A theorem about the postorder numbers in the depth-first tree of a directed graph

Jayadev Misra

March 22, 2020

Abstract

We present a theorem that captures an important property of postorder numbering of the depth-first tree of a directed graph: any path in the graph from a lower to a higher numbered node includes a node that is the least-common ancestor of all nodes in the path. We apply this theorem in the proof of an algorithm for identifying the strongly-connected components.

Keywords: Graph algorithms, reachability in graphs, postorder number, depth-first traversal, strongly-connected components.

1 Background

Depth-first traversal is a fundamental tool in analyzing the structure and properties of both directed and undirected graphs. The traversal induces a tree structure on the nodes of a graph. The postorder numbers of the nodes in the tree are essential in the derivation of a variety of algorithms. We present a theorem that captures an important property of postorder numbering of the depth-first tree of a directed graph: any path in the graph from a lower to a higher numbered node includes a node that is the least-common ancestor of all nodes in the path. We apply this theorem in the proof of an algorithm, independently due to Kosaraju and Sharir [2], for identifying the strongly-connected components.

Conventions and Terminology The reader may get the background material about depth-first traversal from a number of sources; see, for example Cormen et. al. [1]. Write $u \rightarrow v$ to denote that there is a directed edge from node u to v, and $u \rightsquigarrow v$ for a directed path. Add a label to an edge or a path, as in $u \stackrel{p}{\rightarrow} v$ or $u \stackrel{p}{\rightsquigarrow} v$, to identify a specific edge or path. We assume that there is no self loop, $u \rightarrow u$. There is no technical difficulty in admitting self loops, but it makes some of the results easier to state and prove, without loss in generality.

A depth-first tree of a directed graph is constructed through a traversal starting at a node called *root*. Henceforth, a node is to be understood as one reachable from *root*. The postorder number of node u in the depth-first tree is denoted by u_e. Call u to be *lower* than v (or, v *higher* than u) if $u_e < v_e$. Ancestors of a node are defined according to the depth-first tree; we take a node to be an ancestor of itself. A *cross edge* is directed from a node to a non-ancestor.

In subsequent discussions, the terms related to trees, *child* and *ancestor*, refer to the depth-first tree, whereas *edge* and *path* refer to the given graph. □

The following lemma describes a well-known property of postorder numbering of nodes in a tree.

Lemma 1 (Convexity rule) Suppose $x_e \leq y_e \leq z_e$ and z is an ancestor of x. Then z is an ancestor of y. □

It is a well-known property of the depth-first traversal of a directed graph that a cross edge is directed from a higher to a lower node. The following lemma, Edge-ancestor, is a rewriting of this fact.

Lemma 2 (Edge-ancestor) For $u \rightarrow v$: $u_e < v_e \;\equiv v$ is ancestor of u. □

Observe that absence of self loop is essential for this lemma because given $u \rightarrow u$ and that $u_e = u_e$ would imply that u is not its own ancestor, violating the definition of ancestor.

Corollary 1 A cross edge is directed from a higher to a lower node.

2 Path-ancestor Theorem

Theorem 1 (Path-ancestor Theorem) Given $u \overset{p}{\leadsto} v$, where $u_e < v_e$, the highest node in p is the least-common ancestor of all nodes in p. □

It suffices to show that the highest node is an ancestor of all nodes in p, because it is an ancestor of itself. The theorem is valid even for a path that is not simple.

Let h be the highest node in p. Then the path is of the form $u \overset{lp}{\leadsto} h \overset{rp}{\leadsto} v$, where the first occurrence of h is as an extreme node in lp. We prove the result in two parts, that h is an ancestor of nodes in lp (Lemma 3) and in rp (Lemma 4).

Lemma 3 Given $u \overset{lp}{\leadsto} h$ where h is the unique highest node in lp, h is an ancestor of all nodes in lp.

Proof: proof is by induction on the length of lp.

(1) Base case, lp has one edge: then $u \rightarrow h$ and $u_e < h_e$. From edge-ancestor lemma, Lemma 2, h is an ancestor of u. Since h is an ancestor of itself, the result holds.

(2) Inductive case, $u \rightarrow u' \overset{lp'}{\leadsto} h$: Inductively, h is an ancestor of all nodes in lp' including u'. If $u_e < u'_e$, from edge-ancestor lemma, Lemma 2, u'_e is an ancestor of u, hence h is an ancestor of u. If $u'_e < u_e$ then $u'_e < u_e < h_e$ and h is an ancestor of u'. From the Convexity rule, Lemma 1, h is an ancestor of u.

Lemma 4 Given $u \overset{lp}{\rightsquigarrow} h \overset{rp}{\rightsquigarrow} v$ where h is the highest node, h is an ancestor of all nodes in rp including v.

