Optimal routing in the De Bruijn networks

Zhen Liu

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 1130

Programme 3
Réseaux et Systèmes Répartis

OPTIMAL ROUTING IN THE DE BRUIJN NETWORKS

Zhen LIU

Décembre 1989

RR_1130

Optimal Routing in the De Bruijn Networks

Routage Optimal dans les Réseaux de De Bruijn

Zhen LIU

INRIA Centre Sophia Antipolis
2004 route des Lucioles
06560 Valbonne, France

E-mail: liu@mirsa.inria.fr

July 1989

Abstract

In this paper, we consider the problem of optimal routing in an interconnection network, called the de Bruijn network, where the sites are linked in the form of a de Bruijn graph. We provide the distance functions for the undirected as well as the directed de Bruijn graphs. The optimal routing problem is then reduced to that of pattern matching. We use Morris and Pratt's failure function and Weiner's prefix tree to develop algorithms that find the shortest paths in the uni-directional and in the bi-directional de Bruijn networks, respectively. These algorithms are linear in time and in space (in the diameter of the graph).

Résumé

Dans ce papier, on considère le problème du routage dans un réseau d'interconnexion, appelé le réseau de deBruijn. On obtient les fonctions de distance pour les graphes de deBruijn orientés et non-orientés. Le problème du routage optimal est ensuite réduit à celui de la reconnaissance des chaînes de caractères. On utilise la *fonction d'échec* de Morris et Pratt et l'arbre de préfixe de Weiner pour développer des algorithmes de routage dans les réseaux de deBruijn orientés et non-orientés, respectivement. Ces algorithmes trouvent les plus courts chemins dans ces réseaux, et sont linéaires en temps et en espace par rapport au diamètre du graphe.

Keywords : Computer network, de Bruijn graph, routing algorithm, distance function, shortest path, pattern matching, failure function, prefix tree.

Mots-clés : réseau d'interconnexion, graphe de deBruijn, algorithme de routage, fonction de distance, plus court chemin, reconnaissance des chaînes de caractères, fonction d'échec, arbre de préfixe.

1 Introduction and Notation

This paper is concerned with the problem of optimal routing in an interconnection network, referred to as the de Bruijn networks, where the connections between sites can be described by the de Bruijn graph. We are interested in the distance functions and the routing algorithms of such networks. Both uni-directional and bi-directional de Bruijn networks are analyzed.

Graphs are widely used in the design and analysis of computer networks. A vertex in the graph denotes a site in the corresponding network, and an edge denotes a communication link between two sites. If a network is uni-directional, i.e., the communication links in the network are uni-directional, a directed graph can be used. Whereas for bi-directional network undirected graphs are used.

Let $G = (V, E)$ be a directed (or undirected) graph, where V denotes the set of vertices, E the set of arcs (or, for undirected graph, edges), $(i,j) \in E$ indicates that there is an edge from vertex i to vertex j (or, between vertices i and j). In an undirected graph, $(i,j) \in E$ is equivalent to $(j,i) \in E$. The cardinal $N = |V|$ denotes the number of vertices in G. Without loss of generality, we assume that $V = \{1, 2, \cdots, N\}$.

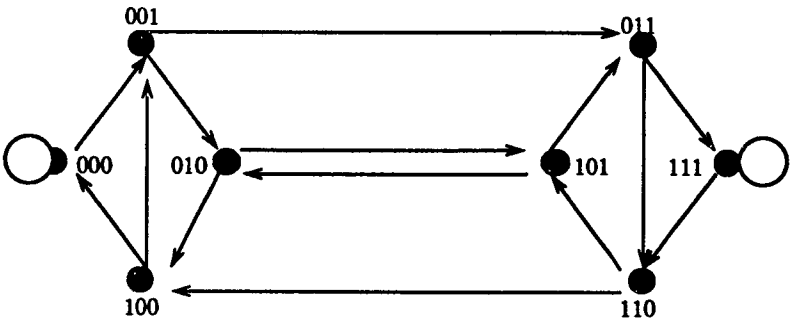
The vertex i is a *neighbor* of the vertex j if $(i,j) \in E$ or $(j,i) \in E$. The *degree* of a vertex is the number of its neighbors. The *degree* of a graph is the maximum degree of the vertices. A *path* from the vertex i to j is an ordered set of vertices $\{i, v_1, v_2, \cdots, v_n, j\}$, such that

$$(i, v_1) \in E, \quad (v_1, v_2) \in E, \quad \cdots, \quad (v_n, j) \in E$$

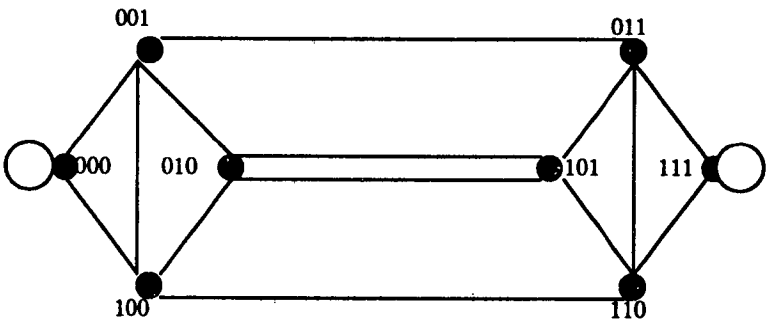
The *length* of a path is the number of edges on this path. The *distance* from i to j is the minimum length of the paths from i to j. The *diameter* of a graph is the maximum distance in the graph.

The de Bruijn graph (cf. de Bruijn [2]), denoted by $DG(d,k)$, has $N = d^k$ vertices with diameter k and degree $2d$. This corresponds to the state graph of a shift register of length k using d-ary digits. A shift register goes from a state to another by doing a shift operation. The multiprocessor system which can be modeled by a de Bruijn graph $DG(d,k)$ is called de Bruijn network, referred to as $DN(d,k)$.

Let $X = (x_1, \cdots, x_k)$, $x_i \in \{0, 1, \cdots, d-1\}$, $1 \le i \le k$, denote a vertex in $DG(d,k)$. For all $a \in \{0, 1, \cdots, d-1\}$, let $X^-(a) = (x_2, \cdots, x_k, a)$, and $X^+(a) = (a, x_1, \cdots, x_{k-1})$ denote the vertices obtained by the left and right shift operations, respectively. The directed $DG(d,k)$ is composed of the vertices $X = (x_1, \cdots, x_k)$, and the edges $X \to X^-(a)$ and $X^+(a) \to X$. The undirected de Bruijn graph can be obtained from the directed one by removing the directions of the edges. In what follows, we call $X^-(a)$ ($X^+(a)$, respectively) a *type-L* (*type-R*, respectively) neighbor of X.



