Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 380 (2007) 37–46

Theoretical Computer Science

www.elsevier.com/locate/tcs

Faster algorithms for finding lowest common ancestors in directed acyclic graphs[☆]

Artur Czumaj[a,*], Mirosław Kowaluk[b], Andrzej Lingas[c]

[a] Department of Computer Science, University of Warwick, Coventry CV4 7AL, United Kingdom
[b] Institute of Informatics, Warsaw University, Warsaw, Poland
[c] Department of Computer Science, Lund University, 22100 Lund, Sweden

Abstract

We present two new methods for finding a lowest common ancestor (LCA) for each pair of vertices of a directed acyclic graph (dag) on n vertices and m edges.

The first method is surprisingly natural and solves the all-pairs LCA problem for the input dag on n vertices and m edges in time $O(nm)$.

The second method relies on a novel reduction of the all-pairs LCA problem to the problem of finding maximum witnesses for Boolean matrix product. We solve the latter problem (and hence also the all-pairs LCA problem) in time $O(n^{2+\lambda})$, where λ satisfies the equation $\omega(1, \lambda, 1) = 1 + 2\lambda$ and $\omega(1, \lambda, 1)$ is the exponent of the multiplication of an $n \times n^{\lambda}$ matrix by an $n^{\lambda} \times n$ matrix. By the currently best known bounds on $\omega(1, \lambda, 1)$, the running time of our algorithm is $O(n^{2.575})$. Our algorithm improves the previously known $O(n^{2.688})$ time-bound for the general all-pairs LCA problem in dags by Bender et al.

Our additional contribution is a faster algorithm for solving the all-pairs lowest common ancestor problem in dags of small depth, where the depth of a dag is defined as the length of the longest path in the dag. For all dags of depth at most $h \leq n^{\alpha}$, where $\alpha \approx 0.294$, our algorithm runs in a time that is asymptotically the same as that required for multiplying two $n \times n$ matrices, that is, $O(n^{\omega})$; we also prove that this running time is optimal even for dags of depth 1. For dags with depth $h > n^{\alpha}$, the running time of our algorithm is at most $O(n^{\omega} \cdot h^{0.468})$. This algorithm is faster than our algorithm for arbitrary dags for all values of $h \leq n^{0.42}$.

Keywords: Directed acyclic graphs; Lowest common ancestors; Matrix multiplication; Time complexity

1. Introduction

The problem of finding a *lowest common ancestor* (LCA) in a tree, or more generally, in a *directed acyclic graph* (dag) is a basic problem in algorithmic graph theory. An LCA of vertices u and v in a dag is an ancestor of both u

☆ This paper is the result of merging the ICALP'05 paper [M. Kowaluk, A. Lingas, LCA queries in directed acyclic graphs, in: Proc. 32nd International Colloquium on Automata, Languages and Programming, ICALP'05, 2005, pp. 241–248] with [A. Czumaj, A. Lingas, Improved algorithms for the all-pairs lowest common ancestor problem in directed acyclic graphs, 2005, Manuscript].

* Corresponding author. Tel.: +44 24 7657 3796; fax: +44 24 7657 3024.
E-mail addresses: Artur.Czumaj@dcs.warwick.ac.uk (A. Czumaj), kowaluk@mimum.edu.pl (M. Kowaluk), Andrzej.Lingas@cs.lth.se (A. Lingas).

0304-3975/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2007.02.053

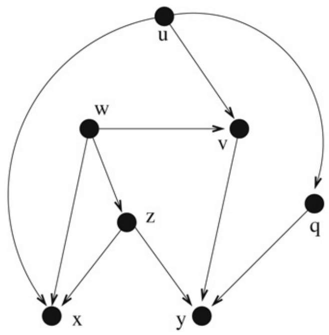

Fig. 1. A dag with 7 vertices. The LCA of vertices x and y are both, vertex u and vertex z; vertex w is a common ancestor of x and y but it is not the LCA of x and y. There is no common ancestor for vertices w and q.

and v that has no descendant which is an ancestor of u and v; see Fig. 1 for example. We consider the problem of preprocessing a dag in a way such that LCA queries can be answered quickly for any pair of vertices.

This problem has been very extensively studied in the literature in the context of (rooted) trees (see, e.g., [3,12,17]), where it appears naturally in various settings and where it found many applications in the design of efficient algorithms and data structures (e.g., for the problem of computing maximum matching in graphs and for various string problems). Harel and Tarjan [12] were the first to show that in rooted trees, LCA queries can be answered in constant time after a linear time preprocessing of the input tree. This work has been later extended in many ways, including the simplified algorithm from [3] and the recent dynamic algorithm [5].

