DFS TREE CONSTRUCTION
ALGORITHMS and CHARACTERIZATIONS
(Detailed Abstract)

by

Ephraim Korach and Zvi Ostfeld

Technical Report #470

November 1987
DFS TREE CONSTRUCTION: ALGORITHMS and CHARACTERIZATIONS

(Detailed abstract)

Ephraim Korach and Zvi Ostfeld

Computer Science Department
Technion - Israel Institute of Technology
Haifa 32000, Israel

ABSTRACT

The Depth First Search (DFS) algorithm is one of the basic techniques which is used in a very large variety of graph algorithms. Every application of the DFS involves, beside traversing the graph, constructing a special structured tree, called a "DFS tree". In this paper, we give a complete characterization of all the graphs in which every spanning tree is a DFS tree. These graphs are called Total DFS Graphs. The characterization we present shows that the family of Total DFS Graphs is quite limited, and therefore the following question is naturally raised: Given a graph G and a spanning tree T, is T a DFS tree of G? We give an algorithm to answer this question in linear time. Next we characterize all graphs in which every DFS tree is a directed path. Finally, we focus on the problem of determining the highest degree that a given vertex may have in a DFS tree in G. The decision problem is: Given a vertex v in a graph G, is there a DFS tree in which v has degree greater than k? We show that this problem is NP-Complete. This last question is important for distributed and for parallel algorithms, where the degree of a vertex in a DFS tree obtained in a certain run might be too high for the vertex to handle because of technical limitations, e.g. there may be a limited number of communication ports to be used in the tree.
1. INTRODUCTION

The Depth First Search (DFS) algorithm is one of the basic techniques which is used in a very large variety of graph algorithms. The history of this algorithm (in a different form) goes back to 1882 when Tremaux' algorithm for the maze problem was first published (see [3, p. 18]).

The impact of DFS grew rapidly since its publication by Tarjan [12]. Notable are also the work of Hopcroft and Tarjan [5], [6] and [7].

In many areas of computer science, this algorithm is used, and lately it also has penetrated the field of parallel and distributed algorithms (e.g. [1], [2], [9] and [11]).

Every use of the DFS, beside traversing the graph, constructs a special structured directed rooted tree, called a "DFS tree", that may be used subsequently.

In this paper, we raise several important questions, regarding the structure of the DFS tree that is obtained, and discuss their solutions. First, we are interested in knowing in which graphs, every spanning tree can be obtained as a DFS tree. In section 2 we give a complete characterization of all the graphs in which every spanning tree is a DFS tree. Such a graph is called a Total DFS Graph (T-DFS-G).

It turns out that the family of Total DFS Graphs is quite limited, and therefore the following question is naturally raised: Given a graph $G$ and a spanning tree $T$, is $T$ a DFS tree of $G$?

This question is answered by a linear time algorithm in section 3.

In section 4 we characterize all graphs in which every DFS tree is a directed path.

Finally, we focus on the problem of determining the highest degree that a given vertex may have in a DFS tree in $G$. The decision question problem is: Given a vertex $v$ in a graph $G$, is there a DFS tree in which $v$ has degree greater than $k$? We show that this problem is NP-Complete. This last question is probably important for distributed and for parallel algorithms, where the degree of a vertex in a DFS tree obtained in a certain run might be too high for the vertex to handle because of technical limitations (e.g. there may be a limited number of communication ports to be used in the tree).

One can think about many other applications of the results presented here. For example, we can solve the following problem: Let $G$ be a graph with a unique minimum spanning tree $T$. We would like to run a DFS in such a way that $T$ will be the DFS tree.

Extensions to the directed case appear in [8].
2. TOTAL DFS GRAPHS

Definition: An undirected connected graph $G$ is called Total DFS Graph (T-DFS-G) if every spanning tree in $G$ is a DFS tree.

Definition: A graph $G$ is called $k$-parallel-path graph if the edges of $G$ can be partitioned into $k$ internally vertex disjoint paths between two vertices.

A $k$-parallel-path graph $G$ is completely defined by a positive integral vector $l = (l_1, \ldots, l_k)$ of length $k$, where $l_i$ is the length of the $i$-th path in $G$ (the number of edges in it).

Notation: A $k$-parallel-path graph with a vector $l = (l_1, \ldots, l_k)$ is denoted by $PPG(l_1, \ldots, l_k)$ where $l_i \geq l_j$ for $i < j$.

