Applications De Bruijn Graphs of Hamiltonian and Eulerian Cycles to Fault Tolerant Networks

Mochamad Suyudi*
Department of Mathematics
Faculty of Mathematics and Natural Sciences
Universitas Padjadjaran,
Bandung - Indonesia
*Email: moch.suyudi@gmail.com

Roslan Bin Hasni @Abdullah
School of Informatics and Applied Mathematics
Universiti Malaysia Terengganu
Terengganu - Malaysia
E-mail: hroslan@umt.edu.my

Sudrajat Supian
Department of Mathematics
Faculty of Mathematics and Natural Sciences
Universitas Padjadjaran,
Bandung - Indonesia
Email: Adjat03@yahoo.com

Asep Kuswandi Supriatna
Department of Mathematics
Faculty of Mathematics and Natural Sciences
Universitas Padjadjaran,
Bandung - Indonesia
Email: ak_supriatna@unpad.ac.id

***Abstract*— The goal of this expository paper is to introduce De Bruijn graphs and discuss their applications to fault tolerant networks. We will begin by examining N.G. de Bruijn's original paper and the proof of his claim that there are exactly $2^{2^{n-1}-n}$ De Bruijn cycles in the binary De Bruijn graph B(2, *n*). In order to study fault tolerance we explore the properties of Hamiltonian and Eulerian cycles that occur on De Bruijn graphs and the type of redundancy that occurs as a result. Lastly, in this paper we seek to provide some guidance into further research on De Bruijn graphs and their potential applications to other areas.**

Keywords*— *De Bruijn graphs; Hamiltonian; Eulerian; cycle

I. INTRODUCTION

De Bruijn graphs are an interesting class of graphs that have applications in a variety of different areas. A De Bruijn graph is a directed graph with d^n nodes labeled by n-tuples over a d-character alphabet (denoted by juxtaposition). The edges are defined to be ordered pairs of the form $((a_1 \ldots a_n), (a_2 \ldots a_n a_{n+1}))$ where a_{n+1} is any character in the alphabet. We will denote the De Bruijn graph $B(d, n)$. In Figure 1 below we see two examples of these graphs. Properties discussed in this paper show that they are a natural choice for constructing interconnection networks, which are networks for which fault tolerance is important. Fault tolerance is a networks ability to maintain communication even if certain communication paths are destroyed. We will study the ability of networks to handle the failure of both nodes and edges.

In this paper we will first present some of the properties of De Bruijn graphs, including ways in which they can be applied both in theory and in practice. Before we examine applications of De Bruijn graphs, we will discuss the original theorem which De Bruijn proved which is the beginning of the study of the structure of De Bruijn Graphs. De Bruijn proved that a De Bruijn graph exhibits a predictable amount of Hamiltonian cycles. His proof centers around the fact that in the binary De Bruijn graph with 2^n nodes there will exist $2^{2^{n-1}-n}$ different Hamiltonian cycles. This trait discovered by De Bruijn is useful for applications of De Bruijn graphs to fault tolerant networks.

De Bruijn graphs are natural choices for a network's connection layout. When examining a network, i.e. a structure with objects connected by communication paths, there are a few desirable traits which make De Bruijn graphs a logical choice for investigation. Esfahanian and Hakimi [4] discuss these traits; they point out that we want the number of connections from each

processor to be relatively small, but we also want there to be short paths between any two processors. We will examine this problem in terms of graphs which use nodes to represent processors and edges to represent the communication paths between these processors.

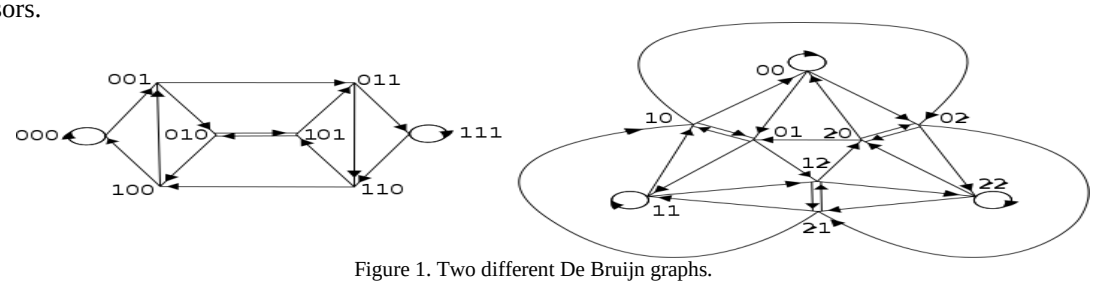


Figure 1. Two different De Bruijn graphs.

We will see that the De Bruijn graphs make excellent choices for a network's structure since they:
(1) have a relatively large number of nodes
(2) have few connections at each node and
(3) still maintain short paths between nodes.
These three traits make De Bruijn graphs excellent graphs for further investigation.
The remainder of this paper is organized as follows. In Section 2 we introduce the De Bruijn graph and other relevant terms. In Section 3 we will examine N.G. de Bruijn's original proof that there are $2^{2^{n-1}-n}$ Hamiltonian cycles in the De Bruijn graph B(2, *n*). Following that, in Section 4 we will focus on fault tolerance. This section has been divided up into two parts. The first is aimed at understanding the type of connectivity that we have if there are node failures and the second is focused on edge failure.

II. PRELIMINARIES

A. *Terminology and Definitions.*

In order to define De Bruijn graphs, we must first begin with the definition of graph.

Definition 1. A graph is a an ordered pair $G = (V,E)$ where V is a set of vertices called nodes, together with a set E of edges which are pairs of nodes. Unless otherwise stated,the graphs mentioned will be directed graphs, i.e. graphs whose edges are an ordered pair of nodes.

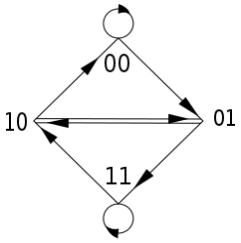
Definition 2. A subgraph, $G' = (V', E')$, of a graph $G = (V, E)$, is a graph on a subset V' of V with the property that every edge $e \in E$ with endpoints in V' is also in E'.

Definition 3. A $(b_1 \ldots b_n)$ is called a *successor* of node $(a_1 \ldots a_n)$ if there is an edge from $(a_1 \ldots a_n)$ to $(b_1 \ldots b_n)$. Likewise, $(a_1 \ldots a_n)$ is said to be a *predecessor* of $(b_1 \ldots b_n)$. We will say two nodes are adjacent nodes if there is an edge between them and that two edges are adjacent edges if they share a common node.

