1998 International Computer Symposium
Workshop on Algorithms
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

On the Shortest Length Queries for Permutation Graphs

H. S. Chao[1] *F. R. Hsu*[2] *R. C. T.Lee*[3]

[1]R&D Division
Syscom Computer Engineering Company
11F, No. 260, Pa Teh Road, Sec 2, Taipei, Taiwan, 104
E-mail: hschao@oodb.syscom.com.tw

[2]Department of Accounting
Providence University
Shalu, Taichung Hsien, Taiwan, 433
E-mail: frhsu@simon.pu.edu.

[3]Office of the President
Providence University
Shalu, Taichung Hsien, Taiwan, 433
E-mail: rctlee@simon.pu.edu.tw

Abstract

The all-pairs shortest path problem is a very important problem for both theoretical researches and practical applications. Given an undirected, unweighted, connected graph of n vertices, computing the lengths of the shortest paths between all pairs of vertices takes $\Omega(n^2)$ time and space, since there are $\Theta(n^2)$ pairs of vertices. In this paper, we present efficient algorithms to solve the query version for the problem of computing the lengths of all-pairs shortest paths for permutation graphs. Given a permutation graph G, our algorithms preprocess G in $O(\log n)$ time using $O(n/\log n)$ processors under the EREW PRAM model such that the shortest length query between any two vertices can be answered in $O(1)$ time using one processor.

1 Introduction

Let $G = (V, E)$ be an undirected, unweighted, connected graph and let $|V| = n$ and $|E| = m$. A path between two vertices s and t is a sequence of vertices $(v_1, v_2, \ldots, v_k)$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i < k$, where $s = v_1$ and $t = v_k$. The length of the path is the number of edges in the sequence. A shortest path between two given vertices is a path with the shortest length. The *all-pairs shortest path* problem is the problem of finding the shortest paths between all pairs of vertices. In this paper, we consider the distance version of the all-pairs shortest path problem; i.e., the problem of finding the lengths of the shortest paths between all pairs of vertices.

Given a undirected, unweighted graph G of n vertices, computing the lengths of the shortest paths between all pairs of vertices in G takes $\Omega(n^2)$ time, since there are $\Theta(n^2)$ pairs of vertices. For general graphs, the best known algorithm was proposed by Seidel [8] and runs in $O(M(n) \log n)$ time, where $M(n)$ is the time necessary to multiply two $n \times n$ matrices of small integers, which is currently $o(n^{2.376})$. Efficient sequential and parallel algorithms have been developed for special classes of graphs such as the interval, circular-arc, and permutation graphs (see, e.g., [4, 6, 7]).

For the interval and circular-arc graphs, Chen and Lee [4] presented a preprocessing algorithm which runs in $O(n)$ time and in $O(\log n)$ time using $(n/\log n)$ CREW PRAM processors in parallel. Their preprocessing algorithm constructs an $O(n)$ space data structure and using the data structure, any shortest length query can be answered in $O(1)$ time using one processor. In [4], Chen and Lee made use of the technique of the level-ancestor query in trees introduced by Berkman and Vishkin [2].

In this paper, permutation graphs are considered. For a permutation graph G with its corresponding permutation $\pi = [\pi(1), \pi(2), \cdots, \pi(n)]$, we propose a preprocessing algorithm which runs in $O(n)$ time and in $O(\log n)$ time using $(n/\log n)$ processors under the EREW PRAM model. Our preprocessing algorithm constructs an $O(n)$ space data structure and using this data structure, we can answer a shortest length query in $O(1)$ time with one processor.

2 Preliminary

Let $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ be a permutation of the numbers $1, 2, \ldots, n$. We can construct a graph $G[\pi] = (V, E)$ with vertex set $V = \{1, \ldots, n\}$ and edge set E:

$$(i,j) \in E \Leftrightarrow (i-j)(\pi^{-1}(i) - \pi^{-1}(j)) < 0,$$

where $\pi^{-1}(i)$ is the position of i in $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$. An undirected graph G is a *permutation graph* [5] if there is a permutation π such that G is isomorphic to $G[\pi]$. In this paper, our input is a permutation graph $G[\pi]$, with its corresponding permutation π.

