On Leaf Node Edge Switchings in Spanning Trees of De Bruijn Graphs

Suman Roy[1(✉)], Srinivasan Krishnaswamy[1], and P. Vinod Kumar[2]

[1] Department of Electronics and Electrical Engineering,
Indian Institute of Technology Guwahati, Guwahati 781039, Assam, India
{suman.roy,srinikris}@iitg.ernet.in
[2] Bharat Broadband Network Limited, Trivandrum, India
saivinod.potnuru@gmail.com

Abstract. An n-th order k-ary de Bruijn sequence is a cyclic sequence of length k^n which contains every possible k-ary subsequence of length n exactly once during each period. In this paper, we show that, if we fix the initial n bits, any n-th order de Bruijn sequence can be transformed to another using a sequence of transformations.

Keywords: De Bruijn sequences · De Bruijn graph
Pseudorandom sequence generator · Shift register

1 Introduction

An n-th order k-ary de Bruijn sequence, $DB_n(k)$, is a periodic sequence of length k^n having every possible k-ary subsequence of length n exactly once in each period. An example of $DB_2(3)$ is 001021122. In [3], it has been shown that there exist $((k-1)!)^{k^{n-1}} k^{k^{(n-1)}-n}$ k-ary de Bruijn sequences of order n. De Bruijn sequences satisfy many statistical properties associated with randomness such as balance property, span-n property, etc. Thus, they find many applications ranging from cryptography and coding theory to communication systems [11,12]. This paper deals with binary de Bruijn sequences although the results can be easily extended for k-ary de Bruijn sequences. Feedback Shift Registers (FSRs) have been used to generate such sequences for many decades [7]. There are a number of algorithms available in literature to generate de Bruijn sequences using shift registers [5]. One way of generating de Bruijn sequences is by joining various FSRs of shorter cycles [4,8]. Given a k-ary de Bruijn sequence of order n, other such sequences can be obtained by using cross-join pairs [6,10]. Recursive algorithms to produce higher order de Bruijn sequences from the lower order de Bruijn sequences have been discussed in [1,9]. In this paper, we use the enumerative construction given in [2] to show that any n-th order de Bruijn sequence can be generated from another by using a set of transformations. Here all de Bruijn sequences that are cyclic shifts of each other shall be considered equivalent.

D. Ghosh et al. (Eds.): ICMC 2018, CCIS 834, pp. 110–117, 2018.
https://doi.org/10.1007/978-981-13-0023-3_11

The remainder of this paper is organized as follows. Section 2 introduces de Bruijn graphs and contains a brief description of the algorithm given in [2]. Section 3 contains the main results of the paper. In Sect. 4, we summarize the results and conclude the paper.

2 Preliminaries

Let $G = (V, E)$ be a directed graph where V and E denote the vertex (or node) set and the edge set in G respectively. Every edge $e \in E$ is directed from the source vertex $s(e)$ to the target vertex $t(e)$. For all $v \in V$, $indeg(v)$ and $outdeg(v)$ are the number of incoming and outgoing edges respectively. An Eulerian cycle (or Eulerian circuit) in a directed graph is a directed cycle which uses every edge $e \in E$ exactly once. A graph that contains an Eulerian cycle is called an Eulerian graph. For a connected directed balanced graph G, there exists at least one Eulerian circuit. De Bruijn sequences are closely associated with special directed Eulerian graphs known as de Bruijn graphs [3]. A binary de Bruijn graph of order n, denoted as G_n, is a directed graph with 2^n vertices, each labeled with a unique n bit string. Each edge of the graph is labeled with a binary string of length $(n + 1)$. The edge labeled as $s_0 s_1 \dots s_n$ connects the source vertex labeled $s_0 s_1 \dots s_{n-1}$ with the target vertex labeled $s_1 s_2 \dots s_n$.

Example 1. Figure 1 represents a second order binary de Bruijn graph G_2.

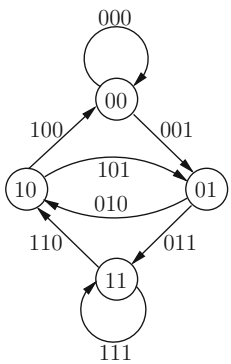

Fig. 1. G_2: De Bruijn graph of order 2.