Recently, Bender et al. initiated investigations of the LCA problem for arbitrary directed acyclic graphs in [4]. They listed a number of natural applications of LCA queries in dags in [4], e.g., in inheritance analysis in programming languages, analysis of genealogical data, and lattice operations in complex systems (for more details, see e.g., [9,15,16], and especially [4] and the references therein). Bender et al. also observed that the all-pairs LCA problem in dags is unfortunately not simpler than that of transitive closure in arbitrary directed graphs and Boolean matrix multiplication [4]. On the other hand, they were first to design substantially subcubic-time solution to the all-pairs LCA problem in dags. Their algorithm for the all-pairs LCA problem in dags runs in time $O(n^{(3+\omega)/2}) = O(n^{2.688})$, where n is the number of vertices and $\omega < 2.376$ is the exponent of the fastest matrix multiplication algorithm [7].

New contributions. We present two new methods of efficiently preprocessing a directed graph on n vertices and m edges in order to answer an LCA query for any pair of vertices in constant time, subsuming the previously known best results from [4].

The first method is surprisingly natural and solves the all-pairs LCA problem for the input dag on n vertices and m edges in time $O(nm)$. For sparse dags, this method is optimal and substantially faster than the known $O(n^{\frac{\omega+3}{2}})$-time general method from [4].

The second method efficiently reduces the all-pairs LCA problem to the problem of finding maximum (index) witnesses for a Boolean matrix product. We solve the latter problem (and hence also the all-pairs LCA problem) in time $O(n^{2+\lambda})$, where λ satisfies the equation $\omega(1, \lambda, 1) = 1 + 2\lambda$ and $\omega(1, \lambda, 1)$ is the exponent of the multiplication of an $n \times n^{\lambda}$ matrix by an $n^{\lambda} \times n$ matrix. By the currently best known bounds on $\omega(1, \lambda, 1)$, the running time of our algorithm is $O(n^{2.575})$. Our algorithm improves the previously known $O(n^{2.688})$ time-bound for the general all-pairs LCA problem in dags by Bender et al. [4].

In addition, we present a faster algorithm for solving the all-pairs lowest common ancestor problem in dags of small depth, where the *depth* of a dag is defined as the length of the longest path in the dag.[1] For all dags of depth at most $h \le n^{\alpha}$, where $\alpha \approx 0.294$, our algorithm runs in time asymptotically the same as that of multiplying two $n \times n$ matrices, that is, $O(n^{\omega})$; we also prove that this running time is optimal even for dags of depth 1. For dags with depth $h > n^{\alpha}$, the running time of our algorithm is at most $O(n^{\omega} \cdot h^{0.468})$. This algorithm is faster than our method for arbitrary dags for all values of $h \le n^{0.42}$.

[1] In [4], an equivalent notion of the *height* of a dag is used. We find however the term *depth* more intuitive.

Organization. Our paper is structured as follows. In Section 2, we present our natural $O(nm)$-time method for the all-pairs LCA problem in dags. In Section 3, we present several concepts and facts used by our matrix based methods for the all-pairs LCA problem in dags. In Section 4, we demonstrate the relationships between the problems of computing common ancestors and LCA in dags and those of finding witnesses and maximum witnesses for Boolean matrix product. In Section 5, we present our $O(n^{2.575})$-time method for the maximum witness problem and the all-pairs LCA problem for dags. In Section 6, we derive a more efficient solution to the all-pairs LCA problem in dags of small depth.

2. Optimal method for sparse dags

First, we shall describe preprocessing for answering queries about the existence of a common ancestor for arbitrary pair of vertices in constant time.

For the input dag, we shall denote by n and m its number of vertices and edges, respectively. Also for a vertex v in the dag, $\mathtt{indeg}(v)$ and $\mathtt{outdeg}(v)$ stand respectively for the in-degree and out-degree of v. If $\mathtt{outdeg}(v) = 0$ then v is called a *terminal vertex* and if $\mathtt{indeg}(v) = 0$ then v is called a *source vertex*.

We may assume without loss of generality that the input dag is connected, since otherwise we can decompose it into connected components and solve the problem for each component separately. For technical reasons, we shall also assume that every vertex is its own ancestor.

The following lemma immediately follows from the definition of a dag.

Lemma 1. *If two vertices have a common ancestor then there is a source vertex that is their common ancestor.*

In the first stage of the preprocessing, for each vertex of the input dag we form a table containing its descendants. In other words, we create the transitive closure of the dag, which obviously can be done in time $O(nm)$. For the sake of Section 3, we describe this stage in more detail below.

We initialize the tables in time $O(n^2)$ and start from the terminal vertices, filling their tables with single vertices in time $O(n)$. Next we iterate the following step: remove the vertices of out-degree 0 with incident edges and fill in the tables for the new vertices v of out-degree 0 by merging the information from the tables associated with the removed direct descendants of v, taking into account the set of direct descendants of v. We also add v to its table. For each vertex v, such an operation takes time $O(n) \times \mathtt{outdeg}(v)$. Thus, for the whole graph it takes $O(nm)$ time.

Lemma 2. *The tables of descendants for all vertices can be formed in time* $O(nm)$.

