A Parallel Algorithm for Constructing a Labeled Tree

Yue-Li Wang, Hon-Chan Chen, and Wei-Kai Liu

Abstract—A tree T is labeled when the n vertices are distinguished from one another by names such as $v_1, v_2, \cdots, v_n$. Two labeled trees are considered to be distinct if they have different vertex labels even though they might be isomorphic. According to Cayley's tree formula, there are n^{n-2} labeled trees on n vertices. Prüfer used a simple way to prove this formula and demonstrated that there exists a mapping between a labeled tree and a number sequence. From his proof, we can find a naive sequential algorithm which transfers a labeled tree to a number sequence and vice versa. However, it is hard to parallelize. In this paper, we shall propose an $O(\log n)$ time parallel algorithm for constructing a labeled tree by using $O(n)$ processors and $O(n \log n)$ space on the EREW PRAM computational model.

Index Terms—Cayley's tree formula, dominance counting problem, labeled trees, parallel algorithms, Prüfer mapping.

1 Introduction

A tree T is *labeled* when the n vertices are distinguished from one another by names such as $v_1, v_2, \cdots, v_n$. Two labeled trees are considered to be distinct if they have different vertex labels even though they might be isomorphic. For example, the two trees T_1 and T_2 of Fig. 1 are labeled, but T_3 is not. Moreover, T_1 and T_2 are two different labeled trees even though they are isomorphic [7].

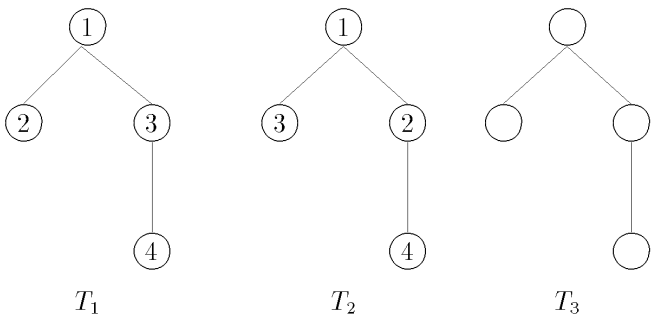


Fig. 1. Labeled and unlabeled trees.

According to Cayley's tree formula [4], there are n^{n-2} different labeled trees on n vertices. In [11], Moon introduced various proofs of Cayley's formula. In [12], Prüfer used a simple way to prove this formula. His main idea is to build a one-to-one correspondence between a labeled tree and a number sequence. He gave an algorithm which transfers a labeled tree to a number sequence and vice versa. Thus, a labeled tree can be encoded to a number sequence. This encoding scheme can be viewed as a data compression technique on trees. However, his algorithm is difficult to parallelize. In this paper, we shall propose a parallel algorithm to transfer a number sequence to a labeled tree. Our approach uses the EREW PRAM (Exclusive-Read-Exclusive-Write Parallel Random Access Machine) computational model. Our algorithm enables very fast parallel construction of random, uniformly distributed labeled tree.

The remainder of this paper is organized as follows. In Section 2, we introduce Prüfer's mapping. Some notations with their properties are introduced in Section 3. In Section 4, we introduce and analyze our parallel algorithm. Section 5 contains the concluding remarks.

2 Preliminaries

In this section, we shall briefly discuss how to obtain a labeled tree from a number sequence by using Prüfer's algorithm [6]. We shall also define some notations which will be used in the remaining sections.

A *p-sequence* is a sequence of length $n-2$ with entries from the set $\{1, 2, 3, \ldots, n\}$. Let $p_1, p_2, \ldots, p_{n-2}$ be a p-sequence. Then, sequence $s_1, s_2, \ldots, s_{n-1}$ is called an *s-sequence* if $s_i = p_i$ for $i = 1, 2, \ldots, n-2$ and $s_{n-1} = n$. For example, 7, 4, 4, 7, 5 is a p-sequence, since its length is 5 and its entries are from the set {1, 2, 3, 4, 5, 6, 7}. And, 7, 4, 4, 7, 5, 7 is an s-sequence, but 9, 2, 3, 4, 4 is not a p-sequence.

