Application of Directed De Bruijn Graphs and Eulerian Circuits for DNA Sequence Assembly in Bioinformatics

Sahla Nailah Salsabilla - 13525134
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: nailahsahla858@gmail.com , 13525134@std.stei.itb.ac.id

***Abstract*—DNA sequence assembly has become an important problem in bioinformatics. When scientists read DNA, the machine can only produce very short pieces of sequences called reads. Because the original DNA is very long, we need a method to put these short reads back together in the correct order. One approach to solve this problem is by using Directed De Bruijn Graphs, which use pieces of DNA as nodes and overlaps as directed edges. In the past, people used Hamiltonian paths, but it was too slow for big data. The De Bruijn Graph uses an Eulerian path approach, which is faster and better for modern computers. This paper will examine how to use the De Bruijn Graph and K-mer fragmentation to assemble a random DNA sequence, as this mathematical method is very useful for processing large biological data in bioinformatics.**

***Keywords*—Bioinformatics, DNA Sequence Assembly, Directed Graph, De Bruijn Graph, Eulerian Path, K-mers.**

I. Introduction

Biology couldn't grow without understanding DNA. DNA contains all the important information about living things. Scientists want to read DNA to find cures for diseases and understand how life works. But, reading DNA is not an easy task. The machines used by scientists, called DNA sequencers, cannot read the long DNA from start to finish. They can only produce very short pieces of the DNA, which are called reads. Imagine having a very thick book, but it is destroyed into millions of small papers. Because the original DNA is very long, it is a big problem to put them back together in the correct order. That's why, scientists are competing to find the best method to solve this giant puzzle.

In order to solve this problem, scientists must use computers to arrange the short reads. This process is called DNA sequence assembly. In the past, people tried to match the pieces one by one using a method called the Hamiltonian path. But, this method is too slow and the computer will freeze because there are millions of pieces from the modern machines. To satisfy the need of the scientists, bioinformatics must provide a faster method to arrange the DNA.

Basically, DNA sequence assembly is an information processing system that attempts to predict the correct long sequence. Many fields of science have used math to increase the speed of this process. One approach of method to solve this is by using a graph which harmonizes the short reads into a clear path. Instead of matching pieces one by one, the computer breaks the short reads into smaller parts called K-mers. Then, it uses a Directed De Bruijn graph, which has these K-mers as nodes and directed edges. This graph has taken the basic concept of Graph Theory, specifically using the Eulerian path to easily find the final DNA sequence without taking too much computer memory.

II. DNA Sequence

Deoxyribonucleic acid (abbreviated DNA) is the molecule that carries genetic information for the development and functioning of an organism. It is made of two linked strands that wind around each other like a twisted ladder, known as a double helix. Attached to the DNA backbone are four bases: adenine (A), cytosine (C), guanine (G), and thymine (T). The sequence of these bases encodes important biological information.

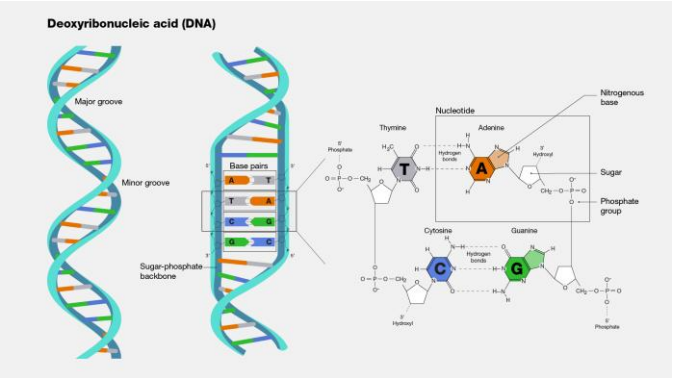


Fig 2.1 DNA Illustration (Source: https://www.genome.gov/genetics-glossary/Deoxyribonucleic-Acid-DNA)

The sequence assembly process is needed because modern DNA sequencing technology cannot read whole genomes in one step. Instead, the sequencing machines can only read small pieces. For example, the older Sanger sequencers can read around 2000 bases. Sanger is very accurate, but it has a great disadvantage: it is very expensive,

making it inappropriate for sequencing whole genomes. Newer machines like NextGen sequencers are cheaper but read very short pieces, around 30 bases.