(a)



(b)

Figure 1: Examples of de Bruijn graphs. (a) Directed $DG(2,3)$. (b) Undirected $DG(2,3)$.

Examples of directed and undirected de Bruijn graphs are given in Figure 1. The reader can see that each vertex is of degree $2d$ and there are Nd arcs (or edges) in the graph. By removing the redundant arcs (or edges), i.e., those with multiple occurrences in the graph, or those linking the same vertices, one can show that, in a directed $DG(d,k)$, there are $N-d$ vertices of degree $2d$ and d vertices of degree $2d-2$, and that in an undirected $DG(d,k)$, there exist $N-d^2$ vertices of degree $2d$, d^2-d vertices of degree $2d-1$ and d vertices of degree $2d-2$.

In the design of computer networks, one intends to increase the number of vertices while keeping a limited degree of graph (due to technical constraints) and a minimal graph diameter (since the transmission of a message in the network may be proportional to its diameter). One of the most attractive features of de Bruijn graphs is that they are nearly optimal graphs that minimize the diameter, given the number of vertices and the degree of the graph (see Imase and Itoh [4]).

Another concern of the design of a network is fault-tolerance, which becomes crucial when the system is large. It is shown in Pradhan and Reddy [8] that de Bruijn networks are able to tolerate up to $d-1$ processor failures.

De Bruijn networks are adequate for various applications. In fact, it is shown, in Samatham and Pradhan [9], that the binary ($d=2$) de Bruijn network allows one to represent various usual architectures such as linear arrays, rings, complete binary trees and shuffle-exchange networks, so that it can be used to solve efficiently many problems.

Other interesting characteristics of the de Bruijn networks include the existence of multiple Hamiltonian paths (de Bruijn [2], Etzion and Lempel [3]), extensibility (Samatham and Pradhan [9]) and regularity which admits easy routing procedures (see Section 3 below).

In this paper, we discuss the problem of optimal routing in the de Bruijn networks. In Section 2, we provide the distance functions for the directed and undirected de Bruijn graphs. In virtue of these distance functions, the problem of finding shortest paths is reduced to the classical pattern matching problem. In Section 3, we develop algorithms, in using the notion of *failure function* introduced by Morris, Pratt [7] and the notion of *prefix tree* by Weiner [10], for the optimal routing in the unidirectional and the bi-directional de Bruijn networks, respectively. The complexities in time and in space of these algorithms are linear in the diameter.

2 Distance Function

Let $DG(d,k)$ denote the de Bruijn graph under consideration, $d \geq 2$, $k \geq 1$. Let $X = (x_1, \cdots, x_k)$ and $Y = (y_1, \cdots, y_k)$ be two arbitrary vertices in $DG(d,k)$, where $x_i, y_i \in \{0, 1, \cdots, d-1\}$, $1 \leq i \leq k$. Denote by $D(X,Y)$ the distance from X to Y (recall that $D(X,Y) = D(Y,X)$ if $DG(d,k)$ is undirected). By convention, we define $D(X,X) = 0$ for all $X \in DG(d,k)$. In the following two subsections, we provide the distance functions for directed and undirected $DG(d,k)$'s, respectively.

Before proceeding with the discussions, we show that $DG(d,k)$ has diameter k. In fact, for all $X, Y \in DG(d,k)$, the following trivial path has length k:

$$X = X_0 \rightarrow X_1 \rightarrow \cdots \rightarrow X_k = Y,$$

where $X_i = X_{i-1}{}^-(y_i)$, $i = 1, 2, \cdots, k$. Therefore $DG(d,k)$ has diameter less than or equal to k. It is easy to see that the distance from $(0, \cdots, 0)$ to $(1, \cdots, 1)$ is k. Thus the diameter of $DG(d,k)$ is k.

2.1 Directed de Bruijn Graphs

Consider first the case where $DG(d,k)$ is a directed de Bruijn graph.

Property 1. *For all X, Y in the directed $DG(d,k)$,*

$$D(X,Y) = k - \max\{s \mid 1 \leq s \leq k,\ x_{k-s+1}x_{k-s+2}\cdots x_k = y_1 y_2 \cdots y_s\} \tag{1}$$

where, by convention, the maximum over an empty set is zero.

Proof. The proof of this fact is easy. Let

$$l = \max\{s \mid 1 \leq s \leq k,\ x_{k-s+1}\cdots x_k = y_1 \cdots y_s\}). \tag{2}$$

If $l = k$, then $X = Y$, so that $D(X,Y) = 0$, and equality (1) holds. Suppose $l < k$. Define the vertices

$$X_1 = X^-(y_{l+1}), \quad X_2 = X_1{}^-(y_{l+2}), \quad \cdots, \quad X_{k-l} = X_{k-l-1}{}^-(y_k).$$

One readily sees that

$$X \rightarrow X_1 \rightarrow \cdots \rightarrow X_{k-l} = Y$$

is a path from X to Y. Thus

$$D(X,Y) \leq k - l. \tag{3}$$

Consider now an arbitrary path

$$X \to Z_1 \to \cdots \to Z_j = Y.$$

We have necessarily,

$$Z_j = Z_{j-1}^{-}(y_k), \quad Z_{j-1} = Z_{j-2}^{-}(y_{k-1}), \quad \cdots, \quad Z_1 = X^{-}(y_{k-j+1}),$$

which implies that

$$Y = (y_1, \cdots, y_k) = Z_j = (x_{j+1}, \cdots, x_k, y_{k-j+1}, \cdots, y_k),$$

so that

$$y_1 y_2 \cdots y_{k-j} = x_{j+1} x_{j+2} \cdots x_k.$$

Using definition (2) yields $k - j \leq l$, so that

$$D(X,Y) \geq k - l. \tag{4}$$

We can thus conclude (1). ∎

Let $\delta(d,k)$ denote the average distance between vertices in the directed de Bruijn graph $DG(d,k)$. It is readily checked from (1) that

$$\delta(d,k) = \sum_{i=1}^{k} i\alpha^{k-i}\bar{\alpha}$$

where $\alpha = 1/d$, $\bar{\alpha} = 1 - \alpha$. Therefore

$$\delta(d,k) = k - (1 - \alpha^k)\alpha/\bar{\alpha} \tag{5}$$

In particular, when $d = 2$, $\alpha = \bar{\alpha} = 1/2$, so that $\delta(2,k) = k - 1 + 1/2^k$.