Now, we describe Prüfer's algorithm. In the following algorithm, the *degree* of vertex v, denoted by $deg(v)$, is the number of edges incident with vertex v.

Algorithm A

Input: An s-sequence $s_1, s_2, \ldots, s_{n-1}$ of length $n-1$.

Output: A labeled tree T.

Method:

Step 1. Let T be a graph with n vertices 1, 2, …, n and no edge.

Step 2. Let $deg(k) = 1 +$ (the number of times k appears in p-sequence), for $k = 1, 2, \ldots, n$.

• *The authors are with the Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan, Republic of China. E-mail: ylwang@cs.ntust.edu.tw.*

Manuscript received 11 Sept. 1995.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 101305.

Step 3. For $i = 1$ to $n - 1$ do

Let j be the least vertex such that $deg(j) = 1$. Construct an edge between vertices j and s_i. That is, let (j, s_i) be an edge of T. Set $deg(j) = 0$ and $deg(s_i) = deg(s_i) - 1$.

Step 4. The resulting graph T is a labeled tree.

End of Algorithm A

The most time-consuming step of Algorithm A is Step 3. By using a heap [8], finding the least vertex with degree one can be done in $O(log\ n)$ time. Hence, Step 3 takes $O(n \log n)$ time. The time-complexity of Algorithm A therefore is $O(n \log n)$.

Let us use an example to illustrate Algorithm A. Given an s-sequence 7, 4, 4, 7, 5, 7, the degree of each vertex i, i = 1, 2, …, 7 is shown in Table 1. Note that 7, 4, 4, 7, 5 is a p-sequence.

TABLE 1
THE DEGREE OF EACH VERTEX

k	1	2	3	4	5	6	7
$deg\ (k)$	1	1	1	3	2	1	3

According to Step 3 of Algorithm A, vertex 1 is the least vertex, with degree one when $i = 1$. An edge is added between vertices 1 and 7. Then, the degrees of both vertices are decreased by 1. Next, vertex 2 will be selected when i = 2. Vertices 2 and 4 will be incident to each other and the degrees of both vertices are decreased by 1. After the algorithm is terminated, the resulting tree is shown in Fig. 2.

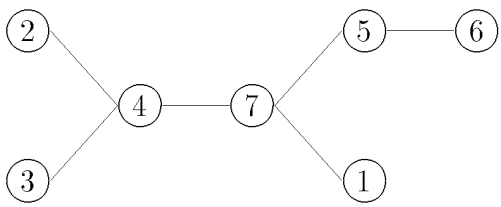


Fig. 2. The corresponding labeled tree T of p-sequence 7, 4, 4, 7, 5.

In Step 3 of Algorithm A, a vertex j will be selected in each iteration, so that an edge will be added between j and s_i. We use r_i to denote the vertex selected in iteration i. Clearly, there are totally $n - 1$ vertices which will be selected. The sequence $r_1, r_2, \ldots, r_{n-1}$ is called an *r-sequence*. In the above example, 1, 2, 3, 4, 6, 5 is the resulting r-sequence. Notice that (s_i, r_i), i = 1, 2, …, $n - 1$, is an edge of the resulting tree.

3 SOME PROPERTIES OF A P-SEQUENCE

In this section, we shall define two functions f_1 and f_2 on a p-sequence which will be used in our algorithm to construct a labeled tree. Besides, we shall describe some properties of these two functions.

For each vertex i, let $f_1(i)$ be equal to the last position of vertex i in the p-sequence. If i is not in the p-sequence, then $f_1(i) = 0$. It means that the degree of vertex i will become one at iteration $f_1(i)$ of Algorithm A. It also means that the position of vertex i in the r-sequence is after $f_1(i)$. Let $f_2(i)$ denote the number of $f_1(j)$, where $f_1(j) \le f_1(i)$ and $j < i$. It means that there are at least $f_2(i)$ vertices which appear before vertex i in the r-sequence. Obviously, the position of vertex i in the r-sequence is not equal to $f_1(i) + 1$ if $f_2(i) > f_1(i)$. For example, let 9, 6, 4, 10, 1, 5, 7, 7 be a p-sequence. Then, $f_1(1) = 5$, since the last position of vertex 1 in the p-sequence is five. The other values of $f_1(i)$, i = 2, 3, …, 10 can be seen in Table 2. Furthermore, $f_2(1) = 0$, since there exists no vertex j which has $f_1(j) \le f_1(1)$ and $j < 1$. However, $f_2(3) = 1$, since there is only one vertex (vertex 2) which has $f_1(2) \le f_1(3)$ and $2 < 3$. The other values of $f_2(i)$ are shown in Table 2.

