Independent Spanning Trees on Crossed Cubes

Jhen-Ding Wang\textsuperscript{1} \hspace{0.5cm} Jou-Ming Chang\textsuperscript{1,†} \hspace{0.5cm} Jinn-Shyong Yang\textsuperscript{2} \hspace{0.5cm} Kun-Fu Ding\textsuperscript{3}

\textsuperscript{1} Institute of Information and Decision Sciences, National Taipei College of Business, Taipei, Taiwan, ROC
\textsuperscript{2} Department of Information Management, National Taipei College of Business, Taipei, Taiwan, ROC
\textsuperscript{3} Department of Industrial Engineering and Management, Ming Chi University of Technology, New Taipei City, Taiwan, ROC

Abstract

A set of spanning trees in a graph is said to be independent (ISTs for short) if all the trees are rooted at the same node \( r \) and for any other node \( v(\neq r) \), the paths from \( v \) to \( r \) in any two trees are node-disjoint except the two end nodes \( v \) and \( r \). For an \( n \)-connected graph, the independent spanning trees problem asks to construct \( n \) ISTs rooted at an arbitrary node of the graph. Recently, Zhang et al. [Y.-H. Zhang, W. Hao, and T. Xiang, Independent spanning trees in crossed cubes, Inform. Process. Lett., 113 (2013) 653–658] proposed an algorithm to construct \( n \) ISTs with a common root at node 0 in an \( n \)-dimensional crossed cube \( \text{CQ}_n \). However, it has been proved by Kulasinghe and Bettayeb [P.D. Kulasinghe and S. Bettayeb, Multiplicity-twisted hypercube with 5 or more dimensions is not vertex transitive, Inform. Process. Lett., 53 (1995) 33–36] that the \( \text{CQ}_n \) (a synonym called multiply-twisted hypercube in that paper) fails to be node-transitive for \( n \geq 5 \). Thus, the result of Zhang et al. does not really solve the ISTs problem in \( \text{CQ}_n \). In this paper, we revisit the problem of constructing \( n \) ISTs rooted at an arbitrary node in \( \text{CQ}_n \). As a consequence, we show that the proposed algorithm can be parallelized to run in \( \mathcal{O}(\log N) \) time using \( N = 2^n \) nodes of \( \text{CQ}_n \) as processors.

\textbf{Keyword}: independent spanning trees; interconnection networks; crossed cubes; multiply-twisted hypercube;

\textsuperscript{*}This research was partially supported by National Science Council under the Grants NSC102-2221-E-141-002 and NSC102-2221-E-141-001-MY3.
\textsuperscript{†}Corresponding author. Email: spade@mail.ntcb.edu.tw

1 Introduction

Constructing multiple spanning trees in networks have been studied from not only the theoretical point of view but also some practical applications such as fault-tolerant broadcasting [1, 15] and secure message distribution [1, 25, 31]. Let \( G \) be a graph with node set \( V(G) \) and edge set \( E(G) \), respectively. Two spanning trees in a graph \( G \) are said to be independent if they are rooted at the same node \( r \) such that, for each node \( v(\neq r) \) in \( G \), the two different paths from \( v \) to \( r \), one path in each tree, are internally node-disjoint. A set of spanning trees of \( G \) is called independent spanning trees (ISTs for short) if they are pairwise independent.

A graph \( G \) is \( k \)-connected if \( |V(G)| > k \) and \( G - F \) is connected for every subset \( F \subseteq V(G) \) with \( |F| < k \), where \( G - F \) denotes the graph obtained from \( G \) by removing \( F \). It was conjectured by Zehavi and Itai [38] that for any \( n \)-connected graph there exist \( n \) ISTs rooted at an arbitrary node. From then on, this conjecture has been shown to be true for \( k \)-connected graphs with \( k \leq 4 \) (see [15], [8, 38] and [9] for \( k = 2, 3, 4 \), respectively) and is still open for \( k \geq 5 \). In particular, this conjecture has been confirmed for several restricted classes of graphs, e.g., graphs related to planarity [13, 14, 22, 23], graphs defined by Cartesian product [3, 24, 26, 27, 30, 33, 37], variations of hypercubes [4–7, 21, 28, 29, 31], special Cayley graphs [17, 18, 25, 32, 35, 36], and chordal ring [16, 34].

