The Serial Transitive Closure Problem for Trees

Maria Luisa Bonet*
Department of Mathematics
University of California, San Diego

Samuel R. Buss*
Department of Mathematics
University of California, San Diego

July 9, 2002

Abstract

The serial transitive closure problem is the problem of, given a directed graph G and a list of edges, called closure edges, which are in the transitive closure of the graph, to generate all the closure edges from edges in G. We give a nearly linear upper bound on the number of steps in optimal solutions to the serial transitive closure problem for the case of graphs which are trees. "Nearly linear" means $O(n \cdot \alpha(n))$ where α is the inverse Ackermann function. This upper bound is optimal to within a constant factor.

Keywords: *transitive closure, graph algorithm, inverse Ackermann function, weak superconcentrators.*
Math. Subject Classification: *68R10, 05C12, 05C85.*

1 Introduction

A directed graph is transitive if, whenever there is an edge from a node X to a node Y and an edge from Y to Z, then there is an edge from X to Z. The transitive closure of G is a smallest transitive, directed graph containing G.

*Supported in part by NSF Grant DMS-8902480.

We write $X \to Y$ to indicate the presence of an edge from X to Y. It is easy to see that any edge in the transitive closure of a graph G can be obtained from the edges of G by a series of zero or more *closure steps*, which are inferences of the form

$$\frac{A \to B \qquad B \to C}{A \to C}$$

In other words, if $A \to B$ and $B \to C$ are edges in the transitive closure of G, then $A \to C$ is too. This is because G plus the edges that can be derived by closure steps from edges in G both must be in any transitive graph containing G and also form a transitive graph on the nodes of G.

The serial transitive closure problem is the problem of deriving a given subset of edges, which we call "closure edges", in the transitive closure of a directed graph. A solution to the serial transitive closure problem is a sequence of closure steps which generates all of the given closure edges and the size of a solution is the number of closure steps in the solution. In this paper, we give upper and lower bounds on the size of optimal solutions to the serial transitive closure problems for directed graphs which are trees. It should be stressed that the set of closure edges can be any subset of the edges in the transitive closure of the graph (but not in the graph). The degenerate case of deriving a single closure edge $A \to B$ is quite simple, since the minimum number of closure steps required will be one less than the length of a shortest path from A to B. The general question of determining the optimal size of a solution is made difficult by the fact that, when a *set* of closure edges is being derived, it may be possible for individual closure steps to aid in the generation of multiple closure edges. In other words, it is not necessary to generate each closure edge independently. It is also important that the set of closure edges will, in general, not be *all* the edges in the transitive closure of the graph; the problem of deriving all the edges in the transitive closure of the graph is uninteresting because, in this case, exactly one closure step is needed per closure edge.

The authors' interest in the serial transitive closure problem arose in the study of the lengths of propositional proofs with and without the deduction rule. The serial transitive closure problem is directly related to the question of how efficiently Frege proof systems can simulate nested deduction Frege proof systems. More information on these proof systems and the application of the serial transitive closure problem to proof lengths can be found in [2, 3, 4]. The present paper is an expansion of portions of [2, 3].

The serial transitive closure problem is potentially applicable to problems in networks. As an example, suppose a communication network is given where the nodes are, say, computers and an edge from X to Y indicates that X is capable of sending messages to Y. Correspondingly, a set of closure edges is a set of edges of desired connections. If the closure edges are in the transitive closure of the network, then it is possible to establish the desired connections by having nodes relay messages (much like real-world nets such as Usenet). A closure step would correspond to determining that, since X can (indirectly) transmit to Y and Y can (indirectly) transmit to Z, X can indirectly transmit to Z. The size of a solution to the serial transitive closure problem would correspond to the number of indirect communication links that must be established to set up the desired connections. (It should be mentioned that this example completely glosses over important issues such as the bandwidth of the connections.)

The serial transitive closure problem is formally defined as follows:

Serial Transitive Closure Problem:

An *instance* consists of

- A directed graph G with m edges and
- A list of n *closure* edges $X_i \to Y_i$ $(i = 1, \ldots, n)$ which are in the transitive closure of G but not in G.

A *solution* is a sequence of edges $U_i \to V_i$ $(i = 1, \ldots, s)$ containing all n closure edges such that each $U_i \to V_i$ is inferred by a single closure step from earlier edges in the solution and/or edges in G. We call s the number of steps of the solution.

Note that the number of steps in a solution counts only closure steps and does not count edges that are already in G. A directed graph is a *tree* if it is a tree in the usual sense, with root at the top and with all edges directed downwards or with all edges directed upwards.

The outline of this paper is as follows. In section 2, we state and prove the main results which give near-linear upper bounds on size of solutions to the serial transitive closure problem for trees. Our near-linear upper bounds are of the form $O(n \cdot \alpha(n))$ where $\alpha(n)$ is the extremely slow-growing inverse Ackermann function. In section 3 we show that the upper bound is optimal for trees, using a theorem used by Tarjan [13] for an algorithm for the Union-Find problem. It is not known whether our upper bound is optimal for the case where the directed graph G is linear, i.e., G is a tree with each nonleaf node having only one child. However, we argue at the end of section 3 that our construction can not be easily improved for the linear case, since our approach gives an explicit construction for weak superconcentrators and there is a lower bound on the size of constant depth superconcentrators [10] which prevents the possibility that a simple modification can lead to an improvement of our construction.

The methods of our paper do not apply to graphs which are not trees, and we do not know any non-trivial upper or lower bounds on the size of solutions of serial transitive closure problems for general graphs.

Our proof methods for our upper bounds (Theorem 3 through Corollary 8) are related to the constructions of weak superconcentrators by [6, 7], and are also similar to prior methods for creating algorithms for range queries on linear lists [15] and on free trees [8]. In fact, as one of the referees pointed out, it is possible to derive Corollary 8 below, as a corollary to the proofs (but not the theorems) contained in [15, 8]. Similar constructions have also been used for adding edges to trees to reduce their diameter [1]. However, it is useful to give direct proofs in this paper since the Serial Transitive Closure Problem is of independent interest (e.g., as applied in [3, 4]).