TABLE 2
THE VALUES OF $f_1(i)$ AND $f_2(i)$ OF THE GIVEN P-SEQUENCE
9, 6, 4, 10, 1, 5, 7, 7

i	1	2	3	4	5	6	7	8	9	10
$f_1(i)$	5	0	0	3	6	2	8	0	1	4
$f_2(i)$	0	0	1	2	4	2	6	2	3	6

The following lemmas describe some properties of the above two functions.

LEMMA 1. *Given a p-sequence of length $n - 2$, let i and j be two distinct vertices with $1 \le i < j \le n$. If $f_1(i) \le f_1(j)$, then i appears before j in the corresponding r-sequence.*

PROOF. By the definition of f_1, the degrees of vertices i and j will become one at iterations $f_1(i)$ and $f_1(j)$, respectively, of Algorithm A. Since $f_1(i) \le f_1(j)$, vertex i is selected either at an iteration number less than or equal to $f_1(j)$ or it is selected after iteration $f_1(j)$. In the former case, i is certainly selected before j. In the latter case, i is still selected before vertex j, since $i < j$. Therefore, vertex i appears before vertex j in the corresponding r-sequence. □

LEMMA 2. *Given a p-sequence of length $n - 2$, let i, $1 \le i \le n - 1$, be a vertex with $f_1(i) \ge f_2(i)$. Then, the position of i in the corresponding r-sequence is $f_1(i) + 1$.*

PROOF. Obviously, this lemma holds when $f_1(i) = f_2(i) = 0$. In the following, we consider the case where $f_1(i) > 0$.

Let $U_1 = \{u \mid 1 \le u < i \text{ and } f_1(u) \le f_1(i)\}$. Note that $|U_1| = f_2(i)$, where $|U_1|$ denotes the number of vertices in the set U_1. By the definition of f_2, all of the vertices in $|U_1|$ will be selected before vertex i. Let U_2 be the set of vertices which are selected before iteration $f_1(i) + 1$ of Algorithm A and whose labels are greater than i.

If vertex i is selected at iteration $f_1(i) + 1$ of Algorithm A, then vertex i must be the least vertex of degree one at iteration $f_1(i) + 1$. For the purpose of contradiction, we assume that vertex i is not the least vertex of degree one at iteration $f_1(i) + 1$ of Algorithm A. That is, there exists at least one vertex in U_1 which is not selected before iteration $f_1(i) + 1$. It implies that $|U_1| + |U_2| \ge f_1(i) + 1$.

Let j be the latest selected vertex among the vertices in U_2 and k be its selected iteration. Clearly, $k \le f_1(i)$. Let $U_3 \subset U_1$ be the set of vertices which are not selected before iteration k. Since $k \le f_1(i)$ and there exists at least one vertex in U_1 whose degree is one at iteration $f_1(i) + 1$, U_3 is not an empty set. Note that there is no

vertex in U_3 whose degree is one before iteration k; otherwise, j would not be selected at iteration k. Moreover, there is exactly one vertex $u \in U_3$ whose degree becomes one at iteration k, i.e., $f_1(u) = k$. In each iteration after iteration k, there is, at most, one vertex in U_3 whose degree can become one. Suppose that v is the vertex in U_3 whose degree most lately becomes one. Then, $f_1(v) \geq f_1(i) + 1$, since $|U_1| + |U_2| \geq f_1(i) + 1$. It contradicts that U_1 is the set of all vertices whose labels are less than i and are of $f_1(u) \leq f_1(i)$, where $u \in U_1$. Therefore, the lemma follows. □

LEMMA 3. *Given a p-sequence of length $n-2$, let i and j be two distinct vertices with $1 \leq i < j \leq n$. If $f_1(i) < f_2(i)$ and $f_1(i) \geq f_1(j)$, then i appears before j in the corresponding r-sequence.*

PROOF. The case where $f_1(i) = f_1(j)$ occurs only when $f_1(i) = 0$. It means that i and j are not in the given p-sequence and their initial degrees are one. It is clear that, in this case, i appears before j in the r-sequence.