Theorem 2.1 (Characterization of Total-DFS-Graphs): A connected simple graph $G$ is a T-DFS-G if and only if $G$ has at most one non-separable component $C$ with at least 3 vertices and $C$ is one of the following $k$-parallel-path graphs:

$$PPG(x, y) \quad x \geq 2, y \geq 1$$
$$PPG(x, 2, 1), PPG(x, 2, 2) \quad x \geq 2$$
$$PPG(2, 2, 1), PPG(2, 2, 2)$$

Next we give the proof of Theorem 2.1. This is the only proof presented. It suppose to illustrate some of the techniques we use. The reader who is not interested in the proof can move immediately to the next section on page 6.

In order to prove the theorem we need the lemmas and propositions stated and proved below.

Lemma 2.2: Let $G$ be a Total DFS graph (T-DFS-G). Then there are two vertices which are in all circuits of $G$.

Definitions: Let $T$ be a spanning tree in $G = (V, E)$ and let $s \in V$. Let $T_s$ be the tree $T$ with an orientation that makes $s$ the root of $T_s$. If there is a directed path in $T_s$ from $a$ to $b$, we say that $a$ is an ancestor of $b$ and $b$ is a descendant of $a$.

Property (PDFS): $T_s$ has the property PDFS if for every edge $(a, b) \in E$ either $a$ is an ancestor of $b$ or a descendant of $b$.

DFS orientation: Let $T$ be a spanning tree in $G = (V, E)$ and let $s \in V$. Let $T_s$ be the tree $T$ with an orientation that makes $s$ the root of $T_s$. If $T_s$ has the property PDFS, then the orientation given to $T$ is called DFS orientation of $T$. 

Technion - Computer Science Department - Technical Report CS0470 - 1987
Proposition 2.3: Let $T$ be a spanning tree of $G$ and let $s \in V$, then $T_s$ is a DFS tree of $G$ if and only if $T_s$ has the property PDFS.

Proof: (\(\Rightarrow\)) see [4, p. 57].

($\Leftarrow$) Let us assume that the edges of $G$ are labeled by numbers from 1 to $|E|$ and w.l.o.g. assume that the tree edges are labeled from 1 to $|V| - 1$. Consider the DFS algorithm with the additional freedom braking rule: "whenever we choose an unused edge we choose the unused one with the smallest possible label".

We shall prove by induction on $|E|$ that

Claim : If $T_s$ is a spanning tree in a given $G$ that has the property PDFS then the above modified DFS algorithm will give $T_s$ provided that we start from $s$.

Clearly, since $G$ contains $T_s$ then $|E| \geq |V| - 1$, and obviously, if $|E| = |V| - 1$ then the modified DFS algorithm (M-DFS) starting from $s$ will give $T_s$, i.e., the claim is true for every graph with $|E| = |V| - 1$.

Assume that the claim is true for all connected graphs with $|E| = M \geq |V| - 1$ and assume $G$ has $|E| = M + 1$. Let $G' = G \setminus e_{M+1}$. By the induction hypothesis, the M-DFS algorithm gives $T_s$ as its DFS tree. Assume $e_{M+1} = (u, v)$ and assume w.l.o.g. that $u$ is an ancestor of $v$.

Consider a run of the M-DFS in $G$ and consider the first time the edge $e_{M+1}$ is used. Until that moment, the run is identified to the run in $G$. If it is used from $v$ then, since $u$ is an ancestor of $v$, it has already been discovered. $e_{M+1}$ is a back edge and the rest of the execution is the same as in $G'$. This implies that the DFS tree will be the same as in $G'$, i.e. $T_s$.

Otherwise, $e_{M+1}$ is used from $u$. At this moment, all other edges of $u$ are already used and in particular, all the edges that lead to the sons of $u$ in $T_s$ are used. It is an easy observation that in this case all the descendants of $u$ have already been discovered and all their edges are used. Therefore, $e_{M+1}$ is already used and this case is not possible. This completes the proof of the claim and hence the proof of the proposition. \(\square\)

Proposition 2.4: Let $T$ be the underlying undirected tree of any DFS tree in a graph $G$ and let $(u, v) = e \in E(G) - E(T)$ be an edge between two leaves of $T$. Then either $u$ is the root of the DFS tree or $v$ is the root.

Proof: Clearly in the DFS tree, the in-degree of every node except for the root is equal to 1 and the in-degree of the root is zero. $e$ is an edge between a node and one of its descendants, and w.l.o.g. assume that $v$ is a descendant of $u$.