A permutation graph can be viewed as an intersection graph, which is illustrated by the *permutation diagram* [5], which is defined as follows: Write the numbers $1, 2, \ldots, n$ horizontally from left to right. Under every i, write the numbers $\pi(i)$. Draw line segments connecting i in the top row and i in the bottom row, for each i. It is easy to see that two vertices i and j of $G[\pi]$ are adjacent if and only if the corresponding line segments of i and j intersect. Figure 1 shows the permutation graph $G[\pi]$ and its corresponding permutation diagram of a permutation $\pi = [5, 2, 1, 8, 3, 6, 12, 4, 10, 14, 7, 9, 17, 15, 11, 13, 16]$.

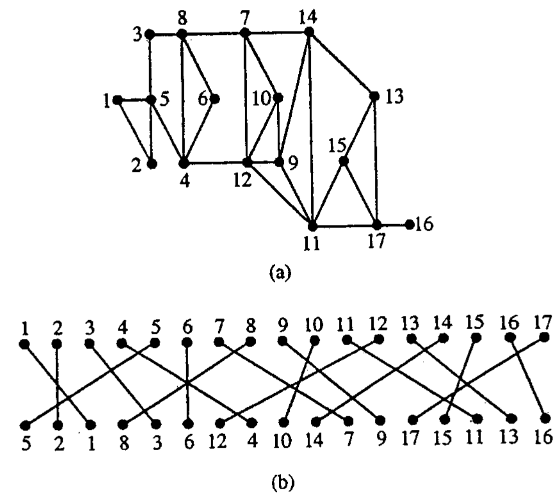

Figure 1: (a) A permutation graph. (b) The permutation diagram.

Consider a permutation graph $G[\pi]$ defined by a permutation $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$. For each vertex i, define:

$$UR(i) = \max(\{i\} \cup \{k|(i,k) \in E\}),$$
$$UL(i) = \min(\{i\} \cup \{k|(i,k) \in E\}),$$
$$LR(i) = \pi(\max(\{\pi^{-1}(i)\} \cup \{\pi^{-1}(k)|(i,k) \in E\})),$$
$$LL(i) = \pi(\min(\{\pi^{-1}(i)\} \cup \{\pi^{-1}(k)|(i,k) \in E\})).$$

For our example in Figure 1, if $i = 10$, we have $UR(10) = 12$ and $UL(10) = 7$ and $LR(10) = 9$ and $LL(10) = 12$. In other words, on the permutation diagram, for a vertex i, $UR(i)$ (resp. $LR(i)$) is the upper (resp. lower) rightmost vertex among the vertices adjacent to vertex i, including i itself. Similarly, $UL(i)$ (resp. $LL(i)$) is the upper (resp. lower) leftmost vertex among the vertices adjacent to vertex i, including i itself.

Lemma 1 *Let $G[\pi]$ be a permutation graph with n vertices. The following equations hold for all*

i, $1 \le i \le n$.

$$UR(i) = \max(\{\pi(1), \pi(2), \ldots, \pi(\pi^{-1}(i))\})$$
$$UL(i) = \min(\{\pi(\pi^{-1}(i)), \ldots, \pi(n-1), \pi(n)\})$$
$$LR(i) = \pi(\max(\{\pi^{-1}(1), \pi^{-1}(2), \ldots, \pi^{-1}(i)\}))$$
$$LL(i) = \pi(\min(\{\pi^{-1}(i), \ldots, \pi^{-1}(n)\}))$$

Proof. We prove the first equation only. The proofs of the other three equations are similar and omitted. Let $\max(\{\pi(1), \pi(2), \ldots, \pi(\pi^{-1}(i))\})$ be j. Since $UR(i)$ is at least i, for any $UR(i) = k \neq i$, $k > i$. By the definition of $UR(i)$, $(i,k) \in E$. Hence $\pi^{-1}(k) < \pi^{-1}(i)$. We have $\pi^{-1}(UR(i)) \le \pi^{-1}(i)$ and then, by the definition of j, $UR(i) \le j$. Consider any vertex l, where $1 \le \pi^{-1}(l) \le \pi^{-1}(i)$. If $(i,l) \in E$, we have $l \le UR(i)$ by the definition of $UR(i)$. If $(i,l) \notin E$, we have $l \le i \le UR(i)$. Hence, we have $l = \pi(\pi^{-1}(l)) \le UR(i)$. Thus, $UR(i) = l = \max(\{\pi(1), \pi(2), \ldots, \pi(\pi^{-1}(i))\})$. □

