Analyzing Optimal Narrative Pathfinding in Detroit: Become Human by Applying Dijkstra's Algorithm on Negative Log-Probability Weighted Directed Acyclic Graphs

Andro Irsa Syafiq - 13525129
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: androirsa@gmail.com , 13525129@std.stei.itb.ac.id

***Abstract*—This paper presents an analysis of optimal narrative pathfinding in Detroit Become Human through the application of Directed Acyclic Graphs and Dijkstra algorithm. The research investigates the use of global player choice statistics to transform traversal probabilities into graph weights using a negative logarithm function. The results highlight the power of graph theory in solving combinatorial probability problems and demonstrate its relevance in the context of state space search and discrete mathematics.**

Keywords—Directed Acyclic Graph; Dijkstra Algorithm; Probability; Detroit Become Human; State Space Search

I. Introduction

Graph theory is a core area of discrete mathematics that supports many fields in computer science, from algorithm optimization to data analysis. Its concepts offer a structured way to represent and reason about interconnected networks and state spaces. As software and digital media grow more complex, applying graph theory helps us systematically organize and navigate massive amounts of information, bridging the gap between theoretical concepts and practical software development.

In the world of interactive gaming, *Detroit: Become Human* serves as an excellent, real-world example of graph theory in action. The game is built upon a massive branching storyline where every critical player decision leads to different outcomes. Because the narrative progresses chronologically and players cannot naturally reverse time, this structure perfectly forms a Directed Acyclic Graph (DAG). By analyzing this graph alongside player statistics, we can uncover interesting patterns, such as the most probable survival routes based on global player habits.

This paper explores the narrative structure of *Detroit: Become Human* using graph theory and pathfinding algorithms, utilizing global player choice percentages sourced from community data. In this context, the "optimal path" is defined as the route with the highest cumulative probability of reaching the most favorable narrative outcome (the 'Best Ending'). Finding the most likely narrative path presents a unique algorithmic challenge. If we simply multiply the choice probabilities along a deep graph, the numbers get too small and the system risks arithmetic underflow. To solve this, a negative logarithm transformation (-log(P)) is applied to the probability weights. This mathematical trick effectively turns a probability multiplication problem into a shortest-path addition problem, allowing us to implement Dijkstra algorithm. Ultimately, this work aims to demonstrate how graph operations can be creatively applied to solve complex structural problems in modern digital media.

II. Theoretical Background

A. Directed Acyclic Graphs (DAG)

A graph G is defined as a pair (V, E), where V represents a set of vertices (nodes) and E represents a set of edges connecting these vertices. In the context of interactive narrative structures like Detroit: Become Human, the game flow can be accurately modeled as a Directed Acyclic Graph (DAG).

In this model, each vertex $v \in V$ represents a specific scene, event, or decision point within the game. Each directed edge $e = (u, v) \in E$ represents a narrative transition or player choice leading from state u to state v. The graph is strictly acyclic because the narrative flows linearly forward in time, players cannot traverse backward to a previous decision point without entirely restarting the chapter.

B. Dijkstra's Algorithm

Dijkstra's algorithm is a fundamental graph search algorithm that solves the single-source shortest-path problem for a graph with non-negative edge path costs. Given a source vertex, the algorithm finds the lowest-cost path to all other vertices (or a specific target vertex) by systematically exploring the most promising adjacent nodes.

Traditionally, the "shortest path" refers to minimizing physical distance or traversal time. However, in this study, the algorithm is adapted to find the "optimal narrative path." The

optimal path is defined not by the fewest number of edges, but by the highest cumulative probability of player choices leading to a targeted "Best Ending" node

C. Arithmetic Underflow in Probability Trees

In order to determine the most likely path taken by the global player base, the cumulative probability of a sequence of independent choices must be calculated. If a path consists of n sequential choices, each with a probability P_i (where $0 < P_i \leq 1$), the total probability P_{total} of reaching the final node is the product of all individual probabilities:

$$P_{total} = \prod_{i=1}^{n} P_i$$

Because P_i is always a fraction or decimal, multiplying numerous probabilities together results in an exponentially shrinking value. In complex narrative chapters involving multiple characters and dozens of decisions, this calculation produces infinitesimally small floating-point numbers. In computational mathematics, this leads to arithmetic underflow, a condition where the computer's memory limits cannot accurately represent the minuscule numbers, rounding them to exactly zero and causing the algorithm to fail.