Definition 4. A De Bruijn graph is a directed graph with d^n nodes labeled by n-tuples over a d-character alphabet (denoted by juxtaposition). The edges are defined to be ordered pairs of the form $((a_1 \ldots a_n), (a_2 \ldots a_n\, a_{n+1}))$ where a_{n+1} is any character in the alphabet. We will denote the De Bruijn graph $B(d, n)$.
Example. The De Bruijn graph $B(3,4)$ has the node (1022) (a 4-tuple with alphabet {0, 1, 2, 3}) which has successors (0220), (0221), (0222) and (0223).
Example. The De Bruijn graph $B(2, 2)$ is given below in Figure 2.

Definition 5. In [4], Esfahanian and Hakimi discuss a modified version of a De Bruijn graph called an undirected De Bruijn graph, denoted $UB(d, n)$. An undirected De Bruijn graph is a De Bruijn graph modified so that:
1) All edges which are self loops are removed.
2) If $\overrightarrow{ab}$ is an edge in $B(d, n)$, then the undirected edge $\overline{ab}$ is an edge in $UB(d, n)$.
3) If $\overrightarrow{ab}$ and $\overrightarrow{ba}$ are both edges in $UB(d, n)$ then there is only the single undirected edge $\overline{ab}$ in $UB(d, n)$.

In order to understand applications to fault tolerant networks, we need to understand some of the terms used to describe the structure of graphs. In particular, we need to define the notions of both in-degree and out-degree, and also Hamiltonian and Eulerian cycles.



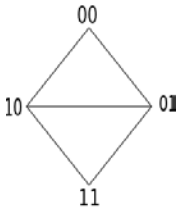


Figure 2. The De Bruijn graph with alphabet {0, 1} whose nodes are 2-tuples. Figure 3. The undirected De Bruijn graph *UB*(2, 2).

Definition 6. The in-degree of a node in a directed graph is the number of edges that end at that node and similarly, the out-degree of a node will be the number of edges that start at that node.
Upon examining Definition 4, we outline a few basic properties of De Bruijn graphs:
1) The number of nodes in $B(d, n)$ is dn since each n-tuple is a node.
2) The number of edges in $B(d, n)$ is d_{n+1} since there are dn nodes and out-degree of d on each node.
3) Every node in $B(d, n)$ has out-degree d since every successor of $(a_1 \ldots a_n)$ has the form $(a_2\ a_3 \ldots a_n b)$ and there are d choices for b.
4) Every node in $B(d, n)$ has in-degree d since if we examine the predecessors of $(a_1 \ldots a_n)$ we see that they will be those which look like $(ba_1 \ldots a_{n-1})$ with any of the d choices for b.
5) Since the edges are defined explicitly, we see that the graph $B(d, n)$ is unique for fixed d and n.

Definition 7. A path is a sequence of edges for which each edge begins where the previous edge in the sequence ended. We will denote a path

$$(a_1 \ldots a_n)\,(a_2 \ldots a_{n+1}) \ldots (a_i \ldots a_{n+i-1})$$

of length i instead by $a_1\ a_2\ a_3 \ldots a_{n+i-1}$. A cycle is a path whose start node and end node are the same. We will denote the corresponding cycle on $B(d,n)$ whose path is $a_1 \ldots a_{n+i-1}\ a_1\ a_2 \ldots a_n$ by $[a_1 \ldots a_{n+i-1}]$.
Example. In Figure 5 we see there is a path 000101 which we can see comes from the long form of the path (000)(001)(010)(101) of $B(2, 3)$ (using underlines to highlight which characters are in our short form).

Definition 8. A graph is connected if for every pair of nodes, a and b, there exists a path from a to b and a path from b to a

Definition 9. A subgraph of a graph G is called a component of G if
1) Any two nodes in the subgraph are connected.
2) Any node in the graph which is connected to a node in the subgraph is also in the subgraph.

Example. In Figure 4, the cycle [000101] is the path 000101000, written in long form as (000) (001) (010) (101) (010) (100) (000).

Definition 10. A Hamiltonian cycle of a graph G is a cycle of G which visits every node exactly once. An Eulerian cycle of G is a cycle of G which traverses every edge exactly once.
Example. The highlighted cycle in Figure 4 is the Hamiltonian cycle [11010001] which is described by starting at the node (110). We also see that [1111000010011010] is an Eulerian cycle on B(2; 3), starting at the node (111).

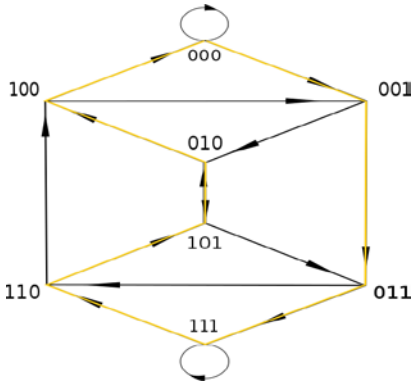


Figure 4. The graph above is the De Bruijn graph $B(2, 3)$ with alphabet {0,1} and nodes which are 3-tuples. The highlighted cycle is a Hamiltonian cycle on the graph.

Definition 11. We will call the sequence of characters $[a_1, a_2, \ldots, a_{m+n}]$ a De Bruijn sequence when $[a_1 \ldots a_{n+m}]$ is a Hamiltonian cycle. Since there are d^n nodes in a De Bruijn graph, then a De Bruijn sequence will necessarily be d^n characters long.

Without use of the geometric representation, a De Bruijn sequence can also be described as a dn character cyclic string in which each possible n-tuple from d characters occurs exactly once. We will say that two sequences $[a_1 \ldots a_{d^n}]$ and $[b_1 \ldots b_{d^n}]$ are equal if there exists some c such that for all $i \in 1, \ldots, d^n$ we have that $a_i = b_{i+c} \pmod{d^n}$. This is equivalent to saying that the sequences correspond to the same cycle on the De Bruijn graph.

Example. Using the highlighted Hamiltonian cycle of $B(2, 3)$ in Figure 5 above, we see that [0, 0, 0, 1, 1, 1, 0, 1] is a De Bruijn sequence.

B. *The Doubling of a De Bruijn Graph.*

An additional interesting property of De Bruijn graphs is that we can expand the De Bruijn graph $B(d, n)$ to the graph $B(d, n + 1)$ with the following procedure.