Hence there is a directed path from $u$ to $v$, and since $u$ is a leaf in $T$ it has only one edge directed out of it. Therefore $u$ is a root. \(\square\)
Proposition 2.5: If $G$ is a T-DFS-G then every connected subgraph of $G$ is a Total DFS graph.

Proof: Let $G'$ be a connected subgraph of $G$ and let $T'$ be any spanning tree in $G'$. Clearly we can extend $T'$ to a spanning tree $T$ of $G$. Since $G$ is T-DFS-$G$ then $T$ is a DFS tree and hence $T$ together with the orientation induced by the DFS run has the property PDFS in $G$.

Let $OT$ be the tree $T$ together with the orientation induced by the DFS run. Then $OT$ has the property PDFS in $G$.

Let $OT'$ be the tree $T'$ with the orientation induced by $OT$. Clearly $OT'$ also has the property PDFS relative to $G'$.

Proposition 2.6: Every two circuits in a Total DFS graph have at least two vertices in common.

Proof: Let $C_1 = (v_1,v_2,...,v_k)$ and $C_2 = (u_1,u_2,...,u_m)$ be any two circuits in $G$.

Since $G$ is a simple graph then $m, k \geq 3$.

Let $\pi$ be the shortest path connecting $C_1$ and $C_2$ and w.l.o.g. assume that the ends of $\pi$ are $v_1$ and $u_1$ (possibly $v_1 = u_1$).

Assume, in contradiction, that $|C_1 \cap C_2| \leq 1$ and consider the subgraph $G' = C_1 \cup C_2 \cup \pi$.

Let $C_1', C_2'$ be the paths obtained from $C_1, C_2$ by deleting the edges $(v_2,v_3), (u_2,u_3)$, respectively.

Let $T' = \pi \cup C_1' \cup C_2'$. Clearly $T'$ has 4 leaves $v_2, v_3, u_2, u_3$ and by Proposition 2.4 one of $v_2$ and $v_3$ must be the root of the DFS tree and also one of $u_2$ and $u_3$ must be the root of the DFS tree which is impossible. Therefore $T'$ is not a DFS tree in $G'$ which implies that $G'$ is not a T-DFS-$G$ and therefore by Proposition 2.5 $G$ is not a DFS tree in contradiction to the assumption. □

Proposition 2.7: If $G$ is a homeomorph of $K_4$ then $G$ is not a Total DFS graph.

Proof: Let $v_1, v_2, v_3$ and $v_4$ be the vertices of degree 3 in $G$. Let $\pi_{ij}, 1 \leq i, j \leq 4$ be the six paths between $v_i$ and $v_j$ homeomorphic to the edges of the $K_4$. Let $l_{12}, l_{23},$ and $l_{13}$ be the edges on $\pi_{12}, \pi_{23}$ and $\pi_{13}$, respectively. Clearly, $G-\{l_{12}, l_{13}, l_{23}\}$ is a spanning tree in $G$ which is not a T-DFS. □

Corollary 2.8: A Total DFS graph does not contain a subgraph homeomorphic to $K_4$. □

Proposition 2.9: Let $G$ be a T-DFS-$G$ with at least 2 circuits, and assume that $a, b \in V$ are contained in all circuits of $G$. Then there are 3 paths, pairwise internally disjoint between $a$ and $b$ in $G$.

Proof: Let $C_1$ and $C_2$ be any two different circuits in $G$. Then $C_1 \Delta C_2$ is a collection of circuits in $G$. If it is a single circuit $c$ then since $a, b \in C$ it contains two disjoint paths between $a$ and $b$. Clearly $C_1 \cap C_2$ is another path between $a$ and $b$ and these 3 paths are pairwise internally disjoint. If $C_1 \Delta C_2$ is more than one circuit then since all the circuits contain both $a$ and $b$, there are at least 3 paths between $a$ and $b$ which are pairwise internally dis-
Proof of Lemma 2.1: By induction on the dimension of the cycle space (see [13]) of $G$.

Let $G$ be a T-DFS-G. Clearly if the cycle space of $G$ is of dimension 2 then by Proposition 2.6 the lemma is true. Assume it is true for all T-DFS-G with cycle space of dimension $k \geq 2$ and for contradiction let us assume that it is not true for a given T-DFS-G with cycle space of dimension $k+1$. Let $T$ be any spanning tree in $G$ and let $e \in E(G) - E(T)$. Then $G - e$ is also a T-DFS-G (by Proposition 2.5) and has cycle space of dimension $k \geq 2$. By the induction hypothesis, there are two vertices $a$ and $b$ which are common to all circuits of $G - e$. By Proposition 2.9 there are 3 paths between $a$ and $b$ which are pairwise internally disjoint, say, $\pi_1$, $\pi_2$ and $\pi_3$. Since we assumed that the claim of the lemma is not true for $G$, there is a circuit $C_e$ containing $e$ that does not contain both $a$ and $b$.

