be stored in $\mathcal{O}(N+n)$ space. Furthermore, using $\mathcal{D}'$ we can retrieve the shortest path tree corresponding to any of its leaves in $\mathcal{O}(n)$ time.*

Proof: The proof is almost identical to that of Lemma 3, so we skip the details. Similar to the proof of Lemma 3, $\mathcal{O}(n+m)$ time is needed for preprocessing. Since every non-leaf node has at least two children, D' has $\mathcal{O}(N)$ nodes (because the nodes in the last layer of $\mathcal{D}'$ correspond to

$V(\mathsf{SPT}(G,r)))$. Therefore, constructing $\mathcal{D}'$ takes $\mathcal{O}(n+N+m)$ time in total. Regarding the space complexity, we store $\mathcal{O}(1)$ information per node of $\mathcal{D}'$ and for each layer $1 \leq \ell \leq \beta - 1$, we store the set Δ_ℓ (as described in Step (iii) of the construction of $\mathcal{D}'$). These sets take $\mathcal{O}(n)$ space in total and allow us to reconstruct each shortest path tree T corresponding to a leaf l of $\mathcal{D}'$ by traversing l upwards towards the root. □

We now present the main result of this section. The proof is similar to that of Theorem 2 using SPTG_HAMIL. Therefore, we omit the proof to avoid redundancy. Note that constructing $\mathcal{D}'$ takes $\mathcal{O}(n + N + m)$ time by Lemma 6. Similar to Theorem 2, the hamiltonian cycles are constructed layer-by-layer in $\mathcal{D}'$ from top to bottom and find the edge flips. This takes $\mathcal{O}(N)$ time in total. The first shortest path tree on the cycle can be obtained in $\mathcal{O}(n)$ time using the sets Δ_ℓ (see Lemma 6). Therefore, the algorithm takes $\mathcal{O}(n + N + m)$ time in total.

Theorem 3. *A hamiltonian cycle of* $\mathsf{SPT}(G,r)$ *can be found in* $\mathcal{O}(m + n + N)$ *time, where* $n = |V(G)|$, $m = |E(G)|$, *and* $N = |V(\mathsf{SPT}(G,r))|$.

3.3 Shortest Paths on $\mathsf{SPT}(G,r)$

Motivated by the literature on reconfiguration problems (see Section 1), in this section, we study the algorithmic problem of finding the shortest path between two shortest path trees of G in $\mathsf{SPT}(G,r)$. Let G be any connected graph, with T_i and T_j being any two distinct shortest path trees of G rooted at r. We present an $\mathcal{O}(|V(G)|)$-time algorithm for finding the shortest path between T_i and T_j in $\mathsf{SPT}(G,r)$.

The pseudocode of this algorithm (SPTG_PATH) is described in Algorithm 2. Given two

Algorithm 2

```
1: procedure SPTG_PATH
2:     Input: The graph G and two shortest path trees T_i and T_j of G rooted at r ∈ V(G)
3:     Output: S(SPT(G,r), T_i, T_j)
4:     Initialize S(SPT(G,r), T_i, T_j) as a sequence containing only T_i
5:     for each ℓ ∈ {ε_G(r), ..., 1} (taken in descending order) do
6:         for each vertex v in L_ℓ(G) do
7:             Let u_i and u_j be the BFS parents of v in T_i and T_j, respectively (with v ∈ N_out(u_i) ∩
       N_out(u_j))
8:             if u_i ≠ u_j then
9:                 Set T_i ← ((T_i ∖ u_i v) + u_j v), where {u_i v, u_j v} ⊆ E(G)
10:                Add T_i to S(SPT(G,r), T_i, T_j)
11:            end if
12:        end for
13:    end for
14:    return S(SPT(G,r), T_i, T_j)
15: end procedure
```

trees T_i and T_j, SPTG_PATH first computes $\epsilon_G(r)$ (the eccentricity of r in G). Then, for each value $\ell \in \{\epsilon_G(r), \ldots, 1\}$ taken in descending order, each vertex v of the ℓ-th BFS layer is considered. For each such vertex v, if the level-$(\ell-1)$ parent of v in T_i is different from its counterpart in T_j, an edge flip is executed and T_i is updated, see Line 9 and Line 10. When SPTG_PATH terminates, every vertex in $V(G)$ has the same BFS parent in T_i and T_j and $T_i = T_j$. We assume that, prior

to SPTG_PATH, the vertices in $L_\ell(G)$ (for each $\ell \geq 1$) have already been computed. Note that given T_i and T_j, any algorithm needs to execute at least $|E(T_i) \setminus E(T_j)|$ edge flips. Therefore, the optimality of SPTG_PATH is due to the following theorem.