Note that $UR(i) \ge i$ and $\pi^{-1}(UR(i)) \le \pi^{-1}(i)$ for any i in $G[\pi]$. Since $UR(i) \ge i$, we have $UR(UR(i)) \ge UR(i)$. Since $\pi^{-1}(UR(i)) \le \pi^{-1}(i)$, we have $UR(UR(i)) \le UR(i)$ by Lemma 1. Hence, for any i, $UR(UR(i)) = UR(i)$. Similarly, $LR(LR(i)) = LR(i)$. Define a vertex i to be an *UR-type* (resp. *LR-type*) vertex if $UR(i) = i$ (resp. $LR(i) = i$). It is obvious that vertex $UR(i)$ is of UR-type and vertex $LR(i)$ is of LR-type. Thus we have the following lemma:

Lemma 2 *Given a permutation graph $G[\pi]$, for each i, $UR(i)$ and $LL(i)$ are UR-type vertices and $LR(i)$ and $UL(i)$ are of LR-type.*

For example, consider the permutation graph in Figure 1. All of the LR-type vertices are boldfaced line segments and its corresponding permutation diagram is shown in Figure 2.

If i is an LR(resp. UR)-type vertex, define $SUC(i)$ to be $UR(i)$(resp. $LR(i)$).

For any function $F : [1..n] \to [1..n]$, and any integer l, $l \ge 0$, define

$$F^{(l)}(i) = \begin{cases} F(F^{(l-1)}) & \text{if } l > 1, \\ F(i) & \text{if } l = 1, \\ i & \text{if } l = 0. \end{cases}$$

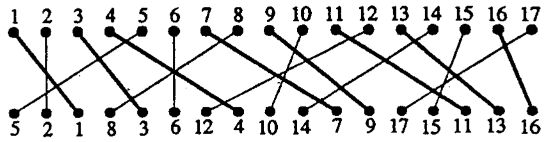

Figure 2: The LR-type vertices in a permutation graph.

For any two vertices s and t in a permutation graph, where $s < t$ and $(s,t) \notin E$, let $UR\_path(s,t)$ be the vertex path $(s, SUC^{(0)}(UR(s)), \ldots, SUC^{(l)}(UR(s)), t)$, where l is the smallest nonnegative integer such that $(t, SUC^{(l)}(UR(s))) \in E$. Similarly, let $LR\_path(s,t)$ be $(s, SUC^{(0)}(LR(s)), \ldots, SUC^{(l)}(LR(s)), t)$, where l is the smallest nonnegative integer such that $(t, SUC^{(l)}(LR(s))) \in E$.

Lemma 3 (Ibarra and Zheng [6]) *Let s and t be any two vertices in a permutation graph $G[\pi]$, where $s < t$ and $(s,t) \notin E$, the shorter one between $UR\_path(s,t)$ and $LR\_path(s,t)$ is the shortest path connecting s and t.* □

Define $d(s,t)$ to be the length of the shortest path between s and t. If $(s,t) \in E$, $d(s,t) = 1$. Otherwise, by Lemma 3, $d(s,t)$ is equal to the length of the shorter path between $UR\_path(s,t)$ and $LR\_path(s,t)$. In the cases of $d(s,t) \le 2$, we can find $d(s,t)$ by simply checking whether s or $UR(s)$ or $LR(s)$ is adjacent to t. Hence, in the following, we consider the cases of $d(s,t) \ge 3$.

According to Lemma 3, we are interested in the distance between $UR(s)$ and t and the distance between $LR(s)$ and t. We first note that instead of finding the distance between $UR(s)$ and t, we can find the distance between $UR(s)$ and $UL(t)$ and the distance between $UR(s)$ and $LL(t)$. Suppose we further prove that instead of finding the distance between $UR(s)$ and $UL(t)$, we may find the distance between $LR(UR(s))$ and $UL(t)$, then there is one advantage that we can utilize. This advantage is that both $LR(UR(s))$ and $UL(t)$ are of LR-type. In the following, we shall finally prove that we only have to find the distances of four shortest paths which start and end with LR-type vertices. If we can answer a shortest length query between any two LR-type vertices in $O(1)$ time, we can answer a shortest length query between any two vertices still in $O(1)$ time.