The \( n \)-dimensional crossed cube \( \text{CQ}_n \), proposed first by Efe [11], is a variant of an \( n \)-dimensional hypercube. One advantage of \( \text{CQ}_n \) is that the diameter is only about one half of the diameter of an \( n \)-dimensional hypercube. For more properties
of $CQ_n$, the reader can refer to [2, 10, 12, 19, 20]. Note that Kulasinghe [19] showed that $CQ_n$ is $n$-connected. Cheng et al. [6] and [5] respectively proposed algorithms to construct $n$ ISTs rooted at an arbitrary node in $CQ_n$. Let $N = 2^n$. The construction scheme of [6] is in a recursive fashion to run in $O(N \log^2 N)$ time. Although the algorithm in [5] can simultaneously construct $n$ ISTs in parallel with time complexity $O(N)$, it is not fully parallelized for the construction of each spanning tree. Recently, Zhang et al. [39] proposed another algorithm that takes $O(N \log N)$ for constructing $n$ ISTs rooted at node 0 in $CQ_n$ and showed that it can be parallelized to run in $O(\log N)$. Because Kulasinghe and Bettayeb [20] had already pointed out that $CQ_n$ (a synonym called multiply-twisted hypercube in that paper) fails to be node-transitive for $n \geq 5$, the construction of [39] that takes node 0 as the common root of spanning trees does not really solve the ISTs problem in $CQ_n$. In this paper, we present a fully parallelized approach for constructing $n$ ISTs rooted at an arbitrary node in $CQ_n$. Our algorithm totally takes $O(N \log N)$ time and can be parallelized to run in $O(\log N)$ time using $N = 2^n$ nodes of $CQ_n$ as processors.

The rest of this paper is organized as follows. Section 2 formally gives the definition of crossed cubes and provides some useful terminologies and notations. Section 3 presents our algorithm for constructing ISTs in $CQ_n$. The final section proves the correctness of the algorithm.

2 Preliminary

In this paper, we use a binary string $x_{n-1}x_{n-2} \cdots x_1x_0$ of length $n$ to label a node $x$ in $CQ_n$. Two binary strings $x = x_0x_1$ and $y = y_0y_1$ are pair-related, denoted $x \sim y$, if and only if $(x, y) \in \{(00, 00), (10, 10), (01, 11), (11, 01)\}$. The $n$-dimensional crossed cube $CQ_n$ is the labeled graph with the following recursively fashion:

$CQ_1$ is the complete graph on two nodes with labels 0 and 1. For $n \geq 2$, $CQ_n$ consists of two subcubes $CQ_{n-1}^0$ and $CQ_{n-1}^1$ such that every vertex in $CQ_{n-1}^0$ and $CQ_{n-1}^1$ is labeled by 0 and 1 in its leftmost bit, respectively. Two nodes $x = 0x_{n-2} \cdots x_1x_0 \in V(CQ_{n-1}^0)$ and $y = 1y_{n-2} \cdots y_1y_0 \in V(CQ_{n-1}^1)$ are joined by an edge if and only if

1. $x_{n-2} = y_{n-2}$ if $n$ is even, and
2. $x_{2i+1}x_{2i} \sim y_{2i+1}y_{2i}$ for $0 \leq i < \lfloor (n - 1)/2 \rfloor$.

Figure 1 shows crossed cubes $CQ_3$ and $CQ_4$.

Let $Z_n = \{0, 1, \ldots, n - 1\}$. Crossed cubes can be defined equivalently as follows:

**Lemma 1.** [11] For all integer $n \geq 1$, two nodes $x = x_{n-1}x_{n-2} \cdots x_0$ and $y = y_{n-1}y_{n-2} \cdots y_0$ are joined by an edge in $CQ_n$ if and only if there exists an integer $i \in Z_n$ such that

1. $x_{n-1}x_{n-2} \cdots x_{i+1} = y_{n-1}y_{n-2} \cdots y_{i+1}$,
2. $x_0 \not= y_0$,
3. $x_{i-1} = y_{i-1}$ if $i$ is odd, and
4. $x_{2j+1}x_{2j} \sim y_{2j+1}y_{2j}$ for $0 \leq j < \lfloor i/2 \rfloor$.

If conditions (1) and (2) of Lemma 1 hold, we say that $x$ and $y$ have the leftmost differing bit at position $i$. In this case, $x$ and $y$ are said to be the $i$-neighbors to each other, and for notational convenience we write $y = N_i(x)$ or $x = N_i(y)$. Moreover, the edge $(x, y)$ is an $i$-dimensional edge of $CQ_n$, and we denote $i = \text{dim}(x, y)$. For example, we consider the node $x = 011011$ in $CQ_6$. Then, $N_i(x)$ for $i = 0, 1, \ldots, 5$ are 011010, 011011, 011101, 010001, 001001, and 111001, respectively.