2 Upper Bounds for Trees

The "near-linear" upper bounds given below are stated in terms of extremely slow growing functions such as log^* and the inverse Ackermann function. The function $\log x$ is the real-valued base two logarithm function. The log^* function is defined so that $\log^* n$ is equal to the least number of iterations of the logarithm base 2 which applied to n yield a value < 2. In other words,

$\log^* n$ is equal to the least value of k such that $n < 2^{2^{\cdot^{\cdot^{2}}}}$ where there are k 2's in the stack. To get even slower growing functions, we define the $\log^{(*i)}$ functions for each $i \geq 0$. The $log^{(*0)}$ function is just the base 2 logarithm function rounded down to an integer and the $log^{(*1)}$ is just the log^* function. For $i > 1$, $log^{(*i)}(n)$ is defined to be equal to the least number of iterations of the $log^{(*i-1)}$ function which applied to n yields a value < 2. The Ackermann function can be defined by the equations:

$$\begin{aligned} A(0,m) &= 2m \\ A(n+1,0) &= 1 \\ A(n+1,m+1) &= A(n, A(n+1,m)) \end{aligned}$$

It is well-known that the Ackermann function is recursive but dominates eventually every primitive recursive function (see [9] for a proof). We next develop the basic properties of the Ackermann function and the $\log^*$ functions; see La Poutré [12] for a similar development (his function $\alpha(i,m)$ is equal to our $\log^{(*i-1)}(m)$).

It is is easy to see, by induction on n, that $A(n,1) = 2$ for all n; because

$$A(n+1,1) \;=\; A(n, A(n+1,0)) \;=\; A(n,1)$$

Also, by induction on m, we have $A(1,m) = 2^m$; since,

$$A(1,m+1) \;=\; A(0, A(1,m)) \;=\; 2 \cdot A(1,m).$$

Likewise, $A(2,m) = 2^{2^{\cdot^{\cdot^{2}}}}$ where there are m 2's in the stack, since

$$A(2,m+1) \;=\; A(1, A(2,m)) \;=\; 2^{A(2,m)}.$$

Proposition 1 *For $n > 1$, $A(n,m)$ is the equal to the least i such that $\log^{(*n-1)}(i) = m$. Hence, $\log^{(*n-1)} A(n,m) = m$.*

Proof The proof is by induction on n. By the above definitions and comments, the lemma holds for $n = 1, 2$. Fix n and assume, as the induction hypothesis for n, that for all i, m, if $A(n,m) \leq i < A(n,m+1)$, then $\log^{(*n-1)}(i) = m$. To prove the corresponding fact for $n+1$, we use induction on m. It is obvious for $m = 0$ since $A(n, 0) = 0$ and $A(n, 1) = 2$. For $m > 0$, we have that

$$\begin{aligned}\log^{(*n-1)} A(n+1, m) &= \log^{(*n-1)} A(n, A(n+1, m-1)) \\ &= A(n+1, m-1)\end{aligned}$$

and, in addition, that

$$\begin{aligned}\log^{(*n-1)}\Big(A(n+1, m+1) - 1\Big) &= \log^{(*n-1)}\Big(A(n, A(n+1, m)) - 1\Big) \\ &= A(n+1, m) - 1.\end{aligned}$$

The last equality is justified by the induction hypothesis that $A(n, A(n+1, m))$ is the least value i such that $\log^{(*n-1)}(i) = A(n+1, m)$. Thus we have shown that, if $A(n+1, m) \leq k < A(n+1, m+1)$, then $A(n+1, m-1) \leq \log^{(*n-1)}(k) < A(n+1, m)$. If follows that, if $A(n+1, m) \leq k < A(n+1, m+1)$, then $\log^{(*n-1)}$ must be applied exactly m times to k to yield a value $< 2 = A(n+1, 1)$; i.e., that $\log^{(*n)}(k) = m$. □

Proposition 2 *For $i \geq 1$, $\log^{(*i-1)}(i) \leq 3$.*

Proof First observe that for all $i \geq 0$ and $x \geq 1$, we have $\log^{(*i)}(x) < x$. And for $i \geq 0$, $\log^{(*i)}(3) < 2$. We now prove the proposition by induction on i. The base case, $i = 1$, is obvious. For the induction step, we have that $\log^{(*i-1)}(i+1) \leq i$ by the first observation,and $\log^{(*i-1)}(i) \leq 3$ by the induction hypothesis. Thus three applications of $\log^{(*i-1)}$ suffice to take $i+1$ to a value less than 2. Hence $\log^{(*i)}(i+1) \leq 3$. □

Definition The *inverse Ackermann* function α is defined so that $\alpha(n)$ is equal to the least value of i such that $A(i, i) > n$. Equivalently, $\alpha(n)$ is equal to the least i such that $\log^{(*i-1)} n < i$.

Main Theorem 3 *Let $i \geq 0$. If the directed graph G is a tree then the serial transitive closure problem has a solution with $O(n + m \log^{(*i)} m)$ steps.*

Main Theorem 4 *If the directed graph G is a tree then the serial transitive closure problem has a solution with $O((n + m) \cdot \alpha(m))$ steps.*

Another definition of the inverse Ackermann function has been given by Tarjan [13] who defines

$$\alpha(m,n) \;=\; \text{least } i \geq 1 \text{ s.t. } A(i, 4\lceil m/n \rceil) > \log n.$$

We shall also prove below that for G a tree, the Serial Transitive Closure has a solution with $O((n+m)\alpha(n+m,m))$ steps.

Theorem 7 below is a restatement of Theorem 3 with explicit constants. The rest of our upper bounds will be corollaries of Theorem 7.

For the proofs of our main theorems we may assume without loss of generality that G is a rooted tree with edges pointing away from the root. We always picture trees with the root at the top, except in the special case of one-trees, which have fanout 1, the root is to the left and edges point to the right. The concepts of *child, father, ancestor* and *descendent* are defined as usual. The *size* of a tree is defined to be the number of edges in the tree (not the number of nodes). If a tree has e edges, then it has exactly $e+1$ nodes. We define a *subtree* of a tree T to be any connected subset of the edges of T and their endpoints so that, such that, for all nodes X and Y in T, if X and Y have the same father, then X is in the subtree iff Y is in the subtree. A subtree is *unscarred* if it consists of all the edges below a given node in the tree. A subtree S may also be obtained by first removing some set of unscarred subtrees and then letting S consist of all the remaining edges below some given remaining node: S is said to have a *scar* at any of its leaf nodes which are roots of earlier removed (nontrivial) subtrees. Two subtrees are said to be *disjoint* if they have no edges in common; disjoint subtrees may share a single node since the root of one may be a scar of the other. If X is a node in T, then T_X denotes the subtree of T rooted at X. The *immediate subtrees* of a tree T are the maximal proper subtrees of T, i.e., the trees T_X for X a child of the root of T. The following lemma is well-known: see, e.g., Brent [5].

