Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th February 2025. Vol. 4. No. 2; e-ISSN: 2808-4519

Implementation of Dijkstra Algorithm in Determining the Fastest Route for Goods Delivery

Ubaidillah Irganata Putra Mamba'ul Ulum[1*], Indyah Hartami Santi[2], Mukh. Taofik Chulkamdi[3]

[1,2,3]Teknik Informatika, Universitas Islam Balitar
Jl. Majapahit No.2- 4, Sananwetan, Kec. Sananwetan, Kota Blitar, Jawa Timur
ubaidillahirganataputra@gmail.com [1]*

Abstract

Goods delivery is a critical component in business, especially for logistics industries such as J&T Express Gedog, Blitar, which serves areas in Kecamatan Kepanjen Kidul, Kanigoro, Nglegok, and Garum. During the delivery process, couriers often face difficulties in determining the fastest route, particularly for those who are not familiar with the delivery areas. These challenges often lead to getting lost and prolonged delivery times. This research aims to implement the Dijkstra algorithm to determine the fastest route for goods delivery. The road network is modeled as a weighted graph, where nodes represent delivery locations and edges represent the distances between nodes. The Dijkstra algorithm is then used to process this data and find the fastest route from the starting point to the destination. The testing results show that the algorithm provides the shortest route in 90% of the 10 delivery scenarios tested, with an average distance saving of 8.55%. Therefore, the Dijkstra algorithm proves to be effective in optimizing goods delivery routes, improving time and distance efficiency during the delivery process.

Keywords: *Dijkstra Algorithm, Fastest Route, Goods Delivery*

1. Introduction

The delivery of goods is one of the most crucial activities in business and trade. In the shipping industry, efficiency and speed are essential to staying competitive in the market. J&T Express is a logistics service provider that specializes in shipping both documents and packages. This company offers the advantage of package pickup services. Shipping services are highly needed by the public, and couriers act as the main link in ensuring that packages reach consumers within the estimated timeframe. One such branch is J&T Express Gedog, located in Sananwetan District, Blitar City. Since it began operations in December 2019, this branch has remained popular due to its strategic location on the outskirts of Blitar City. Currently, J&T Express Gedog employs 34 couriers, covering delivery areas including Kepanjen Kidul District, Kanigoro District, Nglegok District, and Garum District.

Finding the shortest route is a common issue in logistics, as it helps determine the most efficient travel distance [1]. The increasing interest in online shopping has led to a surge in demand for shipping services. Based on interviews conducted with the supervisor of J&T Express Gedog in Blitar and several couriers, one of the key challenges in package delivery is locating the recipient's address. Couriers sometimes lack knowledge about the routes they need to take, causing them to get lost, which leads to delays in package delivery.

Several obstacles affect the ability to find the fastest route, including narrow alleyways in residential areas, traffic congestion, and unforeseen road conditions. New couriers who are unfamiliar with the local routes often face limited information, leading to route selection errors. One common mistake is when couriers have to return to previously passed locations due to a lack of information about the next recipient's nearest location. As a result, they may waste a significant amount of time searching for the next destination, even when it is close to their current location.

This inefficiency negatively impacts customer satisfaction and increases distribution costs, particularly fuel expenses, due to inefficient route selection. Therefore, this study employs Dijkstra's algorithm to assist in determining the fastest route for J&T Express deliveries. Dijkstra's algorithm is a well-known method for finding the shortest or fastest path in a graph [2]. In the context of package delivery, this algorithm can help optimize route selection by considering distance and weight for each available path [3]. By implementing Dijkstra's algorithm in J&T Express, the company can enhance delivery efficiency and speed while reducing overall shipping costs. Consequently, this research aims to implement Dijkstra's algorithm at J&T Express and analyze its performance in determining the fastest delivery routes.

2. Literature Review

2.1. Dijkstra's Algorithm

This algorithm was developed by Edsger W. Dijkstra and published in 1959 in the journal Numerische Mathematik under the title "A Note on Two Problems in Connexion with Graphs." It is often described as a greedy algorithm [4], as illustrated in the book Algorithmics [5]. Dijkstra's algorithm is one of the most popular optimization algorithms for solving shortest path problems in a weighted graph. The algorithm finds the shortest path from a given source vertex to a target vertex in a weighted graph, where all weights are positive. If negative weights are present, the algorithm will return an infinite (∞) value.

Dijkstra's algorithm operates by considering nodes and utilizing a directed graph to determine the shortest route.

