Information Processing Letters 115 (2015) 684–688

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

On testing single connectedness in directed graphs and some related problems

Martin Dietzfelbinger, Raed Jaberi *

Department of Computer Science and Automation, Technische Universität Ilmenau, 98693 Ilmenau, Germany

ARTICLE INFO

Article history:
Received 3 March 2015
Received in revised form 15 April 2015
Accepted 16 April 2015
Available online 20 April 2015
Communicated by Ł. Kowalik

Keywords:
Algorithms
Depth first search
Unique paths
Directed graphs
Connectivity
NP-complete

ABSTRACT

Let $G = (V, E)$ be a directed graph with n vertices and m edges. The graph G is called singly-connected if for each pair of vertices $v, w \in V$ there is at most one simple path from v to w in G. Buchsbaum and Carlisle (1993) [1] gave an algorithm for testing whether G is singly-connected in $O(n^2)$ time. In this paper we describe a refined version of this algorithm with running time $O(s \cdot t + m)$, where s and t are the number of sources and sinks, respectively, in the reduced graph G^r obtained by first contracting each strongly connected component of G into one vertex and then eliminating vertices of indegree or outdegree 1 by a contraction operation. Moreover, we show that the problem of finding a minimum cardinality edge subset $C \subseteq E$ (respectively, vertex subset $F \subseteq V$) whose removal from G leaves a singly-connected graph is NP-hard.

1. Introduction

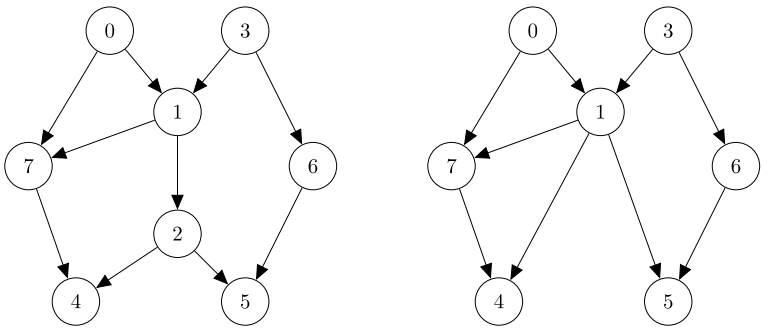
Let $G = (V, E)$ be a directed graph with n vertices and m edges. We assume that the underlying graphs of all directed graphs considered in this paper are connected; thus $m \geq n - 1$. The graph G is called *singly-connected* if for every pair of vertices $v, w \in V$ there is at most one simple path from v to w in G. The problem of testing whether or not G is singly-connected was introduced by Cormen et al. in [3, Ex. 23.3-10] and [4, Ex. 22.3-13]. In [1,2], Buchsbaum and Carlisle presented an algorithm for solving this problem in $O(n^2)$ time. Khuller described in [5] a similar approach for solving the same problem in $O(n^2)$ time. Karlin [6] (also see [8]) also presented a simple $O(n^2)$ algorithm for solving the problem.

Let $G = (V, E)$ be a directed graph. By $G^c = (V^c, E^c)$ we denote the directed graph which is obtained by contracting every strongly connected component of G into a single vertex. The algorithms from [1] and [5] are based on reducing the problem on G to the same problem on the acyclic digraph G^c. We use G^r to denote the digraph obtained from G^c by eliminating all vertices of indegree or outdegree 1 by contraction operations. A vertex x of G^r is called a source if its indegree is 0 and a vertex y of G^r is called a sink if its outdegree is 0. By s and t we denote the number of sources and sinks in G^r, respectively. In this paper we improve the running time of the algorithms from [1,5] from $O(n^2)$ to $O(s \cdot t + m)$. We also give an example for a graph where $s \cdot t$ is much bigger than m. The question posed by Khuller [5] whether the problem of testing single connectedness in directed graphs is solvable in linear time remains open.

As mentioned in [2], it is clear that a singly-connected graph can be obtained from a directed graph which is not singly-connected by removing edges, but the property of singly-connectivity can be ruined by adding edges. We also show that the problem of finding a minimum cardinality edge subset $C \subseteq E$ (respectively, vertex subset $F \subseteq V$) whose removal from G leaves a singly-connected graph is NP-hard.