Theorem 1 *For any two vertices s and t, $s < t$ and $d(s,t) \geq 3$, in a permutation graph $G[\pi]$,*

$$d(s,t) = \min(\{d(UR(s),UL(t)), d(UR(s),LL(t)), d(LR(s),UL(t)), d(LR(s),LL(t))\}) + 2.$$

Proof. Let s' denote $UR(s)$. Without losing generality, suppose $UR\_path(s,t)$ is the shortest path from s to t. Furthermore, suppose $UR\_path(s,t) = (s, SUC^{(0)}(s'), \ldots, SUC^{(l)}(s'), t)$ and $SUC^{(l)}(s')$ is of LR-type. It is clear that $d(s,t) = l+2$. We claim that the vertex path obtained by replacing $SUC^{(l)}(s')$ with $UL(t)$ from $UR\_path(s,t)$, is still a shortest path connecting s and t. It is obvious that $(UL(t),t) \in E$. We only need to prove that $(SUC^{(l-1)}(s'), UL(t)) \in E$. Since $SUC^{(l)}(s')$ is LR-type, $SUC^{(l-1)}(s')$ is of UR-type. Note that $SUC^{(l)}(s') < SUC^{(l-1)}(s')$ and $(SUC^{(l)}(s'),t) \in E$. By the definition of $UL(t)$, $UL(t) \leq SUC^{(l)}(s') < SUC^{(l-1)}(s')$. Since $(SUC^{(l-1)}(s'),t) \notin E$ and $\pi^{-1}(SUC^{(l-1)}(s')) < \pi^{-1}(t)$, we have $\pi^{-1}(UL(t)) \geq \pi^{-1}(t) > \pi^{-1}(SUC^{(l-1)}(s'))$. Therefore, $(SUC^{(l-1)}(s'), UL(t)) \in E$ and the path $(s, SUC^{(0)}(s'), \ldots, SUC^{(l-1)}(s'), UL(t), t)$ is still a shortest path connecting s and t. Furthermore, note that the subpath $(SUC^{(0)}(s'), \ldots, SUC^{(l-1)}(s'), UL(t))$ is also a shortest path connecting $UR(s) = SUC^{(0)}(s')$ and $UL(t)$. Hence, $d(UR(s),UL(t)) = l$. We therefore have $d(UR(s),UL(t))+2 = d(s,t)$. It is impossible that $d(UR(s),UL(t))$ is greater than anyone of $d(UR(s),LL(t))$, $d(LR(s),UL(t))$ and $d(LR(s),LL(t))$. Otherwise, it is directly contradictory to the fact that $d(s,t)$ is the shortest length between s and t. By the above discussion, the equality holds for the supposed case.
The proofs for the remained cases are similar and omitted here. □

Lemma 4 *For any two vertices s and t, $s < t$ and $d(s,t) > 1$, in a permutation graph $G[\pi]$, if s is of UR-type, $d(s,t) = d(LR(s),t)+1$.*

Proof. Since s is UR-type, $UR(s) = s$. Hence, $LR\_path(s,t)$ is shorter than $UR\_path(s,t)$. By Lemma 3, it is obvious that $LR\_path(s,t)$ is the shortest path connecting s and t. Hence, $d(s,t) = d(LR(s),t)+1$. □

Lemma 5 *For any two vertices s and t, $s < t$ and $d(s,t) > 1$, in a permutation graph $G[\pi]$, if t is of UR-type, $d(s,t) = d(s,UL(t))+1$.*

Proof. Without losing generality, suppose $UR\_path(s,t)$ is shorter than $LR\_path(s,t)$. Let $s' = UR(s)$ and $UR\_path(s,t) = (s, SUC^{(0)}(s'), \ldots, SUC^{(l)}(s'), t)$. Since t is of UR-type, $SUC^{(l)}(s')$ must be of LR-type. The remaining proof is similar to that in Theorem 1 and omitted. □

Theorem 2 *For any two vertices s and t, $s < t$ and $d(s,t) \geq 3$, in a permutation graph $G[\pi]$,*

$$\begin{aligned} d(s,t) &= \min(\{d(LR(UR(s)),UL(t))+3, \\ &\quad d(LR(UR(s)),UL(LL(t)))+4, \\ &\quad d(LR(s),UL(t))+2, \\ &\quad d(LR(s),UL(LL(t))+3)\}). \end{aligned}$$

Proof. Note that $UR(i)$ and $LL(i)$ are of UR-type for any vertex i in a permutation graph $G[\pi]$. Apply Lemmas 4 and 5 to Theorem 1. The proof is complete. □

Since both $LR(i)$ and $UL(i)$ are of LR-type for any vertex i in a permutation graph $G[\pi]$, according to Theorem 2, if we can answer a shortest length query between any two LR-type vertices in $O(1)$ time, we can answer a shortest length query between any two vertices in $O(1)$ time.