Procedure 1. Using our standard notation as defined above we can create the De Bruijn graph $B(d, n + 1)$ by the following steps:

1) Construct nodes in $B(d, n + 1)$ from $B(d, n)$ by allowing the edges of $B(d, n)$ to correspond to nodes of $B(d, n+1)$. Specifically, the adjacent nodes $(a_1 \ldots a_n)$ and $(a_2 \ldots a_{n+1})$ form the edge $(a_1 \ldots a_n)(a_2 \ldots a_{n+1})$ in $B(d, n)$ and correspond to node $(a_1 \ldots a_{n+1})$ in $B(d, n + 1)$.
   Example. In the doubling from $B(3, 3)$ to $B(3, 4)$ the edge (201)(011) in $B(3, 3)$ gives the node (2011) in $B(3, 4)$

2) Construct one edge in $B(d, n + 1)$ for each pair of adjacent edges in $B(d, n)$. Here we will say that edges are adjacent if the ending node of one edge is the start node of the next edge. The node $(a_1 \ldots a_{n+1})$ in $B(d, n + 1)$ corresponds to the edge $(a_1 \ldots a_n)(a_2 \ldots a_{n+1})$ in $B(d, n)$, the edges leaving $(a_1 \ldots a_{n+1})$ are of the form $(a_1 \ldots a_{n+1})(a_2 \ldots a_{n+2})$. These d edges correspond to the d adjacent edges $(a_1 \ldots a_n)(a_2 \ldots a_{n+1})$ and $(a_2 \ldots a_{n+1})(a_3 \ldots a_{n+2})$ in $B(d, n + 1)$.
   Example. In the doubling from $B(3, 3)$ to $B(3, 4)$ the edge (1222)(2220) in $B(3, 4)$ came from the adjacent edges (122)(222) and (222)(220) in $B(3, 3)$. See Figure 5 to see an example of the doubling process from $B(2, 2)$ to $B(2, 3)$.

Remark. By Definition 4 and Procedure 1, the number of nodes in $B(d, n + 1)$ (and thus the number of edges in $B(d, n)$) ought to be d^{n+1}. The number of edges in $B(d, n)$ is the number of nodes, d^n, multiplied by the out-degree, d, resulting in d^{n+1} nodes in $B(d, n + 1)$ as desired. Similarly, the number of pairs of adjacent edges in $B(d, n)$ is the product of the number of nodes, the in-degree and the out-degree giving $d^n \cdot d \cdot d = d^{n+2}$ which is the number of edges that are in $B(d, n + 1)$.

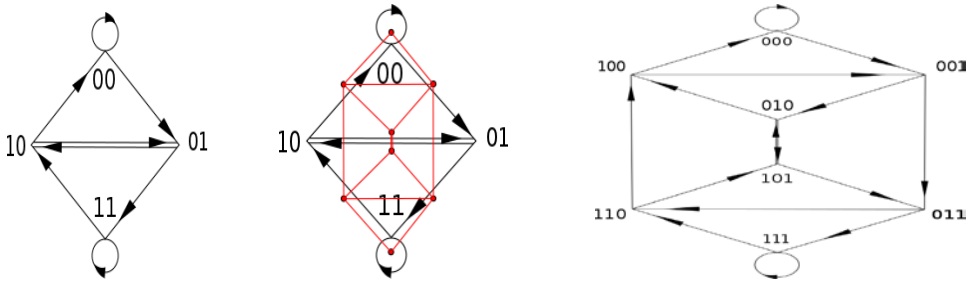


Figure 5. The graph $B(2, 2)$ (left) getting doubled with intermediate step (middle) to $B(2, 3)$ (right).

III. A Combinatorial Problem

De Bruijn's original paper [2] arose from the investigation of a problem concerning the number of unique De Bruijn sequences with respect to n on the De Bruijn graph $B(2, n)$. De Bruijn was presented [2] with the conjecture (see Theorem 1 below) and in his paper he proves the conjecture using an inductive argument on a type of graph which is a more general version of the De Bruijn graph.

Theorem 1. The number of De Bruijn sequences contained in $B(2, n)$ is $2^{2^{n-1}-n}$ for all $n \geq 1$.

Our goal in this section is to prove the number of De Bruijn sequences contained in $B(2, n)$ is $2^{2^{n-1}-n}$. However, we must first establish a few important facts.

Definition 12. A T-net of order m is a graph with m nodes and $2m$ directed edges with the property that both the in-degree and out-degree of each node is 2.

For example, notice that $B(2, n)$ is a T-net of order 2^n since it has 2^n nodes and 2^{n+1} edges. Let N be a T-net and denote the number of Eulerian cycles of N by $|N|$ (If the T-net is not connected then $|N| = 0$). We can now construct a unique doubled T-net by using Procedure 1. This process creates a T-net N^* of order $2m$.

Theorem 2. If N is a T-net of order m and N^* is its doubled T-net, then $|N^*| = 2^{m-1}|N|$.

Proof. We first consider two important cases:

Case 1: N is disconnected.

If N is not connected then there will be two nodes with no path between them. If there was a path in N^* between every two nodes, then this would imply that there is a path in N from every edge (these correspond to a node in N^*) to every other edge. Thus using this "edge path" we could construct a path of nodes between every pair of nodes in N. Therefore if N^* is connected then N is connected and so the contrapositive is true as well. Thus, if N is disconnected, N^* is disconnected. This implies that if N is disconnected, then $|N| = |N| = 0$ and Theorem 2 holds in this case.

Case 2: N is connected and each node has a loop edge.

There is only one such connected T-net for each m since after the m loop edges are counted, the only way to ensure connectivity is to have the remaining edges form a cycle over all m nodes.

Looking at Figure 6 and observing that for any m we will be looking at a regular m-gon with loops at every edge, we see that for each such T-net, a path that visits each edge will never have a choice in which edge is taken, thus $|N| = 1$ in this case. Next, we will show that $|N^*| = 2^{m-1}$. For $m > 1$ we see that since N has m edges which are self loops, then there will be m nodes in N^* which have self loops. This is a direct result from the doubling procedure. We also see that there will be m nodes in N^* which do not have self loops, these will correspond to the edges in N which are not self looping edges.