In the second stage of the preprocessing, we determine, for each vertex v, the set of vertices that have a common ancestor with v. We proceed similarly to the first stage of preprocessing, starting from source vertices instead of the terminal ones. For the source vertices s, the sets are already computed, they are just the sets of descendants of s. Next, we iterate the following step: remove the vertices of in-degree 0 with incident edges and fill the tables for the new vertices v of in-degree 0 by merging the information from the tables associated with the removed direct ancestors of v. For each vertex v, such an operation takes time $O(n) \times \mathtt{indeg}(v)$. Thus, for the whole graph it takes $O(nm)$ time.

By the *depth* of a vertex v in a dag, we shall mean the length of the longest path from a source vertex to v in the dag.

Note that the set of vertices having a common ancestor with a vertex v is the union of the sets of vertices having common ancestors with the ancestors of v (recall that v is also an ancestor of itself). Hence, we obtain the following lemma by induction on the depth of v.

Lemma 3. *For all vertices v, the tables of vertices having a common ancestor with v can be computed in time* $O(nm)$.

In order to answer LCA queries, we need to refine the preprocessing slightly. During the second descending phase of the preprocessing, we additionally enumerate the vertices in their visiting order. Since an ancestor is always visited before its descendant, we obtain the following lemma.

Lemma 4. *A vertex of a higher number cannot be an ancestor of a vertex of a lower number.*

For all vertices v, in the table containing vertices w having a common ancestor with v, we keep also the maximum of the numbers assigned to the common ancestors of v and w. To achieve this, when we merge the information from the tables of direct ancestors of v, we pick the maximum number of a common ancestor of a direct ancestor of v and w. Clearly, the refinement can be accomplished within the same asymptotic time $O(mn)$. By induction, we obtain the following lemma.

Lemma 5. *For all vertices v, the tables of vertices w having a common ancestor with v with a pointer to a lowest common ancestor of v and w can be computed in time $O(nm)$.*

Hence, we obtain immediately the following theorem.

Theorem 6. *The all-pairs LCA problem for a dag on n vertices and m edges can be solved in time $O(n(n+m))$.*

Note that if $m = O(n)$, then our solution is optimal.

The authors of [4] discuss also the so-called *shortest ancestral distance* between a pair of vertices in a dag in [4].[2] For two vertices u and v, they define it as the length of a shortest path between u and v which passes through a common ancestor of u and v (observe that the common ancestor is not necessarily the lowest one[3]). Bender et al. showed that the all-pairs shortest ancestral distances can be computed in time $O(n^{2.575})$ [4].

We can modify our first method to obtain an $O(mn)$-time algorithm to compute the all-pairs shortest ancestral distances as follows. In the ascending phase, for each vertex v, and for each descendent u of v, we additionally compute the shortest directed distance between u and v. This can be easily accomplished within the same asymptotic time $O(mn)$. At the beginning of the descending phase, the previously computed shortest directed distances yield the shortest ancestral distances between sources and their descendants. While descending, the shortest ancestral distances between the parents of the current vertex v and each other vertex u are increased by one. Next, the minimum of them and the shortest directed distance between v and u (it can be infinite) is taken as the shortest ancestral distance between v and u. In this way, the shortest ancestral distance is computed for all pairs of vertices v and u.

Similarly, the so modified descending phase can be also implemented in time $O(mn)$. We conclude with the following theorem.

Theorem 7. *For a dag on n vertices and m edges, the all-pairs shortest ancestral distances can be computed in time $O(n(n+m))$.*

3. Preliminaries for the matrix based methods

In the remaining sections, we shall use several facts and concepts related to matrix multiplication, in particular Boolean matrix products. We shall also consider dags of bounded depth.

We let ω denote the square matrix multiplication exponent, that is, the smallest constant for which the product of two $n \times n$ matrices can be computed in $O(n^{\omega})$ time. The best asymptotic upper bound on ω currently known is $\omega < 2.376$, by Coppersmith and Winograd [7].

The following fact that relates graph problems to fast matrix multiplication is folklore.

Fact 1. *The transitive closure of any directed graph with n vertices, in particular a dag, can be computed in time $O(n^{\omega})$.*

We also need to define the concept of witness and maximum witness in Boolean matrix multiplication.

Definition 8. If an entry $C[i, j]$ of the Boolean product of two Boolean matrices A and B is equal to 1, then any index k such that $A[i, k]$ and $B[k, j]$ are equal to 1 is a **witness** for $C[i, j]$. If k is the largest possible witness for $C[i, j]$, then it is called the **maximum witness** for $C[i, j]$.

The following fact is due to Alon and Naor [1].

Fact 2. *The witnesses of the Boolean product of two $n \times n$ Boolean matrices can be computed in time[4] $\widetilde{O}(n^{\omega})$.*

[2] Their original (cf. [2]) motivation for considering the shortest ancestral distance in sparse dags were applications in the identification of genes associated with genetic diseases [9,16].

[3] One can also consider the so-called shortest ancestral LCA distance where the common ancestor is required to be lowest [4].

[4] Throughout the paper, the notation $\widetilde{O}(f(n))$ stands for $O(f(n) \log^{c} n)$ for some positive constant c.