Let $C_1 = \pi_1 \cup \pi_2$, then by Proposition 2.6, $C_e$ and $C_1$ has at least two vertices in common and w.l.o.g. we may assume that $C_e$ has a vertex $x$ which is internal to $\pi_1$.

Let $C_2 = \pi_2 \cup \pi_3$, then by similar arguments we have that $C_e$ has a vertex $y$ internal to $\pi_2$ or to $\pi_3$.

Let $C_e$ be the part of $C_e$ between $x$ and $y$ that does not contain $b$, and let $C''_e$ be a minimal subpath of $C_e$ that has two ends in two out of $\{\pi_1, \pi_2, \pi_3\}$. Clearly, $C''_e$ is internally disjoint from all the 3 paths $\pi_1$, $\pi_2$, and $\pi_3$, and its ends are on the interior of two of these 3 paths, w.l.o.g. assume that it starts with $s$ on $\pi_1$, ends in $t$ in $\pi_2$ and avoids $\pi_3$.

Then $\pi_1 \cup \pi_2 \cup \pi_3 \cup C''_e$ form an homeomorph of $K_4$ (where $a, b, s, t$ are the 4 points of the $K_4$) contradicting Corollary 2.8. This completes the proof of Lemma 2.2.

Lemma 2.10: The complete set of (simple) $k$-parallel-path graphs which are total DFS graph is the following:

1. all $k$-parallel-path graphs for $k = 1, 2$;
2. for $k = 3$: $PPG(x,2,1), PPG(x,2,2); x \geq 2$
3. for $k = 4$: $PPG(2,2,2,1), PPG(2,2,2,2)$.

Proof: It is easy to check that all graphs in the above list are T-DFS-G. We show that no other graph can be added to the list.

Claim a: for $k \geq 5$ there is no T-DFS-G.

Proof: By Proposition 2.5 it is enough to prove only for $k = 5$. Since there are no parallel edges then there are 4 paths, say $\pi_1$, $\pi_2$, $\pi_3$ and $\pi_4$ of length $\geq 2$. Let $a, b \in V$ be the end vertices of the paths in $G$. Let $l_1 \in \pi_1$ and $l_2 \in \pi_2$ be the edges with end in $a$ and let $l_3 \in \pi_3$, $l_4 \in \pi_4$ be the edges with end in $b$. Clearly, the tree $G - \{l_1, l_2, l_3, l_4\}$ is not a DFS tree in $G$. □
Claim b: For \( k = 3 \) every PPG\((x, y, z)\) where \( x, y \geq 3, z \geq 1 \) is not a T-DFS-G.

Proof: Since there are two paths say \( \pi_1 \) and \( \pi_2 \) of length \( \geq 3 \), there are two edges \( l_1 \in \pi_1 \) and \( l_2 \in \pi_2 \) that are both between vertices of degree 2 and by Proposition 2.4 the tree \( G-\{l_1, l_2\} \) is not a T-DFS.-

Claim c: For \( k = 4 \) we have to consider the following case:

PPG\((xyz)\) \( x \geq 3, y, z \geq 2, \ i \geq 1 \).

Let \( \pi_a, \pi_y, \pi_z, \pi_i \) be the corresponding paths of \( G \) between \( a,b \in V \) and let \( l_1 \in \pi_a \) be an edge between two vertices of degree 2 in \( \pi_a \), let \( l_2 \in \pi_y \) have \( a \) as an end and let \( l_3 \in \pi_z \) have \( b \) as an end.

One can show that the tree \( G-\{l_1, l_2, l_3\} \) is not a T-DFS.-

It is easy to see that a \( k \)-parallel-path graph that does not belong to the set in the statement of the lemma is eliminated from being a T-DFS-G by one of the claims.-

Proof of the Theorem: Clearly every tree is a T-DFS-G.

By Lemma 2.2 if \( G \) has more than one nonseparable component with at least 3 vertices in each then \( G \) is not a T-DFS-G.