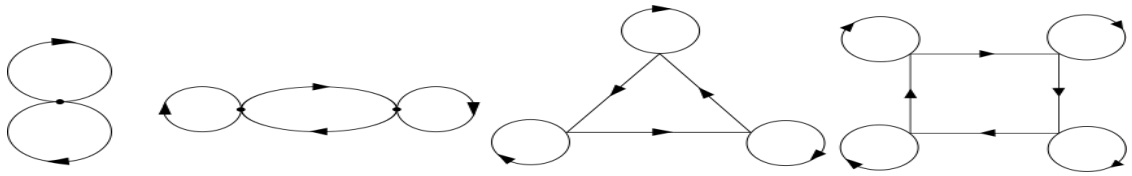


Figure 6. This illustrates Case 2 for $m = 1, 2, 3$ and 4

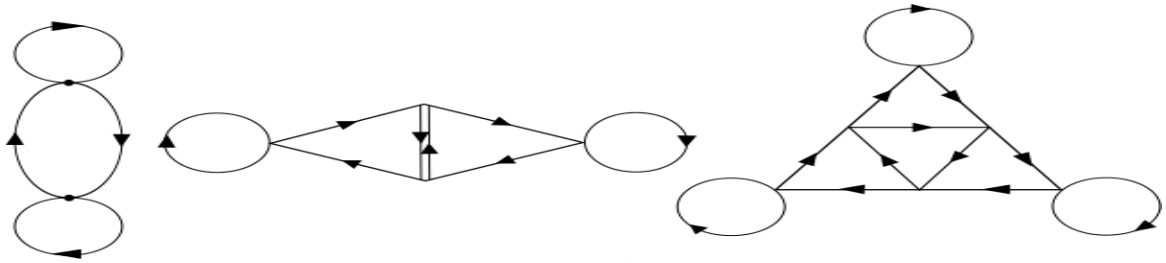


Figure 7. The corresponding N^* graphs for $m = 1$, $m = 2$ and $m = 3$.

We see they will not have these self loops since these edges are not adjacent to themselves since they have different start and end nodes. We observe that any Eularian cycle on N^* will only vary on their choice of edge at the m nodes which do not have self loops. Since each of these m nodes has degree 2 then there will be $2m$ Eularian cycles, one for each set of edge choice at the m nodes. However we have double counted these cycles since any two cycles which take the opposite edge at these m nodes is really the same cycle written in a different order. Thus there are 2^{m-1} Eularian cycles on N^*. Figure 7 shows this case for the first few m's.

Thus Theorem 2 holds in this case as well.

We now proceed by inducting on the order of the T-net, m. When $m = 1$ notice that N is a connected graph and each node has a loop edge, thus it falls into the second case above, so Theorem 2 holds when $m = 1$. We make note that since $m = 1$, we have that $|N| = |N^*|$.

Now assume inductively that for any T-net, *N*, of order 1 to *m* - 1 for *m* - 1 ≥ 1 that $|N^*| = 2^{m-1}|N|$.
Consider a T-net, *N*, of order *m*. If *N* is disconnected or if each node has a loop edge, then we are done. If we are not in either of these two cases, we know that there is some node which does not have a loop edge. Call this node *a*. Since *N* has in-degree and out-degree 2, let the two edges which end at *a* be called ie_1 and ie_2 and let the two edges which start at *a* be called oe_1 and oe_2. We will also refer to the associated predecessors and successors of *a* by *A*, *B*, *C* and *D* as in Figure 8.

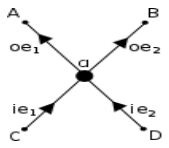


Figure 8. The graph of the node *a* before removal.

We now create a new graph N_1 from *N* by removing the node *a* and all of the edges to and from it, but add an edge from *C* to *A* and another from *D* to *B* as in Figure 9.

Similarly, construct N_2 by removing the node *a* and all of the edges to and from it, but add an edge from *C* to *B* and another from *D* to *A*, also in Figure 10.

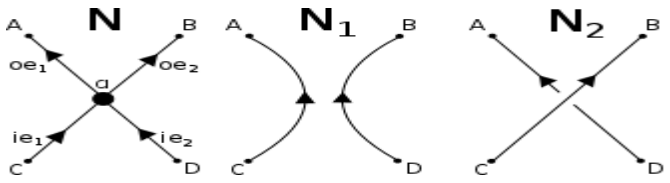


Figure 9. The partial graphs of *N*, N_1 and N_2.

Since our goal is to count Eulerian cycles on the graph, we should note that any Eulerian cycle on our T-net *N* corresponds to an Eulerian cycle on N_1 or on N_2, but not both. This gives us that

(1) $$|N| = |N_1| + |N_2|$$

Using Procedure 1 we double the graphs N_1 and N_2 to create N_1^* and N_2^* respectively. Recall our goal is to show that $|N^*| = 2^{m-1}|N|$. We have by induction that,

$$\begin{aligned} 2^{m-1}|N| &= 2^{m-1}(|N_1| + |N_2|) \\ &= 2.2^{m-2}|N_1| + 2.2^{m-2}|N_2| \\ &= 2.|N_1^*| + 2.|N_2^*| \end{aligned}$$

since both N_1 and N_2 have $m-1$ nodes.
All that remains is to show that,

(2) $$|N^*| = 2.|N_1^*| + 2.|N_2^*|$$

Procedure 1 constructs N^* from *N* and Figures 10 and 11 show parts of these two graphs. The four edges ie_1, ie_2, oe_1 and oe_2 in *N* are replaced by nodes I_1, I_2, O_1 and O_2 respectively in N^*. There are two edges entering the new nodes I_1 and I_2 and also two edges exiting the nodes O_1 and O_2 by construction. Additionally, in N^* there will be edges I_1O_1, I_1O_2, I_2O_1 and I_2O_2.

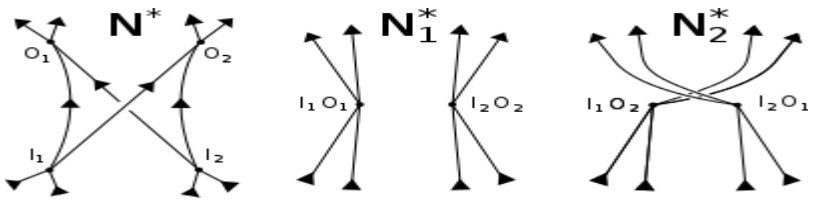


Figure 10. The partial graphs of N^*, N_1^* and N_2^*

We can construct N_1^* and N_2^* from $N_1$and N_2 in a similar manner and once again it will be helpful to reference Figures 9 and 10. The process of combining the edges ie_1 and oe_1 and the edges ie_2 and oe_2 while creating N_1 from *N* corresponds to

merging the two nodes I_1 and O_1 and the two nodes I_2 and O_2 in N_1^* . The construction of N_2^* is similar only instead we combine the edges ie1 and oe_2 and the edges ie_2 and oe_1 to create N_2 from N and thus merge the nodes I_1 and O_2 and the nodes I_2 and O_1. Again, the construction of these graphs can be seen in Figure 10.