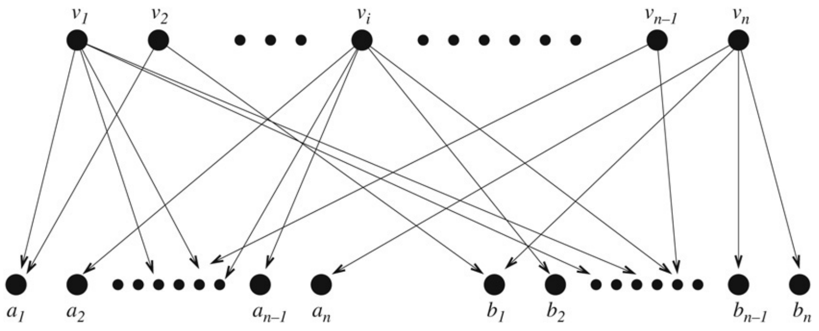

Fig. 2. A scheme of the construction used in the proof of Theorem 12.

As it has been observed in [4], the concept of the level of a vertex in a dag plays an important role in the study of LCA in dags.

Definition 9. The **level** i of a dag is the set of all its vertices of depth i. The **depth of a dag** is the maximum depth of its vertices, or equivalently, the number of its non-empty levels decreased by one.

By using a variant of the Bellman–Ford algorithm for the single-source shortest path problem in dags running in time $O(n + m)$ (see, e.g., Chapter 24.2 in [8]), we obtain the following lemma.

Lemma 10. *For a dag G with n vertices and m edges, one can compute the partition of the vertices of G into the levels and the depth of G in time $O(n + m)$.*

4. Common ancestors versus Boolean matrix product witnesses

In this section, we discuss the relationship between the problems of finding common ancestors and LCA for all pairs of vertices in a dag and the problems of computing witnesses and maximum witnesses of the Boolean product of two Boolean $n \times n$ matrices.

We begin with a reduction of the problem of computing all-pairs common ancestors in a dag to the problem of computing witnesses of the Boolean product of the corresponding Boolean matrices.

Theorem 11. *The problem of computing all-pairs common ancestors in a dag on n vertices can be reduced to the problem of computing witnesses of the Boolean product of two Boolean $n \times n$ matrices in time $O(n^{\omega})$. Similarly, the problem of computing all-pairs LCA in a dag on n vertices can be reduced to the problem of computing maximum witnesses of the Boolean product of two Boolean $n \times n$ matrices in time $O(n^{\omega})$.*

Proof. To prove the first part, proceed as follows. Compute the transitive closure of the input dag. Number the vertices of the dag by their rank in the topological ordering of the transitive closure. Form the ancestor matrix A such that in its ith row, there is 1 on the kth position iff the kth vertex is an ancestor of the ith vertex. Compute the witnesses of the Boolean product of the matrix A and its transpose A^T. For any (i, j) entry of the product matrix, if the entry is positive, then its witness is the number of a common ancestor of the ith and jth vertex.

To prove the second part, compute the maximum witnesses of the Boolean product of the matrix A and its transpose A^T instead of the ordinary ones. Next, observe that a common ancestor of vertices u and v having the largest number among common ancestors of u and v is an LCA of u and v. By this observation, for any (i, j) entry of the product matrix, if the entry is positive, then its maximum witness is the number of an LCA of the ith and jth vertex. □

In turn, we consider a reduction of the problem of computing witnesses to the problem of computing all-pairs common ancestors in a dag of depth 1.

Theorem 12. *The problem of computing witnesses of the Boolean product of two Boolean $n \times n$ matrices can be reduced to the problem of computing all-pairs common ancestors in a dag with $3n$ vertices and depth 1 in time $O(n^2)$.*

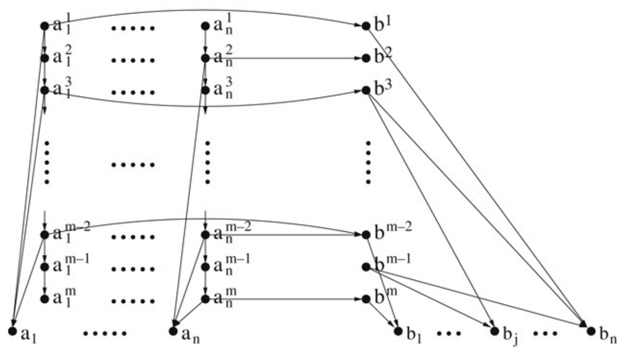

Fig. 3. A scheme of the construction used in the proof of Theorem 14.

Proof. Let A and B be two Boolean $n \times n$ matrices. Construct a two-level dag G with vertices $v_1, v_2, \ldots, v_n$ on the zero level and vertices $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ on the first level. Create an edge (v_k, a_i) iff $A[i,k] = 1$. Analogously, form an edge (v_k, b_j) iff $B[k,j] = 1$ (see Fig. 2).