1998 International Computer Symposium
Workshop on Algorithms
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

3 The Shortest Length Query between Two LR-type Vertices

Given a permutation graph $G[\pi]$, define $SS(i) = SUC(SUC(i))$ for each vertex i.

Lemma 6 *For any two LR-type vertices s and t in $G[\pi]$, if $s < t$ and k is the smallest integer such that $SS^{(k)}(s) \geq t$, then $d(s,t) = 2k$.*

Proof. Since s is of LR-type, $UR(s) = SUC(s)$ and $UR\_path(s,t)$ is the shortest path connecting s and t. Furthermore, let $UR\_path(s,t) = (s, SUC(s), \ldots, SUC^{(l)}(s), t)$, where l is the smallest nonnegative integer such that $(t, SUC^{(l)}(s)) \in E$. We have that all $SUC^{(j)}(s))$'s, $0 \leq j < l$, are smaller than t. Since k is the smallest integer such that $SS^{(k)}(s) \geq t$, we have $2k \geq l$. Since t is of LR-type and $(SUC^{(l)}(s), t) \in E$, $SUC^{(l)}(s)$ is of UR-type, l is an odd number and $SUC^{(l)}(s) > t$. We claim that $SUC^{(l+1)}(s) = SS^{((l+1)/2)} \geq t$. If $SUC^{(l+1)}(s) = t$, the proof is complete. If $SUC^{(l+1)}(s) \neq t$, since $SUC^{(l)}(s)$ is of UR-type, $SUC^{(l+1)}(s) = LR(SUC^{(l)}(s))$. Hence, $\pi^{-1}(SUC^{(l+1)}(s)) > \pi^{-1}(t)$. Since $(SUC^{(l+1)}(s), t) \notin E$, we have $SUC^{(l+1)}(s) = SS^{((l+1)/2)} > t$. Therefore, we have $(l+1)/2 \geq k$. Since l is an odd number and $2k \geq l$ and $(l+1)/2 \geq k$, we have $2k = l + 1$. Then, $d(s,t) = l + 1 = 2k$. The proof is completed. □

Let the number of all LR-type vertices in $G[\pi]$ be q. We can find all LR-type vertices out and organize them into an array A from small to large. Let $A = [A(1), A(2), \ldots, A(q)]$ be the array of all LR-type vertices in $G[\pi]$ such that $A(i) < A(j)$ for any i and j, where $1 \leq i < j \leq q$. The LR-type vertices of our example in Figure 1 are $1, 3, 4, 7, 9, 11, 13$ and 16. The array A is shown in Figure 3. For any LR-type vertex $A(i)$, $SUC(A(i))$ is a UR-type vertex and then $SS(A(i)) = SUC(SUC(A(i)))$ is an LR-type vertex. Define an array $P = [P(1), P(2), \ldots, P(q)]$ such that $P(i) = j$ if $A(j) = SS(A(i))$ for any i, $1 \leq i \leq q$.

Index:	1	2	3	4	5	6	7	8
A:	1	3	4	7	9	11	13	16

Figure 3: The compacted array A of all LR-type vertices.

As shown in [3], a rooted tree, denoted as T_A, is defined by arrays A and P such that T_A is rooted at $A(q)$, where $P(A(q)) = A(q)$, and the parent of $A(i)$, $1 \leq i < q$, is $A(P(i))$. The array P and the tree T_A of our example in Figure 1 are shown in Figure 4.

Index:	1	2	3	4	5	6	7	8
P:	3	4	6	7	7	8	8	8

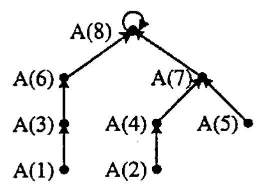

Figure 4: The tree T_A.