We now consider the other case, where $f_1(i) \geq 1$. Suppose, to the contrary, that j appears before i in the resulting r-sequence. Let j be selected at iteration k. Then, k must be less than or equal to $f_1(i)$; otherwise, j would not be the least vertex of degree one at iteration k. Let U denote the set of vertices whose labels are smaller than i and their initial degrees are one or will become one before iteration $f_1(i)$. Clearly, $|U| = f_2(i)$. There are at most $k-1$ vertices which are eligible for selection from U before iteration k of Algorithm A; otherwise, vertex j would not be selected on this iteration. It means that there are at least $|U| - (k-1)$ vertices that must become eligible (i.e., degree one) between iterations k and $f_1(i) - 1$. Therefore, the number of vertices in U, whose degrees do not become one yet before iteration k, is at least $|U| - (k-1) = f_2(i) - k + 1$. The degrees of these $f_2(i) - k + 1$ vertices must all become one between iterations k and $f_1(i) - 1$, i.e. $f_1(i) - 1 - k + 1 = f_1(i) - k$ iterations. However, it is impossible to reduce the degrees of $f_2(i) - k + 1$ vertices into one within $f_1(i) - k$ iterations since

$$\begin{aligned} & f_2(i) - k + 1 - \left(f_1(i) - k\right) \\ &= f_2(i) - k + 1 - f_1(i) + k \\ &= f_2(i) - f_1(i) + 1 \\ &> 1. \end{aligned}$$

It contradicts the assumption that the degrees of all of the vertices in U will become one before iteration $f_1(i)$. This completes the proof. □

COROLLARY 1. *Given a p-sequence of length $n-2$, let i be a vertex of $f_1(i) < f_2(i)$, where $1 \leq i \leq n$. Then, for every vertex j, $i < j \leq n$, j appears after i in the corresponding r-sequence.*

4 A Parallel Algorithm for Constructing a Labeled Tree

Given an s-sequence, if $f_1(i) \geq f_2(i)$ for a vertex i, then, according to Lemma 2, the position of vertex i in the corresponding r-sequence can be easily determined. When $f_1(i) < f_2(i)$, computing the resulting position of vertex i in the r-sequence is not so trivial by applying Corollary 1. In the following, we show how to obtain the resulting position of vertex i in the r-sequence when $f_1(i) < f_2(i)$.

Let $x_1, x_2, \cdots, x_{n-1}$ be a sequence with respect to an s-sequence of length $n-1$, where $x_i = 0$, $1 \leq i \leq n-1$, if there exists a vertex j such that $i = f_1(j) + 1$ and $f_1(j) \geq f_2(j)$; otherwise, $x_i = 1$. Then, $x_1, x_2, \cdots, x_{n-1}$ is called a *position sequence*. Let $y_1, y_2, \cdots, y_{n-1}$ be the prefix sums of a position sequence $x_1, x_2, \cdots, x_{n-1}$. That is $y_1 = x_1$ and $y_k = y_{k-1} + x_k$, for $k = 2, 3, \cdots, n-1$. A vertex v, $1 \leq v \leq n$, is called a *delayed vertex* if $f_2(v) > f_1(v)$. Assume that there are m delayed vertices for a position sequence $x_1, x_2, \cdots, x_{n-1}$. A sequence $z_1, z_2, \cdots, z_m$ is called a *delayed sequence* with respect to an s-sequence if z_i, $i = 1, 2, \cdots, m$, is a delayed vertex and $z_i < z_j$ when $i < j$. Define $f_3(z_i) = k$ if $y_k = i$ and $x_k = 1$ for the delayed vertex z_i, $i = 1, 2, \cdots, m$, where x_k is the kth element of a position sequence and y_k is its corresponding prefix sum.