2.2 Graph

According to [6], a graph is one of the most commonly used data structures. It is used to represent discrete objects and their relationships. The maximum distance from a node to all other nodes in the graph is known as the eccentricity of that node. See Figure 2.6 below for illustration.

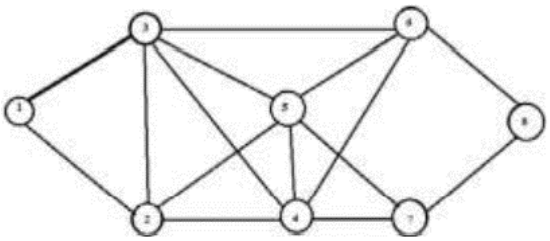

Fig. 1: Illustration Algorithm Dijkstra in Graph

A node with a smaller eccentricity is considered the center of the graph [7]. The minimum eccentricity value among all nodes is regarded as the radius of a connected graph. The functions and uses of graphs include:

1. Graphs are used to represent computational flows.
2. They are utilized in graphical modeling.
3. Graphs are applied in operating systems for resource allocation.
4. Google Maps uses graphs to determine the shortest route.
5. Graphs are used in aviation systems to optimize effective routes.
6. In state-transition diagrams, graphs represent states and their transitions.
7. In circuits, graphs can represent circuit points as nodes and wires as edges.
8. Graphs are used to solve puzzles with only one solution, such as mazes.

3. Research Method

3.1. Research Time and Location

This research was conducted from February to August 2023, covering data collection until the final report preparation, with a total duration of approximately seven months. The research location is J&T Express, Gedog Subdistrict, Sananwetan District, Blitar City.

3.2. Research Type

This study employs a quantitative research method. Quantitative research is defined as a systematic investigation of phenomena by collecting data that is then measured using mathematical, statistical, or computational techniques [8]. Based on the explanation provided in this research, the subject of the study is J&T Express shipping data, including destination addresses. Meanwhile, the research object is the package delivery process utilizing Dijkstra's algorithm, with the final output being the fastest delivery route.

3.3. Data Collection Techniques

3.3.1. Observation

This activity is conducted to analyze the real conditions in the research location, allowing for in-depth information gathering regarding the research object. The obtained results include data on the delivery area coverage and incoming package data for June 2023.

Fig. 2: Observation

3.3.2. Interview

This method was conducted to obtain the necessary information regarding the research topic, which is determining the shortest delivery route for J&T Express Gedog. The interview was carried out through a direct question-and-answer session with the **Supervisor of J&T Express Gedog**.

Figure 3. Interview Documentation

During the interview, the researcher asked several questions to the **Supervisor of J&T Express Gedog**. Below is the list of interview questions presented in **Table 1**.

Table 1: Interview Questions

No	Question
1	Since when has J&T Express Gedog been operating?
2	Are there any challenges in determining the delivery route for couriers?
3	Have there been any efforts to determine the shortest delivery route at J&T Express Gedog?

Table 2: Interview Responses

No	Answer
1	J&T Express Gedog started operating in **December 2019**.
2	Some challenges encountered include **delayed deliveries** due to recipients being unreachable, making it difficult for couriers to locate the recipient's house. Additionally, during special events, couriers often carry **too many packages**, leading to confusion about **which package should be delivered first**.
3	The efforts taken include **contacting recipients first** to confirm their location and using **Google Maps** to navigate.

3.4. Types of Data

1. Primary Data
   Primary data is the main data obtained directly through observations and interviews at J&T Express Gedog.
2. Secondary Data
   Secondary data is obtained from documentation or other sources, such as books, previous research with a similar focus, and other references.

3.5. Data Collection Instruments

The data collection instrument used in this research serves as a guideline to facilitate data collection. The process begins with observations at the research site, followed by interviews to gather relevant information. The data collected during the interview process is then analyzed. Below is a summary of the data collection instruments in Table 3.

Table 3: Data Required in the Research

No	Requirement	Availability
1	Interview Question List	Available
2	Delivery Coverage Data	

Below is a table showing the delivery coverage areas of J&T Express Gedog.