In G_n, each vertex $v = (s_0 s_1 \dots s_{n-1})$ has two out-edges $(s_0 s_1 \dots s_{n-1} 1)$ and $(s_0 s_1 \dots s_{n-1} 0)$; these edges are known as the one-edge and the zero-edge of v respectively. Since de Bruijn graphs are connected and balanced they always contain an Eulerian cycle. Observe that we can obtain an $(n + 1)$-th order de Bruijn sequence by considering the sequence of most significant bits of edges in an Eulerian cycle of G_n. Clearly, there exists a one-to-one correspondence

between Eulerian cycles of G_n and $(n+1)$-th order de Bruijn sequences. We now proceed to briefly describe the process of generating Eulerian cycles of G_n given in [2].

An oriented spanning tree is an acyclic subgraph of a directed graph $G = (V, E)$. It has a vertex $r \in V$ known as root vertex such that $outdeg(r) = 0$ and there exists a path from every vertex $v \in V \backslash \{r\}$ to r. For a directed graph shown in Fig. 1, an oriented spanning tree T rooted at 00 is given in Fig. 2.

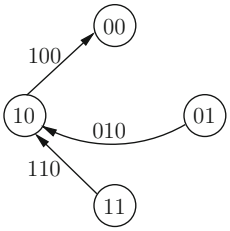

Fig. 2. T: Spanning tree of G_2 rooted at 00.

Let T be a spanning tree of $G_n = (V, E)$ where V is the set of vertices and E is the edge set in G_n. Now, for all $v \in V$, construct the lists l_v having $indeg(v)$ number of edges. This array of lists is known as *tree array*. For all $v \in V \backslash \{r\}$, the last element of l_v will be the unique outgoing edge of v which occurs in the spanning tree T. In case of the root vertex r, last element of l_r is the symbol Ω and the first entry of l_r can be any of its outgoing edges in G_n.

Example 2. Consider the de Bruijn graph G_2 shown in Fig. 1. A spanning tree of G_2 is shown in Fig. 2. A tree array l_v corresponding to T is given as follows:

$$
\begin{aligned}
l_{00} &= \{(000), \quad \Omega\} \\
l_{01} &= \{(011), (010)\} \\
l_{10} &= \{(101), (100)\} \\
l_{11} &= \{(111), (110)\}
\end{aligned}
$$

Now, let T be a spanning tree of G_n rooted at the vertex r and consider its corresponding tree array. One can obtain the Eulerian cycle of G_n as follows. Starting with the unique edge of G_n which does not lie in the tree array, each edge x is followed by the first unused outgoing edge of $t(x)$ in the tree array. This process stops when the only unused entry of the tree array is Ω. Thus, given a spanning tree and a tree array we can generate an Eulerian cycle in G_n. Further, it has been shown in [2] that the correspondence between a spanning tree - tree array pair and Eulerian cycles is one-to-one and one can easily obtain one from the other.

Example 3. Consider the G_2 given in Fig. 1 and its spanning tree T rooted at 00 shown in Fig. 2. A tree array for the spanning tree is given in Example 2.

We start with the edge 001 followed by the edge 011 which is the first unused outgoing edge of the vertex $01 = t(001)$. By repeating this process, we get the Eulerian cycle of G_2 shown in Fig. 3. The corresponding 3-rd order de Bruijn sequence is 00111010.

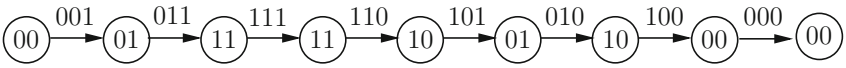

Fig. 3. Eulerian cycle of the de Bruijn graph G_2.

3 Construction of Any Spanning Tree of G_n from a Spanning Tree of G_n

Recall that, a binary de Bruijn graph G_n having a vertex $s = (s_0 s_1 \dots s_{n-1})$ has two out-edges, namely zero-edge and one-edge, which connect s to its successor vertex $(s_1 s_2 \dots s_{n-1} 0)$ and $(s_1 s_2 \dots s_{n-1} 1)$ respectively. These vertices are called conjugates of each other and corresponding edges are called conjugate edges. In a spanning tree of G_n, any vertex s is connected to one of its successor vertices. The all-zero and all-one vertices have only one possible successor in the spanning tree. Therefore, we consider these vertices merged with their respective successor vertices in the spanning tree as a single vertex. For example, consider the spanning tree of G_3 shown in Fig. 4. Here 000 and 001 (similarly, 111 and 110) are jointly treated as a single vertex $000 - 001$ $(111 - 110)$.