2.2 Undirected de Bruijn Graphs

In this subsection, we suppose that $DG(d,k)$ is undirected. Our main result concerning the distance function is the following:

Theorem 2. *For all X, Y in the undirected $DG(d,k)$,*

$$\begin{aligned} & D(X,Y) \\ &= 2k - 1 + \min\left\{ \min_{1 \leq i,j \leq k} (i - j - l_{i,j}(X,Y)), \min_{1 \leq i,j \leq k} (-i + j - r_{i,j}(X,Y)) \right\} \qquad (6) \\ &= 2k - 1 + \min_{1 \leq i,j \leq k} (i - j - \max\{l_{i,j}(X,Y), r_{j,i}(X,Y)\}) \qquad (7) \end{aligned}$$

where

$$l_{i,j}(X,Y) = \max\{s \mid s \le j,\ s \le k-i+1,\ x_i x_{i+1} \cdots x_{i+s-1} = y_{j-s+1} y_{j-s+2} \cdots y_j\} \quad (8)$$

$$r_{i,j}(X,Y) = \max\{s \mid s \le i,\ s \le k-j+1,\ x_{i-s+1} x_{i-s+2} \cdots x_i = y_j y_{j+1} \cdots y_{j+s-1}\} \quad (9)$$

where, by convention, the maximum over an empty set is zero.

Let

$$l(X,Y) = 2k - 1 + \min\left\{\min_{1 \le i,j \le k}(i - j - l_{i,j}(X,Y)), \min_{1 \le i,j \le k}(-i + j - r_{i,j}(X,Y))\right\} \quad (10)$$

In order to prove Theorem 2, we establish the following claim which is equivalent to Theorem 2.

Claim 3. *For all $0 \le n \ge k$, and all $X, Y \in DG(d,k)$, such that $D(X,Y) = n$, the relation*

$$D(X,Y) = l(X,Y) \quad (11)$$

holds.

Proof. We prove the claim by induction on n. For $n = 0$, the fact that $D(X,Y) = 0$ implies $X = Y$. We get immediately $l_{1,k}(X,Y) = k$, so that

$$l(X,Y) = 0 = D(X,Y).$$

Suppose now for some $0 \le n \le k-1$,

$$\forall X, Y \in DG(d,k),\ D(X,Y) \le n: \qquad D(X,Y) = l(X,Y). \quad (12)$$

Consider the pair of vertices $X, Y \in DG(d,k)$ such that $D(X,Y) = n+1$. We first prove that

$$D(X,Y) \le l(X,Y) \quad (13)$$

Examine (10), there exist s, t such that either

$$l(X,Y) = 2k - 1 + s - t - l_{s,t}(X,Y),$$

or

$$l(X,Y) = 2k - 1 - s + t - r_{s,t}(X,Y).$$

In the first case, denote $\theta = l_{s,t}(X,Y)$. We have

$$X = x_1 x_2 \cdots x_{s-1} y_{t-\theta+1} y_{t-\theta+2} \cdots y_t x_{s+\theta} x_{s+\theta+1} \cdots x_k.$$

In term of string, $Y = y_1 y_2 \cdots y_k$ can be obtained by shifting X. We first perform $s-1$ left shift operations on X to obtain an X', then $k-\theta$ right shifts on X' to obtain X'' by inserting successively on the left of X' the digits $y_{t-\theta}, y_{t-\theta-1}, \cdots, y_1, z_1, \cdots, z_{k-t}$, where $z_1, \cdots, z_{k-t}$ are arbitrarily chosen digits with values in $\{0, \cdots, d-1\}$, and finally $k-t$ left shifts on X'' to obtain Y by inserting successively the digits $y_{t+1}, \cdots, y_k$ on the right of X''.

In the second case, we proceed in a similar way. Let $\theta = r_{s,t}(X,Y)$. We have also

$$X = x_1 x_2 \cdots x_{s-\theta} y_t y_{t+1} \cdots y_{t+\theta-1} x_{s+1} x_{s+2} \cdots x_k.$$

We first do $k-s$ right shift operations on X to obtain an X', then $k-\theta$ left shifts on X' to obtain X'' by inserting successively the digits $y_{t+\theta}, y_{t+\theta+1}, \cdots, y_k, z_1, \cdots, z_{t-1}$ on the right of X', where $z_1, \cdots, z_{t-1}$ are arbitrarily chosen digits with values in $\{0, \cdots, d-1\}$, and finally $t-1$ right shifts on X'' to obtain Y by inserting successively the digits $y_{t-1}, \cdots, y_1$ on the left of X''.

The above algorithms show that Y can be reached from X in at most $l(X,Y)$ steps. Therefore, relation (13) holds.

We now prove that

$$D(X,Y) \geq l(X,Y) \tag{14}$$

Observe that all the subpaths of a shortest path are shortest paths. Thus

$$D(X,Y) = \min\left\{ \min_{0 \leq a \leq d-1} (D(X^-(a),Y)+1), \min_{0 \leq a \leq d-1} (D(X^+(a),Y)+1) \right\} \tag{15}$$

Since $D(X,Y) = n+1$, there exists some a_0, $0 \leq a_0 \leq d-1$ such that at least one of the inequalities

$$D(X^-(a_0),Y) \leq n, \quad \text{or} \quad D(X^+(a_0),Y) \leq n$$

holds. Therefore, equation (15) can be rewritten as

$$\begin{aligned} D(X,Y) \;=\; \min\{ & \min_{\{a \mid 0 \leq a \leq d-1,\ D(X^-(a),Y) \leq n\}} (D(X^-(a),Y)+1), \\ & \min_{\{a \mid 0 \leq a \leq d-1,\ D(X^+(a),Y) \leq n\}} (D(X^+(a),Y)+1)\} \end{aligned} \tag{16}$$

Using the inductive assumption (cf. (12)), we get

$$\begin{aligned} D(X,Y) \;=\; \min\{ & \min_{\{a \mid 0 \leq a \leq d-1,\ D(X^-(a),Y) \leq n\}} (l(X^-(a),Y)+1), \\ & \min_{\{a \mid 0 \leq a \leq d-1,\ D(X^+(a),Y) \leq n\}} (l(X^+(a),Y)+1)\} \end{aligned} \tag{17}$$