Table 4: Delivery Coverage Areas

District	Subdistricts
Kepanjen Kidul	Bendo, Kauman, Kepanjen Kidul, Kepanjen Lor, Ngadirejo, Sentul, Tanggung
Kanigoro	Sentul, Tanggung, Gogodeso, Jatinom, Kanigoro, Karangsono, Kuningan, Minggirsari, Papungan, Satreyan, Sawentar, Tlogo
Nglegok	Bangsri, Dayu, Jiwut, Kedawung, Kemloko, Krenceng, Modangan, Nglegok

Table 5: Continued

District	Subdistricts
Nglegok	Ngoran, Penataran, Sumberasri
Garum	Bence, Garum, Karangrejo, Pojok, Sidodadi, Slorok, Sumberdiren, Tawangsari, Tingal

3.6. Research Stages

This research follows several stages to ensure a structured workflow, as described below.
Figure 4. Research Stages

1. Planning
   The research begins with a field study at J&T Express Gedog, where observations and problem identification are carried out. The researcher then determines the appropriate steps to address the identified issue.
   J&T Express Gedog was chosen as the research subject because no previous efforts have been made to determine the shortest package delivery route. This statement is supported by interviews with the Supervisor of J&T Express Gedog.
2. Literature Review
   After identifying the problem, a literature review is conducted to find references related to similar cases. The next step is analyzing the appropriate algorithm for this research. Based on the literature review, Dijkstra's Algorithm is found to be the most relevant for this study [9].
3. Data Collection
   This stage involves collecting all necessary data and information for the research. The collected data is then processed using Dijkstra's Algorithm. Dijkstra's Algorithm works by finding the smallest weight in a weighted graph. The shortest distance is determined between two or more nodes. The final total distance is the smallest value among all possible routes. A literature study is also conducted to gather references that help in solving the shortest route problem in package delivery.
4. Implementation of Dijkstra's Algorithm
   In this stage, the fastest delivery route is calculated based on the collected data. The process follows a flowchart-based approach, where: Each point (location) and its respective weight (distance) are defined. The shortest path is calculated using Dijkstra's Algorithm.
5. Testing and Results
   This section describes the testing process used to evaluate the efficiency of Dijkstra's Algorithm in determining the shortest delivery route compared to the manual routes taken by couriers. The testing includes: Distance Measurement: Comparing the distance covered by couriers manually and the distance calculated using Dijkstra's Algorithm. Success Criteria: The test is considered successful if Dijkstra's Algorithm provides a shorter route than the manual route taken by couriers. Efficiency Calculation Formula: Distance Efficiency (%) = ((Courier Distance - Dijkstra Distance) / Courier Distance) × 100%

4. Result and Discussion

4.1. Results

Dijkstra's Algorithm is used in this research to determine the fastest courier route for package delivery. The input for this route search consists of a starting node and a destination node. In the first step, the courier is required to manually enter the starting point and destination. The Dijkstra Algorithm processes these input nodes by calculating the cost function values and the heuristic values of nodes included in the open set.
The open set consists of nodes that can be traversed. After performing the calculations, the algorithm determines the optimal route, which is then displayed on a map as the selected path using Dijkstra's Algorithm.
The first stage of data collection involves observing and gathering destination package addresses. The researcher then finds the coordinate points of those addresses.
The next step is to determine the distances between coordinate points.
Once the data is collected, the next step is to map the destination coordinates based on field observations. Figures 3 and 4 display the destination package coordinates.

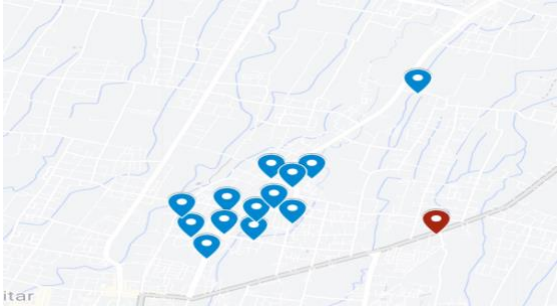

Fig. 3: The recipient's package address in Kepanjen Kidul.

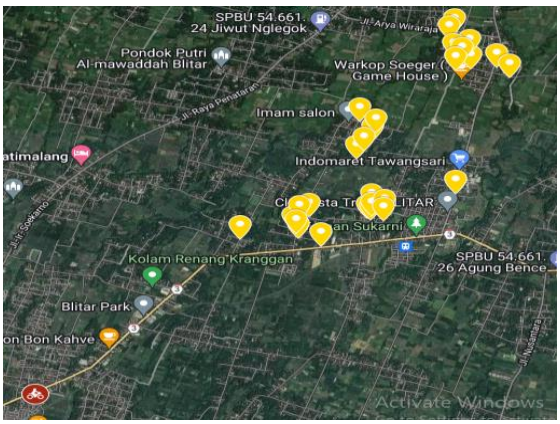

Fig. 4: The recipient's package

At this stage, the process of finding the fastest route is carried out based on the collected data. The Dijkstra algorithm is then applied using a flowchart-based approach.
In the implementation phase, the researcher starts by defining nodes and the weight of each distance.

```
def dijkstra(graph, start, finish):
    # Initialize the initial distance of all nodes as infinity
    distances = {node: float('inf') for node in graph}

    # Shortest path from the start node to other nodes
    path = {node: [] for node in graph}

    # Set the distance from the start node to itself as 0
    distances[start] = 0
```