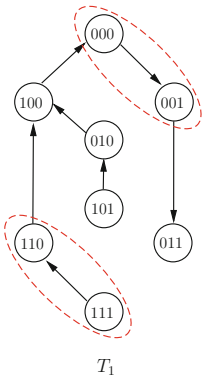

Fig. 4. A spanning tree of G_3 rooted at 011.

Note that fixing the root vertex and its unique outgoing edge that does not lie in the tree array essentially fixes the starting edge of the Eulerian cycle (therefore, the first n-bits of the de Bruijn sequence). Therefore, if we fix the entry

corresponding to the root vertex in the tree array, we have a one-to-one correspondence between de Bruijn sequences with a given initial state and oriented spanning trees of G_n. In the remainder of this section we will show that from a given spanning tree of G_n rooted at a particular vertex we can generate any other spanning tree of G_n having the same root by a sequence of transformations.

The process of replacing the outgoing edge of one vertex in the spanning tree by its conjugate edge is known as edge switching. In a spanning tree T, a leaf node is a node that does not have any incoming edge. For example, consider one of the binary spanning trees of G_2, T, rooted at 00 given in Fig. 2. Here, 01 and 11 are the leaf nodes.

Lemma 1. *Switching the outgoing edge of a leaf node in a spanning tree of G_n generates another spanning tree.*

Proof. Let T be a spanning tree of G_n. Suppose an outgoing edge e of a vertex $v \in T$ is replaced by its conjugate edge e', then this switching results in a cycle only if $t(e)$ is a vertex from which there exists a path to v. Otherwise the resulting graph will be an another spanning tree. Now, if the node $v \in T$ is a leaf node then there exists no path to v from any other vertex. Therefore, when the outgoing edge of a leaf node is switched the resulting graph is another spanning tree of G_n.

Example 4. Consider the spanning tree, T_1, of G_3 as given in Fig. 4. T_1 has two leaf nodes viz. 101 and the merged pair $111 - 110$. Switching of these edges, 101 and $111 - 110$, produce two different spanning trees (see Fig. 5).

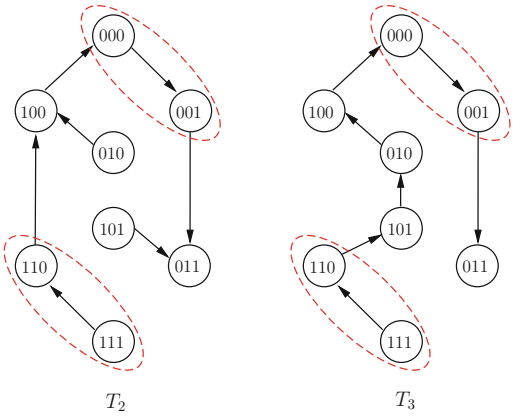

Fig. 5. Spanning trees of G_3 rooted at 011.

In a directed spanning tree T a node x is known as an ancestor of a node v if there exists a path from x to v in T. For example, in Fig. 4 the ancestors of the node 100 are 101, 010 and the merged vertex pair $111 - 110$. Now, we proceed to prove our main result.

Theorem 1. *Given a spanning tree T of G_n rooted at r, any other spanning tree T' of G_n having the same root can be obtained from T by a sequence of leaf nodes edge switching.*

Proof. Let the number of nodes in T whose outgoing edges are same as that in T' be κ. Now, consider a node $v \neq r$ in T whose outgoing edge is different from that in T' and all of whose ancestor nodes have the same out-edges as in T'. Now, we apply post-order depth first traversal on the sub-tree of T rooted at v and switch the nodes in the order in which they occur in this traversal. Clearly, this is a sequence of leaf node edge switchings culminating in the switching of the outgoing edge of v. Once the node v is switched, we switch all the other nodes that we had switched before v in the reverse order. We now have a new graph wherein the number of nodes in T whose outgoing edges are same as that in T' is $\kappa + 1$. We keep repeating this process till the outgoing edges of all vertices in T become same as T'. Thus, we transform the spanning tree T into T'.