In this paper, we also use the following notation. Two paths $P$ and $Q$ joining two distinct nodes $x$ and $y$ are internally node-disjoint, denoted by $P \| Q$, if $V(P) \cap V(Q) = \{x, y\}$. Let $T$ be a spanning tree rooted at node $r$ of $CQ_n$. The parent of a node $x (\not= r)$ in $T$ is denoted by $\text{PARENT}(T, x)$. For $x, y \in V(T)$, the unique path from $x$ to $y$ is denoted by $T[x, y]$. Hence, two spanning trees $T$ and $T'$ with the same root $r$ are ISTs if and only if $T[x, r] \| T'[x, r]$ for every node $x \in V(T) \setminus \{r\}$.
3 An algorithm of Constructing ISTs

Since $CQ_n$ is $n$-connected and we would like to construct $n$ ISTs, the root in each spanning tree must have a unique child. Let $r = r_{n-1}r_{n-2} \cdots r_0$ be the common root of ISTs. For $i \in \mathbb{Z}_n$, we denote $T_i$ as a tree such that $r$ takes its $i$-neighbor as the unique child. Let $N_i(r) = c_{n-1}c_{n-2} \cdots c_0$. A node is called the *surreal* of $N_i(r)$, denoted by $N_i(r) = c_{n-1}c_{n-2} \cdots c_0$, if the following conditions hold:

(1) $c_j = c'_j$ for $j \geq i$ if $i$ is even,
(2) $c_j = c'_j$ for $j > i$ if $i$ is odd, and
(3) $c_{2j+1}c_{2j} \sim c'_{2j+1}c'_{2j}$ for $0 \leq j < [i/2]$.

For each node $x = x_{n-1}x_{n-2} \cdots x_0 \in V(T_i) \setminus \{r\}$, a node $x' = x'_{n-1}x'_{n-2} \cdots x'_{0}$ with respect to $x$ is defined as follows: $x_{2j+1}x_{2j} \sim x'_{2j+1}x'_{2j}$ for $0 \leq j < [n/2]$ and $x_{n-1} = x'_{n-1}$ when $n$ is odd.

Let $I_i(x) = \{ j \in \mathbb{Z}_n; x_j \neq c_j \text{ and } j > i \}$ and $I_i(x') = \{ j \in \mathbb{Z}_n; x'_j \neq c'_j \text{ and } j > i \}$. For two sets of integers $S$ and $T$, define the following function:

$$\beta(S, T) = \begin{cases} 0 & \text{if } S = \emptyset; \\ \beta(\{t \in T : t < \min S\}, S) + 1 & \text{otherwise.} \end{cases}$$

In particular, we let $\alpha_i(x) = \beta(I_i(x), I_i(x'))$. According to the parity of $\alpha_i(x)$, let

$$H_i(x) = \begin{cases} \{ j \in \mathbb{Z}_n : x_j \neq c_j \} & \text{if } \alpha_i(x) \text{ is even}; \\ \{ j \in \mathbb{Z}_n : x_j \neq c'_j \} & \text{otherwise.} \end{cases}$$

We further define the following function:

$$\text{next}(i, x) = \begin{cases} i & \text{if } H_i(x) = \emptyset; \\ \max H_i(x) & \text{if } H_i(x) \neq \emptyset \text{ and } i < \min H_i(x); \\ \max\{ j \in H_i(x) : j \leq i \} & \text{otherwise.} \end{cases}$$

That is, we regard $H_i(x)$ as a cyclic ordered set in decreasing order. If $H_i(x) = \emptyset$ or $i \in H_i(x)$, the function outputs $i$; otherwise, the function outputs the next element in the cyclic order of $H_i(x)$ with respect to $i$.