The process of determining the fastest route is carried out using Google Colab Python, as follows:

a. def dijkstra(graph, start, finish): Defines the Dijkstra function, which takes three parameters:
   graph: The graph to be used.
   start: The starting node.
   finish: The destination node.
   This function returns both the shortest path and the shortest distance between the start node and the finish node within the graph.
   distances = {node: float('inf') for node in graph}: Initializes the initial distance of all nodes in the graph as infinity.
   The distances variable is a dictionary, where nodes are keys, and their distances are values.
b. path = {node: [] for node in graph}: Initializes the shortest path from the start node to all other nodes. The path variable is initialized as an empty list for each node. This path will be updated as the algorithm progresses to store the shortest route from the start node to every other node. The distance from the start node to itself is set to 0 because the distance to itself is always zero. The algorithm then proceeds to search for the shortest path and shortest distance from the start node to the destination node.

```
While graph:
   Min node = None
   For node in graph:
         If min_node is None:
                Min_node = node
         Elif distances[node] < distances[min_node]:
                Min_node = node
   If distances[min_node] == float('inf'):
         break
```

c. During the "while loop", the program searches for the node with the shortest distance in each iteration.
   The loop continues as long as there are unprocessed nodes in the graph (i.e., the graph variable is not empty).
d. At the beginning of each iteration, the min_node variable is initialized as None.
   This variable is used to store the node with the shortest distance from the start node during the iteration.
e. Inside the "for loop", the program checks each node in the graph to find the node with the shortest current distance.
   First, min_node is assigned the first encountered node in the graph.
f. The program compares the distance of other nodes (distances[node]) with the distance of min_node (distances[min_node]).
   If a node has a smaller distance than min_node, the min_node is updated to that node.
   This process continues until the program finds the node with the shortest distance at the current stage.
g. If the distance of min_node is still infinity (float('inf')) after the iteration, it means there are no more reachable nodes from the start node.
   In this case, the algorithm exits the while loop using the break command. This happens when there are no more nodes left to process, meaning the algorithm has successfully found the shortest path.

```
for neighbor, weight in graph[min_node].items():
    new_distance = distances[min_node] + weight
    if new_distance < distances[neighbor]:
        distances[neighbor] = new_distance
        path[neighbor] = path[min_node] + [min_node]

graph.pop(min_node)
```

```
shortest_path = path[finish] + [finish]
shortest_distance = distances[finish]
return shortest_path, shortest_distance
```

h. At each iteration, after finding the node with the shortest distance and storing it in the min_node variable, the program iterates through the neighbors of min_node to update the shortest distance and shortest path.
The "for loop" iterates through each neighbor of min_node, along with the weight of the edge connecting them.

i. The variable new_distance is calculated as the new distance from the start node to that neighbor.
The new distance is obtained by adding the weight of the edge connecting min_node to the neighbor.If the new_distance is smaller than the current distance (distances[neighbor]) from the start node to the neighbor, then:The shortest distance (distances[neighbor]) is updated to new_distance.The shortest path (path[neighbor]) is updated to include min_node, leading to the neighbor.

j. After processing all neighbors of min_node, and updating their shortest distances and paths,
The min_node is removed from the graph using graph.pop(min_node). This is done because min_node has already been processed and no longer needs to be considered in future iterations.

k. Once the "while loop" ends, the program will have:
The shortest path from the start node to the destination node (finish).The shortest distance, stored in the path and distances variables.Finally, the program returns the shortest path and shortest distance from the start node to the destination node using the return statement.

```
    'N16': {'N17': 200},
    'N17': {'N18': 190},
    'N18': {'N19': 140},
    'N19': {'N20': 300, 'N21': 500},
    'N20': {'N21': 100},
    'N21': {'N22': 350},
    'N22': {'N23': 230, 'N27': 1700},
    'N23': {'N24': 1000},
    'N24': {'N25': 100},
    'N25': {'N26': 150},
    'N26': {'N27': 160},
    'N27': {'N26': 160, 'N22': 1700}
}

start_node = 'A'
finish_node = 'N27'
```