Theorem 4. *Let T_i and T_j be any two shortest path trees of G rooted at r. Then, SPTG_PATH finds a path of length exactly $|E(T_i) \setminus E(T_j)|$ between T_i and T_j in $\mathsf{SPT}(G, r)$ in $\mathcal{O}(|V(G)|)$ time.*

Proof: We first show the time complexity. Computing the vertices of $L_\ell(G)$ for all $\ell \geq 1$ can be done in $\mathcal{O}(|V(G)|)$ time in total. SPTG_PATH terminates in $\mathcal{O}(\epsilon_G(r) + \sum_{\ell=1}^{\epsilon_G(r)} |L_\ell(G)|)$ time. Since we have $\epsilon_G(r) \in \mathcal{O}(|V(G)|)$ and $\sum_{\ell=1}^{\epsilon_G(r)} |L_\ell(G)| \in \mathcal{O}(|V(G)|)$, SPTG_PATH runs in $\mathcal{O}(|V(G)|)$ time. To prove the correctness, it suffices to show that every intermediate spanning tree T_i of each iteration (Line 9) is a shortest path tree of G rooted at r. We prove this statement through the following claim.

Claim 3. *The resulting tree T_i after the edge flip in Line 9 of SPTG_PATH is a shortest path tree of G rooted at r.*

Proof: We prove this claim by contradiction. Suppose there exists an edge flip in Line 9 of SPTG_PATH such that the resulting tree T_i after the flip is not a shortest path tree of G. Let $k \geq 1$ be the smallest integer such that the k-th edge flip results in a tree T_i that is not a shortest path tree of G.

Prior to the k-th edge flip in Line 9, both T_i and T_j are shortest path trees of G in Line 7 of SPTG_PATH. Let v, u_i, u_j be as defined in Line 7 of SPTG_PATH. Denote the two components of $T_i \setminus u_i v$ by T_i' and T_i''. Since T_i is a shortest path tree of G before the flip, we have $\{u_i, u_j\} \subseteq V(T_i')$ and $v \in V(T_i'')$. Therefore, $(T_i \setminus u_i v) + u_j v$ forms a spanning tree of G. Moreover, because T_i is a shortest path tree before the flip, it follows that $d_{T_i}(r, u_i) = d_{T_i}(r, u_j) = d_G(r, u_i)$ (Definition 1). Consequently, $d_T(r, w) = d_G(r, w)$ for any vertex $w \in V(G)$, where $T = (T_i \setminus u_i v) + u_j v$. This implies that $T = (T_i \setminus u_i v) + u_j v$ is a shortest path tree of G immediately after the k-th edge flip in Line 9, which contradicts the choice of k as the first *wrong* edge flip. Therefore, the resulting tree T_i after any edge flip in Line 9 of SPTG_PATH remains a shortest path tree of G rooted at r. □

The correctness of SPTG_PATH holds due to Claim 3. □

3.4 Connectivity of BFS-Tree Graphs

As seen in Section 2 (Example 1), BFS-tree graphs are not always connected. Therefore, it is natural to characterize graphs G whose corresponding BFS-tree graph is connected. In this section, we present a necessary condition for the connectivity of BFS-tree graphs. Since every BFS-tree graph is trivially connected when there is only one BFS layer, we assume $\epsilon_G(r) \geq 2$ in the remainder of this section.

We begin by making some observations on shortest path trees.

Observation 3. *Let T be any shortest path tree of a graph G rooted at $r \in V(G)$. For any edge $uv \in E(T)$ we must have $u \in L_{i-1}(G, r)$ and $v \in L_i(G, r)$ for some $i > 0$.*

Lemma 7. *Suppose T_1 and T_2 are two shortest path trees of G rooted at r such that $(T_1 \setminus e_1) + e_2 = T_2$ for edges $e_1, e_2 \in E(G)$. Then, e_1 and e_2 must share an endpoint.*

Proof: Let $e_1 = uv$ with $u \in L_{i-1}(G,r)$ and $v \in L_i(G,r)$ for some $i > 0$ (see Observation 3). Let T_1' and T_1'' be the two components of $T_1 \smallsetminus e_1$ with $u \in V(T_1')$ and $v \in V(T_1'')$. Since T_1 is a shortest path tree of G, it follows that T_1'' is the subtree of T_1 rooted at v. Moreover, e_2 must have an endpoint in T_1'' because $(T_1 \smallsetminus e_1) + e_2 = T_2$ is a spanning tree of G. By Observation 3, we deduce that the endpoints of e_2 must be in two consecutive BFS layers $L_{i-1}(G,r)$ and $L_i(G,r)$, since T_2 is a shortest path tree of G rooted at r. We conclude that e_2 must have v as an endpoint. □

For a BFS-tree T_1 and a vertex $v \in V(T_1)$, we denote the set of the BFS children of v in T_1 by S_v. We show the following lemma.