If \( G \) has only one nonseparable component \( C \) with at least 3 vertices then by Lemma 2.10 and Proposition 2.5 if \( C \) is not one of the list in the statement of the theorem then \( G \) is not a T-DFS-G.

One can see that if \( G' \) is a T-DFS-G and if \( G'' \) is obtained from \( G' \) by adding an edge \( (u,v) \) such that \( u \in V(G'), v \notin V(G') \) then \( G'' \) is also a T-DFS-G. Therefore if \( C \) is one of the list in the statement of the theorem then \( G \) is obtained from it by adding edges as it is described above and therefore \( G \) is a T-DFS-G.-

3. DFS ORIENTATION of a TREE

By Theorem 1, the class of Total-DFS graphs is quite restricted and therefore the question: "Given a graph \( G \) and a spanning tree \( T \) in \( G \), is \( T \) a DFS tree of \( G \)?" becomes important. In this section we solve this question by presenting a linear time (\( O(|E|) \)) algorithm which gets as an input the pair \( (G,T) \) and gives as an output a root \( r \) for the DFS run if \( T \) is a DFS tree of \( G \) (i.e. a DFS orientation of \( T \)) or else answer that \( T \) is not a DFS tree in \( G \).

Algorithm DECIDE

input: A graph \( G \) and a spanning tree \( T \) in \( G \).

output: Decision whether \( T \) is a DFS tree in \( G \), and if yes, a DFS orientation of \( T \).

The algorithm consists of the following four main phases:
Phase 1: Decomposition of \( G \) into its 2-connected components using Tarjan's algorithm [12] and constructing the related super-structure \( S(G) \) (see [4], [10]).

Phase 2: For each 2-connected component \( G_i \) of \( G \) with \( T_i \), the induced subtree of \( T \) in \( G_i \), we solve algorithmically whether \( T_i \) is a DFS tree of \( G_i \) and in case the answer is yes, we get all (1 or 2) possible roots of a DFS run that gives \( T_i \) in \( G_i \). This phase consists of the following main steps:

1) If \( T_i \) is a path then its ends are the 2 possible roots.

2) If \( T_i \) is not a path then there is at most one node which is a candidate for being the root, and this node must be a leaf. We choose an arbitrary leaf \( s \) and check whether \( T_i \) has the property PDFS. If yes, we are done.

3) We range label \( T \) from \( s \) (see algorithm "Range labeling" below).

4) Starting from \( s \) we search for another leaf, say \( r \), to be a candidate for a root (see algorithm SEARCH, below).

5) Checking whether \( T_i \) has the property PDFS. If no, \( T \) is not a DFS tree of \( G \).

Phase 3: Orientation of \( S(G) \) according to phase 2 as follow:

The vertices of \( S(G) \) are partitioned into 2 groups \( A \) and \( C \) where \( A \) is the set of all the articulation points and \( C \) is the set of all vertices that represent the 2-connected components of \( G \).

Let \( c_i \) be a 2-connected component of \( G \) then every edge \((c_i, a_j) \in S(G) \) where \( a_j \in c_i \) is directed from \( c_i \) to \( a_j \). If \( a_m \) is a DFS root of \( T_i \) in \( c_i \) we direct \((a_m, c_i) \) also from \( a_m \) to \( c_i \) (i.e. \((a_m, c_i) \) is bidirected).

Phase 4: The decision is made according to Lemma 3.1 below. If \( c_i \) is the root of \( S(G) \) then the root of \( T_i \) is the output (i.e. the root of a DFS run in \( G \) that gives \( T \)).

Lemma 3.1: \( T \) is a DFS tree of \( G \) if and only if the oriented super-structure has a root in \( C \). \( \Box \)
Algorithm Range Labeling

DFS-labeling: A tree is DFS-labeled if the nodes of the tree are labeled according to a DFS run on the tree.

range-labeling of the edges of a DFS-labeled tree: every edge \( e = (v, w) \) is labeled by \( R(e) = (\min(e), \max(e)) \).

This is the range of the labels of all nodes that are descendant of \( v \) (including \( v \)).

observation: This range contains exactly all nodes that are labeled by all integral points of the range.

Input: a tree \( T \) and a node \( s \in T \).

Output: DFS labeling together with range-labeling of the edges.

1. Mark all the edges "unused". Let \( i := 0 \) and \( v := s \).
2. \( i := i + 1, k(v) := i \).
3. If \( v \) has no unused incident edges, go to Step (6).
4. Choose an unused incident edge \( e = (u, v) \). Mark \( e \) "used". \( f(u) := v, v := u, \min(e = (v, u)) := i + 1 \).
5. go to Step (2).
6. If \( k(v) = 1 \), halt.
7. \( \max(e = (f(v), v)) := i, v := f(v) \) and go to Step (3).