Lemma 5 *Let $N \geq 0$ and T be a tree with $\geq N$ edges. Then there is an unscarred subtree of T which has size $\geq N$ edges such that each of its immediate subtrees has $< N$ edges.*

Lemma 5 is easily proved by taking a minimal subtree with $\geq N$ edges. The next theorem restates and strengthens the case $i = 0$ of Theorem 3 with fairly tight bounds on the constants. The *log* function is base two.

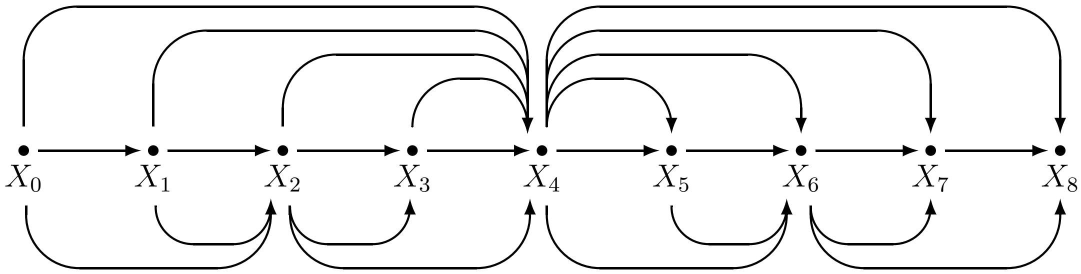


Figure 1

Theorem 6 *If the directed graph G is a tree then the serial transitive closure problem has a solution with $n + m \cdot \lfloor \log m \rfloor$ closure steps.*

Proof We will first derive $\leq m\lfloor \log m \rfloor$ edges, called *auxiliary edges*; each auxiliary edge will be obtained with a single closure step. The choice of auxiliary edges is independent of the closure edges; however, from the edges in G and the auxiliary edges each closure edge can be obtained with (at most) one additional closure step.

For illustration purposes, we first prove the theorem for G a one-tree and then do the general case. Although the general case includes the linear case, the proof of the linear case presents the main ideas more clearly.

Linear Case: Assume G is a one-tree; that is, each node except the leaf has a single child. In this case we may assume the nodes of G are named $X_0, \ldots, X_m$ and that the edges of G are just $X_i \rightarrow X_{i+1}$ for $0 \leq i < m$. The auxiliary edges will be derived in rounds, the first round will in essence split G into two subtrees of $m/2$ edges, the second round splits G into four subtrees of $m/4$ edges, etc., for a total of $\lceil \log m \rceil - 1$ rounds. The process is illustrated for the case $m = 8$ in Figure 1; the upper edges of Figure 1 are derived in the first round and the lower edges in the second round.

Round 1: $X_{\lfloor m/2 \rfloor}$ is the midpoint of the one-tree G. The auxiliary edges added in round 1 are the edges of the form $X_j \rightarrow X_{\lfloor m/2 \rfloor}$ for $0 \leq j < \lfloor m/2 \rfloor$ and the edges of the form $X_{\lfloor m/2 \rfloor} \rightarrow X_k$ for $m/2 < k \leq m$. There are exactly m such edges and they can derived with m closure steps if we derive them in the right order; namely, letting j range from $\lfloor m/2 \rfloor - 1$ down to 0 and letting k range from $\lfloor m/2 \rfloor + 1$ up to m. (Actually only $m - 2$ closure steps are needed since two of the auxiliary edges are already in G.)

Round 2: Round 1 split G into two halves; the midpoints of these two halves are $X_{\lfloor m/4 \rfloor}$ and $X_{\lfloor 3m/4 \rfloor}$. In round two, auxiliary edges to and from these midpoints are derived. Namely, (1) the edges $X_j \to X_{\lfloor m/4 \rfloor}$ with $j < \lfloor m/4 \rfloor$, and (2) the edges $X_{\lfloor m/4 \rfloor} \to X_j$ with $\lfloor m/4 \rfloor < j \le \lfloor m/2 \rfloor$, and (3) the edges $X_j \to X_{\lfloor 3m/4 \rfloor}$ with $\lfloor m/2 \rfloor \le j < \lfloor 3m/4 \rfloor$, and (4) the edges $X_{\lfloor 3m/4 \rfloor} \to X_j$ with $\lfloor 3m/4 \rfloor < j \le m$. There are m such edges and by deriving them in the right order each can be obtained with a single closure step. (Again, taking into account duplicate edges, fewer than m closure steps are needed for round 2.)

Round ℓ: For round number ℓ we add auxiliary edges incident on the nodes $X_{\lfloor k \cdot m/2^\ell \rfloor}$ for odd values of k. Specifically, for each odd value $k < 2^\ell$ the following auxiliary edges are derived: (1) the edges $X_j \to X_{\lfloor k \cdot m/2^\ell \rfloor}$ for $\lfloor (k-1)m/2^\ell \rfloor \le j < \lfloor k \cdot m/2^\ell \rfloor$ and (2) the edges $X_{\lfloor k \cdot m/2^\ell \rfloor} \to X_j$ for $\lfloor k \cdot m/2^\ell \rfloor < j \le \lfloor (k+1)m/2^\ell \rfloor$. Again, there are exactly m such edges (some of them duplicates of edges from G and edges from earlier rounds); this is seen by using the obvious one-to-one correspondence with the edges of G. And by deriving them in the right order, each auxiliary edge is obtained with a single closure step.

Since there are exactly $\lceil \log m \rceil - 1$ rounds and fewer than m closure steps are needed in each round, it is clear that there are $\le m \lfloor \log m \rfloor$ auxiliary edges and that the number of closure steps so far is bounded by $m \lfloor \log m \rfloor$.