D. Negative Log-Probability Weighting

In order to resolve the arithmetic underflow issue and enable the use of Dijkstra's algorithm, which requires additive edge weights, a logarithmic transformation is applied. Utilizing the logarithmic property that converts multiplication into addition, the probability P_i of each edge is transformed into a weight W_i:

$$W_i = -\log(P_i)$$

Since $0 < P_i \leq 1$, the resulting W_i is guaranteed to be a non-negative value, perfectly satisfying the prerequisite for Dijkstra's algorithm. Through this transformation, the product of probabilities is converted into a summation of weights:

$$-\log(P_{total}) = \sum_{i=1}^{n} -\log(P_i)$$

This mathematical trick perfectly aligns with the logic of Dijkstra's algorithm. A highly popular choice (e.g., $P \approx 1$) will yield a weight near zero, representing a low-cost, highly favorable edge. Conversely, a rare choice will yield a high weight, discouraging the algorithm from traversing it. Thus, minimizing the sum of negative logarithms is mathematically equivalent to maximizing the product of the probabilities.

E. Graph Representation

In order to computationally process a Directed Acyclic Graph using Dijkstra's algorithm, the narrative structure must be translated into a machine-readable data structure. In discrete mathematics, there are two primary methods for representing graphs: the Adjacency Matrix and the Adjacency List.

An Adjacency Matrix is a 2D array of size $|V| \; x \; |V|$. If there is a directed edge from vertex u to vertex v, the matrix cell at row u and column v will store the edge weight (in this case, the negative log-probability, W). If no direct transition exists between the two scenes, the cell is assigned a value of infinity (∞), representing an impassable route. While matrix representation provides $O(1)$ time complexity for edge lookups, it consumes $O(|V|^2)$ memory space.

Given that narrative branching in *Detroit: Become Human* creates a relatively sparse graph, where each node only branches into two or three choices rather than connecting to all other nodes, an Adjacency List is far more memory-efficient. An Adjacency List stores an array or dictionary of vertices, where each vertex points to a list of its connected neighbors along with their respective edge weights. This reduces the spatial complexity to $O(|V| + |E|)$, making the pathfinding algorithm significantly faster when traversing the large state-space of the "Crossroads" chapter.