It follows from the definitions (8–9) that for all a, $0 \le a \le d-1$,

$$\begin{aligned}
l_{i,j}(X^-(a),Y) &\le \max\{l_{i+1,j}(X,Y),\ l_{i+1,j-1}(X,Y)+1\}, \quad 1 \le i \le k-1, \quad 2 \le j \le k,\\
l_{k,j}(X^-(a),Y) &\le 1, \quad 1 \le j \le k,\\
l_{i,1}(X^-(a),Y) &\le 1, \quad 1 \le i \le k,\\
r_{i,j}(X^-(a),Y) &\le r_{i+1,j}(X,Y), \quad 1 \le i \le k-1, \quad 1 \le j \le k,\\
r_{k,j}(X^-(a),Y) &\le r_{k,j}(X,Y)+1, \quad 1 \le j \le k,
\end{aligned}$$

which imply

$$\begin{aligned}
i-j-l_{i,j}(X^-(a),Y) &\ge \min\{(i+1)-j-l_{i+1,j}(X,Y)-1,\ (i+1)-(j-1)-l_{i+1,j-1}(X,Y)-1\},\\
&\qquad 1 \le i \le k-1, \quad 2 \le j \le k,\\
k-j-l_{k,j}(X^-(a),Y) &\ge k-j-1 \ge k-j-l_{k,j}(X,Y)-1, \quad 1 \le j \le k,\\
i-1-l_{i,1}(X^-(a),Y) &\ge i-1-1 \ge i-1-l_{i,1}(X,Y)-1, \quad 1 \le i \le k,\\
-i+j-r_{i,j}(X^-(a),Y) &\ge -(i+1)+j-r_{i+1,j}(X,Y)-1, \quad 1 \le i \le k-1, \quad 1 \le j \le k,\\
-k+j-r_{k,j}(X^-(a),Y) &\ge -k+j-r_{k,j}(X,Y)-1, \quad 1 \le j \le k.
\end{aligned}$$

Thus

$$\begin{aligned}
&l(X^-(a),Y)\\
&= 2k-1+\min\left\{\min_{1\le i,j\le k}(i-j-l_{i,j}(X^-(a),Y)),\ \min_{1\le i,j\le k}(-i+j-r_{i,j}(X^-(a),Y))\right\}\\
&\ge 2k-1+\min\left\{\min_{1\le i,j\le k}(i-j-l_{i,j}(X,Y)-1),\ \min_{1\le i,j\le k}(-i+j-r_{i,j}(X,Y)-1)\right\}\\
&= l(X,Y)-1
\end{aligned}$$

Analogously, it is readily checked that

$$\begin{aligned}
l_{1,j}(X^+(a),Y) &\le l_{1,j}(X,Y)+1, \quad 1 \le j \le k,\\
l_{i,j}(X^+(a),Y) &\le l_{i-1,j}(X,Y), \quad 2 \le i \le k, \quad 1 \le j \le k,\\
r_{i,j}(X^+(a),Y) &\le \max\{r_{i-1,j}(X,Y),\ r_{i-1,j+1}(X,Y)+1\}, \quad 2 \le i \le k, \quad 1 \le j \le k-1,\\
r_{1,j}(X^+(a),Y) &\le 1, \quad 1 \le j \le k,\\
r_{i,k}(X^+(a),Y) &\le 1, \quad 1 \le i \le k,
\end{aligned}$$

so that

$$\begin{aligned}
1-j-l_{1,j}(X^+(a),Y) &\geq 1-j-l_{1,j}(X,Y)-1, \qquad 1\leq j\leq k,\\
i-j-l_{i,j}(X^+(a),Y) &\geq (i-1)-j-l_{i-1,j}(X,Y)-1, \qquad 2\leq i\leq k, \quad 1\leq j\leq k,\\
-i+j-r_{i,j}(X^+(a),Y) &\geq \min\{-(i-1)+j-r_{i-1,j}(X,Y)-1,\ -(i-1)+(j+1)-r_{i-1,j+1}(X,Y)-1\},\\
&\qquad 2\leq i\leq k, \quad 1\leq j\leq k-1,\\
-1+j-r_{1,j}(X^+(a),Y) &\geq -1+j-r_{1,j}(X,Y)-1, \qquad 1\leq j\leq k\\
-i+k-r_{i,k}(X^+(a),Y) &\geq -i+k-r_{i,k}(X,Y)-1, \qquad 1\leq i\leq k.
\end{aligned}$$

Hence,

$$\begin{aligned}
&l(X^+(a),Y)\\
&= 2k-1+\min\left\{\min_{1\leq i,j\leq k}(i-j-l_{i,j}(X^+(a),Y)),\ \min_{1\leq i,j\leq k}(-i+j-r_{i,j}(X^+(a),Y))\right\}\\
&\geq 2k-1+\min\left\{\min_{1\leq i,j\leq k}(i-j-l_{i,j}(X,Y)-1),\ \min_{1\leq i,j\leq k}(-i+j-r_{i,j}(X,Y)-1)\right\}\\
&= l(X,Y)-1
\end{aligned}$$

To sumarize,

$$D(X^-(a),Y) \geq l(X,Y)-1, \tag{18}$$

$$D(X^+(a),Y) \geq l(X,Y)-1. \tag{19}$$

Applying these two relations to (17) immediately yields (14). In view of (13) and (14), the relation (11) holds for $n+1$. By induction, (11) holds for all $0\leq n\leq k$. ∎

Corollary 4. *For all $X,Y\in G$,*

$$D(X,Y)=2k-1+\min\left\{\min_{1\leq i\leq j\leq k}(i-j-l_{i,j}(X,Y)),\ \min_{1\leq i\leq j\leq k}(-i+j-r_{i,j}(X,Y))\right\} \tag{20}$$

Proof. The equality follows from Theorem 2 and the facts that

$$\begin{aligned}
i>j\geq 1 &\Rightarrow 2k-1+i-j-l_{i,j}(X,Y)\geq k\geq l_{1,k}(X,Y)\\
j>i\geq 1 &\Rightarrow 2k-1-i+j-r_{i,j}(X,Y))\geq k\geq r_{k,1}(X,Y).
\end{aligned}$$