Now we claim that each of the n closure edges can be obtained with at most a single closure step from the $m \lfloor \log m \rfloor$ auxiliary edges (of course some of the closure edges may also be auxiliary edges). To prove this, suppose $X_i \to X_j$ is a closure edge; of course, $i + 1 < j$. Find the least value of ℓ such that for some odd k, $i \le \lfloor k \cdot m/2^\ell \rfloor \le j$. If either of the inequalities are actually equalities, then $X_i \to X_j$ is an auxiliary edge added in round ℓ and no additional closure step is needed. If both inequalities are strict, then

$X_i \to X_{\lfloor k \cdot m/2^\ell \rfloor}$ and $X_{\lfloor k \cdot m/2^\ell \rfloor} \to X_j$ are both auxiliary edges and from these the closure edge $X_i \to X_j$ can be derived with one closure step.

It follows that all the closure and auxiliary edges are derived with fewer than $n + m\lfloor \log m \rfloor$ closure steps and Theorem 6 is proved for G a one-tree.

General Case: The proof of Theorem 6 for G a tree uses a construction similar to the proof of the linear case. For the general case, we shall use Lemma 5 to split G into multiple subtrees of size less than half the size of G (one of these is scarred); then we similarily split these subtrees into subtrees of size less than one quarter the size of G, etc. As in the linear case, we derive auxiliary edges of G as we split G into subtrees; this process will be done in $\leq \log m$ rounds.

Round 1: By Lemma 5 there is a node X in G such that G_X has $\geq m/2$ edges, but the immediate subtrees of G_X have size $< m/2$ edges. Let $G_1, \ldots, G_k$ be the immediate subtrees of G_X. Let G_0 be the tree obtained by removing G_X from G; i.e., G_0 is the scarred subtree with root at the root of G and with a single scar at X.

During round 1, the following auxiliary edges are derived: (1) for each ancestor Y of X the edge $Y \to X$ is an auxiliary edge, and (2) for each descendent Y of X the edge $X \to Y$ is an auxiliary edge. By deriving auxiliary edges in the correct order (namely, shorter edges first), only one closure step is needed for each auxiliary edge. There are at most m auxiliary edges and thus fewer than m closure steps are needed in round 1.

The subtrees $G_0, \ldots, G_k$ are disjoint and partition the nodes of G. They will be treated in the next round.

The total number of closure steps used to derive auxiliary edges in round 2 is less than $\sum m_i < m$.

Round $\boldsymbol{\ell}$: The previous round $\ell - 1$ resulted in G being split into multiple, disjoint subtrees of size $\leq m/2^{\ell-1}$. In round ℓ, we separately consider each such subtree H which is of size $m_H \geq 2$ and process it in the manner of round 1. Since the subtree H is of size $m_H \geq 2$, Lemma 5 gives a node X in H such that H_X has size $\geq m_H/2$ and each of

H_X's immediate subtrees have size $< m_H/2$. Now auxiliary edges are added from each ancestor of X in H to X and from X to each of its descendents in H; there are $\le m_H$ such auxiliary edges and each can be added with at most one closure step. The immediate subtrees of H_X and the subtree H with H_X removed have size $\le m_H/2$ and will be treated in next round.

Since the total size of all the disjoint subtrees is no more than m edges, fewer than m closure steps are used in this round.

The process of adding auxiliary edges ends when all the subtrees being considered have size < 2; namely after no more than $\lfloor \log m \rfloor$ rounds. Thus at most $m\lfloor \log m \rfloor$ closure steps are needed for deriving auxiliary edges.

As in the linear case, each of the n closure edges can be obtained with at most a single closure step from the $m\lfloor \log m \rfloor$ auxiliary edges. To prove this, suppose Y and Z are nodes in G and $Y \to Z$ is a closure edge; of course, Y is an ancestor of Z. Find the greatest value of ℓ, such that Y and Z are in the same subtree H considered during round ℓ. Of course, the nodes Y and Z are in different subtrees in the next round. Hence the node X chosen to split subtree H in round ℓ has Y as an ancestor and Z as a descendent (or possibly, $X = Y$ and Z is a descendent of X). Thus the edges $Y \to X$ and $X \to Z$ are auxiliary edges derived during round ℓ and the closure edge can be added with a single further closure step.

Thus the total number of inference steps needed for a solution of the serial transitive closure problem is less than $n + m\lfloor \log m \rfloor$ and Theorem 6 is proved. $\Box$

The rest of proof of Theorem 3 proceeds by proving the following theorem by induction on i:

Theorem 7 *Let $i \ge 0$. If the directed graph G is a tree then the serial transitive closure problem has a solution with $(1+2i)(n + m\log^{(*i)} m)$ steps.*

Proof When $i = 0$, the theorem is just a restatement of Theorem 6. So fix $i \ge 1$ and assume the theorem holds for $i-1$. We shall prove the theorem by splitting G into subtrees of size $\log^{(*i-1)} m$, adding auxiliary edges and using the auxiliary edges to derive some of the closure edges; we shall iterate this

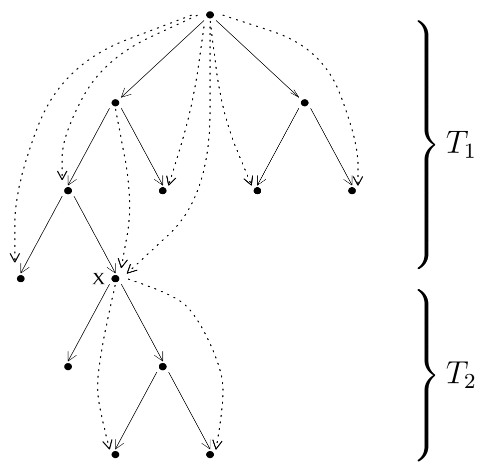


Figure 2

Node X is the root of T_2 and a scar of T_1. Dotted edges are the auxiliary edges derived in Step 1. The edges of the second kind are the two dotted edges with head X; one of these is also of the first kind.

---

process $\log^{(*i)} m$ many times after which the subtrees all have size ≤ 1. The derivation of the closure edges from the auxiliary edges will depend on the induction hypothesis.