(The above algorithm is taken from [4 page 56] endowed with edges labeling by "min" and "max").

Algorithm SEARCH

The algorithm searches for a root candidate.

Input: 2-connected graph \( G \), Range-labeled spanning tree \( T \) in \( G \), a leaf \( s \) where \( k(s) = 1 \) and \( T_s \) does not have the property PDFS.

Output: \( r \in V \) (the only left candidate for being the root of the DFS tree).

1. Mark all the edges "unused".
2. \( \min - root := 2, \max - root := n, v := s \)
3. while there is an unused non tree edge \( e = (u, v) \) do
   begin:
     mark \( e \) "used".
     if \( \min - root \leq k(u) \leq \max - root \)
     then \( \min - root := \min(u, f(u)), \max - root := \max(u, f(u)) \).
   end
(4) let $e = (v, u)$ be the unused tree edge (in this case, only one is left). $v := u$, mark $e$ "used".

(5) if $d_T(v) = 2$ go to step (3). ($d_T(v)$ is the degree of $v$ in the tree $T$)

(6) if $d_T(v) = 1$ then output $v$ and Stop. (The leaf $v$ is the only candidate to become the root).

(7) if $d_T(v) > 2$ find an unused tree edge $e'$ incident to $v$ such that $min(e') \leq min-root$ and $max(e') \geq max-root$

   (there is exactly one edge).

(8) while there is an unused tree edge $e = (v, u) \neq e'$

   begin: mark $e$ "used", perform SIDESEARCH$(u)$. end

(9) go to step (3).

SIDESEARCH $(s)$

(1) $v := s$.

(2) If $v$ has no unused incident edges, go to Step (6).

(3) Choose an unused incident edge $e = (u, v)$. Mark $e$ "used".

   If $e$ is a tree edge $(f(u) := v)$ then $v := u$ and go to Step (2).

(4) ($e = (v, u)$ is a back edge)

   if $min-root \leq k(u) \leq max-root$

   then $min-root := min(u, f(u)), max-root := max(u, f(u))$

(5) go to Step (2)

(6) if $v = s$ RETURN.

(7) $v := f(v)$ and go to Step (2).

Theorem 3.2: Algorithm DECIDE is correct and has linear ($O(|E|)$) time complexity.

4. HAMILTONIAN DFS GRAPHS

Definition: An undirected connected simple graph is called Hamiltonian-DFS-Graph ($H$-DFS-$G$) if and only if every DFS tree in it is an Hamiltonian directed path.

Theorem 4.1: A graph $G$ is H-DFS-G if and only if it is a simple circuit, or $K(n)$, or $K(n,n)$ where $K(n)$ is the complete graph on $n$ vertices and $K(n,n)$ is the complete bipartite graph on $2n$ vertices with two equal parts.

Proof outline:

Proposition 4.2: In every DFS tree $T_s$ of an H-DFS-G there is a back edge from the leaf to the root $s$. □
Lemma 4.3: If \( G=(V,E) \) is a H-DFS-G and has a subgraph which is \( K_3 \) then \( G \) is a complete graph. \( \Box \)

Lemma 4.4: If \( G=(V,E) \) is a H-DFS-G and has a simple circuit of length \( k \) where \( 4<k<n \) then it has a simple circuit of length \( k-2 \). \( \Box \)

Corollary 4.5: If a H-DFS-G is neither a simple circuit nor a bipartite graph, then it is a complete graph. \( \Box \)

Lemma 4.6: If a H-DFS-G is a bipartite which is not a simple circuit, then it is the complete bipartite \( K(n,n) \). \( \Box \)

5. SPECIAL DFS TREES

Lower Bound Degree (LBD) problem: Given a graph \( G(V,E) \) and a vertex \( v \in V \). Is there a DFS tree where the degree of \( v \geq k \geq 3 \)?

This question is important for distributed and for parallel algorithms, where the degree of a vertex in a DFS tree obtained in a certain run might be too high for the vertex to handle because of technical limitations, e.g. there may be a limited number of communication ports to be used in the tree.

Theorem 5.1: The LBD problem is \( NP \)-Complete.

Proof: HAMILTONIAN PATH polynomially transforms to LBD. \( \Box \)

REFERENCES