For example, consider $\text{CQ}_12$ and a node $x = 110001101110$ in $T_4$ rooted at $r = 101101000111$. By definitions, $N_4(r) = 101101011101$, $N_4(r) = 101101010111$ and $x' = 0100111001101$. Since $I_4(x) = \{10,9,8,5\}$ and $I_4(x') = \{11,10,9,8,7,5\}$, we can find $\alpha_4(x)$ as follows:

$$\alpha_4(x) = \beta(\{10,9,8,5\}, \{11,10,9,8,7,5\}) = \beta(\{9,8,7,5\}, \{10,9,8,5\}) + 1 = \beta(\{5\}, \{9,8,7,5\}) + 2 = \beta(\{7,5\}, \{8,5\}) + 3 = \beta(\{7\}, \{5\}) + 4 = \beta(\emptyset, \{5\}) + 5 = 5.$$

Thus, $H_4(x) = \{10,9,8,5,4,3,0\}$ and next(4, x) = 4.

Table 1 shows more examples of $\text{CQ}_n$.

It is clear that, for each node $x \in V(\text{CQ}_n) \setminus \{r\}$, finding $I_i(x)$, $I_i(x')$, $\alpha_i(x)$, $H_i(x)$ and next($i,x$) can be done in $O(n)$ time provided $i$ is given. In what follows, we present a fully parallelized algorithm for constructing $n$ spanning trees with an arbitrary node $r = r_{n-1}r_{n-2} \cdots r_0$ as their common root in $\text{CQ}_n$. For each node $x \in V(\text{CQ}_n) \setminus \{r\}$ with binary string $x = x_{n-1}x_{n-2} \cdots x_0$, the construction can be carried out by describing the parent of $x$ in each spanning tree $T_i$.

#### Algorithm: Constructing-ISTs

**Input:** All nodes of $\text{CQ}_n$ and the common root $r = r_{n-1}r_{n-2} \cdots r_0$.

**Output:** $n$ ISTs $T_0, T_1, \ldots, T_{n-1}$ root at $r$.

1. for $i = 0$ to $n-1$ do in parallel
   /* construct $T_i$ simultaneously */
   2. for each node $x$ in $\text{CQ}_n$ do in parallel
      /* generate parent of each node $x$ simultaneously */
      3. $j = \text{next}(i, x)$
      4. $\text{parent}(T_i, x) = N_j(x)$

Figure 2: Algorithm for constructing $n$ spanning trees in $\text{CQ}_n$.
Figure 3 illustrates the construction of $T_2$ and $T_3$ for $CQ_6$. Henceforth, we adopt the notation $x \rightarrow y$ to mean that $y = \text{parent}(T_1, x) = N_i(x)$ in $T_i$. For instance, we have $T_2[34, 27] = 34 \rightarrow 38 \rightarrow 39 \rightarrow 13 \rightarrow 23 \rightarrow 29 \rightarrow 27$ in Figure 3.

### 4 Correctness and analysis

In this section, we will show the validity of the algorithm. Firstly, we give the following basic property.

**Lemma 2.** For $i \in \mathbb{Z}_n$ and a node $x \in V(CQ_n) \setminus \{r\}$, if $H_i(x) = \emptyset$ then $x = N_i(r)$.

**Proof.** Suppose $H_i(x) = \emptyset$. We claim $\alpha_i(x) = 0$, and thus by Eq. (1), it follows that $x = N_i(r)$. We suppose that, on the contrary, $\alpha_i(x) \neq 0$ (i.e., $I_i(x) \neq \emptyset$). This implies that there is a $k \in \mathbb{Z}_n \setminus \mathbb{Z}_i$ such that $x_k \neq c_k$. Obviously, if $\alpha_i(x)$ is even, then $I_i(x) \subseteq H_i(x)$. This contradicts that $H_i(x) = \emptyset$. On the other hand, from the surrenal of $N_i(r)$, we have $c_j = c'_j$ for all $j \neq i$. Thus, $x_k \neq c'_k$, and it follows that $H_i(x) \neq \emptyset$, a contradiction. $\square$

For two ordered sets $A$ and $B$, we write $A \preceq_{\text{lex}} B$ to mean that $A$ precedes $B$ in lexicographic order. We now prove the reachability between every node $x(\neq r)$ and the root $r$ in $T_i$, thereby proving the existence of a unique path from $x$ to the root in the tree.

**Theorem 3.** Let $r \in V(CQ_n)$ be an arbitrary node. The construction of $T_i$ for $i \in \mathbb{Z}_n$ are spanning trees rooted at $r$.

**Proof.** From CONSTRUCTING-ISTS, since every node $v \in V(CQ_n)$ must be contained in $T_i$, it follows that $T_i$ is a spanning subgraph of $CQ_n$. Let $x = x_{n-1}x_{n-2}\cdots x_0$ be any node of $CQ_n$. We show that $T_i[x, r]$ is the unique path connecting $x$ and $r$ in $T_i$. By Lemma 2, if $H_i(x) = \emptyset$, then $x = N_i(r)$.