∎

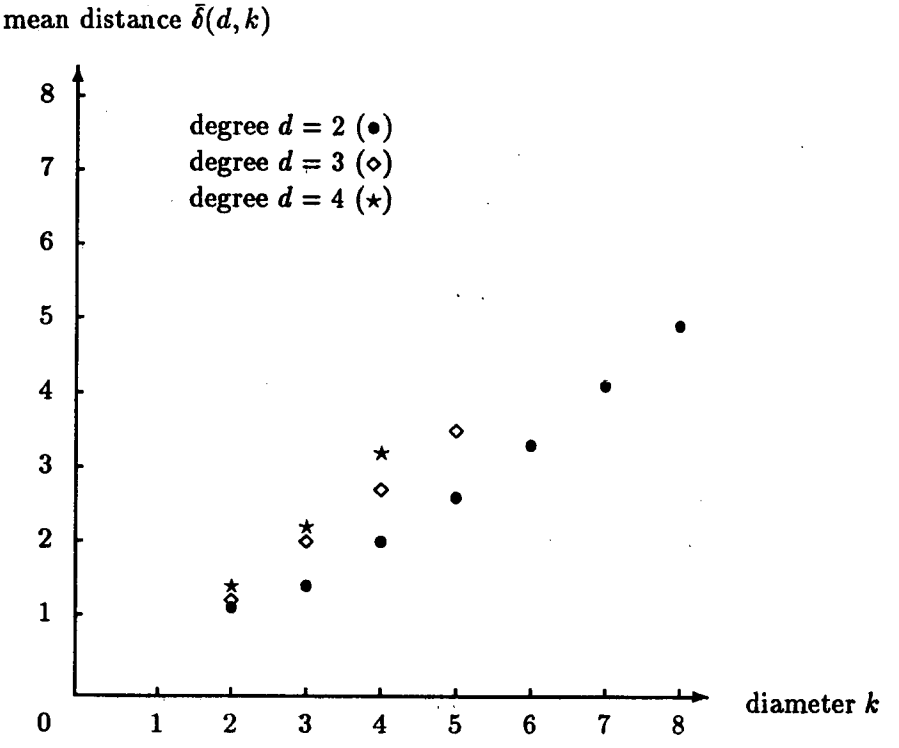


Figure 2: Average distance of undirected de Bruijn graphs

Let $\bar{\delta}(d,k)$ denote the average distance between verticse in the undirected de Bruijn graph $DG(d,k)$. It is not easy to give a simple relation between $\bar{\delta}(d,k)$ and d,k. Numerical results are provided in Figure 2.

3 Optimal Routing Algorithms for the De Bruijn Networks

In de Bruijn networks, the connections between processors are highly regular: They are described by shift operations. This allows one to specify routing paths in a very simple way: A path from X to Y

$$X = X_0 \rightarrow X_1 \rightarrow \cdots \rightarrow X_n = Y$$

of length n can be described by $2n$ digits of d-ary: $a_1b_1a_2b_2\cdots a_nb_n$, where $a_i = 0$ or 1, $0 \leq b_i \leq d-1$, and

$$a_i = 0,\ b_i = u \quad \text{if} \quad X_i = X_{i-1}{}^-(u), \qquad a_i = 1,\ b_i = u \quad \text{if} \quad X_i = X_{i-1}{}^+(u).$$

In words, a_i indicates the type of neighbor, b_i identifies the neighbor of this type. In case the de Bruijn network is uni-directional, the digits $a_1, \cdots, a_n$, which are all equal to zero, can be omitted.

Thus, when a message is generated, it is composed of five fields: control code, source address, destination address, routing path, and the message content. The routing path field is of the form $\{(a_1, b_1), \cdots, (a_n, b_n)\}$, each pair specifying the selection of a particular neighbor. When a site, say X, receives a message, it looks at the routing path field. If it is empty, then the message is destined for this site, and the message is accepted. If, however, the routing path field is not empty, the site removes the first element (pair) (a, b) from the field and transmits the message to the neighbor with address Z:

$$Z = X^-(b) \quad \text{if} \quad a = 0, \qquad Z = X^+(b) \quad \text{if} \quad a = 1.$$

When the network is uni-directional, the value of a is not important, in which case, an element in the field of routing path can simply be the value of b.

In a computer network, the optimal routing consists in finding a shortest routing path from a site to another. In the above section, the length of the shortest paths from an arbitrary vertex X to another arbitrary vertex Y in a de Bruijn graph is analyzed. In this section, we develop the algorithms that generates the shortest paths for any arbitrary pair of source and destination. The complexities in time and in space of these algorithms are also analyzed.

3.1 Algorithms of optimal routing path generation in the de Bruijn networks

Consider first the simpler case: The de Bruijn network $DN(d, k)$ is uni-directional. The reader might have observed that an algorithm which finds the shortest paths was also provided in the proof of Property 1. We rewrite this in a more explicit form. Let $X = (x_1, \cdots, x_k)$, $Y = (y_1, \cdots, y_k)$. In the algorithm, a path $\mathcal{P}$

$$X = X_0 \rightarrow X_1 \rightarrow \cdots \rightarrow X_n = Y,$$

where $X_i = X_{i-1}{}^-(b_i)$, $i = 1, \cdots, n$, is represented by $\mathcal{P} = \{b_1, b_2, \cdots, b_n\}$.

Algorithm 1: Find a shortest path $\mathcal{P}$ from X to Y in the uni-directional de Bruijn network $DN(d, k)$.

begin

1. **if** $X = Y$
2. **then** $\mathcal{P} = \emptyset$
   else
   begin
3. Use **Algorithm 3** (see below) to compute l which is defined by (2);
4. $\mathcal{P} = \{y_{l+1}, y_{l+2}, \cdots, y_k\}$;
   end;

end.

Assume now that the de Bruijn network $DN(d,k)$ is bi-directional. Let $X = (x_1, \cdots, x_k)$, $Y = (y_1, \cdots, y_k)$ be the source and destination, respectively. A path $\mathcal{P}$:

$$X = X_0 \rightarrow X_1 \rightarrow \cdots \rightarrow X_n = Y,$$

where $X_i = X_{i-1}{}^-(b_i)$, or $X_i = X_{i-1}{}^+(b_i)$, $i = 1, \cdots, n$, is represented by $\mathcal{P} = \{(a_1, b_1), (a_2, b_2), \cdots, (a_n, b_n)\}$, where $a_i = 0$ (resp. $a_i = 1$) if $X_i = X_{i-1}{}^-(b_i)$ (resp. $X_i = X_{i-1}{}^+(b_i)$), $i = 1, \cdots, n$.

As in the previous case, an algorithm which finds the shortest paths in the undirected de Bruijn graph was provided in the proof of Theorem 2.