Given a de Bruijn sequence we can construct the corresponding spanning tree and tree array. Now, by randomly performing a sequence of leaf node switches we can get a random spanning tree of G_n and therefore a random de Bruijn sequence.

Example 5. Consider a spanning tree, T_1, of the 3-rd order de Bruijn graph shown in Fig. 6(a). Let T_1' be any other spanning tree of G_3 rooted at the same vertex as shown in Fig. 6(g). Here $\kappa = 5$ and except 101 and 100 all other nodes in T_1 have the same outgoing edges as in T_1'. Now, 101 is a leaf node and switching its edge gives us a new spanning tree shown in Fig. 6(b). This spanning tree has 6 nodes whose outgoing edges are same as in T_1'. Now, the vertex 100 is not a leaf node. By applying depth first traversal algorithm on the sub-graph rooted at node 100 we get the list $\{111-110, 010, 100\}$. We now switch the nodes in the order $111-110, 010$ and 100. This gives us the spanning tree shown in Fig. 6(e). We now again switch the nodes 010 and $111-110$. This gives us the spanning tree T_1'. These switching operations are depicted in Fig. 6(a–g). The corresponding tree arrays and 4-th order de Bruijn sequences of the spanning trees T_1 and T_1' are tabulated in Table 1.

Remark 1. A sequence of Leaf node transformations can be represented by a string of vertices whose outgoing edges are switched. It is interesting to note that the set of such strings form a group under string concatenation where the zero element is the empty string and the inverse of any string is a string where the same nodes occur in the reverse order.

These results can be easily extended for k-ary de Bruijn graphs. In a k-ary de Bruijn graph of order n, $\mathcal{G}_n(k)$, every vertex $v \in \mathcal{V}$ will have k incoming edges and k outgoing edges. Let $\mathcal{T}$ be a spanning tree of $\mathcal{G}_n$. Given a vertex $v \in \mathcal{V} \backslash \{r\}$, the row L_v in the tree array has $indeg(v) = k$ elements. The last element of L_v will be the unique outgoing edge of v that occurs in $\mathcal{T}$. The remaining $k-1$ elements in $\mathcal{G}_n(k)$ can be arranged in any order. Similarly, in case of the root

Table 1. Tree arrays and de Bruijn sequences of T_1 and T_1'

	Spanning tree (T_1)	Spanning tree (T_1')
Tree array	$l_{000} = \{(0000), (0001)\}$ $l_{001} = \{(0010), (0011)\}$ $l_{010} = \{(0101), (0100)\}$ $l_{011} = \{(0110), \quad \Omega\ \}$ $l_{100} = \{(1001), (1000)\}$ $l_{101} = \{(1010), (1011)\}$ $l_{110} = \{(1101), (1100)\}$ $l_{111} = \{(1111), (1110)\}$	$l_{000} = \{(0000), (0001)\}$ $l_{001} = \{(0010), (0011)\}$ $l_{010} = \{(0101), (0100)\}$ $l_{011} = \{(0110), \quad \Omega\ \}$ $l_{100} = \{(1000), (1001)\}$ $l_{101} = \{(1010), (1011)\}$ $l_{110} = \{(1101), (1100)\}$ $l_{111} = \{(1111), (1110)\}$
de Bruijn seq.	0111101011001000	0111101011000010