Thus, $\text{next}(i, x) = i$ and $T_i[x, r] = x \rightarrow r$ is the desired path that connects $x$ and $r$ in $T_i$.

Next, we suppose that $H_i(x) = \{j_{p-1}, j_{p-2}, \ldots, j_0\}$ is nonempty and it is treated as an ordered set such that $j_{p-1} > j_{p-2} > \cdots > j_0$. Clearly, $1 \leq p \leq n$. There are two scenarios as follows:

**Case 1:** $i \notin H_i(x)$ (i.e., $x_i = c_i$). Let $j_k = \text{next}(i, x)$, where $0 \leq k \leq p - 1$. By Eq. (2), we know that $j_{p-1} > j_{p-2} > \cdots > j_{k+1} > i > j_k > \cdots > j_0$. Since $H_i(x) \neq \emptyset$, we assume
that $y(\neq r) = y_{n-1}y_{n-2} \cdots y_0$ is the parent of $x$ in $T_i$. That is, $y = \text{PARENT}(T_i, x) = N_j(x)$. By Lemma 1, the following condition hold: (i) $y_{n-1}y_{n-2} \cdots y_{j+1}y_j = x_{n-1}x_{n-2} \cdots x_{j+1}x_j$; (ii) $y_{j+1} = x_{j+1}$ when $j$ is odd; and (iii) $y_{j+1} + x_{j+1} \sim x_{j+2} + x_j$ for $0 \leq j < [j_k/2]$. We consider the following two subcases:

**Case 1.1:** $\alpha_i(x)$ is even. By Eq. (1), $x_j \neq c_j$ for $j \in H_i(x)$ and $x_j = c_j$ for $j \notin Z_n \setminus H_i(x)$. Thus, we have $I_i(x) = H_i(x) \setminus \{j_k, j_{k-1}, \ldots, j_0\}$. Since $i > j_k$, we have $y_j = x_j$ for every bit at position $j$ with $j > i$. Thus, $I_i(y) = I_i(x)$. In addition, for $j_k < j < i$, we have $y_j = x_j = c_j$. Moreover, $x_{j_k} \neq c_{j_k}$ and $y_{j_k} \neq x_{j_k}$ imply $y_{j_k} = c_{j_k}$. Let $F = \{j \in Z_{j_k} : y_j \neq c_j\}$. Then, we can determine $H_i(y)$ as follows: $H_i(y) = I_i(y) \cup F = (H_i(x) \setminus \{j_k, j_{k-1}, \ldots, j_0\}) \cup F$.

**Case 1.2:** $\alpha_i(x)$ is odd. By Eq. (1), $x_j \neq c_j$ for $j \in H_i(x)$ and $x_j = c_j$ for $j \notin Z_n \setminus H_i(x)$. Thus, we have $I'_i(x) = \{j \in Z_{j_k} : x_j \neq c_j \text{ and } j > i\}$. Clearly, $I'_i(x) = I_i(x)$. In addition, for $j_k < j < i$, we have $y_j = x_j = c_j$. Moreover, $x_{j_k} \neq c_{j_k}$ and $y_{j_k} \neq x_{j_k}$ imply $y_{j_k} = c_{j_k}$. Let $F = \{j \in Z_{j_k} : y_j \neq c_j\}$. Then, we can determine $H_i(y)$ as follows: $H_i(y) = I'_i(y) \cup F = (H_i(x) \setminus \{j_k, j_{k-1}, \ldots, j_0\}) \cup F$.

From above, we can determine $H_i(y)$. In particular, we can show that $H_i(y) \sim_{\text{LEX}} H_i(x)$ and $j_k \notin H_i(y)$. By a similar argument, if $H_i(y) \neq \emptyset$, let $z = \text{PARENT}(T_i, y) = N_j(y)$ be the parent of $y$ in $T_i$, where $j = \text{NEXT}(i, y)$. Again, we can determine $H_i(z)$ and show that $j_k,j_0 \notin H_i(z)$. By this way, we can find a sequence of nodes $y = z, z_1, \ldots, c = N_i(r)$ such that $H_i(c) = \emptyset$. Recall that we have already constructed $T_i[r, c] = c \sim r$ for connecting $c$ and $r$ in $T_i$ before Case 1. Therefore, we obtain the following unique path that connects $x$ and $r$ in $T_i$:

$$T_i[x, r] : x \sim y \sim z \sim \cdots \sim y_{j_k} \sim c \sim r.$$  

**Case 2:** $i \in H_i(x)$ (i.e., $x_i \neq c_i$). Suppose $i = j_k$ for some $k \in \{0, 1, \ldots, p-1\}$. By Eq. (2), we have $\text{NEXT}(i, x) = i$. Let $y = \text{PARENT}(T_i, x) = N_j(x)$. Clearly, $y_i = x_i = c_i$. This shows that the current status of $y$ is in the situation of Case 1. Let $P = T_i[y, r]$ be the path connecting $y$ and $r$ in $T_i$. Therefore, we obtain the unique path $T_i[x, r]$ by concatenating $x \sim y$ and $P$.

According to the proof of Theorem 3, we have the following properties.

**Corollary 4.** For $i \in Z_{n}$, let $T_i[x, r] : v_0(= x) \sim_{j_1} v_1 \sim_{j_2} \cdots \sim_{j_k} v_k \sim r$ be a path constructed from Theorem 3. Then, the following statements hold:

1. $\emptyset = H_i(v_k) \sim_{\text{LEX}} H_i(v_{k-1}) \sim_{\text{LEX}} \cdots \sim_{\text{LEX}} H_i(v_0)$.
2. For $1 \leq \ell < m \leq k$, $j_\ell \notin H_i(v_m)$ (i.e., $j_\ell \neq j_m$).
3. For $2 \leq \ell \leq k$, $j_\ell \neq i$. In particular, it is possible $j_1 = i$.

For instance, if we consider the path $T_3[34, 27] = 34 \sim_2 38 \sim_0 39 \sim_5 23 \sim_4 29 \sim_2 27$ in Figure 3, we can verify from Table 1 as follows: $(H_2(39) = \emptyset) \sim_{\text{LEX}} H_2(23) = \{3\} \sim_{\text{LEX}} H_2(13) = \{4\} \sim_{\text{LEX}} H_2(38) = \{5, 4, 3\} \sim_{\text{LEX}} H_2(34) = \{5, 4, 3, 2, 0\}$. Let $\text{HEIGHT}(T)$ denote the height of a tree $T$. Since $|H_i(x)| \leq n$ for every node $x \in V(CQ_n)$, the following result can be obtained from Corollary 4 directly.

**Corollary 5.** For $i \in Z_{n}$, $\text{HEIGHT}(T_i) \leq n + 1$.

**Theorem 6.** The spanning trees constructed from CONSTRUCTING-ISTS are independent.

**Proof.** We prove the lemma by contradiction. Suppose that the lemma is false. That is, there exist two integers $i, j \in Z_{n}$ and a node $x \in V(CQ_n)$ such that the following two paths constructed in Theorem 3 satisfy $\{x, r\} \not\subseteq P \cap Q$:

$$P = T_i[x, r] : u_0(= x) \sim_{j_0} u_1 \sim_{j_1} u_2 \sim_{j_2} \cdots \sim_{j_k} u_k \sim r$$

and

$$Q = T_j[x, r] : v_0(= x) \sim_{j_0} v_1 \sim_{j_1} v_2 \sim_{j_2} \cdots \sim_{j_n} v_m \sim r.$$  

Suppose that $u_p = v_q$ for $1 \leq p < k$ and $1 \leq q < m$. Let $A = \{j_0, j_1, \ldots, j_{k-1}, 1\}$ and $B = \{q, q+1, \ldots, m-1, j\}$. Since $i \neq j$, by Corollary 4 we have $A \neq B$. Let $d = \max(|A \cup B| \setminus (A \cap B))$. This implies that the $d$th bit of $u_p$ is different from that of $v_q$, which leads to a contradiction. □

According to Theorems 3 and 6, we have the following main result.

**Corollary 7.** Let $N = 2^n$ and $r \in V(CQ_n)$ be an arbitrary node. Algorithm CONSTRUCTING-ISTS can correctly construct $n$ ISTs rooted at $r$ in $O(N \log n)$ time. In particular, the algorithm can be parallelized to run in $O(\log N)$ time using $N$ processors of $CQ_n$.

**References**

[1] F. Bao, Y. Fanyu, Y. Hamada and Y. Igarashi, Reliable broadcasting and secure distributing