III. Method

A. Data Acquisition and Preprocessing

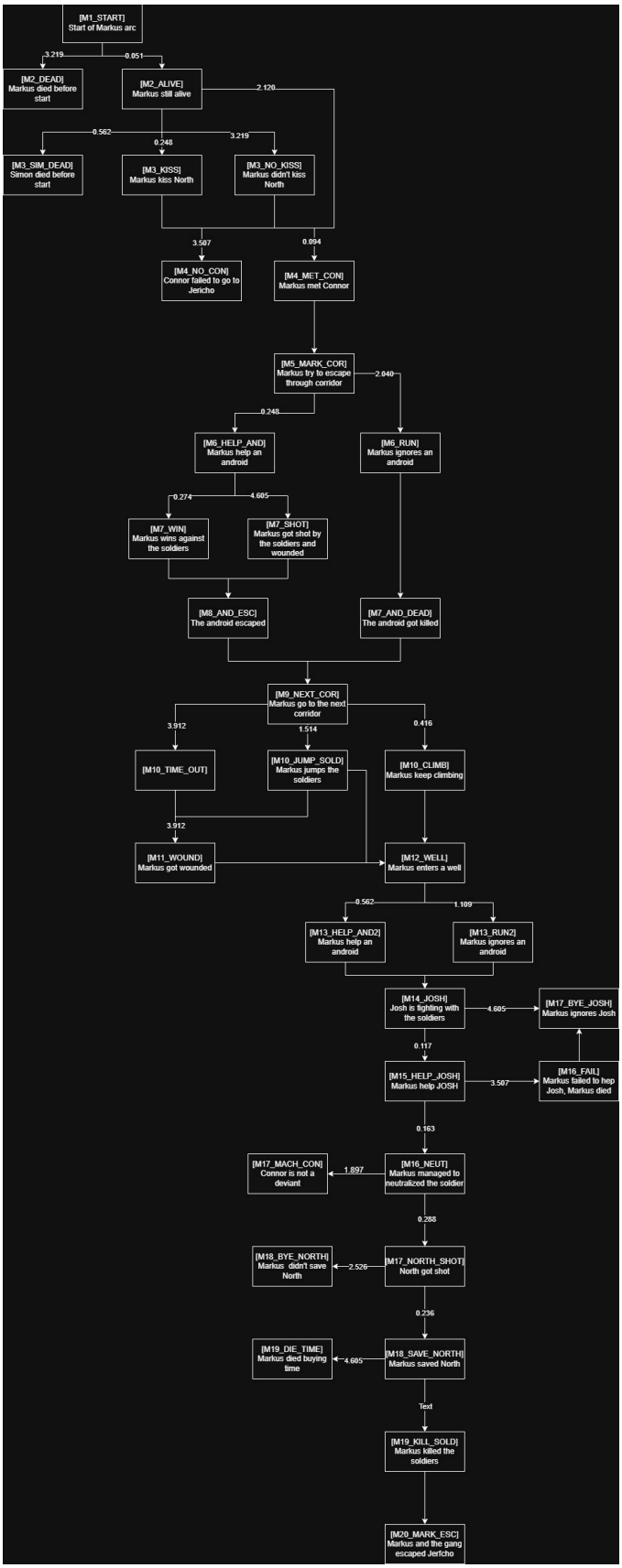
The primary data source for this research is the global player choice statistics for the "Crossroads" chapter in *Detroit: Become Human*. Unlike early-game chapters that feature isolated character arcs, "Crossroads" merges the storylines of all three protagonists (Connor, Kara, and Markus) into a singular, highly complex narrative web.

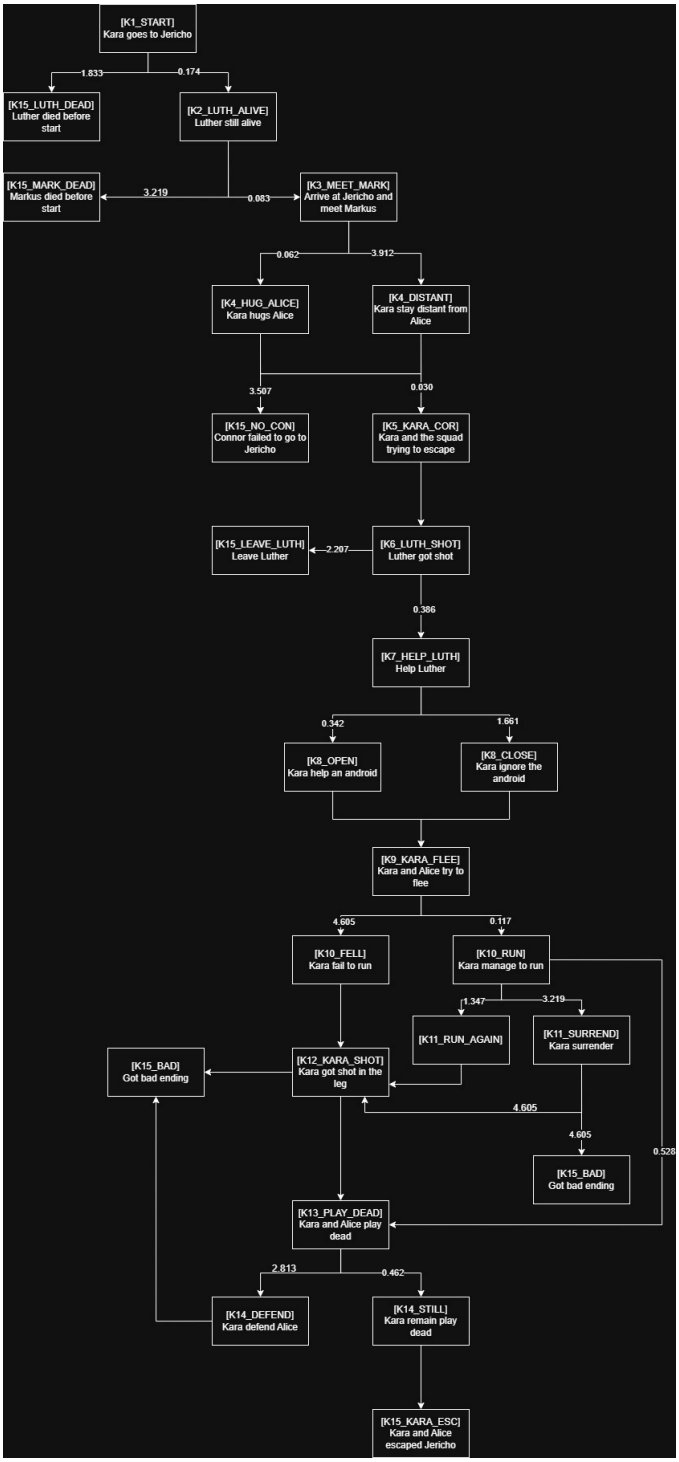
The raw data is parsed to extract the node identifiers, connecting edges, and the percentage of players who chose each specific transition. These percentages are then converted into decimal probabilities P. Any transition with a probability of 0 (impossible routes or locked paths) is removed during the preprocessing phase to maintain data integrity before being fed into the pathfinding algorithm.