Lemma 7 (Chao *et al.* [3]) *$P(i) \leq P(j)$, for each i and j, $1 \leq i < j \leq q$.*

By Lemma 7, it is easy to see that the children of each internal node in T_A occupy consecutive ranges in array A. Hence we can decide whether a given node is the leftmost child of its parent node. We now try to find the ranks among siblings for each node in T_A, except the root. The rank of node $A(i)$ among its siblings will be equal to one plus its index i minus the index of its leftmost sibling. Let Q be an array such that $Q(i) = i$ if $A(i)$ is the leftmost child of its parent, and $Q(i) = 0$, if otherwise, except the root node. Perform a prefix maxima computation on the array Q and let the resulting array be array $PrefixMaxQ$. Construct an array *Rank* such that $Rank(i) = 1 + i - PrefixMaxQ(i)$. Then $Rank(i)$ is the rank of $A(i)$ among its siblings. Figure 5 shows these arrays for our example.

Index:	1	2	3	4	5	6	7	8
Q:	1	2	3	4	0	6	0	-
PrefixMaxQ:	1	2	3	4	4	6	6	-
Rank:	1	1	1	1	2	1	2	-

Figure 5: The computation of the array *Rank*.

With arrays P and $Rank$, the "parent-of" relation and the explicit ordering of children of vertices in T_A are made clear. Let $PreOrder(i)$ denote the pre-order number of $A(i)$ when one traverse T_A. Let $L(i)$ denote the level of $A(i)$ in tree T_A. With arrays P and $Rank$, we can compute $PreOrder(i)$ and $L(i)$ for all i in $O(\log n)$ time using $O(n/\log n)$ processors under the EREW PRAM model by utilizing the Euler tour technique [1]. In our example, the pre-order traversal of T_A would be $16, 11, 4, 1, 13, 7, 3, 9$, $PreOrder = [4, 7, 3, 6, 8, 2, 5, 1]$ and $L = [3, 3, 2, 2, 2, 1, 1, 0]$.

Theorem 3 *For any LR-type vertices $A(i)$ and $A(j)$, $i < j$,*

$$d(A(i), A(j)) = \begin{cases} 2(L(j) - L(i) + 1) \\ \quad \text{if } PreOrder(i) < PreOrder(j), \\ 2(L(j) - L(i)) \\ \quad \text{if otherwise.} \end{cases}$$

Proof. It is easy to see that for nodes $A(k)$ and $A(l)$, $A(k) < A(l)$, at the same level in tree T_A, we have $PreOrder(k) < PreOrder(l)$. Since $i < j$, we have $L(i) \leq L(j)$. Hence , $A(k) = SS^{(L(j)-L(i))}$ is the ancestor of $A(i)$ with the same level of $A(j)$. Note that in a pre-order traversal, every node has larger pre-order number than that of its ancestors. If $PreOrder(i) < PreOrder(j)$, then $PreOrder(k) \leq PreOrder(i) < PreOrder(j)$. Since $A(k)$ and $A(j)$ are at the same level in tree T_A and $PreOrder(k) < PreOrder(j)$, we have $A(k) < A(j)$ and $SS(A(k)) = SS^{(L(j)-L(i))+1}(A(i)) > A(j)$. According to Lemma 6, $d(A(i), A(j)) = 2(L(j) - L(i) + 1)$. If $PreOrder(i) \geq PreOrder(j)$, $PreOrder(k) \geq PreOrder(j)$. According to Lemma 6, $d(A(i), A(j)) = 2(L(j) - L(i))$. The proof is complete. □

Directly from Theorem 3, we have the following corollary:

Corollary 1 *Given the arrays PreOrder and L of the tree T_A of a permutation graph $G[\pi]$, the shortest length query between any two LR-type vertices can be answered in $O(1)$ time.*

4 The Preprocessing and Querying Algorithms.

Given a permutation $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ of a permutation $G[\pi]$ with n vertices, our preprocessing algorithm constructs the following arrays for further use:

1. $UR = [UR(1), UR(2), \ldots, UR(n)]$, where $UR(i) = \max(\{\pi(1), \ldots, \pi(\pi^{-1}(i))\})$.

2. $UL = [UL(1), UL(2), \ldots, UL(n)]$, where $UL(i) = \min(\{\pi(\pi^{-1}(i)), \ldots, \pi(n)\})$.

3. $LR = [LR(1), LR(2), \ldots, LR(n)]$, where $LR(i) = \pi(\max(\{\pi^{-1}(1), \ldots, \pi^{-1}(i)\}))$.