Let C be the Boolean product of matrices A and B. By the definition of G, k is a witness of an entry $C[i,j]$ iff v_k is a common ancestor of the vertices a_i and b_j. □

By combining the first part of Theorem 11 with Theorem 12, we obtain the following corollary.

Corollary 13. *The problem of computing all-pairs common ancestors in a dag on n vertices and the problem of computing witnesses of the Boolean product of two Boolean $n \times n$ matrices are $O(n^{\omega})$-time equivalent.*

Note that the all-pairs common ancestor problem (not the *lowest* common ancestor one) for an arbitrary dag can be solved in time $O(n^{\omega})$ by computing the transitive closure of the dag, using Fact 1, and then computing the witnesses of the Boolean product of the transitive closure matrix and its transpose using Fact 2. This running time is optimal, since Theorem 12 implies that the problem of computing all-pairs LCA in a dag with n vertices requires time $\Omega(n^{\omega})$, even for dags of depth 1.

By using a dag construction much more sophisticated than that used in the proof of Theorem 12, we obtain the following theorem that relates computing *maximum witnesses* to computing the all-pairs LCA in a dag.

Theorem 14. *The problem of computing maximum witnesses of Boolean product of two Boolean $n \times n$ matrices can be reduced to the problem of computing the all-pairs LCA in a dag on $n^2 + 3n$ vertices in time $O(n^2)$.*

Proof. Let A and B be two Boolean $n \times n$ matrices. Construct a dag $G = (V, E)$ where

$$V = \{a_i | 1 \le i \le n\} \cup \{a_i^k | 1 \le i,\ k \le n\} \cup \{b^k | 1 \le k \le n\} \cup \{b_i | 1 \le i \le n\}$$

and

$$\begin{aligned} E &= \{(a_i^k, a_i) | 1 \le i,\ k \le n \,\&\, A[i,k] = 1\} \cup \{(a_i^k, a_i^{k+1}) | 1 \le i,\ k \le n\} \\ &\cup \{(a_i^k, b^k) | 1 \le i,\ k \le n \,\&\, A[i,k] = 1\} \cup \{(b^k, b_j) | 1 \le i,\ j \le n \,\&\, B[k,j] = 1\}. \end{aligned}$$

(See Fig. 3.) Since $\#V = n^2 + 3n$ and $\#E = O(n^2)$ hold, the construction can be done in quadratic time.

Our proof relies on the two following observations easily following from the construction.

1. k is a witness for the entry $C[i,j]$ of the resulting product matrix C if and only if there is an edge from a_i^k to a_i and a directed path from a_i^k through (a_i^k, b^k) to b_j.
2. If v is a common ancestor of a_i and b_j then $v = a_i^l$ holds for some $l \in \{1, \ldots, n\}$ and there exists $l' \in \{l, \ldots, n\}$ such that there is an edge from $a_i^{l'}$ to a_i and a directed path from $a_i^{l'}$ through $(a_i^{l'}, b^{l'})$ to b_j.

It follows from the first observation that if k is a witness for $C[i,j]$, then a_i^k is a common ancestor of a_i and b_j. Furthermore, by the second and first observations, we infer that if k is a maximum witness, then k is a maximum index such that a_i^k is an ancestor of a_i and b_j. Since there is no $l < k$ such that a_i^l is a descendant of a_i^k in G, we

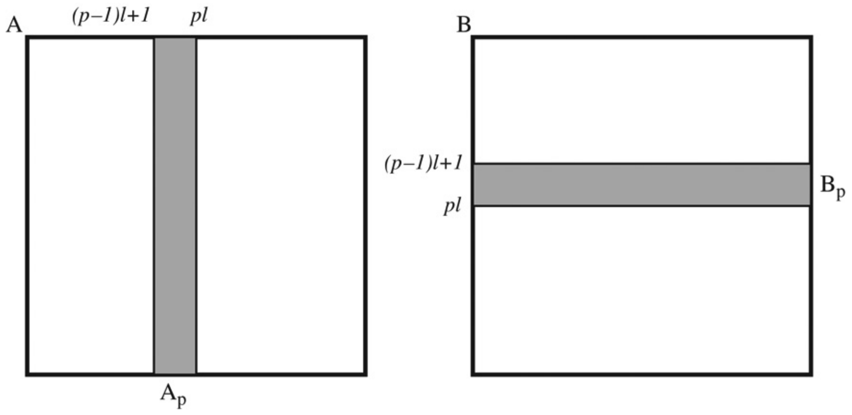

Fig. 4. Rectangular matrices A_p and B_p.

conclude by the second observation that if k is a maximum witness for $C[i,j]$, then a_i^k is a lowest common ancestor of a_i and b_j.

Conversely, consider a common ancestor v of a_i and b_j. By both observations, $v = a_i^l$ for some $l \in \{1,\dots,n\}$ and there is $l' \geq l$ such that $a_i^{l'}$ is a common ancestor of a_i and b_j and l' is witness for $C[i,j]$. Next, if v is a lowest common ancestor of a_i and b_j then $l = l'$ and l must be the maximum witness, since otherwise we obtain a contradiction by the first observation and the fact that if $l < k$, then a_i^l is an ancestor of a_i^k.