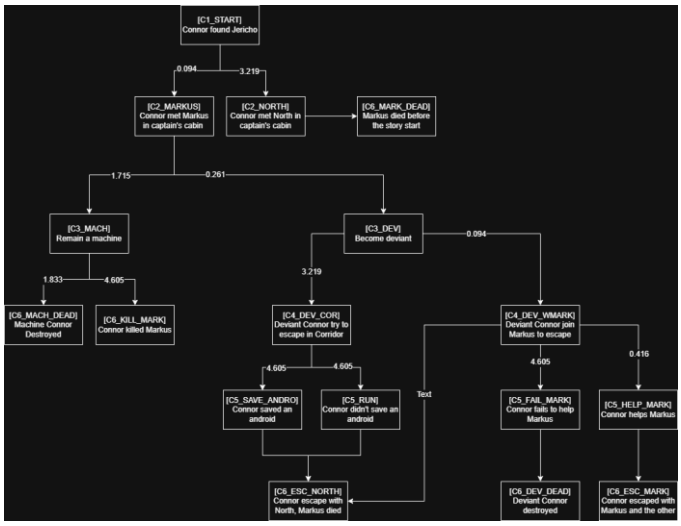
B. Graph Modeling and Visualization

The preprocessed data is mapped into a Directed Acyclic Graph (DAG). The vertices (V) represent pivotal scenes (e.g., "Connor confronts Markus," "Kara sneaks past patrols," or "Markus detonates the bomb"). The directed edges (E) represent the player's choices.

Given the intersecting nature of the three protagonists, a single decision made by one character can lock or unlock entire sub-graphs for another. To ensure computational efficiency when handling this large network, the graph is represented in the program's memory using an Adjacency List. Each node acts as a dictionary key pointing to a list of its reachable neighbors, minimizing spatial complexity.


[M1_START]
Start of Markus arc
3.219
0.051
[M2_DEAD]
Markus died before start
[M2_ALIVE]
Markus still alive
2.120
0.562
0.248
3.219
[M3_SIM_DEAD]
Simon died before start
[M3_KISS]
Markus kiss North
[M3_NO_KISS]
Markus didn't kiss North
3.507
0.094
[M4_NO_CON]
Connor failed to go to Jericho
[M4_MET_CON]
Markus met Connor
[M5_MARK_COR]
Markus try to escape through corridor
2.040
0.248
[M6_HELP_AND]
Markus help an android
[M6_RUN]
Markus ignores an android
0.274
4.605
[M7_WIN]
Markus wins against the soldiers
[M7_SHOT]
Markus got shot by the soldiers and wounded
[M8_AND_ESC]
The android escaped
[M7_AND_DEAD]
The android got killed
[M9_NEXT_COR]
Markus go to the next corridor
3.912
1.514
0.416
[M10_TIME_OUT]
[M10_JUMP_SOLD]
Markus jumps the soldiers
[M10_CLIMB]
Markus keep climbing
3.912
[M11_WOUND]
Markus got wounded
[M12_WELL]
Markus enters a well
0.562
1.109
[M13_HELP_AND2]
Markus help an android
[M13_RUN2]
Markus ignores an android
[M14_JOSH]
Josh is fighting with the soldiers
4.605
[M17_BYE_JOSH]
Markus ignores Josh
0.117
[M15_HELP_JOSH]
Markus help JOSH
3.507
[M16_FAIL]
Markus failed to hep Josh, Markus died
0.163
[M17_MACH_CON]
Connor is not a deviant
1.897
[M16_NEUT]
Markus managed to neutralized the soldier
0.288
[M18_BYE_NORTH]
Markus didn't save North
2.526
[M17_NORTH_SHOT]
North got shot
0.236
[M19_DIE_TIME]
Markus died buying time
4.605
[M18_SAVE_NORTH]
Markus saved North
Text
[M19_KILL_SOLD]
Markus killed the soldiers
[M20_MARK_ESC]
Markus and the gang escaped Jerfcho