Notice now that an Eulerian cycle on N^* is defined by four paths which have no common edges, each of which begins at either O_1 or O_2 and ends at I_1 or I_2. These paths are edge disjoint because an Eulerian cycle visits each edge exactly once and these paths are all part of the same Eularian cycle on N. We are assured that each edge except the four edges I_1O_1, I_1O_2, I_2O_1 and I_2O_2 will be included in exactly one of these paths since an Eulerian cycle traverses every edge. We will be counting the number of Eulerian cycles on N^*, $N_1^*$and N_2^* which contain all four of those paths but differ in their choice of connections of the nodes I_1, I_2, O_1 and O_2.

We will adopt N.G. de Bruijn's notation and construct three more graphs N^{**}, $N_1^{**}$and N_2^{**} which represent these four paths (discussed above) by single edges instead. Notice that these new graphs are not the doubled versions of N^*, $N_1^*$and N_2^*. The additional graphs here will help count the number Eularian cycles in N^* as we proceed by induction. Also, we will now denote$|N^*|$, $|N_1^*|$and $|N_2^*|$as n, n_1 and n_2.

In order to show (2) with the new notation, we need to show

(3) $$n = 2n_1 + 2n_2.$$

We consider the following cases:

(1) Both paths in N^* starting from O_1 end at diferent nodes.

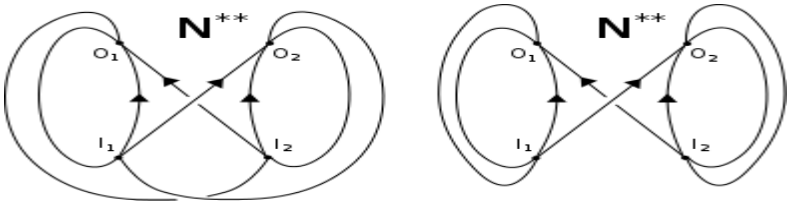


Figure 11. Case 1 (left) and Case 2 (right)

As is indicated by the Figure 11 we see that the four paths are: the path from O_1 to I_1, from O_1 to I_2, from O_2 to I_1 and from O_2 to I_2.

Upon observing the graph N^{**} in Figure 11 we see that there are four Eulerian cycles. Examining N_1^{**} and N_2^{**} in Figure 12, which are the same graph, we seethat there is exactly one Eulerian cycle. Thus, in this case we establish our claim, since

$$n = 4 = 2 \,.\, 1 + 2 \,.\, 1 = 2n_1 + 2n_2:$$

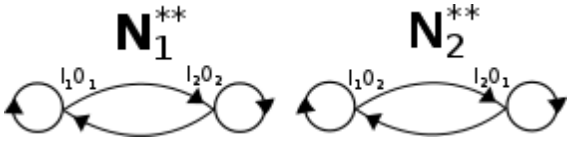


Figure 12. Case 1: N_1^{**} (left) and N_2^{**} (right)

(2) Both paths in N^* starting from O_1 end at the same node. Notice that it doesn't matter if the two paths end at I_1 or I_2 as the resulting graphs $N_1^{**}$and N_2^{**} will be the same except they will change names. Here we assume that the two paths starting at O_1 end at I_1.

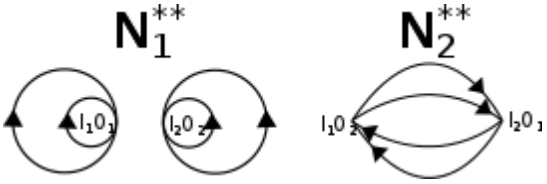


Figure 13. Case 2: N_1^{**} (left) and N_2^{**} (right)

In this case, we notice observe from Figure 13 that since N_1^{**} is disconnected, then $n_1 = 0$. We see number of Euler cycles in N_2^{**} is two and so again we establish the equality that $n = 4 = 2 \cdot 0 + 2 \cdot 2 = 2n_1 + 2n_2$. This completes our claim and we see that the equality is established.
With the truth of this claim established for any set of four paths, we see that (2) is now also clear, thus establishing Theorem 2. We can now establish the truth of Theorem 1 with a few observations.
(1) A De Bruijn graph $B(2, n)$ is a T-net of order 2^n
(2) There is only one Hamiltonian cycle on B(2, 1), namely [0011].
(3) The doubling procedure on $B(2, n)$ gives $B(2, n + 1)$.
(4) Since an Eularian cycle visits every edge and a Hamiltonian cycle visits every node, an Eularian cycle on N represents a Hamiltonian cycle on N^*, the doubled graph of N.

Proof. Let $H(N)$ denote the number of Hamiltonian cycles on N. We prove Theorem 1 by induction on n. Let $n = 1$. By item (2) above,

$$H(B(2, 1)) = 1 = 2^0 = 2^{2^{1-1}-1}.$$

Now to proceed inductively, let $n \geq 1$ and assume that $H(B(2, n)) = 2^{2^{n-1}-n}$. Then,

$$\begin{aligned} H(B(2, n + 1)) &= |B(2, n)| \\ &= |B(2, n-1)^*| \\ &= 2^{2^{n-1}-1}|B(2, n-1)| \\ &= 2^{2^{n-1}-1}H\big(B(2, n)\big) \\ &= 2^{2^{n-1}-1}2^{2^{n-1}-n} \\ &= 2^{2^{n-1}-1+2^{n-1}-n} \\ &= 2^{2^n-n-1} \\ &= 2^{2^{(n+1)-1}-(n+1)} \end{aligned}$$

Thus our induction holds and the proof of Theorem 1 is complete.

IV. Fault Tolerance

This section is motivated by applications of De Bruijn graphs, and while we will be examining the mathematical properties relevant to these applications, it is worthwhile to understand some motivation. De Bruijn graphs have a few properties which make them natural choices for the structure of a communication network. Esfahanian and Hakimi [4] describe applications of De Bruijn graphs to multiprocessor systems and Collins [1] describes how the De Bruijn graph gives the structure for a fully parallel Viterbi decoder. De Bruijn graphs have been natural choices for these types of network topologies for a few important reasons as follows.