Let us begin by describing the reduction process which will be used iteratively. The input to the reduction process is a subtree T of G; we assume T has $M > 1$ edges. The output of the reduction process will consist of a set of node-disjoint subtrees of T and the derivation of the closure edges whose endpoints are in T but are in different subtrees output by T. The reduction process has three steps:

Step 1: In the first step, T is partitioned into subtrees and auxiliary edges are derived:

By iteratively applying Lemma 5, T can be split into a finite set of subtrees $T_0, \ldots, T_k$ so that (1) the edges of the T_j's partition the edges of T, and (2) for each $j > 0$, T_j has size $\geq \log^{(*i-1)} M$ edges and (3) for all $j \geq 0$, each immediate subtree of T_j has $< \log^{(*i-1)} M$ edges, and (4) T_0 has root at at the root of T and the rest of the T_j's have a root which is a scar of another subtree in the partition. Clearly there will be at most $\left\lceil M/\log^{(*i-1)} M \right\rceil$ many subtrees in the partition.

The following auxiliary edges are derived in Step 1: (1) for each T_j with root X, the edges $X \to Z$ for all other nodes Z of T_j are *auxiliary edges of the first kind*; and (2) for each j and p such that the root Y of T_p is a scar of T_j, the edges $X \to Y$ for all ancestors X of Y in T_j are *auxiliary edges of the second kind*. Figure 2 illustrates the choice of auxiliary edges. It is easy to see that by deriving shorter edges first, each auxiliary edge can be derived by single closure step. Further, we claim that there are $\leq 2M$ auxiliary edges. It is easy to see that there $\leq M$ auxiliary edges of the first kind, since each node in T is at the head of at most one such auxiliary edge. To bound the edges of the second kind, note that if T_j has the root Y of T_p as a scar then the ancestors of Y in T_j consist of the root of T_j and some of the nodes in one of the immediate subtrees of T_j. The edge from the root of T_j to Y is also an edge of the first kind and has already been derived. Hence there are $< \log^{(*i-1)} M$ auxiliary edges from nodes inside T_j to Y. Also, the root of T_p can not be the root of T (i.e., $p \neq 0$) so there are at most $M/\log^{(*i-1)} M$ different trees T_p to consider. Taking the product of the number of subtrees and the number of edges, we have that there are less than M auxiliary edges of the second kind.

Step 2: In this step we merely describe the output of the reduction process. The *output trees* are precisely the set of immediate subtrees of $T_0, \ldots, T_k$. Note that the output trees are disjoint and partition the nodes of T other than the root node of T, but do not contain all the edges of T. Each output tree has $< \log^{(*i-1)} M$ edges.

Step 3: In the third step we derive every closure edge $X \to Y$ with X and Y in T but in different output trees. Let N be the number of such closure edges. We derive these closure edges by setting up a new instance of the serial transitive closure problem and applying the induction hypothesis. The new instance will consist of a directed graph G' which has as nodes the roots of the trees $T_0, \ldots, T_k$ and has as edges the auxiliary edges from Step 1 which connect these roots. The closure edges of the new instance are the edges $X' \to Y'$ which are obtained by the following method: for each closure edge $X \to Y$ (of the original problem) such that X is a node in T_j and Y is a node in T_p with $j \neq p$, let Y' be the root of T_p and let X' be the (scarred) leaf of T_j such that Y is a descendent of X'. It will be important that $X \to X'$ and $Y' \to Y$ are auxiliary edges (of the second and first kind respectively).

Clearly the new instance of the serial transitive closure problem has $< M/\log^{(*i-1)} M$ edges in G' and $\le N$ closure edges. By the induction hypothesis, it has a solution of size less than or equal to

$$(1+2(i-1))\left[N+\frac{M}{\log^{(*i-1)} M}\log^{(*i-1)}\left(\frac{M}{\log^{(*i-1)} M}\right)\right]$$

which is trivially bounded by

$$(1+2(i-1))\cdot[N+M].$$

Given a solution to the new serial transitive closure problem, for all X, X', Y and Y' as above, we can derive the closure edge $X \to Y$ in two closure steps from the auxiliary edges $X \to X'$ and $Y' \to Y$ and the closure edge $X' \to Y'$ of the new problem.

To conclude the description of the reduction process, we note that the total number of closure steps needed in the reduction process is bounded by

$$2M+(1+2(i-1))(N+M)+2N,$$

which is more suggestively written as

$$(1+2i)(N+M).$$

The overall procedure for proving Theorem 7 can now be very simply explained in terms of iterating the above reduction process:

Round 1: Apply the reduction process to the whole tree G. This derives n_1 closure edges (n_1 is the value of N from the reduction process) and outputs a set of subtrees of G which partition the non-root nodes of G and are each of size $< \log^{(*i-1)} m$ edges. The total number of closure steps in round 1 is bounded by

$$(1+2i)(n_1+m).$$

Round $\boldsymbol{\ell}$: The previous round generated a set of node-disjoint subtrees each of size less than

$$\underbrace{\log^{(*i-1)}(\log^{(*i-1)}(\cdots(\log^{(*i-1)}}_{\ell-1\text{ times}}(m))\cdots)).$$

Apply the reduction process (steps 1-3) to all of these subtrees which contain more than one edge; the overall result is that some number n_ℓ of closure edges are derived and that a set of node-disjoint output trees each of size less than

$$\underbrace{\log^{(*i-1)}(\log^{(*i-1)}(\cdots(\log^{(*i-1)}}_{\ell \text{ times}}(m))\cdots))$$

is generated. The total number of closure steps in round ℓ is less than

$$(1+2i)(n_\ell + m).$$

The rounds are iterated until all the subtrees have size ≤ 1; namely, in no more than $\log^{(*i)} m$ rounds. At the end every closure edge has been derived. The total number of closure steps used is bounded by

$$\sum_{\ell=1}^{\log^{(*i)} m} (1+2i)(n_\ell + m)$$

and since $\sum n_\ell = n$, the total number of closure steps is bounded by

$$(1+2i)(n + m \log^{(*i)} m).$$

That completes the proofs of Theorems 7 and 3. $\Box$

We are now ready to prove Main Theorem 4:

Proof By Theorem 7, the serial transitive closure problem for the graph G has a solution with $O((1+2i)(n+m\log^{(*i)} m)$ steps, for any value of i. Let $i = \alpha(m)$: by the definition of the function α, we have $\log^{(*i-1)} m < i$. By Proposition 2, it follows that $\log^{(*i)} m \leq 4$. Hence the serial transitive closure problem of G has a solution of size bounded by

$$(1+2\alpha(m))(n+4m) \quad = \quad O((n+m)\alpha(m)).$$

Q.E.D. Theorem 4.