For example, given an s-sequence 9, 6, 4, 10, 1, 5, 7, 7, 10, its position sequence $x_1, x_2, \cdots, x_9$ is 0, 1, 0, 0, 1, 0, 0, 1, 0. The prefix sums $y_1, y_2, \cdots, y_9$ of the position sequence are 0, 1, 1, 1, 2, 2, 2, 3, 3, respectively. The delayed sequence z_1, z_2, z_3 of the s-sequence is 3, 8, 9. Then, $f_3(3)$, $f_3(8)$, and $f_3(9)$ are 2, 5, and 8, respectively.

Now, we describe our algorithm for constructing a labeled tree as follows.

Algorithm B

Input: An s-sequence $s_1, s_2, \cdots, s_{n-1}$.

Output: A labeled tree.

Method:

Step 1. Let $T(V, E)$ be a graph with $V = \{1, 2, \ldots, n\}$ and E is an empty set, where V and E are the vertex and edge, respectively, sets of T.

Step 2. Compute $f_1(i)$ and $f_2(i)$, for $i = 1, 2, \cdots, n-1$.

Step 3. For $i = 1, 2, \cdots, n-1$
if $f_1(i) \geq f_2(i)$, then $r_{f_1(i)+1} = i$.

Step 4. For each delayed vertex i, $1 \leq i \leq n-1$,
compute $f_3(i)$ and let $r_{f_3(i)} = i$.

Step 5. For $i = 1, 2, \cdots, n-1$
$E = E \cup (s_i, r_i)$. The resulting graph T is a labeled tree.

End of Algorithm B

Steps 1, 3, and 5 can be done in $O(1)$ time by using $O(n)$ processors on an EREW PRAM model. By applying parallel sorting algorithms [9], computing $f_1(i)$, $i = 1, 2, \cdots, n-1$, in Step 2 takes $O(\log n)$ time and $O(n)$ processors. Computing $f_2(i)$, for $i = 1, 2, \cdots, n-1$, can be done in $O(\log n)$ time using $O(n)$ processors and $O(n \log n)$ space by viewing this problem in terms of the dominance counting problem [1], [2], [3], as will be described later. In Step 4, computing $f_3(i)$ for all of the delayed vertices can be done in $O(\log n)$ and $O(n)$ processors by using the parallel prefix technique [9], [10]. Notice that Step 4 only computes $f_3(i)$ for $i = 1, 2, \cdots, n-1$, since vertex n is never selected in the sequential algorithm. Therefore, Algorithm B takes $O(\log n)$ time using $O(n)$ processors and $(n \log n)$ space on an EREW PRAM model.

We also use an example to illustrate Algorithm B. Given an s-sequence 9, 6, 4, 10, 1, 5, 7, 7, 10, ten isolated vertices are labeled from 1 to 10 in Step 1. Step 2 computes $f_1(i)$ and $f_2(i)$, for $i = 1, 2, \cdots, 9$. The values of $f_1(i)$ and $f_2(i)$ are shown in Table 2. In Step 3, the positions of vertices 1, 2, 4, 5, 6, and 7 in the corresponding r-sequence can be obtained directly, since all of them have $f_1(i) \geq f_2(i)$. Thus, the positions of vertices 1, 2, 4, 5, 6, and 7 in the r-sequence are 6, 1, 4, 7, 3, and 9, respectively, since $f_1(1) = 5$, $f_1(2) = 0$, $f_1(4) = 3$, $f_1(5) = 6$, $f_1(6) = 2$, and $f_1(7) = 8$. The corresponding position sequence $x_1, x_2, \cdots, x_9$, with respect to the s-sequence, is 0, 1, 0, 0, 1, 0, 0, 1, 0. The prefix sums $y_1, y_2, \cdots, y_9$ of the position sequence are 0, 1, 1, 1, 2, 2, 2, 3, and 3, respectively. Vertices 3, 8, and 9 are delayed vertices since all of them have $f_2(i) > f_1(i)$. The delayed sequence z_1, z_2, z_3 with respect to the s-sequence is 3, 8, 9. Thus, $f_3(3)$, $f_3(8)$, and $f_3(9)$ are equal to 2, 5, and 8, respectively. The resulting r-sequence is 2, 3, 6, 4, 8, 1, 5, 9, 7. In Step 5, the edges (s_i, r_i) for $i = 1, 2, \cdots, 9$ are obtained. The resulting labeled tree is shown in Fig. 3.