We mentioned in the introduction the first two following properties and now with our focus on fault tolerance we _nd this third property to be valuable as well. For a De Bruijn graph with dn nodes there is a
(1) small in-degree and out-degree, which means fewer connections to each node (processor)
(2) small distance between nodes (between any two nodes there does not need to exist more than n edges)
(3) large number of paths between nodes.

These properties are the motivation for investigating applications of De Bruijn graphs, and furthermore, since we are interested in the graphs for communication network applications the following questions will be important to study.
(1) What sort of connectivity can we expect to see if a given node is removed?
(2) What sort of connectivity can we expect to see if an edge is removed?
(3) How many edges can be removed while still maintaining communication paths?

These are the main questions that inspire our mathematical investigation of De Bruijn graphs. As we study the application of De Bruijn graphs we will make these questions more precise.

We will first investigate the types of graph structures that result from node failures. A node failure is represented by a De Bruijn graph which has that node removed as well as all of the edges which begin or end at that particular node.

To investigate these structures, we will prove a few lemmas.

Lemma 1. For any node, a of a De Bruijn graph $B(d, n)$, all of the predecessors of a have the same set of successors.
Proof. Let a be the node $(a_1 \ldots a_n)$, then any predecessor of a looks like$(x a_1 \cdots a_{n-1})$ for x in our alphabet, and then the successor of any node of this form is$(a_1 \cdots a_{n-1} y)$, with y in our alphabet.

Example. In *B*(3, 3) if we look at the node (201) then the set of predecessors will be the set {(020), (120), (220)} and the set of successors for each of those nodes is the same, they are {(200); (201); (202)}.

Lemma 2. Let *a* be a node in *B*(*d*, *n*) of the form (*a* ⋯ *a*), then if a node *b* is neither a successor nor a predecessor of *a*, then *a* and *b* have no common successors and they have no common predecessors (however, a node which is a successor of *a* could be a predecessor of *b* and vice versa).

Proof. Any successor of *a* is of the form (*a* ⋯ *a x*) where *x* is a character in our alphabet. If *a* and *b* share the successor (*a* ⋯ *ax*) then *b* is a predecessor of this node. Any predecessor of (*a* ⋯ *a x*) is of the form (y*a* ⋯ *a*) for *y* also a character in our alphabet. Since *b* is a predecessor of (*a* ⋯ *a x*), then *b* = (y*a* ⋯ *a*) for some choice of *y*. However, we see by construction of *B*(*d*, *n*) that *b* = (y*a* ⋯ *a*) is a predecessor of *a*. Thus if *a* and *b* share a successor, then *b* is a predecessor of *a*.

The proof that if *a* and *b* share a predecessor then *b* is a successor of *a* is similar.

Observation 1. Most of the nodes of a De Bruijn graph have disjoint sets of successors and predecessors. The exception to this is if for a given node $(a_1 \ldots a_n)$ it is the case that for some choice of *x* and *y*, $(xa_1a_2 \cdots a_{n-1}) = (a_2 \cdots a_n y)$. There are three cases where this occurs:

(1) When we examine the node (*a* ⋯ *a*).
Ex) (111) in *B*(3, 3) has itself as both a predecessor and a successor.

(2) When n is even, then when we examine the node (*abab* ⋯ *ab*).
Ex) (2020) in *B*(3, 4) has the node (0202) as both a predecessor and a successor.

(3) When n is odd, then when we examine the node (*abab* ⋯ *a*).
Ex) (404) in *B*(5, 3) has the node (040) as both a predecessor and a successor.

Definition 13. Faulty nodes in a graph will be those which are removed from the graph. We will denote the set of faulty nodes in a graph to be the set *F*. We will refer to the De Bruijn graph *B*(*d*, *n*) with these nodes removed as $B(d, n)_F$.

Remark. When removing *F* from a De Bruijn graph, we will find that this removal of the individual nodes might result in a graph on which there exists no Hamiltonian cycles. We see that Figure 14 is an example of this since after removing the points (001), (010) and (100) we no longer even have a connected graph.

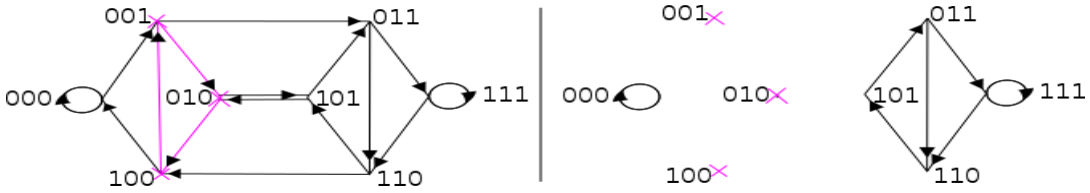


Figure 14. *B*(2, 3) (left) with the removal of the points (100), (010) and (001) (right).

It is with this motivation that we introduce this next definition and will later examine a procedure that constructs a cycle on $B(d, n)_F$ for a given *F*, which covers the maximum number of nodes.

Definition 14. A necklace of a node, denoted $Neck\big((a_1 \cdots a_n)\big)$, is the set of nodes which are in the cycle $[a_1 \cdots a_{n+n}]$

Example. If we examine *B*(3, 5) then *Neck*(21102) = {21102, 11022, 1022, 02211, 22110}.

We notice that this is a set of *n* nodes, but if we examine instead the necklace *Neck*(2121) in *B*(3, 4) we see that this set contains only the nodes (2121) and (1212).

Definition 15. The node connectivity of a graph is defined to be the minimum number of nodes whose removal results in a disconnected graph.

Example. We notice from Figure 14 that the node connectivity of *B*(2, 3) is at most 3, but upon examination we see that we still have a disconnected graph if only (100) and (001) are removed. This implies that the node connectivity of *B*(2, 3) is 2 since there is no node whose removal results in a disconnected graph.

With the defnition of node connectivity and the properties of De Bruijn graphs explored, we are now equipped to answer the question,"What kind of node connectivity can we expect to see on an undirected De Bruijn graph?"

