Diterima : 12 Januari 2025
Disetujui : 17 Januari 2025
Hal : 74-81

Al-Aqlu: Jurnal Matematika, Teknik dan Sains
https://jurnal.yalamqa.com/index.php/aqlu
e-ISSN : 2985-4369
Vol. 3 , No. 1, Januari 2025

OPTIMASI JALUR PERJALANAN ANTARA JAKARTA DAN SURABAYA MENGGUNAKAN ALGORITMA DIJKSTRA

[Optimizing Travel Routes Between Jakarta and Surabaya Using the Dijkstra Algorithm]

Deddy Rahmadi[1)]*, Tarisha Najwa Putri[2)], Derrida Atsmarotul Ilmi[3)], Sinta Miftakhul Rohmah[4)] , Dela Nuraini Safinka[5)]

UIN Sunan Kalijaga Yogyakarta

deddy.rahmadi@uin-suka.ac.id (corresponding)

ABSTRAK

Algoritma Dijkstra merupakan metode populer untuk menentukan jalur terpendek dalam graf berbobot positif dan digunakan dalam berbagai aplikasi seperti sistem navigasi dan jaringan komputer. Penelitian ini menganalisis efektivitas algoritma Dijkstra dalam mencari rute terpendek antar kota di Indonesia, dengan menggunakan contoh penerapan antara Kota Jakarta dan Surabaya. Setiap kota dipandang sebagai simpul dan setiap jalan sebagai sisi dalam graf, dengan jarak atau waktu tempuh sebagai bobot. Implementasi dilakukan dengan bahasa pemrograman Python menggunakan data sekunder yang terdiri dari 20 kota. Hasil penelitian menunjukkan bahwa algoritma Dijkstra mampu menentukan jalur terpendek antara dua kota dengan jarak terpendek yang ditemukan, yakni 664,08 km antara Jakarta dan Surabaya. Studi ini menggarisbawahi pentingnya pengoptimalan algoritma Dijkstra agar lebih efisien dalam jaringan besar, serta aplikasinya yang luas untuk efisiensi di berbagai bidang, termasuk transportasi dan jaringan komputer.

Kata kunci: *Algoritma; dijkstra; graf; jarak; program*

ABSTRACT

The Dijkstra algorithm is a popular method for determining the shortest path in a positively weighted graph and is widely used in various applications such as navigation systems and computer networks. This study analyzes the effectiveness of the Dijkstra algorithm in finding the shortest route between cities in Indonesia, using an example application between Jakarta and Surabaya. Each city is represented as a node and each road as an edge in the graph, with distance or travel time as the weight. The implementation is carried out in Python, using secondary data comprising 20 cities. The results indicate that the Dijkstra algorithm is capable of determining the shortest path between two cities, with the shortest distance found being 664.08 km between Jakarta and Surabaya. This study highlights the importance of optimizing the Dijkstra algorithm for greater efficiency in large networks, as well as its broad applications for improving efficiency in various fields, including transportation and computer networks.

Keywords: *Algorithm; dijkstra; graph; distance; program*

PENDAHULUAN

Di era perkembangan digital yang semakin pesat, sistem informasi navigasi, jaringan komputer, dan berbagai aplikasi berbasis graf lainnya juga semakin berkembang. Permasalahan pencarian jalur terpendek menjadi hal yang sangat penting, karena merupakan dasar untuk efisiensi dalam berbagai aplikasi

nyata yang melibatkan jaringan, pemetaan, dan navigasi. Salah satu metode yang sering digunakan untuk menyelesaikan masalah ini adalah metode dijkstra, yang memungkinkan pencarian jalur terpendek dari satu titik ke titik lainnya dalam graf berbobot positif. Namun kekurangan dari implementasi dijkstra adalah efisiensi waktu dan kompleksitas algoritma dalam menghadapi graf dengan jumlah simpul dan sisi yang besar.

Algoritma dijkstra telah terbukti menjadi solusi dalam permasalahan pada graf berbobot positif. Dalam penelitian ini, wawasan utama yang akan dikaji meliputi pengertian dijkstra, algoritma dijkstra dalam megolah data, program komputer mengenai dijkstra, implementasi studi kasus kedalam program dijkstra berikut dengan pembahasannya dalam optimisasi dan maps, serta kelebihan dan kekurangan cara kerja dari sistem dijkstra.

Penelitian ini bertujuan untuk menganalisis efektivitas dan efisiensi algoritma dijkstra dalam menyelesaikan permasalahan pencarian jalur terpendek pada graf berbobot positif, mengidentifikasi faktor-faktor yang mempengaruhi algoritma dijkstra pada graf dengan berbagai ukuran dan kompleksitas, mengkaji dan mengimplementasikan optimasi algoritma untuk meningkatkan efisiensi dalam penerapannya pada graf besar dan optimisasi dalam maps, dan membandingkan kinerja algoritma dijkstra dengan algoritma alternatif dalam hal akurasi dan efisiensi.

Algoritma dijkstra ditemukan pada tahun 1956 oleh ilmuan komputer Belanda Edsger W. Dijkstra saat rehat minum kopi selama dua puluh menit, saat berbelanja dengan tunangannya di Amsterdam(Girsang, 2017). Algoritma dijkstra menemukan jalur terpendek dari satu titik sudut ke semua titik sudut lainnya. Hal itu dilakukan dengan berulang kali memilih titik simpul terdekat yang belum di lalui dan menghitung jarak ke semua titik simpul yang belum dikunjungi. Djisktra berfokus pada teori algoritma dan struktur data, khususnyya dalam pengembangan algoritma yang efisien dan sederhana. Djisktra juga berperan penting dalam prinsip dasar ilmu komputer, yaitu struktur data dan logika matematika untuk optimisasi algoritma. Disana ditekankan pentingnya koreksi formal dalam pengembangan perangkat lunak, mendorong penggunaan metode yang ketat dan terstruktur untuk memastikan bahwa algoritma yang dirancang bekerja sebagaimana mestinya. Djikstra memperkenalkan beberapa prinsip fundamental yang hingga kini menjadi dasar teori diantaranya: Algoritma jalur terpendek (memilih titik terdekat dari titik awal dan memperbaharui jarak setiap titik dengan yang lain secara berulang), program correctness (menghindari kesalahan sebelum implementasi, dan mengurangi resiko), dan pemrograman struktural (mendorong penggunaan struktur kontrol yang jelas dan menghindari penggunaan goto statement)(Al Hakim et al., 2021). Djikstra menekankan pentingnya pendekatan formal dalam desain algoritma dan struktur data, menggabungkan prinsip matematika dan logika untuk memastikan efisiensi, dan kebenaran dalam pemrograman. Dalam kajiannya, Rahmadi dkk menjelaskan bahwa algoritma Dijkstra sangat relevan untuk masalah jalur terpendek pada graf berbobot positif, tetapi tidak cocok untuk MST karena konsepnya tidak melibatkan jalur antar simpul melainkan pembentukan pohon yang mencakup semua simpul. Sebaliknya, algoritma seperti Kruskal dan Prim lebih tepat untuk menyelesaikan masalah MST, di mana prioritas utama adalah memilih sisi dengan bobot terkecil sambil menghindari pembentukan siklus.

Dengan adanya penelitian ini, agar algoritma dijkstra dapat membantu menentukan jalur terpendek atau tercepat dalam menentukan rute data yang optimal sehingga dapat meningkatkan kecepatan kinerja dalam berbagai bidang. Dan juga diharapkan mampu mengembangkan metode optimasi pada algoritma dijkstra sehingga lebih cepat dan efisien, bahkan dalam jaringan yang besar dan kompleks seperti jaringan kota dan jaringan komputer. Semoga penelitian ini dapat menjadi dasar penelitian selanjutnya, dan dapat di implementasikan ke hal lainnya untuk meningkatkan keamanan dan dampak yang signifikan dalam berbagai sektor bidang.

METODE PENELITIAN

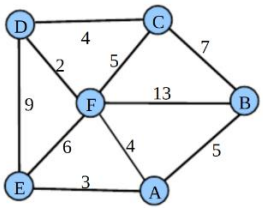
Penelitian ini menggunakan metode penelitian kuantitatif dengan pendekatan komputasional dan algoritmik karena berfokus pada penerapan algoritma djikstra dalam skenario rute terpendek antar kota dengan node. Dalam hal ini terjadi proses sistematis yang menggabungkan pemodelan matematis, simulasi program, dan analisis kuantitatif.

Setiap kota dalam jaringan dianggap sebagai simpul (node), dan setiap jalan yang menghubungkan dua kota adalah sisi (edge) dalam graf. Bobot pada sisi merepresentasikan jarak atau waktu tempuh antar

kota. Sampel yang diambil yakni data sekunder dengan menggunakan teknik simple random sampling yang bersumber dari internet dengan total data 20 kota di Indonesia dengan data berisi nama kota, garis lintang, dan garis bujur. Data diubah menjadi bentuk graf yang merepresentasikan jaringan kota, dengan simpul untuk kota dan sisi untuk jalan yang menghubungkannya. Algoritma dijkstra di implementasikan menggunakan program phython untuk mencari hasil rute terpendek yang ditemukan dengan data sekunder.

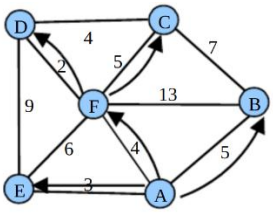
Ilustrasi Graf

1. Graf tak berarah



Gambar 1. Graf Tak Berarah

Solusi dalam bentuk visual :

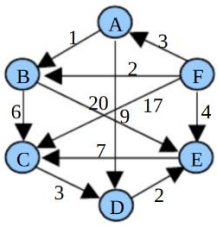


Gambar 2. Solusi Graf Tak Berarah Dalam Bentuk Visual

Penyelesaian jalur terpendek untuk simpul A pada graf adalah :

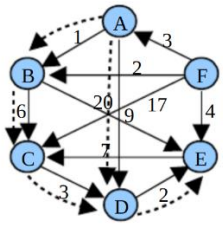
a. Tujuan : simpul B
   Lintasan terpendeknya yaitu simpul A langsung menuju simpul B dengan jarak 5.
b. Tujuan : simpul C
   Lintasan terpendeknya yaitu simpul A menuju simpul F terlebih dahulu setelah itu ke simpul C, jarak dari simpul A ke C yakni 9.
c. Tujuan : simpul D
   Lintasan terpendeknya yaitu simpul A menuju simpul F terlebih dahulu setelah itu ke simpul D, jarak dari simpul A ke D yakni 6.
d. Tujuan : simpul E
   Lintasan terpendeknya yaitu simpul A langsung menuju simpul E dengan jarak 3.
e. Tujuan : simpul F
   Lintasan terpendeknya yaitu simpul A langsung menuju simpul F dengan jarak 4.

2. Graf berarah



Gambar 3. Graf Berarah

Solusi dalam bentuk visual :



Gambar 4. Solusi Graf Berarah Dalam Bentuk Visual

Penyelesaian jalur terpendek untuk simpul A pada graf adalah :

a. Tujuan : simpul B
   Lintasan terpendeknya yaitu simpul A langsung menuju simpul B dengan jarak 1.
b. Tujuan : simpul C
   Lintasan terpendeknya yaitu simpul A menuju simpul B terlebih dahulu, setelah itu lanjut ke simpul C, jarak dari simpul A ke C yakni 7.
c. Tujuan : D
   Lintasan terpendeknya yaitu simpul A langsung menuju simpul D dengan jarak 9.
d. Tujuan : E
   Lintasan terpendeknya yaitu simpul A menuju simpul B terlebih dahulu, setelah itu ke simpul C, lanjut ke simpul D, dan terakhir E, jarak dari simpul A ke E yakni 12.
e. Tujuan : F
   Lintasan terpendek untuk simpul A ke F tidak ada, karena jika simpul A menuju ke F dibutuhkan melewati semua simpul yaitu dari A-B-C-D-E-F, yang artinya itu bukan lintasan terpendek.

Algoritma Dijkstra digunakan untuk menyelesaikan masalah lintasan terpendek dari satu sumber pada graf yang dapat berupa graf berarah maupun tak berarah. Dalam konteks sumber tunggal, satu titik dipilih sebagai titik awal, dan algoritma ini akan mencari jalur terpendek dari titik tersebut ke semua titik lainnya. Algoritma Dijkstra tidak cocok untuk graf yang memiliki sisi dengan bobot negatif. Untuk graf yang demikian, algoritma Bellman-Ford, yang dibahas di bagian selanjutnya, dapat digunakan sebagai alternatif (Anonim, n.d.-a).

Untuk menemukan lintasan terpendek, Algoritma Dijkstra memerlukan informasi mengenai titik mana yang menjadi sumber, cara untuk menandai titik yang telah dikunjungi, serta catatan jarak terpendek ke setiap titik selama eksplorasi graf. Algoritma ini akan memperbarui jarak tersebut saat menemukan jalur yang lebih pendek.

Pengaplikasian dari masalah lintasan terpendek dapat dilihat dalam kehidupan sehari-hari (Anonim, n.d.-c), seperti menentukan suatu rute yang paling efisien dari satu kota ke kota lain, menentukan jalur komunikasi paling optimal dalam menghubungkan antara dua terminal computer, dan merencanakan rute penerbangan ynag paling efisien dalam penerbangan lintas negara.

HASIL DAN PEMBAHASAN

Dalam karya ilmiah ini, Algoritma Dijkstra digunakan untuk mencari jarak terpendek antara dua node. Dalam contoh implementasi kali ini, akan dibuat program sebagai bentuk penerapan Algoritma Dijkstra dalam mencari jarak terpendek antara dua kota yang ada di Indonesia, yaitu Kota Jakarta dan Kota Surabaya. Data yang akan digunakan berupa data sekunder, data yang diambil dari salah satu sumber yang ada di internet. Diambil 20 sample kota yang ada di Indonesia dengan data berisi nama kota, garis lintang, dan garis bujur dari masing-masing kota, sebagai berikut :

Tabel 1. Hasil Obsrvasi Objek

City	Lantitude	Logitude
Jakarta	-6.175	106.8275
Surabaya	-7.2458	112.7378
Medan	3.5894	98.6739
Malang	-7.98	112.62
Bekasi	-6.2349	106.9923
Bekasi Kota	-6.2333	107
Tangerang	-6.1703	106.6403
Depok	-6.394	106.8225
Semarang	-6.99	110.4225
Palembang	-2.9861	104.7556
Sangereng	-6.2889	106.7181

City	Lantitude	Logitude
Makassar	-5.1619	119.4362
Bagam	1.13	104.0531
Sumedang	-6.84	107.9208
Batam	1.13	104.0531
Cilacap	-7.7167	109.017
Bandar Lampung	-5.45	105.2667
Bogor	-6.5966	106.7972
Gubeng	-7.2729	112.7493
Cakung	-6.2138	106.9473

Sumber **:** https://simplemaps.com/data/id-cities (Anonim, n.d.-b)

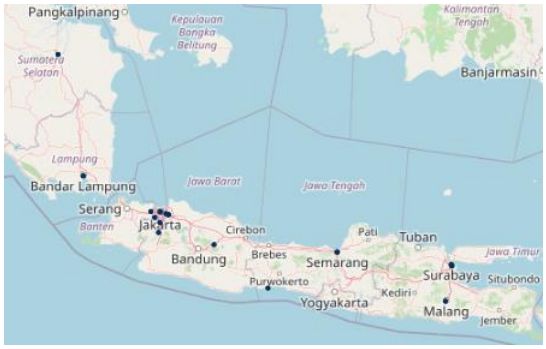
Input

```
import pandas as pd
import numpy as np
pd.set_option('display.max_columns', None)
pd.options.display.max_rows = 500

import matplotlib.pyplot as plt
import plotly.express as px
import geopy.distance

data = {'city': ['Jakarta', 'Surabaya', 'Medan', 'Malang', 'Bekasi', 'Bekasi Kota','Tangerang','Depok','Semarang','Palembang','Sangereng', 'Makassar','Bagam', 'Sumedang', 'Batam', 'Cilacap', 'Bandar Lampung','Bogor', 'Gubeng', 'Cakung'],
        'latitude': [-6.175,-7.2458,3.5894,-7.98,-6.2349,-6.2333,-6.1703,-6.394,-6.99,-2.9861,-6.2889,-5.1619,1.13,-6.84,1.13,-7.7167,-5.45,-6.5966,-7.2729,-6.2138],
                                                    'longitude': [106.8275,112.7378,98.6739,112.62,106.9923,107,106.6403,106.8225,110.4225,104.7556,106.7181,119.4362,104.0531,107.9208,104.0531,109.017,105.2667,106.7972,112.7493,106.9473]}
df_cities = pd.DataFrame(data).set_index('city')
df_cities["coords"] = list(zip(df_cities["latitude"], df_cities['longitude']))

fig = px.scatter_mapbox(df_cities, lat = "latitude", lon = "longitude",
                        hover_name = df_cities.index,
                        zoom = 5, height = 400, width = 950,
                        color_discrete_sequence = ['#00355f'])
fig.update_layout(mapbox_style = "open-street-map")
fig.update_layout(margin = {"r":1,"t":1,"l":1,"b":1})
fig.show()
def create_graph(df_cities, max_distance_between_nodes):
    graph = {}
    for idx, row in df_cities.iterrows():
    centre_node = idx
    centre_coords = row["coords"]
    centre_lat = row["latitude"]
    centre_lon = row["longitude"]

        df_temp = df_cities[(df_cities["latitude"] != centre_lat) & (df_cities["longitude"] != centre_lon)].copy()
        df_temp["distance"] = df_temp["coords"].apply(lambda x: geopy.distance.geodesic(x, centre_coords).km)
        df_temp = df_temp[df_temp["distance"] < max_distance_between_nodes]
```

```
            graph[centre_node] = dict(zip(df_temp.index, df_temp["distance"]))

        graph = {k: v for k, v in graph.items() if v}
        nodes = list(graph.keys())
            return graph, nodes

def dijkstra_algorithm(start_node, end_node, nodes, graph):

    unmarked_nodes = nodes.copy()
    shortest_path = {node: np.inf for node in unmarked_nodes}
    shortest_path[start_node] = 0

    previous_nodes = {}

    while unmarked_nodes:
            current_marked_node = min(unmarked_nodes, key = lambda node: shortest_path.get(node, float('inf')))

            neighbor_nodes = graph[current_marked_node].keys()
            for neighbor in neighbor_nodes:
                                        value_on_hold = shortest_path[current_marked_node] + graph[current_marked_node][neighbor]
            if value_on_hold < shortest_path.get(neighbor, float('inf')):
                shortest_path[neighbor] = value_on_hold
                previous_nodes[neighbor] = current_marked_node

    unmarked_nodes.remove(current_marked_node)

    path = []
    node = end_node
    while node != start_node:
            path.append(node)
            node = previous_nodes[node]
    path.append(start_node)

print(f"The shortest distance is {shortest_path[end_node]:.2f}km.")
    path = list(reversed(path))
    return path

start_node = "Jakarta"; end_node = "Surabaya"; max_distance_between_nodes = 100000000000000 # kilometres

try:
            graph, nodes = create_graph(df_cities = df_cities, max_distance_between_nodes = max_distance_between_nodes)
    path = dijkstra_algorithm(start_node, end_node, nodes, graph)

    '"Plot the result"'
    df = pd.DataFrame()
    for row in range(len(path)):
        df.loc[row, "latitude"] = df_cities.loc[path[row], "latitude"]
        df.loc[row, "longitude"] = df_cities.loc[path[row], "longitude"]
```

```
    fig = px.scatter_mapbox(df, lat = "latitude", lon = "longitude",
                            zoom = 5, height = 400, width = 950,
                            color_discrete_sequence = ['#00355f'])

    fig.add_trace(px.line_mapbox(df, lat = "latitude", lon = "longitude").data[0])
    fig.update_layout(mapbox_style="open-street-map")
    fig.update_layout(margin={"r": 1, "t": 1, "l": 1, "b": 1})
    fig.show()

except:
        print(f"{start_node} and {end_node} is not connected by your graph, please increase max_distance_between_nodes and try again")(Zhao, 2024)
```

Output



Gambar 5. Node Sample 20 Kota di Indonesia

Gambar 6. Jarak Dua Kota Di Indonesia, Kota Jakarta dan Kota Surabaya

```
The shortest distance is 664.08km.
```

Dari program di atas, didapat jarak terpendek Kota Jakarta dan Kota Surabaya adalah 664.08 km.

PENUTUP

Simpulan

Algoritma Dijkstra digunakan untuk mencari jalur terpendek. Dalam paper ini, Dijkstra digunakan untuk mencari jarak terpendek antara kota-kota di Indonesia, contoh jarak terpendek Jakarta dan Surabaya. Algoritma Dijkstra digunakan karena kemampuannya dalam menentukan rute optimal dalam graf berbobot positif. Studi ini menjelaskan bahwa setiap kota diwakili sebagai simpul dan setiap jalan sebagai sisi dalam graf, dengan bobot yang mewakili jarak antar kota.

Algoritma ini diimplementasikan menggunakan bahasa pemrograman Python, dengan hasil menunjukkan jarak terpendek antara Jakarta dan Surabaya adalah sekitar 664,08 km. Penelitian ini menyoroti pentingnya optimasi algoritma Dijkstra dalam jaringan besar dan kompleks, seperti jaringan kota atau komputer, serta manfaatnya untuk meningkatkan efisiensi rute dalam sistem navigasi, jaringan komputer, dan aplikasi graf lainnya

Saran

Algoritma Dijkstra disarankan dalam menentukan rute terpendek suatu wilayah atau daerah, sehingga tepat juga dimanfaatkan dalam pembuatan atau membuka jalan baru pada wilayah yang masih baru.

DAFTAR PUSTAKA

Al Hakim, R. R., Satria, M. H., Arief, Y. Z., Pangestu, A., & Jaenul, A. (2021). Penggunaan Algoritma Dijkstra untuk Berbagai Masalah : Mini Review. *Researchgate.Net*, 1–10. https://rinarxiv.lipi.go.id/lipi/preprint/view/186

Anonim. (n.d.-a). *DSA Dijkstra's Algorithm*. https://www-w3schools-com.translate.goog/dsa/dsa_algo_graphs_dijkstra.php?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=id&_x_tr_pto=tc

Anonim. (n.d.-b). *Indonesia Cities Database*. *https://simplemaps.com/data/id-cities*

Anonim. (n.d.-c). *THE SHORTEST PATH PROBLEM*. 0–4.

Girsang, A. S. (2017). *Algoritma Dijkstra*. https://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/

Rahmadi, D., & Herdianti, R. (2024). Penerapan Minimum Spanning Tree dalam Menentukan Rute Terpendek pada Wisata di Kota Wonogiri. *Basis : Jurnal Ilmiah Matematika, 3*(2), 31-39. doi:10.30872/basis.v3i2.1390

Rahmadi, D., Ramadhani, I. N., Dheana, C. E., & Mustamin, M. A. (2024). Modeling Network Problem using Metric Dimension: Applied Algorithm on Corona Graph. *Mathematical Journal of Modelling and Forecasting*, *2*(1), 20–26. https://doi.org/10.24036/mjmf.v2i1.21

Rahmadi, D., Maharani, N. P., Syifa, M. R., Sama, S. A., & Ardiansyah, G. F. (2024). Optimasi Pemasangan Kabel Internet Antar Daerah Kabupaten Sleman Menggunakan Minimum Spanning Tree. *Journal of Mathematics Theory and Applications*, *2*(2), 24–33. https://doi.org/10.32938/j-math22202424 - 33

Rahmadi, D. (2024). MIXED METRIC DIMENSION OF DOUBLE FAN GRAPH. *Jurnal Diferensial*, *6*(1), 52-56. https://doi.org/10.35508/jd.v6i1.12526

Rahmadi, D. (2023). On Finding Shortest Path Over Vocational High School in Yogyakarta Based on Graph Theory Algorithm. *Mathematical Journal of Modelling and Forecasting*, *1*(2), 10–14. https://doi.org/10.24036/mjmf.v1i2.14

Rahmadi, D., & Sandariria, H. (2023). Penerapan Minimum Spanning Tree dalam Menentukan Rute Terpendek Distribusi Naskah Soal USBN di SMA Negeri se- Sleman. *Basis : Jurnal Ilmiah Matematika, 2*(1), 66-71. doi:10.30872/basis.v2i1.1084

Albar, W., Rahmadi, D., & Dewi, K. (2023). The Implementation of Minimum Spanning Tree in Finding Algebraically the Shortest Path of National-Exam-Sheet Distribution in All Senior High Schools over Bantul Regency. *Basis : Jurnal Ilmiah Matematika, 2*(1), 78-82. doi:10.30872/basis.v2i1.1111

Rahmadi, Deddy. (2022). The k-Metric Dimension of Double Fan Graph. Quadratic: Journal of Innovation and Technology in Mathematics and Mathematics Education. 2. 31-35. 10.14421/quadratic.2022.021-05.

Sudibyo, N. A., Setyawan, P. E., & Hidayat, Y. P. S. R. (2020). Implementasi Algoritma Dijkstra Dalam Pencarian Rute Terpendek Tempat Wisata Di Kabupaten Klaten. Riemann: Research of Mathematics and Mathematics Education, 2(1), 1–9.

Sudibyo, N. A., Purwanto, T., & Rahmadi, D. (2020). Minimum Spanning Tree Pada Distribusi Bahan Naskah USBN SD/MI Di Kabupaten Sragen. *Riemann: Research of Mathematics and Mathematics Education*, *2*(2), 64-69.

Zhao, J. (2024). *Dijkstra's Algorithm for fun*. https://www.kaggle.com/code/jingzhouzhao/dijkstra-s-algorithm-for-fun