Lemma 8. *Let T_1 and T_2 be two* BFS*-trees of G rooted at r such that $(T_1 \smallsetminus e_1) + e_2 = T_2$ with $e_1, e_2 \in E(G)$. Furthermore, let $e_1 = vz$ and $e_2 = uz$ with $\{v,u\} \subseteq L_{i-1}(G,r)$ and $z \in L_i(G,r)$ for some $i > 0$ (Observation 3 and Lemma 7). Then, $|S_v \cap N_{out}(u)| = 1$, where $S_v = N_{T_1}(v) \cap N_{out}(v)$.*

Proof: Note that $|S_v \cap N_{\text{out}}(u)| \geq 1$ because $z \in S_v \cap N_{\text{out}}(u)$. Therefore, it suffices to show $|S_v \cap N_{\text{out}}(u)| \leq 1$. For the sake of contradiction, suppose $|S_v \cap N_{\text{out}}(u)| \geq 2$ and let $z' \in (S_v \cap N_{\text{out}}(u)) \smallsetminus \{z\}$. By definition, $(T_1 \smallsetminus vz) + uz = T_2$. Therefore, we have $z \in N_{T_2}(u) \cap N_{\text{out}}(v)$ and $z' \in N_{T_2}(v) \cap N_{\text{out}}(u)$. This contradicts the fact that T_2 is a BFS-tree of G. To see why, suppose there is a BFS traversal on G that produces T_2. If the traversal visits v before u, then $z \notin N_{T_2}(u)$, contradicting $z \in N_{T_2}(u) \cap N_{\text{out}}(v)$. Conversely, if the traversal visits u before v, then $z' \notin N_{T_2}(v)$, contradicting $z' \in N_{T_2}(v) \cap N_{\text{out}}(u)$. □

We present the following necessary condition for the connectivity of $\mathsf{BFST}(G,r)$ in terms of the BFS layering of G. To motivate this result, consider the graph G depicted in Figure 2(b), whose BFS-tree graph is not connected. Let $S = L_2(G)$ with $|N_{\text{in}}(S)| = 2$. Notice that for each $u \in N_{\text{in}}(S)$, we have $|S \cap N_{\text{out}}(u)| = 2$. The following result formally generalizes this observation, proving that if a graph G has such a set S, then its corresponding BFS-tree graph is disconnected.

Theorem 5. *Let $\{L_1(G), \ldots, L_d(G)\}$ be the* BFS *layering of G rooted at r, where $d \geq 2$. If $\mathsf{BFST}(G,r)$ is connected, then the following holds:*

$$\forall i \in \{1, \ldots, d-1\},\ \forall v \in L_i(G),\ \forall S \subseteq N_{out}(v), \text{ if } |N_{in}(S)| \geq 2 \text{ then}$$

$$\exists u \in N_{in}(S) \smallsetminus \{v\} \text{ such that } |S \cap N_{out}(u)| = 1$$

Proof: For the sake of contradiction, suppose $\mathsf{BFST}(G,r)$ is connected and the following holds:

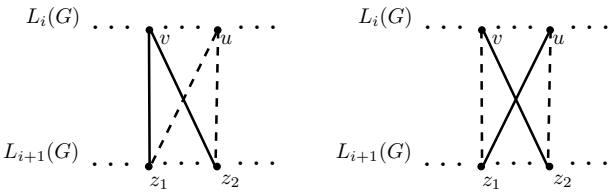
$$\exists i \in \{1, \ldots, d-1\},\ \exists v \in L_i(G),\ \exists S \subseteq N_{\text{out}}(v),\ \text{such that } |N_{\text{in}}(S)| \geq 2$$

$$\text{and } \forall u \in N_{\text{in}}(S) \smallsetminus \{v\},\ |S \cap N_{\text{out}}(u)| \geq 2$$

Let S and v be as defined above. Moreover, let T and T' be two BFS-trees rooted at r in which v is the first and the last visited vertex of layer i in the breadth-first search starting from r, respectively. Observe that $N_{\text{out}}(v) \subseteq N_T(v)$. Since $\mathsf{BFST}(G,r)$ is connected, there must be a path between T and T' in $\mathsf{BFST}(G,r)$. Let $P = T_1, \ldots, T_k$ denote this path with $T_1 = T$, $T_k = T'$, and $T_{j-1}T_j \in E(\mathsf{BFST}(G,r))$ for $j \in \{2, \ldots, k\}$. Let $1 < \ell \leq k$ be the smallest integer such that $(T_{\ell-1} \smallsetminus vz_1) + uz_1 = T_\ell$ with $u \in N_{\text{in}}(S) \smallsetminus \{v\}$ and $z_1 \in S$. By our choice of T and T', Observation 3, Lemma 7, and the fact that $|N_{\text{in}}(S)| \geq 2$, there must exist such an integer ℓ. Since ℓ is the smallest such integer, we have $S \subseteq N_{T_{\ell-1}}(v)$. Furthermore, we have $|S \cap N_{\text{out}}(u)| \geq 2$ by assumption and there exists a vertex $z_2 \in S \smallsetminus \{z_1\}$ with $\{z_1, z_2\} \subseteq N_{\text{out}}(u)$. On the other hand, we have

$$\{z_1, z_2\} \subseteq S \subseteq S_v, \text{ where } S_v = N_{T_{\ell-1}}(v) \cap N_{\text{out}}(v). \tag{1}$$

By (1), we deduce $\{z_1, z_2\} \subseteq S_v \cap N_{\text{out}}(u)$ and $|S_v \cap N_{\text{out}}(u)| \geq 2$. This observation contradicts Lemma 8, see Figure 9. □



(a) $T_{\ell-1}$ (b) T_ℓ

Figure 9: The figure used in the proof of Theorem 5.