DNA sequence assembly is a process where these short DNA fragments (called reads) are merged into a longer DNA sequence to reconstruct the original DNA. The longer sequence result is called a "contig". To process these short reads in a computer, we must break them down again into even smaller strings with a fixed length. This fixed length is called k, and the string pieces are called k-mers. For example, if we have a string "ATGGC" and we choose k=3, the k-mers are "ATG", "TGG", and "GGC". This step is important to connect the biological data to a mathematical graph.

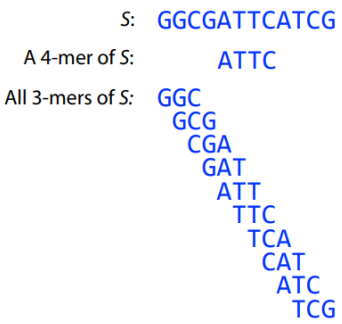


Fig 2.2 K-mer DNA string (Source: https://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf)

III. Graph Theory

In Discrete Mathematics, a graph G is defined as G = (V, E), where V is a non-empty set of vertices (or nodes), and E is a set of edges that connect a pair of vertices. Based on the edges, graphs are divided into several types. A "simple graph" is a graph with no multiple edges or loops. If there are multiple edges connecting the exact same two nodes, it is called a "multigraph". If a graph also has an edge that starts and ends at the same node (loop), the graph is called a "pseudograph".

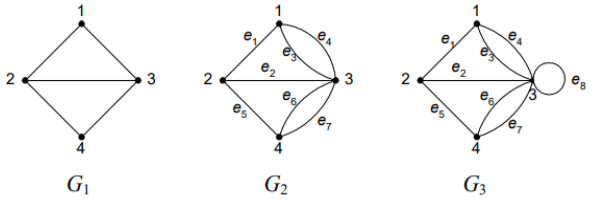


Fig 3.1. G1) simple graph, G2) multigraph, G3) pseudograph (Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/20-Graf-Bagian1-2026.pdf)

Degree, notated as d(v), is the number of edges connected to a node.

A. Directed Graph

A graph where every edge has a direction or orientation is called a "directed graph". In a directed graph, the degree is divided into two types: in-degree and out-degree. In-degree d_{in} is the number of arrows entering a node, and out-degree d_{out} is the number of arrows leaving a node.

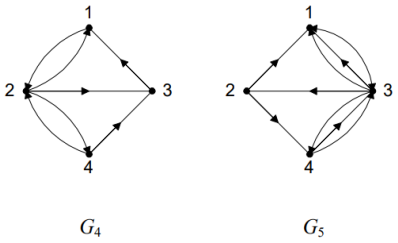


Fig 3.2. Graph with in-degree and out-degree (Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/20-Graf-Bagian1-2026.pdf)

For example, if we observe Node 2 in the graph G4, there are exactly 2 arrows pointing towards it (from Node 1 and Node 4), so its in-degree is 2 d_{in} (2) = 2. At the same time, there are 3 arrows going out from Node 2 (pointing to Node 1, Node 3, and Node 4), so its out-degree is 3 d_{out} (2) = 3.

B. Adjacency Matrix

To process a graph in a computer program, we cannot just use a visual drawing. We need to translate the nodes and edges into numbers. The most common way to do this is by using an Adjacency Matrix.

An adjacency matrix, notated as A = [a_{ij}], is a square table where the rows and columns represent the nodes of the graph. For a normal simple graph, the rule to fill this matrix is very simple:

- a_{ij} = 1, if node i and node j are connected by an edge
- a_{ij} = 0, if node i and node j are not connected

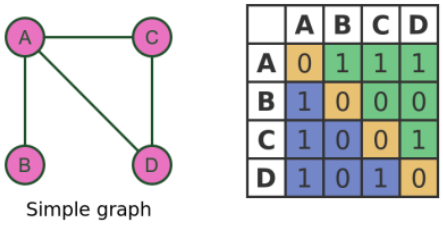
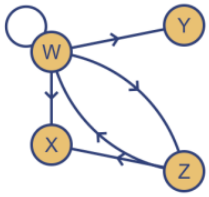


Fig 3.3 Simple graph adjacency matrix (Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/21-Graf-Bagian2-2026.pdf)

For a directed graph, the direction of the arrow is very important. We only put a number in the matrix if there is an arrow pointing exactly from the row node (node i) to the column node (node j). Additionally, if the graph has a "loop" (an arrow that starts and ends at the exact same node), it will be written on the main diagonal of the matrix. For example, if node W has a loop, the matrix will have the number 1 at row W and column W.