Theorem 3 (Thm 2 in [4]). The node connectivity of an undirected De Bruijn graph, *UB*(*d*, *n*), is $2d-2$. Furthermore, for any arbitrary set of node failures *F* with $|F| \leq 2d-3$, the maximum length of the shortest path between any two nodes is less than or equal to 2*n*.

A proof of this theorem can be found in [4], but we omit it here for brevity, and instead look at some examples which actualize some of these bounds.

Remark 1. For most nodes of *B*(*d*, *n*) we see that the set of successors and the set of predecessors are disjoint (see Observation 1) but when we look at the node $(a \cdots a)$ we see that it has d successors and d predecessors. However, since we are examining the undirected graph we see that a node is not considered adjacent to itself, so the node $(a \cdots a)$ has $2d-2$ adjacent nodes. If we remove all of them, then we are clearly left with a disconnected graph as there are no nodes remaining which are adjacent to $(a \cdots a)$, thus exhibiting the lower bound, $2d-2$.

Remark 2. The actualization of the second bound presented on the maximum length of the shortest path between nodes is a bit harder construct. The di_culty in construc- tion is explained when Esfahanian and Hakima [4] present a slightly better bound of min (2*n*, *n* + 5 + log *n*) which has an extensive proof that will not be presented in this paper.

We will now shift our focus back to directed De Bruijn graphs and the approaches taken to find bounds on node connectivity.

Lemma 3. Let *x* be a node in *B*(2, *n*), then *B*(2, *n*) - *Neck*(*x*) has a component with at least $2^n - n - 1$ nodes.

Before beginning the proof of this lemma, it is helpful to examine the following properties of necklaces and binary De Bruijn graphs (i.e. those of the form *B*(2, *n*)).

Definition 16. Let *x* be a node in *B*(2, *n*), then the weight of *x*, denoted *wt*(*x*), is the number of 1's in *x*.

With this in mind, we notice the following observations on necklaces and weights.

Observation 2. For every node *y*∈*Neck*(x) we see that *wt*(*y*) = *wt*(*x*).

Observation 3. There is only one necklace of weight 0 and *n* as *Neck*(0 . . . 0) = {(0 . . :. 0)} and similarly *Neck*(1 . . . 1) = {(1 . . . 1)}.

Observation 4. There is only one necklace of weight 1 and *n*-1 as *Neck*((1 0 . . . 0)) is the set of nodes which contain a single 1 and *Neck*((0 1 . . . 1)) is the set of nodes which contain a single 0.

Observation 5. There can be multiple necklaces of weights between 1 and *n*-1.

Example. In *B*(2, 4) we see that *Neck*((1100)) is not the same as *Neck*((1010))

With these facts in mind, we can proceed with the proof of Lemma 3.

Proof. Let $x = (x_1 \cdots x_n)$ be a faulty node in *B*(2, *n*) with *wt*(*x*) = *k*. We observe that all of the nodes of weight greater than *k* are connected since if we have two such nodes *y* and *z* with *wt*(y), *wt*(z) > *k* then there is a path from *y* to *z* which avoids *Neck*(*x*). One such path is the path:

$$(y_1 \ldots y_n)(y_2 \ldots y_n 1)(y_3 \ldots y_n 11) \ldots (1 \ldots 1)(1 \ldots 1z_1)(1 \ldots 1z_1z_2) \ldots (1z_1 : : : z_{n-1})(z_1 \ldots z_n).$$

We see that every node in that path has weight greater than k since the weight is non- decreasing while we add 1's to the end and then non-increasing when we replace them with the z_i's until we get to *z* which has weight greater than *k*. Similarly, there is a path between any two nodes which have weight less than *k* using the same method with 0's instead.

We will now prove Lemma 3 in cases on the weight of the faulty node.

- Case *wt*(*x*) = 0 (or *n*). There is only one node of weight 0 and therefore all of the nodes have weight greater than this faulty node. We therefore see that *B*(2, *n*)-*Neck*(x) is entirely connected, and thus there is a connected component of size $2^n - 1$.
- Case *wt*(*x*) = 1 (or $n-1$). We see that *B*(2, *n*) - *Neck*(*x*) has no path from (0 . . . 0) to any other node but that each other node is connected to any other node since again all of the remaining nodes have weight greater than 1. Thus the largest component has size $2^n - |Neck(10 \ldots 0)| - 1 = 2^n - n - 1$
- Case $1 < wt(x) < n-1$. We see that this case only arrises when $n \geq 4$ since for $n < 4$ each weight is either in case 1 or case 2 above. By construction we can see there are at least two necklaces of weight *k*, for example *Neck*((0 . . . 01 . . . 1)) and *Neck*((0 . . . 0101 . . . 1)). Now let *y* with *wt*(*y*) = *k* and *y* not in *Neck*(*x*). Then there is a path from any node with weight less than *k* to *y* and there is a path from any node of weight greater than *k* to *y* using the same pathing method as above. We see that both these paths avoid *Neck*(*x*) since our path from lesser weight has non-decreasing weight and once it has reached a node of weight *k* it will be a node in *Neck*(*y*). Thus *B*(2, *n*) - *Neck*(*x*) is connected and then, $2^n - |Neck(x)| = 2^n - n$.

This completes our cases and we see that for all $x \in B(2, n)$, the largest connected component of $B(2, n)$ - $Neck(x)$ has at least $2^n - n - 1$ nodes.

Theorem 4. The De Bruijn graph $B(2, n)$ with a single node failure has a cycle with at least $2^n - n - 1$ nodes.
Remark 3. Notice that this statement is very similar to Lemma 3 but it says something stronger. We claim instead that not only is there a connected component of size $2^n - n - 1$ but that this component actually contains a cycle.

We will not discuss this proof in detail, but the method is to apply Lemma 3 and then to apply the following algorithm to construct the cycle. The proof of the correctness of this algorithm is in [9].

Let x be a faulty node and let $F = Neck(x)$. The following algorithm constructs the largest cycle in the component of $B(2, n)_F$ which contains the node z. This node is a parameter of the algorithm and so it is important that we choose z to be in the largest component before starting the algorithm. This can be done by choosing $wt(z) \neq 0, n$.