Fig. 5: Updating shortest path dan shortest distance

l. The graph is defined as a dictionary with nodes such as 'A', 'N1', 'N2', ..., 'N27' as the keys and the weights between nodes as the values.
   a) This graph represents a series of nodes connected by weighted edges.
   b) Each node has neighbors with associated weights.
   c) For example, node 'A' has a neighbor 'N1' with a weight of 2700, node 'N1' has neighbors 'N2' with a weight of 800 and 'N3' with a weight of 1100, and so on.

m. Next, the program will execute the Dijkstra algorithm to find the shortest path and shortest distance from node 'A' (start node) to node 'N27' (finish node) in the graph.

```
shortest_path, shortest_distance = dijkstra(graph, start_node, finish_node)
print(f"Jalur terpendek dari {start_node} ke {finish_node}: {shortest_path}")
print(f"Jarak terpendek: {shortest_distance}")
```

Fig 6: Print Result

n. First, the program calls the function "dijkstra(graph, start_node, finish_node)" with the parameters:
   a) "graph": the weighted graph,
   b) "start_node": the starting node,
   c) "finish_node": the destination node.
   This function will return the shortest path and shortest distance from the starting node to the destination node.
   The results of the shortest path and shortest distance from the dijkstra function are stored in the variables "shortest_path" and "shortest_distance".
   The program then prints the results of the shortest path and shortest distance using the "print" statement.

```
Jalur terpendek dari A ke N27: ['A', 'N1', 'N2',
Jarak terpendek: 10734
```

Fig 7: Results of Shortest Route Search with Python

The results of testing between the actual route taken by the courier and the optimal route calculated using the Dijkstra algorithm. The testing was conducted on 10 couriers delivering goods in four regions: Garum, Kanigoro, Kepanjen Kidul, and Nglegok. The comparison was made based on the distance traveled and the time spent on the actual route and the route generated by the Dijkstra algorithm.
Below are the route results for 5 couriers:

Table 6: Router Courier

No	Region	Actual Route	Actual Distance (km)	Dijkstra Route	Dijkstra Distance (km)	Distance Efficiency (%)
1	Kanigoro	A-N1-N2-N4-N3-N5-N6-N7-N8-N9-N10	18.42	A-N1-N3-N4-N5-N6-N7-N8-N9-N2-N10	18.5	-0.43
2	Kanigoro	A-N1-N2-N3-N4-N5-N6-N7-N8-N9-N10	7.885	A-N2-N1-N3-N4-N5-N6-N7-N8-N9-N10	7.195	8.75

No	Region	Actual Route	Actual Distance (km)	Dijkstra Route	Dijkstra Distance (km)	Distance Efficiency (%)
3	Garum	A-N1-N3-N5-N6-N7-N9-N10	11.2	A-N1-N2-N4-N6-N7-N9-N10	9.15	18.30
4	Kepanjen Kidul	A-N5-N2-N1-N3-N4-N6-N7-N8-N9-N10-N11-N12-N13	5.64	A-N1-N3-N4-N6-N7-N8-N9-N2-N5-N10-N11-N12-N13	4.96	12.06
5	Nglegok	A-N1-N4-N3-N5-N2-N6-N9-N10-N12-N11-N13-N7-N8	14.68	A-N2-N1-N4-N3-N5-N6-N9-N10-N12-N11-N13-N7-N8	14.22	3.13

From the table above, it can be seen that the average efficiency across all routes is 8.55%. This shows that, on average, couriers travel 8.55% further compared to the shortest route calculated using the Dijkstra algorithm.

4.2. Discussion

The test results show that the Dijkstra algorithm consistently demonstrates higher efficiency in determining the optimal route for parcel delivery compared to the manual route chosen by couriers. The average distance savings reached 8.55%, depending on the geographical conditions and the testing scenarios used. These findings highlight the advantage of the Dijkstra algorithm in reducing travel distance and improving logistics efficiency, as emphasized by comprehensive studies like [10], which underscores its significant potential for route optimization.