We believe the necessary condition of Theorem 5 is also sufficient, see Conjecture 1 in the conclusion.

3.5 Chromatic Number of BFS-Tree graphs

In this section, we present some bounds on the chromatic number of shortest path tree graphs and BFS-tree graphs. A *proper vertex coloring* is a coloring of vertices such that no two adjacent vertices have the same color. The smallest number of colors needed to color a graph G, denoted by $\chi(G)$, is its chromatic number. A *proper edge coloring* of a graph is a coloring of edges such that no two adjacent edges are assigned the same color. Here, two distinct edges are considered to be adjacent when they share an endpoint. The smallest number of colors needed in a proper edge coloring of a graph G is its chromatic index or edge chromatic number. We denote the edge chromatic number of a graph G by $\chi'(G)$.

Using an analogous method to Theorem 2 in [15] and by applying Lemma 7, we show the following bounds. For convenience, we review the parts of the proof of Theorem 2 in [15] that are relevant to our proof.

Proposition 1. $\chi(\mathsf{BFST}(G,r)) \leq \chi(\mathsf{SPT}(G,r)) \leq \chi'(G)$.

Proof: Since every BFS-tree of G is also a shortest path tree of G, we have $\mathsf{BFST}(G,r) \subseteq \mathsf{SPT}(G,r)$ which implies that $\chi(\mathsf{BFST}(G,r)) \leq \chi(\mathsf{SPT}(G,r))$. To see $\chi(\mathsf{SPT}(G,r)) \leq \chi'(G)$, consider a proper edge coloring $c : E(G) \to \{1,\ldots,\chi'(G)\}$ of the graph G. Now, we assign to each node T of $\mathsf{SPT}(G,r)$ the sum of the values of the edges of T modulo $\chi'(G)$. Let T_1 and T_2 be two nodes of $\mathsf{SPT}(G,r)$. If T_1 is adjacent to T_2 in $\mathsf{SPT}(G,r)$, then there are two edges uv and wz in G such that $(T_1 \setminus uv) + wz = T_2$. By Lemma 7, we can assume $z = u$, which implies that $c(uv) \neq c(wz)$. This implies that T_1 and T_2 are assigned different colors, since all other edges of T_1 and T_2 are identical. □

Lemma 9 (Vizing's Theorem [30, 31])**.** *Let G be a graph. Then, $\Delta(G) \leq \chi'(G) \leq \Delta(G)+1$.*

It follows from Vizing's theorem that the chromatic number of shortest path tree graphs is at most the maximum degree plus one.

Corollary 1. $\chi(\mathsf{BFST}(G,r)) \leq \chi(\mathsf{SPT}(G,r)) \leq \Delta(G)+1$

We now present a lower bound on the chromatic number of any shortest path tree graph. Let G be any graph. The *clique number* of G, denoted by $\omega(G)$, is the size of the maximum clique of G. Recall the definition of $N_{\text{in}}(v)$ from Notation 2. We first present the following lower bound on the clique number of any shortest path tree graph.

Proposition 2. *If* $|V(G)| > 1$, *then* $\max_{v \in L} |N_{in}(v)| \leq \omega(\mathsf{SPT}(G, r))$, *where* L *denotes the vertices of the furthest* BFS *layer from* r.

Proof: Let $v \in L$ be any vertex in the last BFS layer of G. Furthermore, let T be any shortest path tree of $G \setminus v$ rooted at r, and let $N_{\text{in}}(v) = \{v_1, \ldots, v_{|N_{\text{in}}(v)|}\}$. For every $v_j \in N_{\text{in}}(v)$, we denote the edge $vv_j \in E(G)$ by e_j. Observe that $T \cup e_1, \ldots, T \cup e_{|N_{\text{in}}(v)|}$ are all shortest path trees of G. Moreover, since these trees all differ by exactly one edge, the subgraph of $\mathsf{SPT}(G, r)$ induced by $\{T \cup e_1, \ldots, T \cup e_{|N_{\text{in}}(v)|}\}$ is a clique of order $|N_{\text{in}}(v)|$. This analysis applies to any $v \in L$, and for any such v, $\mathsf{SPT}(G, r)$ has a clique of order $|N_{\text{in}}(v)|$. Thus, $\max_{v \in L} |N_{\text{in}}(v)| \leq \omega(\mathsf{SPT}(G, r))$. □

Proposition 2 and Corollary 1 yield the following bounds on the chromatic number of any shortest path tree graph because the chromatic number of any graph is lower bounded by its clique number.

Corollary 2. *Let* L *denote the vertices of the furthest* BFS *layer of* G *from* r. *If* $|V(G)| > 1$, *we have*

$$\max_{v \in L} |N_{in}(v)| \leq \chi(\mathsf{SPT}(G, r)) \leq \Delta(G) + 1.$$

4 DFS-Tree Graphs

In this section, we study DFS-tree graphs. As discussed in Section 2, DFS-tree graphs are not necessarily connected (see Example 2). We first show a necessary condition for the connectivity of DFS-tree graphs. We then show that this condition is not sufficient by providing an example. Finally, we prove an upper bound on the chromatic number of any DFS-tree graph.