Algorithm 1 Constructs a Hamiltonian cycle in the component of B(2; n)F which contains the node z

---

C_z := (z)
mark z
x := z
while there is an edge from x to an unmarked successor y do
  insert y after x in C_z
  mark y
  $x := y$
end while
for each node x in C_z with an unmarked successor y do
  C_y := Cycle(y)
  $C_z := C_z$ JOIN C_y
end for
return C_z

Example. In order to understand this algorithm a little better, we will go through an example of it running on $B(2, 4)$ with faulty node (1100). First, we will remove the necklace of the faulty node (1100). This set is

$$F = Neck((1100)) = \{(1100), (1001), (0011, (0110)\}.$$

We know from the proof of Lemma 3 that for this particular F, $B(2, 4)_F$ is connected and so we can choose any node z at which to start the algorithm.

We start at the node z = (1101), and then build C_z by successively appending nodes to the end of C_z always choosing from among the successors the first node lexicographically which is unmarked and also not in F. We will continue in this manner until we are at a point in which there is no successor which is both unmarked and not in F.

The successors of (1101) are (1010) and (1011). At this point only (1101) is marked and neither of these successors are in F, thus our first step will be to add the node (1010) to C_z as it is the first, lexicographically, among the successors of (1101). We notice here that at this point, choosing the successor will generally mean choosing the node which adds a 0 to the end because of the lexicographical ordering. This gives,

$$C_z = (1101)(1010):$$

After this we continue adding nodes to C_z in this fashion so that we obtain

$$C_z = (1101)(1010)(0100)$$
$$= (1101)(1010)(0100)(1000)$$
$$= (1101)(1010)(0100)(1000)(0000).$$

We notice here that the successors of (0000) are (0000) and (0001). The node (0000) has already been added to C_z and so it is considered marked and we must instead choose to add the node (0001). This gives us

$$C_z = (1101)(1010)(0100)(1000)(0000)(0001)$$

and then we continue adding nodes again to obtain

$$C_z = (1101)(1010)(0100)(1000)(0000)(0001)(0010)$$
$$= (1101)(1010)(0100)(1000)(0000)(0001)(0010)(0101)$$
$$= (1101)(1010)(0100)(1000)(0000)(0001)(0010)(0101)(1011).$$

It is at this point that since the successors of (1011) are (0110) and (0111), we choose to add (0111) since (0110) is in the removed set *F*. We now have

$$C_z = (1101)(1010)(0100)(1000)(0000)(0001)(0010)(0101)(1011)(0111).$$

Continuing this process, we add (1110) and then see that with the addition of (1110) that of the two successors, (1101) is marked and (1100) is in *F*. Thus we reach the end of the "while" loop with

$$C_z = (1101)(1010)(0100)(1000)(0000)(0001)(0010)(0101)(1011)(0111)(1110):$$

We now enter into the "for" loop and try to find the first node which has an unmarked successor not in *F*. The first few nodes do not have this property, but we find that the node (0111) has successor (1111) which is unmarked and not in *F*.

We now use Algorithm 1 on the node (1111) to construct C_y. We will set y = (1111) and then construct C_y while maintaining the same list of marked nodes as before. Since the successors of (1111) are (1111) which is now marked and (1110) which is also marked we see that C_y = (1111) after the "while loop" completes. Now, after completing the construction of C_y, we then go back to our C_z and join it with C_y by inserting C_y after (0111) in C_z. This gives us the cycle

$$C_z = (1101)(1010)(0100)(1000)(0000)(0001)(0010)(0101)(1011)(0111)(1111)(1110).$$

This C_z is a cycle on the 12 remaining nodes in $B(2, 4)$ and is therefore a Hamiltonian cycle on the component containing z = (1101).

Remark 4. It may appear to be coincidence that this algorithm returns a cycle instead of a path, but if we recall that each predecessor of a node has all same successors, then since each of the predecessors of our start node have been visited, then so, too, must all the successors of those predecessors. This means that since the first node in C_z is the only one which has not had an in-going edge and the last node is the only one without an out-going edge, then this last node must be a predecessor of the original node.

V. Concluding Statements

In Section 3 we examined N. G. de Bruijn's original proof in [2] that there are $2^{2^{n-1}-n}$ Hamiltonian cycles in the De Bruijn graph $B(2, n)$.

We then saw in Section 4 that in the case of a single node failure on $B(2, n)$ that we could use the technique of removing a necklace to show that these graphs admit cycles of length greater than or equal to $2^n - n - 1$.

ACKNOWLEDGMENT

Thank you for the program of the academic leadership grant (ALG), Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran (Indonesia), which has been providing facilities to conduct a research and publication.

REFERENCES

[1] Oliver Collins, Sam Dolinar, Robert McEliece, and Fabrizio Pollara. A vlsi decomposition of the De Bruijn graph. J. ACM, 39:931{948, October 1992.
[2] N. G. de Bruijn. A combinatorial problem. Nederl. Akad. Wetensch., Proc., 49:758{764 = Indagationes Math. 8, 461{467 (1946), 1946.
[3] J.-C. Delvenne and R. Carli. Optimal strategies in the average consensus problem. In Decision and Control, 2007 46th IEEE Conference on, pages 2498 {2503, dec. 2007.
[4] Abdol-Hossein Esfahanian and S. Louis Hakimi. Fault-tolerant routing in De Bruijn communication networks. IEEE Trans. Comput., 34(9):777{788, 1985.
[5] W. C. Hu_man and V. Pless. Fundamentals of error-correcting codes. Cambridge Univ. Press, 2003.
[6] Zolt_an K_asa. On arc-disjoint hamiltonian cycles in De Bruijn graphs. CoRR, abs/1003.1520, 2010.
[7] R. Rowley and B. Bose. Edge-disjoint hamiltonian cycles in De Bruijn networks. In Distributed Memory Computing Conference, 1991. Proceedings., The Sixth, pages 707 {709, apr-1 may 1991.
[8] Robert Rowley and Bella Bose. On the number of arc-disjoint Hamiltonian circuits in the De Bruijn graph. Parallel Process. Lett., 3(4):375{380, 1993.
[9] Robert A. Rowley and Bella Bose. Fault-tolerant ring embedding in De Bruijn networks. Computers, IEEE Transactions on, 42(12):1480 {1486, dec 1993.
[10] Vladimir Raphael Rosenfeld. Enumerating De Bruijn sequences. In MATCH Communications in Mathematical and in Computer Chemistry, page 2002.

BIOGRAPHY

Mochamad Suyudi, is a lecturer at the Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran. Bachelor in Mathematics at the Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran, and Master in Mathematics at the Faculty of Mathematics and Natural Sciences, Universitas Gajah Mada. Currently pursuing PhD program in the field of Graphs at Universiti Malaysia Terengganu.