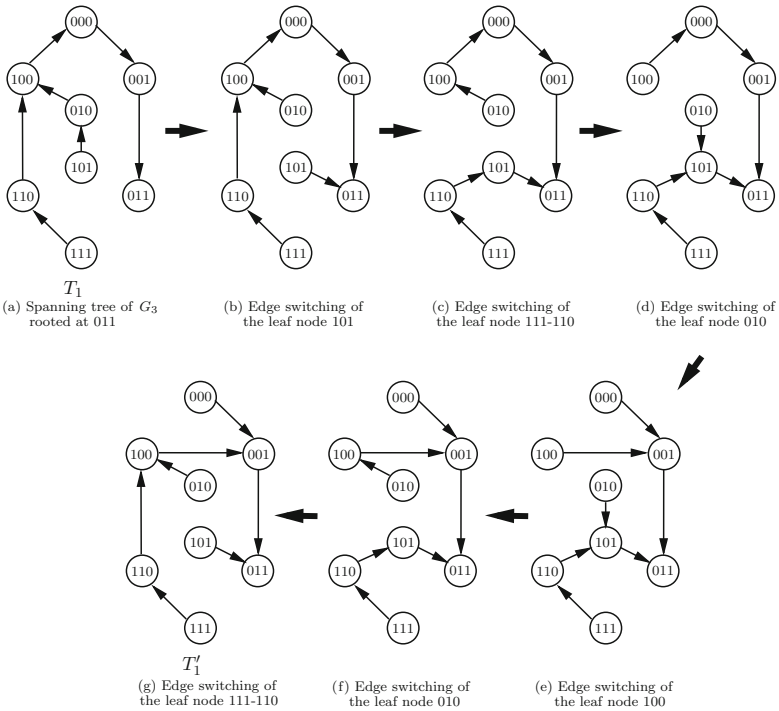

Fig. 6. Switching of leaf node edges in a spanning tree of G_3.

vertex r, we consider Ω as the last entry of L_r and the other $k-1$ entries of L_r can be chosen from any of its outgoing edges in $\mathcal{G}_n(k)$. From this tree array we can construct an Eulerian cycle of a k-ary de Bruijn graph of order n using the method given in Sect. 2. Observe that the proof of Theorem 1 would also be valid in this case.

4 Conclusion and Future Work

In this paper, we have shown that one can go from a given de Bruijn sequence to another by a sequence of leaf node edge switchings. By randomly choosing the sequence of leaf node edge switches we can generate a random spanning tree of G_n and therefore a random de Bruijn sequence.

Since every n-th order de Bruijn sequence can also be generated from a single n-th order de Bruijn sequence by using a sequence of cross joining operations, it would be interesting to find a correspondence between a sequence of leaf node switchings given in this paper and a sequence of cross joining operations. Further, it can be investigated if this method can be efficiently translated into a nonlinear feedback function for FSRs.

Acknowledgment. The authors are grateful to Prof. Harish K. Pillai, Department of Electrical Engineering, Indian Institute of Technology Bombay, without whom this work would never have been possible.

References

1. Annexstein, F.S.: Generating de Bruijn sequences: an efficient implementation. IEEE Trans. Comput. **46**(2), 198–200 (1997)
2. Bidkhori, H., Kishore, S.: A bijective proof of a theorem of Knuth. Comb. Probab. Comput. **20**(1), 11–25 (2011)
3. De Bruijn, N.G.: A Combinatorial Problem. Koninklijke Nederlandsche Akademie Van Wetenschappen, vol. 49, no. 6, pp. 758–764, June 1946
4. Etzion, T., Lempel, A.: Algorithms for the generation of full-length shift-register sequences. IEEE Trans. Inf. Theory **30**(3), 480–484 (1984)
5. Fredricksen, H.: A survey of full length nonlinear shift register cycle algorithms. SIAM Rev. **24**(2), 195–221 (1982)
6. Fredricksen, H.M.: Disjoint cycles from the de Bruijn graph. Technical report, DTIC Document (1968)
7. Golomb, S.W., et al.: Shift Register Sequences. Aegean Park Press, Laguna Hills (1982)
8. Jansen, C.J.A.: Investigations on nonlinear streamcipher systems: construction and evaluation methods. Ph.D. thesis, Technische Universiteit Delft (1989)
9. Lempel, A.: On a homomorphism of the de Bruijn graph and its applications to the design of feedback shift registers. IEEE Trans. Comput. **100**(12), 1204–1209 (1970)
10. Mykkeltveit, J., Szmidt, J.: On cross joining de Bruijn sequences. In: Topics in Finite Fields, vol. 632, pp. 335–346 (2015)
11. Schneier, B.: Applied Cryptography: Protocols, Algorithms, and Source Code in C. Wiley, New York (2007)
12. Spinsante, S., Andrenacci, S., Gambi, E.: De Bruijn sequences for spread spectrum applications: analysis and results. In: 18th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2010, pp. 365–369, September 2010