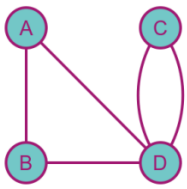


Directed graph with loop

	W	X	Y	Z
W	1	1	1	1
X	0	0	0	0
Y	0	0	0	0
Z	1	1	0	0

Fig 3.4 Directed graph adjacency matrix
(Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/21-Graf-Bagian2-2026.pdf)

Furthermore, because the De Bruijn graph is a multigraph, there is one more important rule. If there are multiple edges connecting the exact same two nodes, the matrix element is filled with the total number of those multiple edges. By using this matrix, computers can easily store the complex graph data and calculate the in-degree and out-degree of every node just by summing the numbers in the columns and rows.



Multigraph

	A	B	C	D
A	0	1	0	1
B	1	0	0	1
C	0	0	0	2
D	1	1	2	0

Fig 3.5 Multigraph adjacency matrix
(Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/21-Graf-Bagian2-2026.pdf)

C. *De Bruijn Graph*

De Bruijn Graph is a directed multigraph used to assemble DNA. In this graph, we do not use full reads as nodes. Instead, the nodes are built from the prefixes and suffixes of the k-mers. Because a prefix or a suffix is formed by removing exactly one letter from the k-mer, the length of every node is:

$$\text{prefix / suffix length} = k - 1 \qquad (1)$$

To understand how the nodes are formed, let's assume we choose a length of k = 3. The rule is to take each length-3 input string (k-mer) and split it into two overlapping substrings of length k - 1 = 3 - 1 = 2. We call these the left 2-mer and the right 2-mer.

The left 2-mer acts as the prefix node, and the right 2-mer acts as the suffix node, while the directed edges represent the 3-mers themselves. Because different k-mers can have the exact same prefix and suffix, there can be multiple edges connecting the same two nodes. That is why the De Bruijn graph is defined as a directed multigraph.

While DNA sequences are typically linear, they can be treated as Eulerian Circuits by adding a virtual 'return edge' from the terminal node to the starting node. This mathematical transformation allows us to apply the properties of an Eulerian Circuit to a linear DNA assembly problem, making the algorithmic traversal more robust.

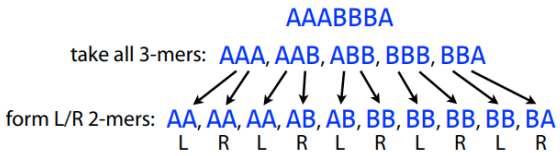


Fig 3.3 DNA string to 2-mers De Bruijn Graph (Source: https://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf)

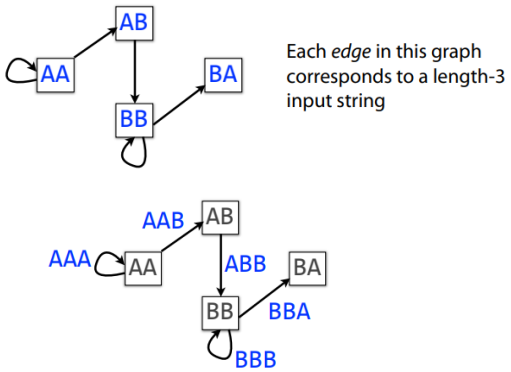


Fig 3.4 De Bruijn Graph (Source: https://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf)

For a more rigorous understanding, the De Bruijn graph can be defined formally. Given a set of DNA reads R over an alphabet $\Sigma = \{A, C, G, T\}$ and a fixed integer p (the length of the k-mer), the order-p De Bruijn graph $Gp(R) = (U, V)$ is constructed as follows:

- Nodes (U): Each node is a (p-1)-mer that appears as the prefix or suffix of some p-mer in the dataset. Formally, $U = \{ d \in \Sigma^{p-1} \mid \exists f \in F : d = \text{prefix}(f, p-1) \vee d = \text{suffix}(f, p-1) \}$.
- Edges (V): Each directed edge corresponds to a p-mer and links the overlapping (p-1)-mer prefix and suffix. Formally, $V = \{ (\text{prefix}(f, p-1), \text{suffix}(f, p-1)) \mid f \in F \}$.