4.1 Connectivity of DFS-Tree Graphs

In this subsection, we show that if G has specific connectivity conditions, then the connectivity of $\mathsf{DFST}(G, r)$ implies the hamiltonicity of G.

We start with the following lemma, which states that every pair of flippable edges in a DFS-tree must share an endpoint.

Lemma 10. *Suppose* T_1 *and* T_2 *are two nodes of* $\mathsf{DFST}(G, r)$ *such that* $(T_1 \setminus e_1) + e_2 = T_2$. *Let* $e_1 = uv$ *with* $d_{T_1}(r, u) < d_{T_1}(r, v)$. *Then,* e_1 *and* e_2 *must share* u *as an endpoint.*

Proof: Suppose $e_2 = xy$. Since x and y are comparable in T_1 with r as the root, $d_{T_1}(r, x) \neq d_{T_1}(r, y)$. Therefore and without loss of generality, assume $d_{T_1}(r, x) < d_{T_1}(r, y)$.

We show $u = x$. For the sake of contradiction, suppose $u \neq x$. Let T_1' and T_1'' denote the components of $T_1 \setminus e_1$ and without loss of generality, assume that $r \in V(T_1')$. Observe that the edge $e_2 = xy$ belongs to the cut between $V(T_1')$ and $V(T_1'')$ and since x is closer to the root than y, $x \in V(T_1')$. Therefore, we have $\{u, x\} \subseteq V(T_1')$ and $\{v, y\} \subseteq V(T_1'')$. Since xy is a non-tree edge of T_1, x and y must be comparable in T_1 with r as the root. It follows from the properties of T_1' and T_1'' that $S(T_1, x, y)$ goes through u, implying that x lies on $S(T_1, r, u)$. Observe that $S(T_2, r, v)$

goes through x and avoids u. Similarly, $S(T_2, r, u)$ is the same as its counterpart in T_1 and avoids v. Therefore, u and v are not comparable with r as the root in T_2. Since uv is a non-tree edge in T_2, it follows that T_2 cannot be a DFS-tree of G rooted at r, a contradiction. □

Recall that a *branch vertex* in a tree is a vertex of degree at least 3, and that DFS-trees can be rooted at multiple vertices (Definition 3). For any graph H with $u, v \in V(H)$, the *intermediate vertices* of $S(H, u, v)$ are the vertices of $S(H, u, v)$ excluding u and v.

Lemma 11. *Let G be a connected graph with $t_1, t_2 \in V(G)$ and let T be a DFS-tree of G rooted at t_1 and t_2 simultaneously. If z is both an intermediate vertex of $S(T, t_1, t_2)$ and a branch vertex of T, then it is also a cut vertex of G.*

Proof: Assume to the contrary that z is not a cut vertex of G. Since z is a branch vertex of T, $T \setminus z$ is a forest with at least three components. Suppose T_1 and T_2 are two components of $T \setminus z$ with $t_i \in V(T_i)$ for $i \in \{1, 2\}$. Since z is not a cut vertex of G, there exists an edge $e = xy \in (E(G) \setminus E(T))$ such that $y \in V(T_1) \cup V(T_2)$ and $x \in V(T_3)$ for some component T_3 of $T \setminus z$ with $T_3 \notin \{T_1, T_2\}$. Any such edge xy results in a contradiction since x and y would be non-comparable with either t_1 or t_2 as the root. To see why and without loss of generality, suppose $y \in V(T_1)$ and $x \in V(T_3)$. Since $t_2 \in V(T_2)$, $S(T, t_2, x)$ and $S(T, t_2, y)$ go through z and avoid y and x, respectively (see Figure 10 for an illustration). Since xy is a non-tree edge of T, this observation contradicts the fact that T is a DFS-tree of G rooted at t_2. □

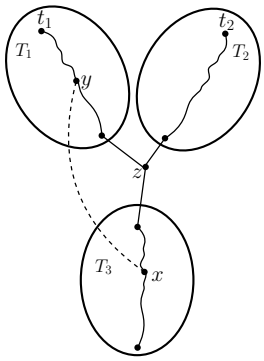


Figure 10: The figure used in the proof of Lemma 11.

The following lemma, used later in the proof of Theorem 6, has appeared in various forms throughout the literature.

Lemma 12. *[1, Lemma 5.5] Let T be a DFS-tree of G rooted at r. If r is not a cut vertex of G, then $\deg_T(r) = 1$.*

Lemma 12 entails the following corollary.