We conclude that k is a maximum witness for $C[i,j]$ if and only if a_i^k is a lowest common ancestor of a_i and b_j. It remains to note that a_i and b_i can have at most one lowest common ancestor. □

Note that the dag constructed in Theorem 14 has a quadratic number of vertices. Therefore, by combining Theorem 11 with Theorem 14, we cannot extend Corollary 13 to include an $O(n^{\omega})$-time equivalence between the problem of computing the all-pairs LCA in a dag on n vertices and the problem of computing maximum witnesses of the Boolean product of two Boolean $n \times n$ matrices.

Bender et al. [4] showed that the problem of computing the all-pairs LCA in a dag with n vertices is not easier than that of computing the transitive closure of a directed graph with n vertices. Our reduction of the all-pairs LCA problem in dags to that of computing maximum witnesses of the Boolean product of two Boolean matrices (Theorem 11) demonstrates that the latter problem does not have a smaller asymptotic complexity.

5. $O(n^{2.575})$-time method for maximum witnesses and the all-pairs LCA problem in dags

By Corollary 13, it is sufficient to compute maximum witnesses of the Boolean product of two Boolean $n \times n$ matrices in order to solve the all-pairs LCA problem in dags. In this section, we present a substantially subcubic algorithm for the maximum witness problem using fast rectangular matrix multiplication.

Let ℓ be a positive integer smaller than n. Partition the matrix A into $n \times \ell$ sub-matrices A_p, and the matrix B into $\ell \times n$ sub-matrices B_p, such that $1 \leq p \leq n/\ell$, and the sub-matrix A_p covers the columns $(p-1)\,\ell + 1$ through $p\,\ell$ of A, whereas the sub-matrix B_p covers the rows $(p-1)\,\ell + 1$ through $p\,\ell$ of B. (For an example, see Fig. 4.)

For $p = 1,\dots,n/\ell$, let C_p be the Boolean product of A_p and B_p. Then, $C_p[i,j] > 0$ iff there is an index k, $(p-1)\,\ell < k \leq p\,\ell$, such that $A[i,k] = B[k,j] = 1$. Therefore the following claim follows.

Claim 15. *Suppose that a* $C[i,j]$ *entry of the Boolean product* C *of* A *and* B *is positive. Let* p' *be the maximum value of* p *such that* $C_{p'}[i,j] > 0$. *The maximum witness of* $C[i,j]$ *belongs to the interval* $[(p'-1)\,\ell + 1, p'\,\ell]$.

By this claim, after computing all the products $C_p = A_p \cdot B_p$, $1 \leq p \leq n/\ell$, we need $O(n/\ell + \ell)$ time per positive entry of C to find the maximum witness: $O(n/\ell)$ time to determine p' and then $O(\ell)$ time to locate the maximum witness.

Let $\omega(1,r,1)$ denote the exponent of the multiplication of an $n \times n^r$ matrix by an $n^r \times n$ matrix. It follows that the total time taken by our method for maximum witnesses is

$$O((n/\ell) \cdot n^{\omega(1,\log_n \ell,1)} + n^3/\ell + n^2\ell).$$

By setting r to $\log_n \ell$, our upper bound transforms to $O(n^{1-r+\omega(1,r,1)} + n^{3-r} + n^{2+r})$. Note that by assuming $r \geq \frac{1}{2}$, we can get rid of the additive n^{3-r} term. Hence, by solving the equation $1 - \lambda + \omega(1, \lambda, 1) = 2 + \lambda$ implying $\lambda \geq \frac{1}{2}$ by $\omega(1, \lambda, 1) \geq 2$, we obtain our main result.

Theorem 16. *Let λ be such that $\omega(1, \lambda, 1) = 1 + 2\lambda$. The maximum witnesses for all positive entries of the Boolean product of two $n \times n$ Boolean matrices can be computed in time $O(n^{2+\lambda})$.*

Coppersmith [6] and Huang and Pan [13] proved the following fact.

Fact 3 (*[6,13]*). *Let $\omega = \omega(1, 1, 1) < 2.376$ and let $\alpha = \sup\{0 \leq r \leq 1 : \omega(1, r, 1) = 2 + o(1)\} > 0.294$. Then $\omega(1, r, 1) \leq \beta(r)$, where $\beta(r) = 2 + o(1)$ for $r \in [0, \alpha]$ and $\beta(r) = 2 + \frac{\omega-2}{1-\alpha}(r - \alpha) + o(1)$ for $r \in [\alpha, 1]$.*

Note that by Fact 3, the solution λ of the equation $\omega(1, \lambda, 1) = 1+2\lambda$ is satisfied by $\lambda = \frac{1-\alpha(\omega-1)}{2(1-\alpha)-(\omega-2)} + o(1) < 0.575$ (cf. [18]). Therefore, we obtain the following concrete corollary.