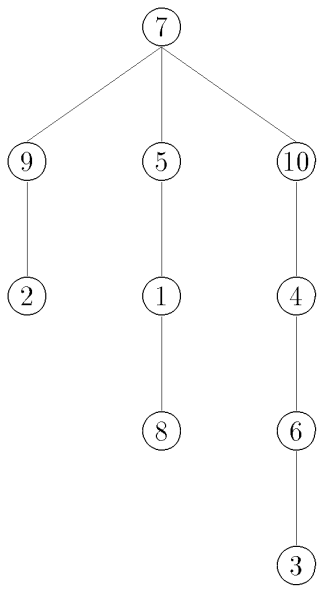


Fig. 3. The corresponding labeled tree of the s-sequence 9, 6, 4, 10, 1, 5, 7, 7, 10.

The correctness of Algorithm B is described as follows.

THEOREM 1. *The output of Algorithm B is the same as that of Algorithm A.*

PROOF. By Lemmas 1, 2, and 3, Algorithm B indeed produces the same labeled tree as Algorithm A. □

Now we describe how to transform the computation of $f_2(i)$, $i = 1, 2, \ldots, n-1$, in Step 2 of Algorithm B into the dominance counting problem. A point (x_1, y_1) is dominated by point (x_2, y_2) in the plane if $x_1 \leq x_2$ and $y_1 \leq y_2$. The dominance counting problem is to count for each point the points dominated by it [2], [3]. For each vertex i and $f_1(i)$, $i = 1, 2, \cdots, n-1$, we can get a corresponding point $(i, f_1(i))$ in the plane. For example, see Fig. 4. The value of $f_2(i)$ is the number of points dominated by point $(i, f_1(i))$, $i = 1, 2, \cdots, n-1$.

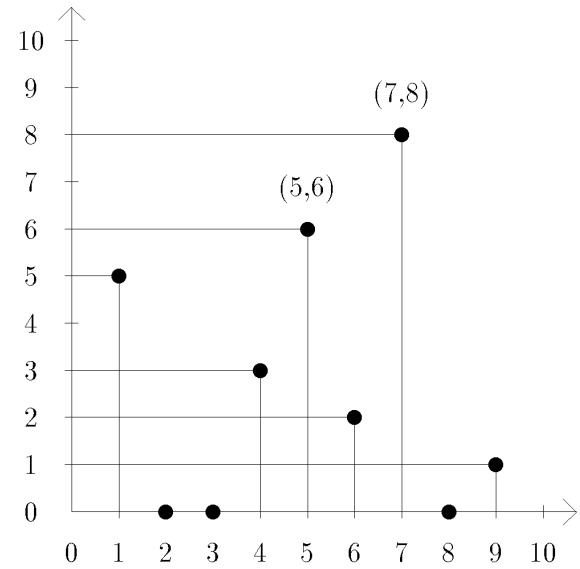


Fig. 4. The corresponding points in the plane of s-sequence 9, 6, 4, 10, 1, 5, 7, 7, 10.

In [1], Atallah et al. gave a parallel algorithm which runs in $O(\log n)$ time using $O(n)$ processors on a CREW (Concurrent-Read-Exclusive-Write) PRAM model to solve the dominance counting problem. In [5], Cole used the sampling technique to implement his merging procedure in the EREW PRAM model. Applying his technique on the dominance counting problem results in EREW PRAM algorithm with the same asymptotic bound as CREW PRAM model, except that the space bound for the problem becomes $O(n \log n)$.

We summarize our result as the following theorem.