1. Route Selection Based on Optimal Distance Weights; The Dijkstra algorithm is designed to find the shortest route between two points in a network based on distance weights. Each node in the road network has a weight value that represents the distance or travel time. The algorithm systematically evaluates each possible route and selects the path with the smallest total weight, ensuring that the generated path is the most optimal in terms of travel distance. In contrast, couriers often use routes they intuitively know or based on experience, which may not be the shortest objectively. Couriers may overlook smaller roads or alternative paths that are less familiar but actually shorter in distance.
2. Consistency in Route Selection; The Dijkstra algorithm processes routes consistently based on available map data and distance, unaffected by subjective factors such as habits, personal preferences, or past experiences. Meanwhile, couriers may face factors such as the habit of choosing larger or perceived faster roads, even if they are objectively longer. When the algorithm is implemented, all routes are optimized evenly, resulting in consistent distance savings, as seen in testing scenarios in urban and suburban areas.

5. Conclusion

The Dijkstra algorithm has been successfully implemented to determine the fastest route in a road network for parcel delivery. The implementation process involves modeling the road network as a weighted graph, where nodes represent delivery locations, and edges represent the distances between nodes. The Dijkstra algorithm is capable of processing this data and producing the fastest route from the starting point to the destination, considering all possible paths. The testing results show that this algorithm delivers the shortest route 90% of the time in 10 delivery scenarios.

Testing the efficiency of the Dijkstra algorithm in determining the fastest route demonstrates that it is more efficient than the manual routes chosen by couriers. In terms of distance, the Dijkstra algorithm is able to save an average of 8.55% in travel distance and provides the shortest route in 90% of the 10 tests. Therefore, the Dijkstra algorithm has proven effective in optimizing parcel delivery routes. Future research can evaluate the performance of the Dijkstra algorithm in more complex scenarios, such as parcel delivery with a larger number of locations or adding considerations for other factors like traffic conditions, fuel costs, or delivery times. In this case, an efficient and reliable approach will be necessary to ensure that the Dijkstra algorithm remains effective in finding the fastest route in more realistic and complex situations.

References

[1] N. A. Sudibyo, P. E. Setyawan, and Y. P. S. R. Hidayat, "Implementasi Algoritma Dijkstra dalam Pencarian Rute Terpendek Tempat Wisata di Kabupaten Klaten," Riemann Res. Math. Math. Educ., vol. 2, no. 1, pp. 1–9, 2020.
[2] A. Cantona, F. Fauziah, and W. Winarsih, "Implementasi Algoritma Dijkstra Pada Pencarian Rute Terpendek ke Museum di Jakarta," J. Teknol. dan Manaj. Inform., vol. 6, no. 1, pp. 27–34, 2020, doi: 10.26905/jtmi.v6i1.3837.
[3] K. Hermanto and T. D. Ermayanti, "Analisa Optimasi Rute Transportasi Antar Jemput Siswa Menggunakan Model CGVRP dan Algoritma Dijkstra di SDIT Darus Sunnah," UJMC (Unisda J. Math. Comput. Sci., vol. 5, no. 2, pp. 19–28, 2019.
[4] E. W. Dijkstra, "A note on two problems in connexion with graphs," in Edsger Wybe Dijkstra: his life, work, and legacy, 2022, pp. 287–290.
[5] G. Brassard and P. Bratley, Fundamentals of algorithmics. Prentice-Hall, Inc., 1996.
[6] S. Baco, R. Syarifuddin, N. I. B, N. Samsul Alan, and Sulfikri, "Perancangan Aplikasi Objek Wisata Alam Di Kabupaten Maros Menggunakan Karakter 3D Adobe Premiere Pro Dengan Algoritma Dijkstra," J. Teknol. dan Komput., vol. 2, no. 02, pp. 172–177, 2022, doi: 10.56923/jtek.v2i02.103.
[7] R. Diestel, Graph theory. Springer (print edition); Reinhard Diestel (eBooks), 2024.
[8] H. Syahrizal and M. S. Jailani, "Jenis-jenis penelitian dalam penelitian kuantitatif dan kualitatif," QOSIM J. Pendidikan, Sos. \& Hum., vol. 1, no. 1, pp. 13–23, 2023.
[9] E. Ismantohadi and I. Iryanto, "Penerapan Algoritma Dijkstra Untuk Penentuan Jalur Terbaik Evakuasi Tsunami--Studi Kasus: Kelurahan Sanur Bali," JTT (Jurnal Teknol. Ter., vol. 4, no. 2, pp. 72–78, 2018.
[10] D. R. Wijaya, A. Athallah, T. N. Noor'afina, P. A. Telnoni, and S. D. Budiwati, "Cargo Route Optimization Using Shortest Path Algorithms: Runtime and Validity Comparison," J. Comput. Sci., vol. 19, no. 11, pp. 1369–1379, 2023, doi: 10.3844/jcssp.2023.1369.1379.