Corollary 17. *The maximum witnesses for all positive entries of the Boolean product of two $n \times n$ Boolean matrices can be computed in time $O(n^{2.575})$.*

By combining Theorem 11 with Theorem 16 and Corollary 17, we obtain also an $O(n^{2.575})$-time solution to the all-pairs LCA problem in dags.

Theorem 18. *Let $\omega(1, \lambda, 1) = 1 + 2\lambda$. The all-pairs LCA problem for an arbitrary dag with n vertices can be solved in time $O(n^{2+\lambda})$, which is bounded above by $O(n^{2.575})$.*

6. All-pairs LCA in dags of bounded depth

In this section, we describe an algorithm for solving the all-pairs LCA problem for an arbitrary dag G of depth bounded by h. The algorithm has a similar flavor to that discussed in the previous section, but now we will additionally use the fact that the depth is bounded to speed up the process.

First, we compute the transitive closure of G and create the ancestor matrix A as in the proof of Theorem 11. Next, using Lemma 10, we partition G into $h + 1$ levels and extend the partial order induced by this partition to a linear order, and number the vertices according to the linear order so the numbering is increasing with respect to vertex depth. These steps can be performed in time $O(n^{\omega})$.

Observe that the numbering naturally decomposes into $h + 1$ continuous intervals in one-to-one correspondence with the levels of G. Our approach relies on the following generalization of the observation that the common ancestor of vertices u and v that has the highest number (in the topological ordering) is a lowest common ancestor of u and v.

Claim 19 (*[4]*). *Any common ancestor of vertices u and v which is of highest level among the common ancestors of u and v is a lowest common ancestor of u and v.*

Claim 19 implies directly that the maximum witnesses of the product of the Boolean matrix of ancestors A and its transpose A^T yield the solution to the all-pairs LCA problem for a dag. However, since it is expensive to compute maximum witnesses, we modify this construction and reduce the problem to that of computing *witnesses* (instead of *maximum* witnesses).

Relying on Claim 19 and the bounded depth h, we decompose A and its transpose A^T into $h + 1$ rectangular sub-matrices in one-to-one correspondence with the level intervals, and compute witnesses for the products of the pairs of sub-matrices in A and A^T corresponding to the same level interval. Now, as in the previous section, we observe that if vertices i and j have a common ancestor at level ℓ, then if we multiply the sub-matrix of A corresponding to level ℓ by the sub-matrix of A^T corresponding to level ℓ, then the resulting matrix C_ℓ will have $C_\ell[i, j] > 0$ and the witness for $C_\ell[i, j] > 0$ will be a vertex from level ℓ that is an ancestor of both i and j.

Let us estimate the time taken by finding the witnesses for the $h + 1$ products of sub-matrices of A and A^T. Recall the definition of function β in Fact 3 due to Coppersmith, and Huang and Pan. By using the straightforward reduction of Boolean matrix multiplication to the arithmetic one, and by generalizing the derandomization method of Alon and Naor [1] for witnesses of Boolean matrix multiplication to include rectangular matrices, we obtain the following lemma.

Lemma 20. *The witnesses of the Boolean product of two Boolean matrices of sizes $n \times n^r$ and $n^r \times n$ can be computed in time $\widetilde{O}(n^{\beta(r)})$.*

For $k = 0, \dots, h$, let ℓ_k be the number of vertices on level k in the dag G. By Lemma 20, the total time taken for computing the witnesses for $h + 1$ sub-matrix products is

$$\widetilde{O}\left(\sum_{k=0}^{h} n^{\beta(\log_n \ell_k)}\right). \tag{1}$$

Finally, once we have determined witnesses for every pair of vertices numbered i and j and for every level ℓ, it remains to find, for each pair i and j, the largest witness among the witnesses for the pairs i, j in the products of the sub-matrices. It takes $O(h)$ time per pair i, j and hence, $O(n^2 h)$ time in total.

Next, by straightforward calculations, the Jensen inequality implies that the value of (1) is maximized if the levels are of equal size $n/(h + 1)$. (Indeed, we observe that the function $n^{\beta(x)-2}$ is concave and therefore $\sum_{k=0}^{h} n^{\beta(\log_n \ell_k)} = n^2 \sum_{k=0}^{h} n^{\beta(\log_n \ell_k)-2} \le n^2 (h+1)\, n^{\beta(\log_n n/(h+1))-2} = (h+1)\, n^{\beta(\log_n n/(h+1))}$.) Hence, we obtain the following theorem.

Theorem 21. *The all-pairs LCA problem for an arbitrary dag with n vertices and depth n^q can be solved in time $\widetilde{O}(n^{\omega} + n^{q+\beta(1-q)})$.*

In particular, for dags of depth at most $n^{\alpha} \approx n^{0.294}$, the running time is $\widetilde{O}(n^{\omega})$.