4. $LL = [LL(1), LL(2), \ldots, LL(n)]$, where $LL(i) = \pi(\min(\{\pi^{-1}(i), \ldots, \pi^{-1}(n)\}))$.

5. $IndexA[1..n]$: $IndexA(i) = j$, if $A(j) = i$, and $IndexA(i) = 0$, if otherwise.

6. $PreOrder[1..q]$: $PreOrder(i)$ is the pre-order number of $A(i)$ in tree T_A for $1 \leq i \leq q$, where q is the number of LR-type vertices in $G[\pi]$.

7. $L[1..q]$: $L(i)$ is the level of $A(i)$ in tree T_A for $1 \leq i \leq q$, where q is the number of LR-type vertices in $G[\pi]$.

Theorem 4 *All of the arrays listed above can be computed in $O(\log n)$ time using $O(n/\log n)$ processors under the EREW PRAM model.*

Proof. The first four arrays can be computed by utilizing the parallel prefix or suffix computations [1]. The array $A[1..q]$ and tree T_A (i.e., the arrays $P[1..q]$ and $Order[1..q]$) can be computed by using the algorithms in Chao *el al.* [3]. The computation of array $IndexA[1..n]$ is very simple while array A has been computed. The arrays $PreOrder[1..q]$ and $L[1..q]$ can be computed by utilizing the Euler tour technique [1] and the parallel prefix computations. All of the above computations can be computed in $O(\log n)$ time using $O(n/\log n)$ processors under the EREW PRAM model. □

According to Theorem 3, $LRSP(s,t)$ correctly finds $d(s,t)$ of two LR-type vertices s and t. According to Theorem 2, *ShortestLength*(s,t) correctly finds $d(s,t)$ for any two vertices s and t in $G[\pi]$.

```
FUNCTION LRSP(s,t) : d(s,t)
  if s < t
    then i ← IndexA(s) and j ← IndexA(t)
    else i ← IndexA(t) and j ← IndexA(s);
  if PreOrder(i) < PreOrder(j)
    then return (2(L(j) − L(i) + 1))
    else return (2(L(j) − L(i)));

FUNCTION ShortestLength(s,t) : d(s,t)
  if (s,t) ∈ E
    then return 1;
  if (LR(s),t) ∈ E or (UR(s),t) ∈ E
    then return 2;
  d ← min{LRSP(LR(UR(s)),UL(t)) + 3,
    LRSP(LR(UR(s)),UL(LL(t))) + 4,
    LRSP(LR(s),UL(t)) + 2,
    LRSP(LR(s),UL(LL(t)) + 3)};
  return d;
```

Corollary 2 *Given the permutation* $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ *of a permutation graph* G *with* n *vertices, our preprocessing algorithm runs in* $O(n)$ *time sequentially and in* $O(\log n)$ *time using* $O(n/\log n)$ *processors under the EREW PRAM model. Using an* $O(n)$ *space data structure in a preprocessing algorithm, any shortest length query between two vertices can be answered in* $O(1)$ *time using one processor.*

References

[1] S. G. Akl. *Parallel computation: models and methods.* Prentice Hall, Upper Saddle River, New Jersey, 1997.

[2] O. Berkman and U. Vishkin. Finding level-ancestors in trees. *Journal of Computer and System Sciences*, 48:214–230, 1994.

[3] H. S. Chao, F. R. Hsu, and R. C. T. Lee. An optimal EREW parallel algorithm for computing breadth-first search trees on permutation graphs. *Information Processing Letters*, pages 311–316, 1997.

[4] D. Z. Chen and D. T. Lee. Solving the all-pair shortest path problem on interval and circular-arc graphs. In *IPPS'94: 8th International Parallel Processing Symposium*, pages 224–228, 1994.

[5] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.

[6] O. H. Ibarra and Q. Zheng. An optimal shortest path parallel algorithm for permutation graphs. *Journal of Parallel and Distributed Computing*, 24:94–99, 1995.

[7] R. Ravi, M. V. Marathe, and C. Pandu Rangan. An optimal algorithm to solve the all-pair shortest path problem on interval graphs. *Networks*, 22:21–35, 1992.

[8] R. Seidel. On the all-pair-shortest-path problem. *Journal of Computer and System Sciences*, 51:400–403, 1995.