Algorithm 2: Find a shortest path $\mathcal{P}$ from X to Y in the bi-directional de Bruijn network $DN(d,k)$.

begin

1. **if** $X = Y$
2. **then** $\mathcal{P} = \emptyset$
   else
   begin
3. Use **Algorithm 3** (see below) to compute $l_{i,j}(X,Y)$, $1 \le i,j \le k$,

which are defined by (8); and let

$$D_1 = 2k - 1 + s_1 - t_1 - \theta_1 = \min_{1 \le i,j \le k} 2k - 1 + i - j - l_{i,j}(X,Y),$$

},

where $\theta_1 = l_{s_1,t_1}(X,Y)$;

4. Use **Algorithm 3** to compute $r_{i,j}(X,Y)$, $1 \le i,j \le k$,

which are defined by (9); and let

$$D_2 = 2k - 1 - s_2 + t_2 - \theta_2 = \min_{1 \le i,j \le k} 2k - 1 - i + j - r_{i,j}(X,Y),$$

where $\theta_2 = r_{s_2,t_2}(X,Y)$;

5. **if** $D_1 = D_2 = k$

6. **then** $\mathcal{P} = \{(0,y_1),(0,y_2),\cdots,(0,y_k)\}$;

7. **else if** $D_1 \le D_2$

8. **then**

$\mathcal{P} = \{(0,u_1),\cdots,(0,u_{s_1-1}),(1,y_{t_1-\theta_1}),(1,y_{t_1-\theta_1-1}),\cdots,(1,y_1),(1,v_1),\cdots,(1,v_{k-t_1}),(0,y_{t_1+1}),\cdots,(0,y_k)\}$,

where $u_1,\cdots,u_{s_1-1},v_1,\cdots,v_{k-t_1}$ are arbitrarily chosen d-ary digits;

9. **else**

$\mathcal{P} = \{(1,u_1),\cdots,(1,u_{k-s_2}),(0,y_{t_2+\theta_2}),(0,y_{t_2+\theta_2+1}),\cdots,(0,y_k),(0,v_1),\cdots,(0,v_{t_2-1}),(1,y_{t_2-1}),\cdots,(1,y_1)\}$,

where $u_1,\cdots,u_{k-s_2},v_1,\cdots,v_{t_2-1}$ are arbitrarily chosen d-ary digits;

end ;

end.

Observe that it would be convenient to introduce a special symbol "$*$" in the routing path specification to indicate that $(0,*)$ (resp. $(1,*)$) is an arbitrary neighbor of type-L (resp. type-R). This would allow the site which transmits the message to be able to select freely one of the neighbors of the specified type, so that the traffic could be more or less balanced. If such a symbol is used, the site specifications (a,u_i) or (a,v_i), $a = 0,1$, in lines 8 and 9 of the above algorithm, can be replaced by $(a,*)$.

We must now develop algorithms for the computation of l (defined by (2)) and $l_{i,j}(X,Y)$, $r_{i,j}(X,Y)$ (defined by (8-9)). Observe first that $l = r_{k,1}$, and that the computations of $r_{i,j}(X,Y)$ are analogous to those of $l_{i,j}(X,Y)$. Therefore, we focus on the computations of $l_{i,j}(X,Y)$, $1 \le i,j \le k$. Observe also that $l_{i,j}(X,Y)$ is the length of the longest substring of X which starts with x_i and matches a substring of Y which terminates with y_j. Thus, these computations reduce to a pattern matching problem. We will call $l_{i,j}(X,Y)$, or simply $l_{i,j}$, the matching function. Two different algorithms will be presented.

3.2 Computing matching functions by failure function

The following algorithm slightly generalizes the algorithm of the computation of *failure functions* which was previously designed by Morris and Pratt [7], for the problem of pattern matching in a string.

Let $X = (x_1, \cdots, x_k)$, $Y = (y_1, \cdots, y_k)$. Let $c_{i,j}$, $j \geq i$, denote the failure function of the pattern $x_i x_{i+1} \cdots x_k$:

$$c_{i,j} = \max\{s \mid 1 \leq s \leq j-i,\ x_i x_{i+1} \cdots x_{i+s-1} = x_{j-s+1} x_{j-s+2} \cdots x_j\}$$

where, by convention, the maximum over an empty set takes value zero.

Algorithm 3: Compute $l_{i,1}(X,Y), l_{i,2}(X,Y), \cdots, l_{i,k}(X,Y)$.

begin

1. $c_{i,i} = 0$;
2. **for** $j = i+1$ **to** k **do**
   begin
3. $h = c_{i,j-1}$;
4. **while** $h > 0$ **and** $x_{i+h} \neq x_j$ **do** $h = c_{i,i+h-1}$;
5. **if** $h = 0$ **and** $x_{i+h} \neq x_j$
6. **then** $c_{i,j} = 0$
7. **else** $c_{i,j} = h+1$;
   end;
8. **if** $x_i = y_1$ **then** $l_{i,1} = 1$ **else** $l_{i,1} = 0$;
9. **for** $j = 2$ **to** k **do**
   begin
10. **if** $l_{i,j-1} = k-i+1$ **then** $h = c_{i,k}$ **else** $h = l_{i,j-1}$;
11. **while** $h > 0$ **and** $x_{i+h} \neq y_j$ **do** $h = l_{i,i+h-1}$;
12. **if** $h = 0$ **and** $x_{i+h} \neq y_j$
13. **then** $l_{i,j} = 0$
14. **else** $l_{i,j} = h+1$;
    end;

end.

One can show by induction on j (in a similar way as Aho, Hopcroft and Ullman [1]) that Algorithm

3 correctly computes the failure functions $c_{i,i}, c_{i,i+1}, \cdots, c_{i,k}$ and the matching functions $l_{i,1}, l_{i,2}, \cdots, l_{i,k}$.

It is easy to see that Algorithm 3 is of the same order of complexity in time and in space as Morris and Pratt's algorithm, so that the complexities in time and in space of Algorithm 3 are $O(k)$.

In Algorithm 1, lines 1, 2 and 4 cost at most $2k$ units of time. Line 3 calls Algorithm 3 once, so it takes $O(k)$ time unit. Hence the time complexity of Algorithm 1 is $O(k)$. It is immediate that the space complexity of Algorithm 1 is $O(k)$.

In Algorithm 2, lines 1, 2, 6, 8 and 9 cost at most $2k$ units of time. Lines 3 and 4 make use of Algorithm 3 for at most $2k$ times. This is of quadratic cost $O(k^2)$. These lines also search for the two pairs (s_1, t_1) and (s_2, t_2) which yield minimal distances D_1 and D_2. Such a search cost at most $2k^2$ units of time. Lines 5 and 7 are of constant cost. To summarize, the time complexity of Algorithm 2 is $O(k^2)$.