We next give a bound on the number of steps in terms of Tarjan's inverse Ackermann function:

Corollary 8 *If the directed graph G is a tree then the serial transitive closure problem has a solution with $O((n+m)\alpha(n+m,m))$.*

Proof We argue similarly to the above proof, but now let i equal $\max\{1, \alpha(n+m,m)\}$. Thus, by the definition of $\alpha(,)$, we have $A(i, 4\lceil\frac{n+m}{m}\rceil) > \log m$. By Proposition 1, $\log^{(*i-1)}(\log m) < 4\lceil\frac{n+m}{m}\rceil$, and hence $\log^{(*i)}(\log m) < 4\lceil\frac{n+m}{m}\rceil$. Since $i \geq 1$, $\log^{(*i)}(m) \leq 4\lceil\frac{n+m}{m}\rceil$. Thus, by Theorem 7, the serial transitive closure problem has a solution with size bounded by

$$(1+2\alpha(n+m,m))\left(n+m\left(4\left\lceil\frac{n+m}{m}\right\rceil\right)\right) \quad = \quad O((n+m)(\alpha(n+m,m)))$$

whenever $\alpha(n+m,m) > 0$. $\square$

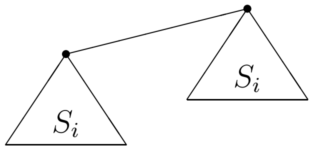
3 Lower Bounds

3.1 The Lower Bound for Trees

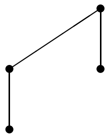
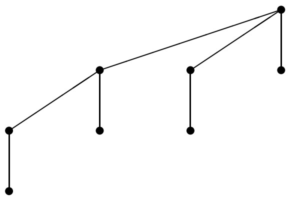
In this section we prove that our method of solving the Serial Transitive Closure problem for trees is optimal to within a constant factor; in particular, linear size solutions are not always possible. Our proof is based on a theorem of Tarjan [13] giving a lower bound on the worst case runtime of an algorithm for the union-find problem. Tarjan's lower bound applied only to a particular type of algorithm for the union-find problem (Tarjan [14], Fredman-Saks [11] and La Poutré [12] have since given lower bounds for a much wider class of algorithms). Essentially, Tarjan gave a lower bound for algorithms that rely exclusively on manipulating pointers (with certain constraints). Since the Serial Transitive Closure problem is framed so as to deal only with edges and closure steps, it is not so surprising that we are able to modify Tarjan's original construction so as to give a lower bound for the Serial Transitive Closure problem.

Our lower bound is obtained by constructing instances of the Serial Transitive Closure problem which require at least $\frac{1}{25}(n+m)\cdot\alpha(n+m)$ closure steps in any solution. The construction is based on Tarjan's lower bound; we shall review the relevant definitions and constructions but do not repeat Tarjan's proof.

Following Tarjan, define S_1 be the tree with two nodes: the root and one leaf. Define S_{i+1} to be the tree constructed by making two copies of S_i and making the root of one copy a child of the root of the other copy; pictorially, S_{i+1} is:



For example, S_2 and S_3 are the trees

S_2 :  S_3 : 

Clearly S_i has 2^i nodes of which 2^{i-1} are leaves. Let T be a tree. A *g-find* is a pair of nodes (a, b) in T such that b is a leaf and a is an ancestor of b. A (permissible) sequence of g-finds is a sequence (a_i, b_i) of g-finds such that the b_i's are distinct and such that, for all $i < j$, a_j is not a descendent of a_i. G is a *shortcut graph* of T if G is a directed graph on the nodes of T with each edge of T contained as a directed edge in T such that, if T is viewed as a directed graph with edges directed downward, the graph G is a subgraph of the transitive closure of T. If (a_i, b_i), $i = 1, \ldots, N$, is a sequence of g-finds, then the associated shortcut graphs $G_0, \ldots, G_N$ are defined by letting G_0 have the edges from T (directed downwards), and letting G_i be G_{i-1} plus all edges (c, d) such that c and d are on the path in T from a_i to b_i and such that d is a descendent of c. The *cost* of a particular g-find (a_i, b_i) in the sequence is defined to be the distance from a_i to b_i in the shortcut graph G_{i-1}.

Tarjan defines a version of the Ackermann function which is slightly different from ours; however, it is easy to see that Tarjan's Ackerman function equals ours, except for the values of $A(i, 0)$.

The next theorem of Tarjan will lead to our lower bounds:

Theorem 9 *For all $k \geq 1$ and $i > A(4k, 4)$, there is a sequence of 2^{i-2} g-finds in S_i such that each g-find has cost $\geq k$.*

For a proof of Theorem 9, apply Theorem 15 of [13] with $s = 1$. We can now prove our lower bound:

Theorem 10 *Let $k \geq 1$. There is an instance of the Serial Transitive Closure problem in which the graph G has $m = 2^{A(4k,4)+1}$ nodes and for which there are $n = 2^{A(4k,4)-1}$ closure edges such that any solution of of the Serial Transitive Closure problem requires at least $(k-1) \cdot 2^{A(4k,4)-1}$ closure steps.*

Proof Let the directed graph G be the tree S_i (with edges directed downwards), where $i = A(4k, 4) + 1$. Clearly G has $m = 2^{A(4k,4)+1}$ nodes. The instance of the Serial Transitive Closure problem is obtained by taking as closure edges, the $n = 2^{A(4k,4)-1}$ g-finds given by Theorem 9. Let $G_0, \ldots, G_n$ be the associated shortcut graphs.