Corollary 3. *Suppose r is not a cut vertex of G and let T be a DFS-tree of G rooted at r. Then, $T \setminus r$ is a DFS-tree of $G \setminus r$ rooted at t, where $rt \in E(T)$.*

Theorem 6. *Suppose $G \setminus r$ is 2-connected and $\deg_G(r) \geq 2$. If $\mathsf{DFST}(G, r)$ is connected, then G is hamiltonian.*

Proof: Let $N_G(r) = \{t_1, \ldots, t_{\deg_G(r)}\}$ and let T be any DFS-tree of G rooted at r. By Lemma 12, $\deg_T(r) = 1$ and T has exactly one edge from $\{rt_1, \ldots, rt_{\deg_G(r)}\}$. We define an equivalence relation $\sim$ on the DFS-trees of G rooted at r. Let T_1 and T_2 be any two nodes of $\mathsf{DFST}(G, r)$. We have $T_1 \sim T_2$ if and only if $N_{T_1}(r) = N_{T_2}(r)$. It is straightforward to see that $\sim$ is an equivalence relation because it is reflexive ($T_1 \sim T_1$ for any DFS-tree T_1 of G), symmetric ($T_1 \sim T_2 \iff T_2 \sim T_1$ for any pair of distinct DFS-trees T_1 and T_2), and transitive (if $T_1 \sim T_2$ and $T_2 \sim T_3$, then $T_1 \sim T_3$ for distinct DFS-trees T_1, T_2, and T_3). Let $V_1, \ldots, V_{\deg_G(r)}$ denote the equivalence classes, such that for each node T of each equivalence class V_j, we have $rt_j \in E(T)$. Since $\mathsf{DFST}(G, r)$ is connected, there must be an edge $T_iT_j \in E(\mathsf{DFST}(G, r))$ with $T_i \in V_i$ and $T_j \in V_j$ for two distinct equivalence classes V_i and V_j. From the definition of DFS-tree graphs, there must be two edges $e_i, e_j \in E(G)$ such that $(T_i \setminus e_i) + e_j = T_j$. By Lemma 10 and the fact that $T_i \in V_i$ and $T_j \in V_j$, e_i and e_j must be incident and both have r as an endpoint. Let $e_i \coloneqq rt_i$ and $e_j \coloneqq rt_j$.

Observation 4. $T_i \setminus r = T_j \setminus r$.

Observation 4 follows from the simple fact that $V(T_i \setminus r) = V(T_j \setminus r)$ with $(T_i \setminus rt_i) + rt_j = T_j$. By Corollary 3, $T_i \setminus r$ and $T_j \setminus r$ are DFS-trees of $G \setminus r$ rooted at t_i and t_j, respectively. By Observation 4, $T_i \setminus r$ is a DFS-tree of $G \setminus r$ rooted at both t_i and t_j. Since $G \setminus r$ has no cut vertices, we observe that every intermediate vertex on $S(T_i \setminus r, t_i, t_j)$ is of degree 2 in $T_i \setminus r$. Moreover, by Lemma 12 we conclude that both t_i and t_j are of degree 1 in $T_i \setminus r$. Since $T_i \setminus r$ is a spanning tree of $G \setminus r$, it follows that $S(T_i \setminus r, t_i, t_j)$ is a hamiltonian path of $G \setminus r$, and by adding the edges rt_i and rt_j into this path we get a hamiltonian cycle of G. □

Example 4. The necessary condition of Theorem 6 for the connectivity of DFS-tree graphs is not sufficient. As depicted in Figure 11, there exists a hamiltonian graph G and a vertex $r \in V(G)$ with $\deg_G(r) \geq 2$ such that $G \setminus r$ is 2-connected but $\mathsf{DFST}(G, r)$ is not connected.

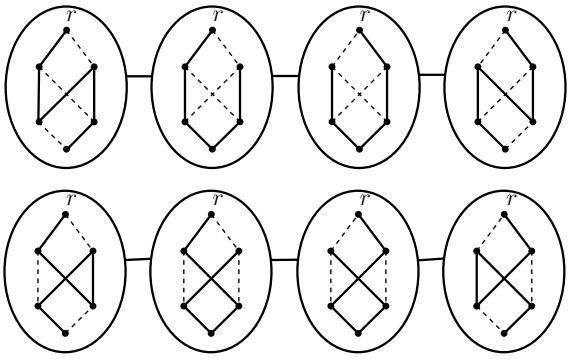


Figure 11: The counterexample described in Example 4.

4.2 Chromatic Number of DFS-tree graphs

This subsection presents an upper bound on the chromatic number of any DFS-tree graph. The upper bound is described in Proposition 3 and Corollary 4. We skip the proofs as they are identical to those of Proposition 1 and Corollary 1, respectively (see Lemma 10).

Proposition 3. $\chi(\mathsf{DFST}(G, r)) \leq \chi'(G)$.

Corollary 4. $\chi(\mathsf{DFST}(G, r)) \leq \Delta(G) + 1$.

5 Summary and Open Problems

In this paper, we introduced three variants of tree graphs, namely shortest path tree graphs, BFS-tree graphs, and DFS-tree graphs. We studied several properties of these graphs, including hamiltonicity, connectivity, and chromatic numbers.

Table 1: A summary of existing results for the hamiltonicity of the different variants of tree graphs

Type of Tree Graph	**Hamiltonicity**	**Ref.**
Tree graphs	✓	[10]
$\mathsf{SPT}(G, r)$	✓	Theorem 1
$\mathsf{BFST}(G, r)$	✗	Example 1
$\mathsf{DFST}(G, r)$	✗	Example 4

Several problems are left open. Let G be any graph with $r \in V(G)$.

1. The flip graph on triangulations is a graph where the vertices correspond to non-isomorphic combinatorial triangulations, and edges connect pairs of triangulations that can be transformed into each other by flipping a single edge. Frati, in [19], showed that the diameter of the flip graph is at least $\frac{7}{3}n + \Theta(1)$. Are there bounds on the diameters of $\mathsf{SPT}(G, r)$ and also $\mathsf{DFST}(G, r)$?

2. Kaneko and Yoshimoto in [23] proved that if G is 2-connected with minimum degree δ, then the leaf graph of G is $(2\delta - 2)$-connected. A natural question that follows is whether there exists a function f such that if G is k-connected with minimum degree δ, then $\mathsf{SPT}(G, r)$ (or $\mathsf{DFST}(G, r)$) is $f(k, \delta)$-connected.

3. We have observed that BFS-tree graphs are not always connected. Can the connectedness issue in BFS-tree graphs be resolved by increasing the vertex or edge connectivity of G? Another approach to ensure the connectedness of $\mathsf{BFST}(G, r)$ is to relax the definition of the BFS-tree graph. We suggest the following definition.

   Definition 5. Let G be a connected graph with a root r and an integer $k \geq 1$. The k-BFS-*tree graph* of G with reference to r (denoted by k-$\mathsf{BFST}(G, r)$) is a graph whose vertices are the set of all possible BFS-trees with the root r, and two distinct vertices T_1 and T_2 are adjacent if $T_1 \setminus (E(T_1) \setminus E(T_2)) \cup (E(T_2) \setminus E(T_1)) = T_2$ with $|E(T_1) \setminus E(T_2)| = |E(T_2) \setminus E(T_1)| \leq k$.

   We now pose the following question: What is the minimum number k such that the k-BFS-tree graph of G is hamiltonian? It turns out that the $\Delta(G)$-BFS-tree graph of G is indeed hamiltonian.

4. Since BFS-tree graphs are not necessarily connected, we presented a necessary condition for their connectivity in Theorem 5. However, we were unable to find counterexamples for the sufficiency of this condition. We believe this necessary condition is also sufficient and conjecture the following.

   Conjecture 1. Let G be a graph with $r \in V(G)$. If G satisfies the conditions of Theorem 5, then $\mathsf{BFST}(G, r)$ is connected.

Much of our initial research efforts for this paper were dedicated to proving Conjecture 1, which unfortunately ended up being unsuccessful.

5. Can the connectedness of $\mathsf{DFST}(G, r)$ be characterized using a finite set of forbidden minors?

6. In this paper, we fix a root r of a graph G and consider all BFS-trees of G rooted at r. If we extend this to allow all BFS-trees of G with respect to different roots and define a tree graph based on these BFS-trees, is the resulting tree graph connected? Furthermore, is it hamiltonian? The same question can be posed when we consider all DFS-trees with respect to all possible roots.

Acknowledgments

We would like to thank the referees for careful reading and suggestions to improve the paper. We also thank Pat Morin for insightful discussions during this work's preparation.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[2] O. Aichholzer and K. Reinhardt. A quadratic distance bound on sliding between crossing-free spanning trees. *Comput. Geom.*, 37(3):155–161, 2007. doi:10.1016/j.comgeo.2004.12.010.

[3] J. Asplund, K. D. Edoh, R. Haas, Y. Hristova, B. Novick, and B. Werner. Reconfiguration graphs of shortest paths. *Discret. Math.*, 341(10):2938–2948, 2018. URL: https://doi.org/10.1016/j.disc.2018.07.007, doi:10.1016/J.DISC.2018.07.007.

[4] N. Behrooznia and T. Mütze. Listing spanning trees of outerplanar graphs by pivot exchanges. *arXiv preprint arXiv:2409.15793*, 2024.

[5] H. B. Bjerkevik, L. Kleist, T. Ueckerdt, and B. Vogtenhuber. Flipping non-crossing spanning trees. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2313–2325. SIAM, 2025. doi:doi.org/10.1137/1.9781611978322.77.

[6] J. A. Bondy and U. S. R. Murty. *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer, New York, 2008. doi:10.1007/978-1-84628-970-5.

[7] N. Bousquet, T. Ito, Y. Kobayashi, H. Mizuta, P. Ouvrard, A. Suzuki, and K. Wasa. Reconfiguration of spanning trees with degree constraints or diameter constraints. *Algorithmica*, 85(9):2779–2816, 2023. URL: https://doi.org/10.1007/s00453-023-01117-z, doi:10.1007/S00453-023-01117-Z.

[8] B. Cameron, A. Grubb, and J. Sawada. Pivot gray codes for the spanning trees of a graph ft. the fan. *Graphs and Combinatorics*, 40(4):78, 2024. doi:10.1007/s00373-024-02808-2.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, third edition, 2009.