It is not difficult to see that lines 3 and 4 of Algorithm 2 can be implemented in linear space $O(k)$ with quadratic cost in time. Indeed, it suffices to store for all $i = 1, 2, \cdots, k$, the values $c_{i,j}$ and $l_{i,j}$, in $c_{1,j}$ and $l_{1,j}$, respectively, $j = 1, 2, \cdots, k$. We rewrite line 3 of Algorithm 2 as follows:

$D_1 = k$;
for $i = 1$ **to** k **do**
  begin
    Use **Algorithm 3** to compute $l_{i,1}, \cdots, l_{k,j}$;
    Let $s - t - l_{s,t} = \min_{1 \le j \le k} i - j - l_{i,j}$;
    if $D_1 > 2k - 1 + s - t - l_{s,t}$
      then $s_1 = s$, $t_1 = t$, $D_1 = 2k - 1 + s_1 - t_1 - l_{s_1,t_1}$;
  end

The computation of $r_{i,j}$, s_2, t_2 and D_2 can be carried out in the same way. Therefore, the complexity in space of Algorithm 2 is $O(k)$.

3.3 The use of prefix trees

Owing to Theorem 2, the optimal routing problem reduces to that of pattern matching. The above two subsections provide a linear and a quadratic in time (in the diameter) algorithms for optimal routing in uni-directional and bi-directional de Bruijn networks, respectively. We now use the notion

of *prefix tree*, introduced by Wiener [10], to obtain a linear algorithm for the bi-directional de Bruijn network.

Some definitions are in order. Let $S = a_1 a_2 \cdots a_n$ be a string of length n over some alphabet $\mathcal{A}$. A *position* in the string is an integer between 1 and n. A symbol $a \in \mathcal{A}$ occurs in position i of string S if $a = a_i$. A substring P *identifies* position i in S if $S = Q_1 P Q_2$, $|Q_1| = i - 1$, and there is no $Q_1' \neq Q_1$ such that $S = Q_1' P Q_2'$. In other words, the only occurrence of P in S begins at position i. Let $\perp$ be a symbol not in $\mathcal{A}$, called the *endmarker*, and let $a_{n+1} = \perp$. It then follows that each position i of the string $S\perp = a_1 \cdots a_n a_{n+1}$ is identified by at least one substring, namely $a_i \cdots a_n a_{n+1}$. We call the sortest substring that identifies position i in $S\perp$ the *prefix identifier* of position i in $S\perp$, denoted by $P(i)$.

The *prefix tree* for the string $S\perp$ is a labeled out-tree T such that

1. For each interior vertex $v \in T$, the edges leaving v have distinct labels in $\mathcal{A} \bigcup \{\perp\}$;
2. T has $n+1$ leaves labeled $1, 2, \cdots, n, n+1$, corresponding to the positions of the string $S\perp$;
3. The sequence of labels of edges on the path from the root to the leaf labeled i is $P(i)$, the prefix identifier of position i in $S\perp$.

The use of endmarker guarantees the existence of a unique prefix tree for any given string. Many pattern matching problems can be solved by prefix trees. For instance, if one wants to find a longest repeated substring in S, it suffices to locate the interior vertex of the prefix tree of $S\perp$ with the maximal depth (the *depth* of a vertex in an out-tree is the length of the path from the root to the vertex). The sequence of labels of the edges on the path from the root to this vertex corresponds to the longest substring, and the leaves in the subtree obtained with this vertex as the root corresponds to the positions where the longest substring occurs.

Note that a prefix tree for a string of length n may have $O(n^2)$ vertices. However, we can *compact* a prefix tree by condensing every *chain* (a chain is a path all of whose vertices have exactly one son) in the prefix tree into a single vertex. One can easily show that the *compact prefix tree* has $O(n)$ vertices. Each vertex of the compact prefix tree is associated with the depth of the vertex in the initial prefix tree; the vertices obtained by condensing chain are associated with the depths of the deepest vertices on the chains of the initial prefix tree. A linear (in time and in space) algorithm that constructs a compact prefix tree from any given string over a fixed alphabet is provided by Weiner [10].

We are now in a position to show that a shortest path between two arbitrary vertices X and Y in

the bi-directional de Bruijn network $DN(d,k)$ can be found in $O(k)$ in time and in space.

Let $X = x_1x_2\cdots x_k$, $Y = y_1y_2\cdots y_k$, and $\bar{X} = x_kx_{k-1}\cdots x_1$, $\bar{Y} = y_ky_{k-1}\cdots y_1$. Let also $S = X\perp\bar{Y}\top$ and $\bar{S} = \bar{X}\perp Y\top$ be two strings, where $\perp$ and $\top$ are two different endmarkers, $\perp,\top \notin \{0,1,\cdots,d-1\}$.

Algorithm 4: Using prefix tree to find a shortest path $\mathcal{P}$ from X to Y in the bi-directional de Bruijn network $DN(d,k)$.

(the same as Algorithm 2 except that lines 3 and 4 are replaced by the following)

3.0 Use Weiner's Algorithm to construct the compact prefix tree T for string S;

let the vertices in T be labeled in such a way that the leaves are labeled $1,2,\cdots,2k+2$, corresponding to the positions in S, the root is labeled 0, and the interior vertices other than the root are labeled with different negative integers;

let $D(v)$ denote the depth associated with vertex v;

3.1 For all $v \in T$, compute $p(v)$ and $q(v)$ such that

$$p(v) = \begin{cases} v, & 1 \le v \le k, \\ 2k+2, & k+1 \le v \le 2k+2, \\ \min_{u\in s(v)} p(u), & v \le 0, \end{cases}$$

$$q(v) = \begin{cases} 2k+2, & 1 \le v \le k+1, \quad \text{or} \quad v = 2k+2, \\ v-k-1, & k+2 \le v \le 2k+1, \\ \min_{u\in s(v)} q(u), & v \le 0, \end{cases}$$

where $s(v)$ denotes the set of sons of $v \in T$.