In this formal notation, for two nodes $v_1 = a_1 \ldots a_{p-1}$ and $v_2 = a_2 \ldots a_p$, a directed edge $v_1 \rightarrow v_2$ exists whenever their (p-2)-overlap matches. This mathematical structure allows for the union of graphs when dealing with a collection of strings, where nodes with identical (p-1)-mer labels are merged to form a unified graph.

D. Eulerian Walk versus Hamiltonian Path

In the early stages of bioinformatics, researchers attempted to solve the DNA sequence assembly problem by finding a Hamiltonian Path, a path that visits every node exactly once. However, identifying a Hamiltonian path is classified as an NP-complete problem, meaning it is computationally intensive and inefficient for the massive datasets generated by modern sequencing technologies.

To overcome this limitation, De Bruijn graphs employ the Eulerian Walk approach, which focuses on visiting every directed edge exactly once instead of the nodes. A directed graph G = (V, E) contains an Eulerian path if and only if the following conditions are satisfied:

- in-degree ($d_{in}(v)$) and out-degree ($d_{out}(v)$) for every node $v \in V$ must be evaluated.
- There exists at most one node where $d_{out}(v) - d_{in}(v) = 1$. This node serves as the starting point of the path.
- There exists at most one node where $d_{in}(v) - d_{out}(v) = 1$. This node serves as the end point of the path.
- For all other nodes $v \in V$ that are neither the start nor the end node, the graph must satisfy the condition $d_{out}(v) = d_{in}(v)$.

If these conditions are met, the graph is considered traversable. This ensures that a computer algorithm can successfully reconstruct the original DNA sequence by visiting every edge exactly once, without missing any fragment or repeating any input.

While DNA sequences are typically linear, they can be treated as Eulerian Circuits by adding a virtual 'return edge' from the terminal node to the starting node. This mathematical transformation allows us to apply the properties of an Eulerian Circuit to a linear DNA assembly problem, making the algorithmic traversal more robust.

IV. DNA Sequence Assembly with De Bruijn Graphs

To understand how the De Bruijn Graph works computationally in bioinformatics, we propose a structured simulation using a dummy DNA sequence. This section will thoroughly explain the step-by-step mathematical process of DNA sequence assembly, starting from raw string fragmentation to the final algorithmic reconstruction using an Eulerian path.

A. Data Initialization

In real biological experiments, sequencing machines cannot read a whole genome from start to finish. They output scattered, short overlapping strings called "reads". For this simulation, let us assume an original unknown DNA sequence: "ATGGCGTGCA".

From this sequence, the machine generates three short reads. The input reads for our system are:

1. Read 1: ATGGC (Length = 5)
2. Read 2: GGCGT (Length = 5)
3. Read 3: CGTGCA (Length = 6)

B. K-mer Fragmentation Process

To build a De Bruijn graph, we must break the reads down into k-mers. In this model, we determine the fixed length k = 3. The number of k-mers produced from a sequence of length L can be calculated using the mathematical formula:

$$\text{Number of k-mers produced} = L - k + 1 \quad (2)$$

Apply (2) to our reads:

1. Read 1 (ATGGC, L=5):

   Number of k-mers produced = 5 - 3 + 1 = 3 pieces of 3-mers.

   ATG, TGG, GGC

2. Read 2 (GGCGT, L=5):

   Number of k-mers produced = 5 - 3 + 1 = 3 pieces of 3-mers.

   GGC, GCG, CGT

3. Read 3 (CGTGCA, L=6):

   Number of k-mers produced = 6 - 3 + 1 = 4 pieces of 3-mers.

   CGT, GTG, TGC, GCA

After combining all extractions and removing the exact duplicates, we define our set of unique 3-mers as:

M = {ATG, TGG, GGC, GCG, CGT, GTG, TGC, GCA}

These unique 3-mers will act as the directed edges in our graph.

C. Node Extraction and Graph Construction

In a De Bruijn graph, the nodes (V) are strictly defined as the prefixes (left part) and suffixes (right part) of the k-mers. Since our k = 3, the length of the nodes will be k - 1 = 2. The complete extraction of the 2-mers from our set M is summarized in Table I.