* Corresponding author.
E-mail addresses: martin.dietzfelbinger@tu-ilmenau.de (M. Dietzfelbinger), raed.jaberi@tu-ilmenau.de (R. Jaberi).

http://dx.doi.org/10.1016/j.ipl.2015.04.008
0020-0190/



(a) G^c (b)

Fig. 1. (a) $G = (V, E)$, vertex 2 has indegree 1. (b) The remaining graph after contracting edge $(1, 2)$.

Papers [1] and [5] show the existence of a procedure for solving the problem of testing whether or not a directed graph $G = (V, E)$ is singly-connected, which works as follows: It either correctly determines that G is not singly-connected or outputs "Test whether G^c is singly-connected". In the latter case we have that G is singly-connected if and only if G^c is singly-connected. This procedure without testing of G^c has running time $O(m)$. Paper [1] shows that the acyclic graph G^c is singly-connected if and only if for every vertex $w \in V^c$ the vertex set of the DFS tree rooted at w has neither cross edges nor forward edges. Testing single connectedness of G^c in this way takes $O(n^2)$ time [1,5] since there are n calls to DFS, one for each vertex of G^c, and one can stop as soon as a forward edge or cross edge appears. The procedure leads to the situation that we only have to consider acyclic graphs. We give another reduction, which is to be applied after the procedure described above, can be carried out in time $O(m)$, and yields a reduced graph G^r, with the property that G is singly-connected if and only if G^r is. In G^r no vertex has indegree or outdegree 1.

2. Improved handling of acyclic graphs

Assume that the input graph $G = (V, E)$ is acyclic. We propose two types of improvement over the algorithms in [1,5] for acyclic graphs:

1. eliminating vertices with indegree or outdegree 1,
2. starting DFS only from sources.

In the first step, called preprocessing step, we modify G as follows. We consider the vertices in bottom-up order. For each vertex $v \in V$ with indegree 1 we shrink u, v, where $(u, v) \in E$, by replacing each edge $(v, w) \in E$ by (u, w) and removing the vertex v and the edge (u, v) from G (see Fig. 1 for one such contraction). Of course, if v has outdegree 0, it can simply be deleted. Let G' be the resulting graph. To G' we apply a similar procedure top down to eliminate all nodes of outdegree 1. The resulting graph is called G^r.

Lemma 2.1.

(a) *G^r can be computed in time $O(m)$.*
(b) *G is singly-connected if and only if G^r is singly-connected.*

Proof. (a) We represent the acyclic graph G by adjacency lists in which each vertex $v \in V$ has a circular doubly linked list L_v containing all the successors of v in G. Eliminate all vertices of indegree 1, as follows:

(i) Carry out a depth first search, which will yield a topological order of the vertices of G. Treat the vertices v in reverse topological order (according to increasing postorder numbers) as follows: If v has indegree 1, merge v with its predecessor u by linking L_v into L_u in constant time and deleting v. It is clear this can be done in time $O(m)$. (Actually, the merging itself takes time $O(n)$, only finding vertices with indegree 1 needs $O(m)$ time.)
(ii) Eliminate all vertices of outdegree 1. This is done exactly as in (i), just using the reversed graph of G'.

(b) Each step preserves the property of single connectedness. □

Note that if multiple edges arise in the preprocessing step, then G is not singly-connected. Check once at the very end, in time $O(m)$, if G^r has multiple edges. If so, return "not singly-connected".

From here on we only consider the reduced graph G^r, in which all non-sources have indegree at least 2 and all non-sinks have outdegree at least 2. In the second step we perform a DFS only for the sources of G^r. The correctness of this step is based on the following lemma.

Lemma 2.2. *Let $G = (V, E)$ be a directed acyclic graph such that G does not have any multiple edges. Then G is singly-connected if and only if for every source $v \in V$ the vertex set of the DFS tree T_v with root v has only tree edges.*

Proof. "⇐": Assume that G is not singly-connected. Then by definition, there are two vertices $v, w \in V$ such that there exist at least two simple paths p_1, p_2 from v to w in G. Then there is a vertex u that lies on both paths p_1, p_2