We now must show that, because each g-find has cost k, any solution to this Serial Transitive Closure problem requires at least $k-1$ steps per closure edge. To prove this, let $e_1, \ldots, e_s$ be a sequence of edges which comprise a solution to the Serial Transitive Closure problem. For each $i \leq s$, define the *rank* of e_i to be the least value r_i such that e_i is an edge in G_{r_i}; let $r_i = n+1$ if e_i is not in G_n. Now reorder the edges $e_1, \ldots, e_s$ according to their rank, keeping edges of the same rank in the same relative order. We claim that the reordered edges are also a solution to the Serial Transitive Closure problem. This claim is easily proved by noting that if an edge $e_i = (c_1, c_2)$ is inferred from edges (c_1, c_3) and (c_3, c_2) and if e_i is in G_{r_i} then the other two edges must also be in G_{r_i} (by the definition of the shortcut graphs) and thus have ranks $\leq r_i$. Finally, we claim that that for all $r \leq n$ there are at least $k-1$ edges with rank r. To prove this, consider the edges of rank r; these edges are inferred by closure steps from the edges of rank $< r$ and one of the edges is the r-th closure edge. But because the cost of the r-th g-find is $\geq k$, at least $k-1$ closure steps are required to derive the r-th closure edge from the edges of rank $< r$.
Q.E.D. Theorem 10

Let $f(m, n)$ be the maximum number of edges required to solve Serial Transitive Closure problems for **trees** with m edges and n closure edges.

Lemma 11 *For infinitely many values of* n, $f(4n-1,n) \geq \dfrac{n}{5}\alpha(5n)$.

Proof Let $k \geq 5$ and $n = 2^{A(4k,4)-1}$ and $i = A(4k,4)+1$. Since S_i has $4n$ nodes, it has $m = 4n-1$ edges. By Theorem 10, $f(m,n) \geq (k-1)n$. It will suffice to show $\alpha(5n) \leq 5k-5$. Now,

$$\begin{aligned} A(5k-5,5k-5) &> A(5k-5,5) \\ &= A(5k-6, A(5k-5,4)) \\ &\geq A(5k-6, A(4k,4)) \\ &> 2^{A(4k,4)+2} \qquad\qquad \text{since } 5k-6 > 2 \\ &> 5n \end{aligned}$$

from whence, by the definition of α, $\alpha(5n) \leq 5k-5$. $\square$

Since $m = 4n-1$ and hence $5n > m+n$, the proof of Lemma 11 immediately implies:

Theorem 12 $f(m,n) \geq \frac{1}{25}(n+m)\cdot\alpha(n+m)$ *for infinitely many values of* n *with* $m = 4n-1$.

3.2 Relations to Weak Superconcentrators

The above lower bound for the serial transitive closure problem applies to the case where G is a general tree. Essentially the same lower bound applies to binary trees, since any tree can be converted into a binary tree by expanding any node that has more than two children into multiple nodes with two children each. This expansion will at most double the number of edges in the tree and will not make the solutions to the serial transitive closure problem smaller.

However, we know no way to extend the above argument to give a nonlinear lower bound for the size of solutions to the serial transitive closure problem for the case where the directed graph is linear, i.e., the case where each node in G has at most one incoming edge and at most one outgoing edge (this was the simple case considered in the preliminary portion of the proof of theorem 6). On the other hand, there is a connection between constant depth weak superconcentrators and our proof of the main theorems, that suggests that our upper bound of $n \cdot \alpha(n)$ is the best that can be obtained for the linear case with our techniques. Namely, our proof of Theorem 7, for the case where G is linear, implicitly contains a construction of weak superconcentrators of optimal size (to within a constant factor). We shall outline this construction below; a similar construction has already been given by [6, 7].

It is also possible to use Yao's lower bound on the size of (t, m)-structures [15] to get a nonlinear lower bound on the number of auxiliary edges needed for our construction. However, the lower bounded obtained in this way is not as good as the lower bound we obtain below via weak superconcentrators.

We recall the definition of a weak superconcentrator. A network is a directed graph with nodes $a_0, \ldots, a_m$ designated as inputs and with nodes $b_0, \ldots, b_m$ designated as outputs. The inputs have no incoming edges and the outputs have no outgoing edges. A network is *synchronous* if it is possible to associate with each node a *depth*, such that each input has depth 0 and each edge in the network goes from a depth d node to a depth $d+1$ node, and such that the output nodes have a common depth. The depth of a synchronous circuit is defined to be the depth of the output nodes. The network is said to be a *weak* $(m+1)$*-superconcentrator* if the following property holds: if $0 \leq i_1 \leq j_1 < i_2 \leq j_2 < \ldots i_k \leq j_k \leq m$ are integers, then the network contains k many node-disjoint paths from a_{i_r} to b_{j_r}, for $r = 1, \ldots, k$. It is a theorem of Dolev, Dwork, Pippenger and Widgerson [10] that depth $2i+2$ synchronous weak $(m+1)$-superconcentrators must have at least $\Omega(m \log^{(*i)}(m))$ many nodes.† (Some related, but weaker, lower bounds are given by Bodlaender-Tel-Santoro [1]).

Recall that in the proof of Theorem 7, we added auxiliary edges independently of the choice of closure edges and then derived the closure edges with the aid of the auxiliary edges. The net effect is, after the proof by induction has been unwound, that there were a total of $(1+2i)m\log^{(*i)}(m)$ auxiliary edges added such that, for any two nodes A and B with A an ancestor of B, there exists a path from A to B of length at most $(2i+2)$.‡ This path consists

†The Ackermann function $A(i,x)$ used in [10] is the same as Tarjan's Ackermann function, and hence is the same as our $A(i,x)$ for $x \geq 1$. The function $\lambda(i,x)$ used in [10] satisfies $\log^{(*i)}(x) = \lambda(i+1,x)$ for all $i \geq 0$ and $x > 0$.

‡To justify the length $(2i+2)$ recall that at most $(2i+1)$ closure steps were needed

entirely of auxiliary edges and edges in G, of course. We claim that these auxiliary edges can be made into a weak supercentrator of depth $(2i+2)$. To do this we make $(2i+3)$ disjoint copies of G, called $G_0, \ldots, G_{2i+2}$ and construct a network on the nodes in the union of these graphs.