For larger values of the depth n^q, the running time of this algorithm is

$$\widetilde{O}\left(n^{\omega+q(1-\frac{\omega-2}{1-\alpha})+o(1)}\right) \approx O\left(n^{2.376+0.468\,q}\right).$$

Remark 22. Note that for all values of $q \le \alpha \approx 0.294$, the running time of our algorithm from Theorem 21 is asymptotically optimal due to our result in Theorem 12. Even for larger values of q, up to $q \le 0.42$, our algorithm from Theorem 21 is faster than the general one from Theorem 18.

7. Final remarks

The problems of finding the LCA are classical and central in the area of algorithms and data structures [4,15,17]. In spite of the long history of studies devoted to LCA problems, we have succeeded in designing two quite natural methods for finding the LCA in dags that considerably subsume the previously known best results [4].

Our second method relies on the presented close relationship between the problem of computing the maximum witnesses of the Boolean matrix product and that of finding LCAs for all pairs of vertices in a dag and the use of fast algorithms for rectangular matrix multiplication.

It is an intriguing open problem whether the complexity gap between the problems of computing witnesses and computing maximum witnesses of Boolean matrix product, or similarly, the problems of finding common ancestors and finding LCAs in a dag, can be further decreased.

The problem of finding maximum witnesses of a Boolean matrix product seems to be of interest in its own right. At first glance, it seems that the recursive $O(n^{\omega+\epsilon})$-time method of Galil and Margalit [11] for the witnesses of Boolean matrix product could be adapted to produce the maximum witnesses by considering the fragments containing maximum witnesses in subproblems without substantially altering its asymptotic time. However, the aforementioned method may permute rows or columns in recursive steps, which may disturb the search for maximum witnesses. Thus, the problem of whether or not our $O(n^{2.575})$-time method is optimal thus remains open.

It is also an interesting question as to whether or not the instances of the problem of finding maximum witnesses of the Boolean matrix product that occur in our reduction from the LCA problem in dags are computationally easier than the general ones.

For further reading

[10,14]

Acknowledgments

The authors are grateful to Pavel Sumazin for inspiration, to Leszek Gąsieniec and Martin Wahlén for some discussions, and to Raphael Yuster for pointing out the incorrectness of the prior proof of Theorem 14.

A. Czumaj's research was supported in part by NSF ITR grant CCR-0313219. M. Kowaluk's research was supported by KBN grant 4T11C04425. A. Lingas's research was supported in part by VR grant 621-2002-4049.

References

[1] N. Alon, M. Naor, Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions, Algorithmica 16 (1996) 434–449.

[2] A. Becker, D. Geiger, A.A. Schaeffer, Automatic selection of loop breakers for genetic linkage analysis.

[3] M.A. Bender, M. Farach-Colton, The LCA problem revisited, in: Proc. 4th Latin American Symposium on Theoretical Informatics, LATIN'00, 2000, pp. 88–93.

[4] M.A. Bender, G. Pemmasani, S. Skiena, P. Sumazin, Finding least common ancestors in directed acyclic graphs, in: Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'01, 2001, pp. 845–853.

[5] R. Cole, R. Hariharan, Dynamic LCA queries in trees, SIAM Journal on Computing 34 (4) (2005) 894–923.

[6] D. Coppersmith, Rectangular matrix multiplication revisited, Journal of Symbolic Computation 13 (1997) 42–49.

[7] D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progression, Journal of Symbolic Computation 9 (1990) 251–290.

[8] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 2nd ed., McGraw-Hill Book Company, Boston, MA, 2001.

[9] R.W. Cottingham Jr., R.M. Idury, A.A. Shäffer, Genetic linkage computations, American Journal of Human Genetics 53 (1993) 252–263.

[10] A. Czumaj, A. Lingas, Improved algorithms for the all-pairs lowest common ancestor problem in directed acyclic graphs, 2005, Manuscript.

[11] Z. Galil, O. Margalit, Witnesses for Boolean matrix multiplication and for transitive closure, Journal of Complexity 9 (1993) 201–221.

[12] D. Harel, R.E. Tarjan, Fast algorithms for finding nearest common ancestors, SIAM Journal on Computing 13 (2) (1984) 338–355.

[13] X. Huang, V.Y. Pan, Fast rectangular matrix multiplications and applications, Journal of Complexity 14 (1998) 257–299.

[14] M. Kowaluk, A. Lingas, LCA queries in directed acyclic graphs, in: Proc. 32nd International Colloquium on Automata, Languages and Programming, ICALP'05, 2005, pp. 241–248.

[15] M. Nykänen, E. Ukkonen, Finding lowest common ancestors in arbitrarily directed trees, Information Processing Letters 50 (6) (1994) 307–310.

[16] A.A. Shäffer, S.K. Gupta, K. Shriram, R.W. Cottingham Jr., Avoiding recomputation in linkage analysis, Human Heredity 44 (1994) 225–237.

[17] R.E. Tarjan, Applications of path compression on balanced trees, Journal of the ACM 26 (4) (1979) 690–715.

[18] U. Zwick, All pairs shortest paths using bridging sets and rectangular matrix multiplication, Journal of the ACM 49 (3) (2002) 289–317.