TABLE I. NODE EXTRACTION FROM K-MERS

K-mer	Left 2-mer (prefix)	Right 2-mer (suffix)
ATG	AT	TG
TGG	TG	GG
GGC	GG	GC
GCG	GC	CG
CGT	CG	GT
GTG	GT	TG
TGC	TG	GC
GCA	GC	CA

From this extraction, we can properly define the graph G = (V, E). The set of vertices is:

V = {AT, TG, GG, GC, CG, GT, CA}

The set of directed edges is:

E = {ATG, TGG, GGC, GCG, CGT, GTG, TGC, GCA}

D. Degree Analysis and Adjacency Matrix

To prove that our graph can be solved computationally, we must analyze the connections between the nodes. A graph can be represented efficiently in a computer using an Adjacency Matrix. Below is the adjacency matrix for our De Bruijn graph, where 1 indicates a directed edge from the row node to the column node, and 0 indicates no edge.

	AT	TG	GG	GC	CG	GT	CA
AT	0	1	0	0	0	0	0
TG	0	0	1	1	0	0	0
GG	0	0	0	1	0	0	0
GC	0	0	0	0	1	0	1
CG	0	0	0	0	0	1	0
GT	0	1	0	0	0	0	0
CA	0	0	0	0	0	0	0

Fig 4.1 Adjacency Matrix of De Bruijn Graph DNA Sequence

The mathematical relationships defined in Fig 4.1 can be visually represented as a directed multigraph, which illustrates the flow of the Eulerian walk such as:

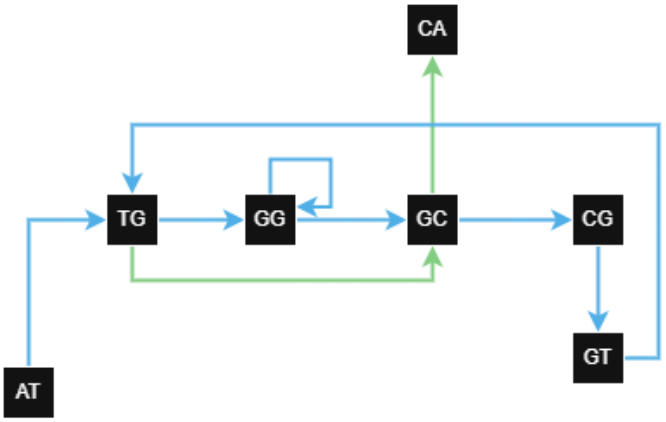


Fig. 4.2. De Bruijn Graph visualization for DNA sequence "ATGGCGTGCA". The nodes represent 2-mers, while the directed edges represent 3-mers. The blue arrows indicate the primary traversal path starting from the "AT" node. The green arrows represent the branching cycle (the secondary path) that redirects the traversal through the "GC" node to successfully complete the Eulerian walk.

To determine the topological properties, we calculate the in-degree (d_{in}) and out-degree (d_{out}) for every node by summing the matrix columns and rows:

- Node AT: $d_{in} = 0$, $d_{out} = 1$
- Node TG: $d_{in} = 2$, $d_{out} = 2$
- Node GG: $d_{in} = 1$, $d_{out} = 1$
- Node GC: $d_{in} = 2$, $d_{out} = 2$
- Node CG: $d_{in} = 1$, $d_{out} = 1$
- Node GT: $d_{in} = 1$, $d_{out} = 1$
- Node CA: $d_{in} = 1$, $d_{out} = 0$

If we observe the analysis above, nodes like TG and GC have degrees of 2. This creates a branching cycle in the graph. This mathematical proof shows that real DNA graphs are not just straight lines, and they need a specific algorithm to be traversed.

E. Assembly using Eulerian Walk Algorithm

To assemble the final DNA sequence, the computer must find an Eulerian walk. According to Graph Theory, a directed graph has an Eulerian walk if and only if:

1. At most one node has $d_{out} - d_{in} = 1$ (Starting Node).
2. At most one node has $d_{in} - d_{out} = 1$ (Ending Node).
3. All other nodes are perfectly balanced ($d_{out} = d_{in}$).

Based on our degree analysis in Section D:

- Node AT has $d_{out}(1) - d_{in}(0) = 1$. Therefore, AT is mathematically proven as the Starting Node.
- Node CA has $d_{in}(1) - d_{out}(0) = 1$. Therefore, CA is proven as the Ending Node.
- All other nodes are balanced (e.g., TG is 2=2, GG is 1=1).