To construct the weak $(m+1)$-superconcentrator of depth $2i+2$, let G be a linear tree with nodes $X_0, \ldots, X_m$ and edges $X_j \to X_{j+1}$, for $0 \le j \le m$. Form $(2i+3)$ disjoint copies of G denoted $G_0, \ldots, G_{2i+2}$ and let X_j^k be the copy of X_j in G_k. The weak superconcentrator will be a directed graph on the nodes X_j^k; the X_j^0's are the input nodes, the X_j^{2i+2}'s are the output nodes and each X_j^k will be of depth k. First, the weak superconcentrator will have the edges $X_j^k \to X_j^{k+1}$ for all j, k. There are obviously $2(i+1)(m+1)$ edges of this first kind. Second, the weak superconcentrator will have edges corresponding to the auxiliary edges from the proofs of Theorems 6 and 7. Recall that Theorem 7 was proved by induction on i; by considering the auxiliary edges added to G in the proof of Theorem 6 and in all the induction steps of the proof of Theorem 7, it is easy to see that there are, in total, $\le (1+2i)m\log^{(*i)} m$ auxiliary edges added to G. We must explain how the auxiliary edges are translated into edges in the weak superconcentrator — the essential idea is that each induction step of the proof of Theorem 7 adds two more layers of connections in the weak superconcentrator. To make this more precise, recall that in proving Theorem 7 for i we added $2m\log^{(*i)}(m)$ many auxiliary edges and invoked Theorem 7 for $i-1$ multiple times. If this proof by induction is 'unwound', then each of these invocations of Theorem 7 for $i-1$ adds many auxiliary edges and further invokes Theorem 7 for $i-2$ multiple times, etc. The proof of Theorem 7 for $i=0$ was just the proof of Theorem 6 (especially, the linear case); this of course also added auxilliary edges. Now consider all the auxiliary edges that are added during the unwinding of the proof of Theorem 7 for i; each auxiliary edge is associated with a value $i' \le i$ by considering which case of Theorem 7 introduced the auxiliary edge (edges can be associated with more that one value of i' and in this case we count the edge multiple times). The weak superconcentrator contains all edges $X_j^{i-i'} \to X_s^{i-i'+1}$ and $X_j^{i+i'+1} \to X_s^{i+i\prime+2}$ where $X_j \to X_s$ is an auxiliary edge associated with i'. Since each auxiliary edge is put twice into the weak

to derive any possible closure edge from the auxiliary edges — this corresponds to a path of length $(2i+2)$.

superconcentrator[§], there are $2(1+2i)m\log^{(*i)}(m)$ such edges put into the network.

Thus, in total, there are only $O(i \cdot m \cdot \log^{(*i)} m)$ edges in the weak $(m+1)$-superconcentrator of depth $2i+2$. Hence our method of proof for the upper bound of Theorem 7 constructs optimal size weak superconcentrators and it seems, therefore, that no simple modification of the proof method can give a better upper bound for the linear case. Nonetheless, it is open whether our upper bound for the linear case of the Serial Transitive Closure problem is optimal — for instance, it might be possible to give an improved construction by not choosing the auxiliary edges independently of the closure edges. The point is that it is not *a priori* necessary to construct a weak superconcentrator in order to get a solution to a particular instance of the Serial Transitive Closure problem. The lower bound methods of [13] and [10] do not appear to give a definitive lower bound for the linear case.

Acknowledgements

We wish to thank P. Pudlák for suggesting the connection to weak superconcentrators. We also thank one of the referees for bringing the works of Yao, Chazelle, and Bodlaender-Tel-Santoro to our attention and suggesting some improvements to the paper, including Theorem 8.

References

[1] H. L. Bodlaender, G. Tel, and N. Santoro, *Trade-offs in non-reversing diameter*, Tech. Rep. RUU-CS-89-22, Dept. of Computer Science, Utrecht Univ., 1989.

[2] M. L. Bonet, *The Lengths of Propositional Proofs and the Deduction Rule*, PhD thesis, U.C. Berkeley, 1991.

[3] M. L. Bonet and S. R. Buss, *On the deduction rule and the number of proof lines*, in Proceedings Sixth Annual IEEE Symposium on Logic in Computer Science, 1991, pp. 286–297.

[§] It is possible to improve the construction to put each auxiliary edge only once in the weak superconcentrator.

[4] ——, *The deduction rule and linear and near-linear proof simulations*, Journal of Symbolic Logic, 58 (1993), pp. 688–709.

[5] R. P. Brent, *The parallel evaluation of general arithmetic expressions*, J. Assoc. Comput. Mach., 21 (1974), pp. 201–206.

[6] A. K. Chandra, S. Fortune, and R. Lipton, *Lower bounds for constant depth circuits for prefix problems*, in 10th International Colloquium on Automata, Languages and Progamming, Springer-Verlag Lecture Notes in Computer Science #154, 1983, pp. 109–117.

[7] ——, *Unbounded fan-in circuits and associative functions*, in Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 1983, pp. 52–60.

[8] B. Chazelle, *Computing on a free tree via complexity-preserving mappings*, Algorithmica, (1987), pp. 337–361.

[9] M. D. Davis and E. J. Weyuker, *Computability, Complexity, and Languages*, Academic Press, 1983.

[10] D. Dolev, C. Dwork, N. Pippenger, and A. Widgerson, *Superconcentrators, generalizers and generalized connectors with limited depth*, in Proceedings 15-th Annual ACM Symposium on Theory of Computing, 1983, pp. 42–51.

[11] M. L. Fredman and M. E. Saks, *The cell probe complexity of dynamic data structures*, in Proceedings 21-st Annual ACM Symposium on Theory of Computing, 1989, pp. 345–354.

[12] J. L. Poutré, *Lower bounds for the union-find and the split-find problem on pointer machines*, in Proceedings of the 22th Annual ACM Symposium on Theory of Computing, 1990, pp. 34–44.

[13] R. E. Tarjan, *Efficiency of a good but not linear set union algorithm*, J. Assoc. Comput. Mach., 22 (1975), pp. 215–225.

[14] ——, *A class of algorithms which require nonlinear time to maintain disjoint sets*, J. Comput. System Sci., (1979), pp. 110–127.

[15] A. C. Yao, *Space-time tradeoffs for answering range queries (extended abstract)*, in Proceedings of the 14th Annual ACM Symposium on Theory of Computing, 1982, pp. 128–136.