[K1_START]
Kara goes to Jericho
1.833
0.174
[K15_LUTH_DEAD]
Luther died before start
[K2_LUTH_ALIVE]
Luther still alive
[K15_MARK_DEAD]
Markus died before start
3.219
0.083
[K3_MEET_MARK]
Arrive at Jericho and meet Markus
0.062
3.912
[K4_HUG_ALICE]
Kara hugs Alice
[K4_DISTANT]
Kara stay distant from Alice
3.507
0.030
[K15_NO_CON]
Connor failed to go to Jericho
[K5_KARA_COR]
Kara and the squad trying to escape
[K15_LEAVE_LUTH]
Leave Luther
2.207
[K6_LUTH_SHOT]
Luther got shot
0.386
[K7_HELP_LUTH]
Help Luther
0.342
1.661
[K8_OPEN]
Kara help an android
[K8_CLOSE]
Kara ignore the android
[K9_KARA_FLEE]
Kara and Alice try to flee
4.605
0.117
[K10_FELL]
Kara fail to run
[K10_RUN]
Kara manage to run
1.347
3.219
[K11_RUN_AGAIN]
[K11_SURRENDD]
Kara surrender
[K15_BAD]
Got bad ending
[K12_KARA_SHOT]
Kara got shot in the leg
4.605
4.605
0.528
[K15_BAD]
Got bad ending
[K13_PLAY_DEAD]
Kara and Alice play dead
2.813
0.462
[K14_DEFEND]
Kara defend Alice
[K14_STILL]
Kara remain play dead
[K15_KARA_ESC]
Kara and Alice escaped Jericho



IV. Result and Analysis

A. Execution of Dijkstra's Algorithm

To find the Global Best Ending, the Divide and Conquer strategy was applied to the three character graphs: Connor, Kara, and Markus. Dijkstra's algorithm evaluated the edges by minimizing the cumulative negative log-probability ($W = -\log(P)$). Edges with no explicit branching weights represent direct narrative flows and are assigned a weight of 0.

Below are the traces of the optimal paths identified by the algorithm for each character.

1. Optimal Path for Connor

The algorithm successfully navigated Connor's decision tree to reach the target node [C6_ESC_MARK] (Deviant Connor escapes with Markus). The algorithm effectively avoided the heavy-weighted trap [C3_MACH] (1.715).

Path Sequence: [C1_START] - [C2_MARKUS] ($W=0.094$) - [C3_DEV] ($W=0.261$) - [C4_DEV_WMARK] ($W=0.094$) - [C5_HELP_MARK] ($W=0.416$) - [C6_ESC_MARK] ($W=0.000$).

Total Cumulative Cost: 0.865

2. Optimal Path for Kara

For Kara, the algorithm found the safest route to [K15_KARA_ESC] by meticulously balancing survival decisions. The algorithm bypassed high-risk routes such as [K10_FELL] (4.605) and successfully calculated that playing dead was mathematically cheaper than running again.