[10] R. L. Cummins. Hamilton circuits in tree graphs. *IEEE Trans. Circuit Theory*, CT-13:82–90, 1966. URL: https://doi.org/10.1109/tct.1966.1082546, doi:10.1109/TCT.1966.1082546.

[11] H. de Fraysseix and P. O. de Mendez. Trémaux trees and planarity. *European Journal of Combinatorics*, 33(3):279–293, 2012. doi:10.1016/j.ejc.2011.09.012.

[12] E. D. Demaine, D. Eppstein, A. Hesterberg, K. Jain, A. Lubiw, R. Uehara, and Y. Uno. Reconfiguring undirected paths. In *Algorithms and data structures*, volume 11646 of *Lecture Notes in Comput. Sci.*, pages 353–365. Springer, Cham, 2019. URL: https://doi.org/10.1007/978-3-030-24766-9_26, doi:10.1007/978-3-030-24766-9_26.

[13] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Berlin, Heidelberg, 6 edition, 2025. doi:10.1007/978-3-662-70107-2.

[14] R. Diestel and I. Leader. Normal spanning trees, aronszajn trees and excluded minors. *Journal of the London Mathematical Society*, 63(1):16–32, 2001.

[15] V. Estivill-Castro, M. Noy, and J. Urrutia. On the chromatic number of tree graphs. *Discrete Math.*, 223(1-3):363–366, 2000. doi:10.1016/S0012-365X(00)00092-3.

[16] S. Even. *Graph algorithms*. Cambridge University Press, 2011. doi:10.1017/cbo9781139015165.

[17] R. Fabila-Monroy, D. Flores-Penaloza, C. Huemer, F. Hurtado, J. Urrutia, and D. R. Wood. On the chromatic number of some flip graphs. *Discrete Mathematics & Theoretical Computer Science*, 11(Graph and Algorithms), 2009. doi:10.46298/dmtcs.460.

[18] M. F. Foregger. Hamiltonian decompositions of products of cycles. *Discrete Mathematics*, 24(3):251–260, 1978. URL: https://www.sciencedirect.com/science/article/pii/0012365X78900961, doi:10.1016/0012-365X(78)90096-1.

[19] F. Frati. A lower bound on the diameter of the flip graph. *Electron. J. Combin.*, 24(1):Paper No. 1.43, 6, 2017. doi:10.37236/5489.

[20] C. A. Holzmann and F. Harary. On the tree graph of a matroid. *SIAM J. Appl. Math.*, 22:187–193, 1972. doi:10.1137/0122021.

[21] T. Ito, E. D. Demaine, N. J. A. Harvey, C. H. Papadimitriou, M. Sideri, R. Uehara, and Y. Uno. On the complexity of reconfiguration problems. *Theor. Comput. Sci.*, 412(12-14):1054–1065, 2011. URL: https://doi.org/10.1016/j.tcs.2010.12.005, doi:10.1016/J.TCS.2010.12.005.

[22] T. Kamae. The existence of a Hamilton circuit in a tree graph. *IEEE Trans. Circuit Theory*, CT-14:279–283, 1967. doi:10.1109/tct.1967.1082707.

[23] A. Kaneko and K. Yoshimoto. The connectivities of leaf graphs of 2-connected graphs. *J. Combin. Theory Ser. B*, 76(2):155–169, 1999. doi:10.1006/jctb.1998.1895.

[24] G. Kishi and Y. Kajitani. On hamilton circuits in tree graphs. *IEEE Transactions on Circuit Theory*, 15(1):42–50, 1968. doi:10.1109/tct.1968.1082762.

[25] L. Kleist, P. Kramer, and C. Rieck. On the connectivity of the flip graph of plane spanning paths. In *Graph-Theoretic Concepts in Computer Science: 50th International Workshop, WG 2024, Gozd Martuljek, Slovenia, June 19–21, 2024, Revised Selected Papers*, page 327. Springer Nature, 2024. doi:10.1007/978-3-031-75409-8_23.

[26] G. Liu. On connectivities of tree graphs. *Journal of Graph Theory*, 12(3):453–459, 1988. doi:10.1002/jgt.3190120318.

[27] M. Messinger and A. Porter. Eulerian k-dominating reconfiguration graphs. *Discrete Mathematics & Theoretical Computer Science*, 27(Graph Theory), 2025. doi:10.46298/dmtcs.13438.

[28] A. E. Mouawad, N. Nishimura, V. Raman, N. Simjour, and A. Suzuki. On the parameterized complexity of reconfiguration problems. *Algorithmica*, 78(1):274–297, 2017. URL: https://doi.org/10.1007/s00453-016-0159-2, doi:10.1007/S00453-016-0159-2.

[29] H. Shank. A note on hamilton circuits in tree graphs. *IEEE Transactions on Circuit Theory*, 15(1):86–86, 1968. doi:10.1109/tct.1968.1082765.

[30] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz*, 3:25–30, 1964. In Russian.

[31] V. G. Vizing. The chromatic class of a multigraph. *Cybernetics*, 1(3):32–41, 1965.