Because all conditions are met, the algorithm begins traversing from the starting node AT. The rule is to visit every directed edge in E exactly once. The step-by-step traversal is as follows:

1. Start at AT, traverse edge ATG to TG.
2. From TG, traverse edge TGG to GG.
3. From GG, traverse edge GGC to GC.
4. From GC, traverse edge GCG to CG.
5. From CG, traverse edge CGT to GT.
6. From GT, traverse edge GTG back to TG (completing the cycle).
7. From TG, traverse the remaining edge TGC to GC.
8. From GC, traverse the final edge GCA to CA (Ending Node).

The final Eulerian path is:

AT ➔ TG ➔ GG ➔ GC ➔ CG ➔ GT ➔ TG ➔ GC ➔ CA

To get the final DNA string, we take the starting node AT and append the last character of every subsequent node in the path:

AT + G + G + C + G + T + G + C + A = ATGGCGTGCA

The Eulerian walk algorithm has successfully solved the branching cycle and reconstructed the short fragments back into the original DNA sequence perfectly.

V. Conclusion

Based on the study of De Bruijn graphs for DNA sequence assembly, several important conclusions can be drawn:

1. DNA sequence assembly is a fundamental problem in bioinformatics because modern sequencing technologies can only produce fragmented, short reads. The assembly process is essential to reconstruct the original long DNA sequence, which serves as the blueprint for living organisms.
2. Bruijn graph proves to be an effective mathematical solution for this problem. By representing DNA reads as directed edges and their overlapping prefixes and suffixes as nodes, the assembly problem is transformed into a structured graph problem, specifically a directed multigraph, which handles overlapping sequences more efficiently than traditional Hamiltonian path approaches.
3. use of an Eulerian walk algorithm significantly enhances the efficiency of the assembly process. Since the algorithm only requires visiting every edge exactly once to reconstruct the original sequence, it provides a faster and more scalable solution for massive genomic data compared to the computationally expensive Hamiltonian path method.
4. This study demonstrates that discrete mathematical structures, such as adjacency matrices and degree analysis, are not just theoretical concepts but practical tools that underpin essential modern scientific technologies. The successful reconstruction of the dummy sequence "ATGGCGTGCA" through Eulerian traversal confirms the robustness of this graph-based approach.

Acknowledgment

The author would like to thank first of all, to Allah for all the guidance and strength throughout the process of learning and writing this paper. The author would like to express her deepest gratitude to Mr. Rinaldi Munir, the lecturer of Discrete Mathematics IF2120, for his invaluable knowledge, insights, and continuous guidance throughout the semester. He has greatly inspired the author to understand the depth and beauty of graph theory.

Furthermore, the author would like to thank her family and friends for their support and encouragement. Their presence has been a constant source of motivation. Last, the author hopes that this paper can serve as a meaningful contribution to her peers and provide a clearer perspective on the application of graph theory in bioinformatics. It is the author's hope that this work marks the beginning of a deeper journey into the world of computer science and innovative technology.

References

[1] R. Munir, "Teori Graf," Bahan Kuliah IF2120 Matematika Diskrit, Program Studi Teknik Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir. [Accessed: June 18].

[2] B. Langmead, "De Bruijn Graph assembly," Lecture Notes, Whiting School of Engineering, Johns Hopkins University, 2026. [Online]. Available: https://cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf. [Accessed: June 18, 2026].

[3] DNA Baser, "What is DNA Sequence Assembly?," DNA Sequencing Software, 2026. [Online]. Available: https://www.dnabaser.com/articles/what%20is%20sequence%20assembly.html. [Accessed: June 18, 2026].

[4] National Human Genome Research Institute, "Deoxyribonucleic Acid (DNA)," Genetic Glossary, National Institutes of Health (NIH), 2026. [Online]. Available: https://genome.gov/genetics-glossary/Deoxyribonucleic-Acid-DNA. [Accessed: June 18, 2026

[5] A. D. Couto, F. R. Cerqueira, R. S. Ferreira, and A. P. Oliveira, "Proposal of a New Method for de Novo DNA Sequence Assembly Using de Bruijn Graphs," International Conference on Computational Intelligence and Bioengineering, 2026. [Online]. Available: https://www.researchgate.net/. [Accessed: June 18, 2026].

[6] C. J. Prybol, A. T. Hammack, E. A. Ashley, and M. P. Snyder, "A Novel Approach for Accurate Sequence Assembly Using de Bruijn graphs," Department of Genetics, Stanford University School of Medicine, 2026. [Online]. Available: https://www.researchgate.net/. [Accessed: June 18, 2026].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2025

Sahla Nailah Salsabilla (13525134)