THEOREM 2. *Algorithm B constructs a labeled tree in $O(\log n)$ time by using $O(n)$ processors on CREW PRAM model. Algorithm B can also solve the problem in EREW PRAM model with the same asymptotic bound as CREW PRAM model, except that the space bound becomes $O(n \log n)$.*

5 CONCLUDING REMARKS

In this paper, we propose a parallel algorithm for transferring an s-sequence to a labeled tree. It takes $O(\log n)$ time by using $O(n)$ processors and $O(n \log n)$ space on the EREW PRAM model. Another interesting problem is to design an efficient parallel algorithm for transferring a labeled tree to its corresponding number sequence. We are now trying to solve this problem.

ACKNOWLEDGMENTS

This work was supported by the National Science Council, Republic of China, under Contract NSC-84-2213-E-011-008.

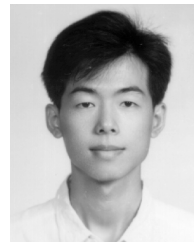
REFERENCES

[1] M.J. Atallah, R. Cole, and M.T. Goodrich, "Cascading Divide-and-Conquer: A Technique for Designing Parallel Algorithms," *SIAM J. Computing*, vol. 18, no. 3, pp. 499-532, June 1989.

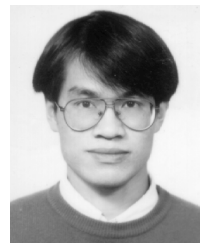
[2] M.J. Atallah and S.R. Kosaraju, "An Efficient Algorithm for Maxdominance, with Applications," *Algorithmica*, vol. 4, pp. 221-236, 1889.
[3] M.T. De Berg, S. Carlsson, and M.H. Overmars, "A General Approach to Dominance in the Plane," *J. Algorithms*, vol. 13, pp. 274-296, 1992.
[4] A. Cayley, "A Theorem on Trees," *Quarterly J. Math.*, vol. 23, pp. 376-378, 1989.
[5] R. Cole, "Parallel Merge Sort," *Proc. 27th IEEE Symp. Foundations of Computer Science*, pp. 511-516, 1986.
[6] R. Gould, *Graph Theory*. Benjamin Cummings. 1988.
[7] F. Harary, *Graph Theory*. Reading, Mass.: Addison-Wesley, 1969.
[8] E. Horowitz and S. Sahni, *Fundamentals of Data Structures in Pascal*, third edition. Computer Science Press, 1990.
[9] J. Jájá, *Introduction to Parallel Algorithms.* Addison-Wesley, 1992.
[10] C.P. Kruskal, L. Rudolph, and M. Snir, "The Power of Parallel Prefix," *IEEE Trans. Computers*, vol. 34, pp. 965-968, 1985.
[11] J.W. Moon, *Counting Labeled Trees*. Montreal: Canadian Mathematical Congress, 1970.
[12] H. Prüfer, "Neuer Beweis eines satzes über Permutationen," *Archiv der Mathematik und Physik*, vol. 27, pp. 742-744, 1918.

Yue-Li Wang received the BS and MS degrees from Tamkang University, Taiwan, Republic of China, in 1975 and 1979, respectively, and the PhD degree from National Tsing Hua University, Taiwan, in 1988, all in computer science.

He joined the National Taiwan Institute of Technology in 1990. He is presently the head of the Department of Information Management, National Taiwan University of Science and Technology (the original National Taiwan Institute of Technology). Before joining the National Taiwan University of Science and Technology, he worked in the Advanced Technology Center, Electronics Research and Service Organization, Industrial Technology Research Institute. His research interests include graph theory, computational geometry, and the design and analysis of computer algorithms.



Hon-Chan Chen received the BS and MS degrees in information management from National Taiwan Institute of Technology, Taipei, Taiwan, Republic of China, in 1992 and 1994, respectively. He is currently a PhD student in the Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan. His research interests include algorithms, parallel processing and graph theory.



Wei-Kai Liu received the BS and MS degrees in information management from National Taiwan Institute of Technology, Taipei, Taiwan, Republic of China, in 1993 and 1995, respectively. He is currently an engineer at Taiwan Semiconductor Manufacturing Company. His research interests include parallel processing and graph theory.