Path Sequence: [K1_START] - [K2_LUTH_ALIVE] ($W=0.174$) - [K3_MEET_MARK] ($W=0.083$) - [K4_HUG_ALICE] ($W=0.062$) - [K5_KARA_COR] ($W=0.030$) - [K6_LUTH_SHOT] ($W=0.000$) - [K7_HELP_LUTH] ($W=0.386$) - [K8_OPEN] ($W=0.342$) - [K9_KARA_FLEE] ($W=0.000$) - [K10_RUN] ($W=0.117$) - [K13_PLAY_DEAD] ($W=0.528$) - [K14_STILL] ($W=0.462$) - [K15_KARA_ESC] ($W=0.000$).

Total Cumulative Cost: 2.184

3. Optimal Path for Markus

Markus's graph possesses the deepest state-space. The algorithm optimally traced the path to [M20_MARK_ESC], minimizing risk by consistently choosing to save his companions, as these popular choices yielded significantly lower Dijkstra weights compared to abandoning them.

Path Sequence: [M1_START] - [M2_ALIVE] ($W=0.051$) - [M3_KISS] ($W=0.248$) - [M4_MET_CON] ($W=0.094$) - [M5_MARK_COR] ($W=0.000$) - [M6_HELP_AND] ($W=0.248$) - [M7_WIN] ($W=0.274$) - [M8_AND_ESC] ($W=0.000$) - [M9_NEXT_COR] ($W=0.000$) - [M10_CLIMB] ($W=0.416$) - [M12_WELL] ($W=0.000$) - [M13_HELP_AND2] ($W=0.562$) - [M14_JOSH] ($W=0.000$) - [M15_HELP_JOSH] ($W=0.117$) - [M16_NEUT] ($W=0.163$) - [M17_NORTH_SHOT] ($W=0.288$) - [M18_SAVE_NORTH] ($W=0.236$) - [M19_KILL_SOLD] ($W=0.000$) - [M20_MARK_ESC] ($W=0.000$).

Total Cumulative Cost: 2.697

B. Synthesis of the Global Best Ending

Based on the Divide and Conquer framework, the total minimal cost required to achieve the Global Best Ending where all three protagonists survive and accomplish their optimal objectives is the sum of their individual path costs:

$$W_{total} = W_{Connor} + W_{Kara} + W_{Markus}$$

$$W_{total} = 0.865 + 2.184 + 2.697 = 5.746$$

By converting this total weight back into a cumulative probability using the inverse logarithmic function (P = e^{-W_{total}}), it is mathematically proven that traversing the absolute perfect narrative path in the "Crossroads" chapter has an exact sequence probability of 0.00319 (or approximately 0.32%).

This demonstrates that while individual decisions may seem highly probable, perfectly chaining dozens of correct decisions across three intersecting characters without making a single fatal error is statistically rare, validating the immense narrative complexity of *Detroit: Become Human* and the effectiveness of Dijkstra's algorithm in solving it.

V. Conclusion

This study successfully demonstrates the application of Dijkstra's algorithm in navigating the massive state-space of the "Crossroads" chapter in Detroit: Become Human. By

transforming the traversal probabilities into negative logarithms, the algorithm circumvented the arithmetic underflow problem and accurately identified the mathematically optimal narrative path.

The Divide and Conquer approach revealed that achieving the Global Best Ending—where Connor, Kara, and Markus all survive and fulfill their optimal arcs—requires a precise sequence of decisions with a cumulative probability of merely 0.32%. This extremely low probability highlights the intense branching complexity of interactive narratives. Ultimately, this research proves that discrete mathematics and graph theory provide a robust and highly effective framework for computationally solving and optimizing complex gameplay strategies.

ACKNOWLEDGMENT

The author would like to express the deepest gratitude to God Almighty for the blessing and grace. Sincere appreciation is extended to the author's parents for their endless support. Special thanks to Dr. Ir. Rinaldi Munir, M.T., the lecturer for the Discrete Mathematics course, for providing the foundational knowledge of graph theory that made this research possible. Finally, the author thanks all peers and friends who offered valuable discussions and encouragement throughout the completion of this paper.

REFERENCES

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
[2] R. Munir, "Graf - Bagian 1 & 2," Course Materials for IF1220 Discrete Mathematics, Institut Teknologi Bandung, 2024.
[3] Quantic Dream, *Detroit: Become Human*. Paris: Sony Interactive Entertainment, 2018.

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

Andro Irsa Syafiq - 13525129