Proof: Note that all nodes of lp and rp, including h, may occur multiple times in rp. Assign consecutive positive indexes to the nodes in rp, from v to h, starting with index 1 for v. We show that h is an ancestor of the node of index k, for all k where $k \geq 1$. Proof is by induction on k. From Lemma 3 h is an ancestor of u.

(1) Base case, $k = 1$: We have $u_e < v_e \leq h_e$ and h is an ancestor of u. Applying the Convexity rule, Lemma 1, h is an ancestor of v.

(2) Inductive case, $k > 1$: Let w be the node of index k, so the path is $u \overset{lp}{\rightsquigarrow} h \overset{rp'}{\rightsquigarrow} w \overset{rp''}{\rightsquigarrow} v$. If $w = u$ then h is an ancestor of w because h is an ancestor of u. So, assume $w \neq u$. Consider two cases.

(2.1) $u_e < w_e$: Then $u_e < w_e \leq h_e$. Applying the Convexity rule, Lemma 1, h is an ancestor of w.

(2.2) $u_e > w_e$: Then $w_e < u_e < v_e$. Let h' be the highest node in rp'', the path from w to v. Its index is less than k because, from $w_e < v_e \leq h'_e$, $h' \neq w$. Inductively, h is an ancestor of h'. Apply Lemma 3 on $w \rightsquigarrow h'$ to conclude that h' is an ancestor of w. Therefore, h is an ancestor of w. □

3 An application: Strongly-connected Components

An excellent example of the usefulness of depth-first traversal is in identifying the strongly-connected components of a directed graph. The following algorithm appears in an unpublished manuscript, dated 1978, by Kosaraju, and independently in Sharir [2].

Algorithm outline The algorithm runs in two phases. In phase 1, do a depth-first traversal of the given graph G and assign postorder numbers to nodes. In phase 2, identify the strongly-connected components as follows. Construct G^{-1} from G by reversing the directions of all edges of G. If there is a path $v \rightsquigarrow u$ in G^{-1}, where v has a higher postorder number than u in G (computed in phase 1), then (1) there is a path $u \rightsquigarrow v$ in G, and (2) from the path-ancestor theorem, v is an ancestor of u in the depth-first tree in G, so $v \rightsquigarrow u$ exists in G. Therefore, u and v are strongly-connected. The postorder numbers in G are used to guide phase 2; this is the only connection between the two phases. We describe the algorithm more formally, next.

Henceforth, the number of a node is its postorder number in G. The strongly-connected components $C_0, C_1, \cdots C_n$ are constructed in sequence. Component C_0 consists of the highest numbered node, r_0, and the set of nodes reachable from it in G^{-1}, C_1 consists of the highest numbered node r_1 that is not in C_0 and its reachable nodes in G^{-1} that are not in C_0, and so forth, continuing until every node belongs to some component. Specifically,

• (SCC) Let r_j be the highest numbered node that is not in any C_i, $0 \le i < j$. Then C_j is the set of reachable nodes from r_j in G^{-1} which do not belong to any C_i, $0 \le i < j$. That is,

$$r_j = v \text{ where } v_e = \{max\ x : (\forall i : 0 \le i < j : x \notin C_i) : x_e\}$$
$$C_j = \{u |\ r_j \rightsquigarrow u \text{ in } G^{-1}, (\forall i : 0 \le i < j : u \notin C_i)\}$$

Theorem 2 Each C_j, $0 \le j \le n$, is a strongly-connected component.

Proof: The proof of the theorem is in two parts.

1. Each C_j is strongly-connected: We show that every node u in C_j, $u \ne r_j$, is strongly-connected to r_j. Then every pair of nodes in C_j are strongly-connected through r_j.

there is $u \rightsquigarrow r_j$ in G	, $r_j \rightsquigarrow u$ exists in G^{-1}	(A)
r_j is higher than u	, choice of r_j in (SCC)	
r_j is an ancestor of u in G	, from path-ancestor theorem	
$r_j \rightsquigarrow u$ exists in G	, from above	
$u \rightsquigarrow r_j \rightsquigarrow u$ in G	, combining above with (A)	

2. Each C_j is a strongly-connected component: We show that u and v in different components are not strongly-connected. Suppose $u \rightsquigarrow v$, $u \in C_j$ and $v \in C_k$ where $j < k$. Then, $r_j \rightsquigarrow u \rightsquigarrow v$, so v is reachable from r_j. Also, $v \in C_k$ means $v \notin C_i$, $0 \le i < k$, so $v \notin C_i$, $0 \le i < j$. So, $v \in C_j$, according to rule (SCC). □

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. McGraw Hill and MIT press, third edition, 2009.

[2] Micha Sharir. A strong-connectivity algorithm and its applications to data flow analysis. *Computers and Mathematics with Applications*, 7(1):67–72, 1981.