3.2 Find the interior vertex $w \in T$ such that

$$p(w)+q(w)-D(w) = \min_{\{v|p(v)+q(v)\le 2k\}} p(v)+q(v)-D(v),$$

where the set $\{v|p(v)+q(v) \le 2k\}$ is not empty, it contains at least the root of T;

3.3 Let $s_1 = p(w)$, $t_1 = k+1-q(w)$, $D_1 = k-2+p(w)+q(w)-D(w)$;

4.0 Use Weiner's Algorithm to construct the compact prefix tree $\bar{T}$ for string $\bar{S}$;

let the vertices in $\bar{T}$ be labeled in such a way that the leaves are labeled $1,2,\cdots,2k+2$, corresponding to the positions in $\bar{S}$, the root is labeled 0, and the interior vertices other than the root are labeled with different negative integers;

let $\bar{D}(v)$ denote the depth associated with vertex v;

4.1 For all $v \in \bar{T}$, compute $\bar{p}(v)$ and $\bar{q}(v)$ such that

$$\bar{p}(v) = \begin{cases} v, & 1 \le v \le k, \\ 2k+2, & k+1 \le v \le 2k+2, \\ \min_{u \in \bar{s}(v)} \bar{p}(u) & v \le 0, \end{cases}$$

$$\bar{q}(v) = \begin{cases} 2k+2, & 1 \le v \le k+1, \quad \text{or} \quad v = 2k+2, \\ v-k-1, & k+2 \le v \le 2k+1, \\ \min_{u \in \bar{s}(v)} \bar{q}(u) & v \le 0, \end{cases}$$

where $\bar{s}(v)$ denotes the set of sons of $v \in \bar{T}$.

4.2 Find the interior vertex $\bar{w} \in \bar{T}$ such that

$$\bar{p}(\bar{w}) + \bar{q}(\bar{w}) - \bar{D}(\bar{w}) = \min_{\{v | \bar{p}(v) + \bar{q}(v) \le 2k\}} \bar{p}(v) + \bar{q}(v) - \bar{D}(v),$$

4.3 Let $s_2 = k + 1 - \bar{p}(\bar{w})$, $t_2 = \bar{q}(\bar{w})$, $D_2 = k - 2 + \bar{p}(\bar{w}) + \bar{q}(\bar{w}) - \bar{D}(\bar{w})$;

The complexity in space of Algorithm 4 is trivially $O(k)$, owing to the fact that the compact prefix trees T and $\bar{T}$ of strings S and $\bar{S}$ have $O(k)$ vertices. Line 3.0 uses Weiner's algorithm for the construction of the compact prefix tree, so it is linear in time $O(k)$. Line 3.1 costs $O(k)$ in time, as the computations of $p(v)$ and $q(v)$ can be completed by a simple visit in the compact prefix tree T. Similarly, line 3.2 costs $O(k)$ in time, which is the cost of a visit in the compact prefix tree T. Line 3.3 is of constant cost. Lines 4.0–4.3 are of the same cost as 3.0–3.3. Therefore, Algorithm 4 is linear in time $O(k)$.

The correctness of this algorithm is guaranteed by the following proposition.

Proposition 5. *Let w be defined by line 3.2 of Algorithm 4. Then*

$$2k - 1 + \min_{1 \le i,j \le k} i - j - l_{i,j}(X, Y) = k - 2 + p(w) + q(w) - D(w) \tag{21}$$

Proof. Denote by $T(v)$ the subtree of T obtained by taking $v \in T$ as the root. Let $L(v)$ be the substring of S that is obtained by concatenating the labels of the edges in the path from the root to v in the initial prefix tree. By definition, $p(v)$ (respectively $q(v) + k + 1$) represents the leaf of $T(v)$ with the smallest label, and hence corresponds to the smallest position $p(v)$ in X (resp. $q(v)$ in $\bar{Y}$) which has the prefix $L(v)$, provided $1 \le p(v) \le k$ (resp. $1 \le q(v) \le k$).

For all $1 \leq h_1 < h_2 \leq 2k+2$, let $f(h_1, h_2)$ be the vertex of T such that $T(f(h_1, h_2))$ is the smallest subtree that contains the leaves labeled h_1 and h_2. It follows from the construction of the compact prefix tree T of the string $S = X \perp \bar{Y} \top$, that $D(f(h_1, h_2))$ represents the length of the longest common substrings started from the positions h_1 and h_2 in S, respectively. Hence, for all $v \in T$, and all $1 \leq i, j \leq k$, such that $f(i, 2k+2-j) = v$, the relation

$$l_{i,j}(X, Y) = D(v)$$

holds. More over, $i \geq p(v)$, $2k+2-j \geq k+1+q(v)$, so that

$$2k-1+i-j-l_{i,j}(X,Y) \geq k-2+p(v)+q(v)-D(v) \geq k-2+p(w)+q(w)-D(w),$$

which completes the proof. ∎

4 Remarks

We have presented the routing algorithms in a succinct form. In order to gain efficiency, some mechanical transformations on the programs are necessary, see Knuth [5], Knuth, Morris, and Pratt [6] for discussions on the possible approaches. Appropriately implemented, the constant factors of our linear algorithms are low enough to make these algorithms of practical use.

It is clear that when the diameter k of the de Bruijn network is small, the use of conceptually simpler pattern matching algorithms, which would yield routing algorithms of complexity $O(k^2)$ or even $O(k^3)$, may not be worse than the linear algorithms.

Acknowledgements : The author is grateful to Philippe Mussi for useful discussions on this work, and to Michel Syska for his numerical computations.

References

[1] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[2] N. G. de Bruijn, "A Combinatorial Problem", *Proc. Akademe Van Wetenschappen*, Vol. 49, part 2, pp. 758-764, 1946.

[3] T. Etzion, A. Lempel, "Algorithms for the Generation of Full-Length Shift-Register Sequences", *IEEE Trans. on Information Theory*, Vol. IT-30, No. 3, pp. 480-484, 1984.

[4] M. Imase, M. Itoh, "Design to Minimize Diameter on Building-Block Network", *IEEE Trans. on Computers*, Vol. C-30, No. 6, pp. 439-442, 1981.

[5] D. E. Knuth, "Structured Programming with **go to** statements," *Computing Surveys*, Vol. 6, pp. 261-301, 1974.

[6] D. E. Knuth, J. H. Morris, V. R. Pratt, "Fast Pattern Matching in Strings", *SIAM J. Comp.*, Vol. 6, No. 2, pp. 323-350, 1977.

[7] J. H. Morris, V. R. Pratt, "A Linear Pattern Matching Algorithm," Technical Report No. 40, Computing Center, University of California, Berkeley, 1970.

[8] D. K. Pradhan, S. M. Reddy, "A Fault-Tolerant Communication Architecture for Distributed Systems", *IEEE Trans. on Computers*, Vol. C-31, No. 9, pp. 863-870, 1989.

[9] M. R. Samatham, D. K. Pradhan, "The De Bruijn Multiprocessor Network: A Versatile Parallel Processing and Sorting Network for VLSI", *IEEE Trans. on Computers*, Vol. C-38, No. 4, pp. 567-581, 1989.

[10] P. Weiner, "Linear Pattern Matching Algorithms," *Proc. IEEE 14th Annual Symposium on Switching and Automata Theory*, pp. 1-11, 1973.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique