EXPLORE

Jurnal Sistem Informasi dan Telematika
(Telekomunikasi, Multimedia dan Informatika)

The Comparison of Dijsktra's Algorithm and Flyod Warshall's Algorithm to Determine The Shortest Path of Traditional Markets in Bandar Lampung City

Poetri Hana Nurhandayani, Wamiliana*, Misgiyati, Notiragayu, dan Muslim Ansori
Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Lampung
Bandar Lampung, Indonesia
poetri2001hana03nurhandayani08@gmail.com, wamiliana.1963@fmipa.unila.ac.id,
misgiyati@fmipa.unila.ac.id, notiragayu@fmipa.unila.ac.id, muslim.ansori@fmipa.unila.ac.id

Abstract- Determining the shortest path between one location to another location is one of the problems that usually has to be faced in daily life. The shortest path will affect the shorter time needed, less money used, and others. In this study, we will discuss the comparison of Dijsktra's Algorithm and Floyd Warshal's Algorithm to determine the shortest path of the biggest and the most famous traditional market in Bandar Lampung City, which is Bambu Kuning traditional market, to 26 other traditional markets in Bandar Lampung city. The Phyton programming language is used to implement both algorithms on the data of 27 traditional markets. The results show that both algorithms give the same solution. However, the processing time of Dijsktra's Algorithm (0.1063 seconds) is faster than Floyd Warshal's algorithm (0.1691 seconds)

Keywords: Shortest Path, Dijkstra's Algorithm, Floyd Warshall's Algorithm, traditional market

Abstrak- Penentuan lintasan terpendek dari satu lokasi ke lokasi lainnya seringakali dihadapi dalam kehidupan sehari-hari. Lintasan terpendek akan mengakibatkan waktu yag diperlukan lebih singkat, dana yang lebih sedikit, dan sebagainya. Pada artikel ini akan dibandingkan Algoritma Dijsktra dan Algoritma Floyd Warshall dalam menentukan lintasan terpendek dari pasar tradisional yang paling besar dan terkenal di Kota Bandarlampung, yaitu Pasar Tradisional Bambu Kuning, ke 26 pasar trasional lainnya yang ada di kota Bandarlampung. Pemigraman Phyton digunakan untuk mengimplementasikan kedua algoritma dengan menggunakan data 27 pasar tradisional tersebut. Hasil yang didapat menunjukkan bahwa kedua algoritma memberikan solusi yang sama, akan tetapi dari waktu proses Algoritma Dijstra (0,1063 detik) lebih cepat dari Algoritma Floyd Warshall (0,1691 detik).

Kata Kunci: Lintasan Terpendek, Dijskrta's Algorithm, Floyd Warshall's Algorithm, pasar tradisional

1. Introduction

A traditional market is a location where traders and buyers meet directly to do business activities for buying and selling transactions. There are 27 traditional markets in the city of Bandar Lampung. Among those 27 traditional markets, several markets are famous for their specialty, such as the traditional market which sells various types of fish (Gudang Lelang Market), the market which sells numerous kinds of vegetables and fruits (Farmer's markets/Pasar Tani), as well as the market which sells many types of goods.

The shortest path from Bambu Kuning traditional market to 26 other traditional markets in the city of Bandar Lampung was found in this study using Floyd Warshall's and Dijkstra's algorithms. The solution and running time of the two algorithms will be compared.

In a weighted graph, the shortest path between any two vertices can be found using Dijkstra's algorithm. The weight value of each edge connecting a pair of vertices in the graph, is equal to the distance between that pair of vertices. A weight must have a positive value (weight ≥0) **[1]**. In 1956 the Dijkstra's Algorithm was discovered by Edger Wybe Dijkstra in 1956 and published in 1959, three years later.

EXPLORE : ISSN: 2087-2062, Online ISSN: 2686-181X / DOI: http://dx.doi.org/10.36448/jsit.v15i1.3714

2. Literature Review

A. Graph

A structure that consists of V, a non-empty set of vertices, and set E is a graph G(V,E). $V=\{v_1, v_2, \ldots, v_n\}$, and $E = \{e_{ij} \mid i, j \in V\}$of edges which connect the vertices in E. Associated with every edge there is a weight c_{ij}, where the weights are nonnegative. An edge that has the same endpoint is called a loop. Parallel edges are two or more edges that join the same pair of vertices. A graph with no parallel edges or a loop is called a simple graph [2]. A vertex in a graph can represent a city, computer, warehouse, location, station, and others; while an edge can represent a road, cable, train track, and others.

B. The Shortest Path

A walk is a finite sequence of alternating vertices and edges, that begin and end with a vertex. Every edge in a walk is next to the vertex immediately before and after it. A path is a walk in which no vertex is passed more than once. A path is called a closed path if the original and terminal vertex are the same. In a graph, it is possible to find more than one path between a pair of vertices, except if the graph is a tree. Some methods were already developed to determine the shortest path, including Dijkstra's Algorithms and Flyod Warshall's Algorithm.

C. Dijkstra's Algorithm

A shortest path between two vertices in a graph can be found using the Dijkstra's Algorithm. This algorithm was developed by Dijkstra in 1959 [3]. In this algorithm, the weights are assumed to be nonnegative. In general, Dijkstra's Algorithm can be illustrated as follows [4]:

Let N= the set of vertices where the shortest path has been found, s is the initial/original vertex.

Initiation: $N=\{ s \}$, d_i = the distance from s to i, $i \in V$, where V is the set of all vertices under consideration (need to be determined the shortest path from s}. Note that $d_s = 0$ because the distance of vertex s to itself is zero, $d_j = c_{sj}$ for every $j \neq s$, c_{sj} is the weight of edge e_{ij}.

The Dijkstra'Algorithm is divided into two stages:

I. Determine $i \notin N$ from so that $d_i = min\ d_j$, for $j \notin N$. Add i to N. If N contains all the nodes, stop

II. For each node $j \notin N$
$d_j = \min (d_j , d_i+c_{ij})$ Go to I. Note that d_i+c_{ij} is the distance from s to j through node i in N.

Dijkstra's Algorithm has been applied in many cases, for examples: Aulia et al [1] used it in determining the shortest distance from The Education Authorities of Lampung Province to 30 out of 229 Public Senior High Schools in 15 subdistricts/municipalities in Lampung Province. Two public Senior High Schools represent every subdistrict/municipality. Putriani et al [3] used Dijkstra' Algorithm to find the shortest distance between Lampung Government Hospital (Abdul Muluk hospital), to 41 other hospitals in Lampung. Cantona et al [5] explore Dijkstra' Algorithm in determining the shortest path to museums in Jakarta, and Kai et al [6], implemented Dijkstra's Algorithm in analyzing the system of emergency response.

D. Floyd Warshall's Algorithm

Another algorithm for determining the shortest path between two vertices is Floyd Warshall's Algorithm. Originally, this algorithm determined the shortest path in a directed graph, however, it was also able for an undirected graph. The Floyd Warshall's Algorithm is given as follows [7]:

1. Create an adjacency matrix using distance/weight from the data of the locations under consideration.
2. Use every vertex k in V as an intermediary vertex to update the matrix. For each $k \in V$, update distances/weight by using $d_{ij} = \min (d_{ij} , d_{ik}+c_{kj})$. The shortest path connecting every two vertices is found by observing the vertex one at a time and updating the weights and distances accordingly.
3. Two possibilities are conceivable for any pair (i,j) of vertices:
   a. k is not an intermediary vertex between i and j. In this case, the shortest path between i and j does not contain vertex k.
   b. k is not an intermediary vertex between i and j. In this case, $d_{ij} = \min d_{ik}+c_{kj}$.

There are some researchers had been applied the Floyd Warshall's Algorithm, including [3], [7-10]

E. The data

The data of the distances among 27 locations of traditional markets in Bandar Lampung is used. The data was taken on Tuesday, 4th of April 2023 using Google Map. Table 1 shows the names and locations of 27 traditional markets in Bandar Lampung.

Table 1 The data of 27 Traditional Markets in Bandar Lampung

No	Name of Traditonal Market	Location
1.	Bambu Kuning	Jl. Imam Bonjol No.1, Kec. Tanjung Karang Pusat
2.	Pasir Gintung	Jl. Pisang, Kec. Tanjung Karang Pusat
3.	Pasar Tengah	Jl. Padang, Kec. Tj. Karang Pusat
4.	Pasar Simpur	Jl. Raden Intan No.32, Enggal
5.	Pasar Tamin	Jl. Antara, Kelapa Tiga, Kec. Tanjung Karang Barat
6.	Pasar Koga	Jl. Teuku Umar, Sidodadi, Kec. Kedaton

EXPLORE : ISSN: 2087-2062, Online ISSN: 2686-181X / DOI: http://dx.doi.org/10.36448/jsit.v15i1.3714

7.	Pasar Tugu	Jl. Hayam Wuruk No.68, anjung Agung Raya, Kec. edamaian
8.	Pasar Gotong Royong	Jl. Wolter Monginsidi No.63 A, Gotong Royong, Kec. Tanjung Karang Pusat
9.	Pasar Way Halim	Gg. Kedua, Kec. Kedaton
10.	Pasar Kangkung	Kec. Telukbetung Selatan
11.	Pasar Way Halim Permai	Kec. Kedaton
12.	Pasar Buah	Sumur Putri, Kec. Teluk Betung Utara
13.	Pasar Labuhan Ratu	Jl. Untung Suropati No.30, Labuhan Ratu, Kec. Kedaton
14.	Pasar Tani	Jl. Teuku Cik Ditiro, Kec. Kemiling
15.	Pasar Bukit Kemiling Permai	Jl. Bukit Kemiling Permai Raya, Kec. Kemiling,
16.	Pasar Cimeng	Jl. Kh. Hasyim Ashari, Talang, Kec. Teluk Betung Selatan,
17.	Pasar Kota Karang	Kota Karang, Kec. Teluk Betung Barat
18.	Pasar Rajabasa	Jl. Kapten Abdul Haq, Kec. Rajabasa
19.	Pasar Gudang Lelang	Jl. Ikan Bawal, Kec. Teluk Betung Selatan
20.	Pasar Sukaraja	Jl. Yos Sudars, Kec. Teluk Betung Selatan
21.	Pasar Untung Suropati	Jl. Untung Suropati, Labuhan Ratu, Kec. Tanjung Senang
22.	Pasar Untung	Jl. R.A. Basyid No.100, Labuhan Dalam, Kec. Tanjung Senang
23.	Pasar Rajabasa Raya	Jl. H. Komarudin baronggan Kotok No.2 a, Rajabasa Raya, Kec. Rajabasa,
24.	Pasar Wana Asri	Beringin Raya, Kec. Kemiling
25.	Pasar Way Dadi	Jl. Pembangunan A, Way Dadi, Kec. Sukarame,
26.	Pasar Way Kandis	Jl. Pulau Damar, Kec. Tanjung Senang
27.	Pasar Korpri	Jl. Ryacudu, Harapan Jaya, Kec. Sukarame

Based on the data in Table 2, we constructs the graph of those 27 traditional markets in Bandar Lampung as shown in Figure 1.

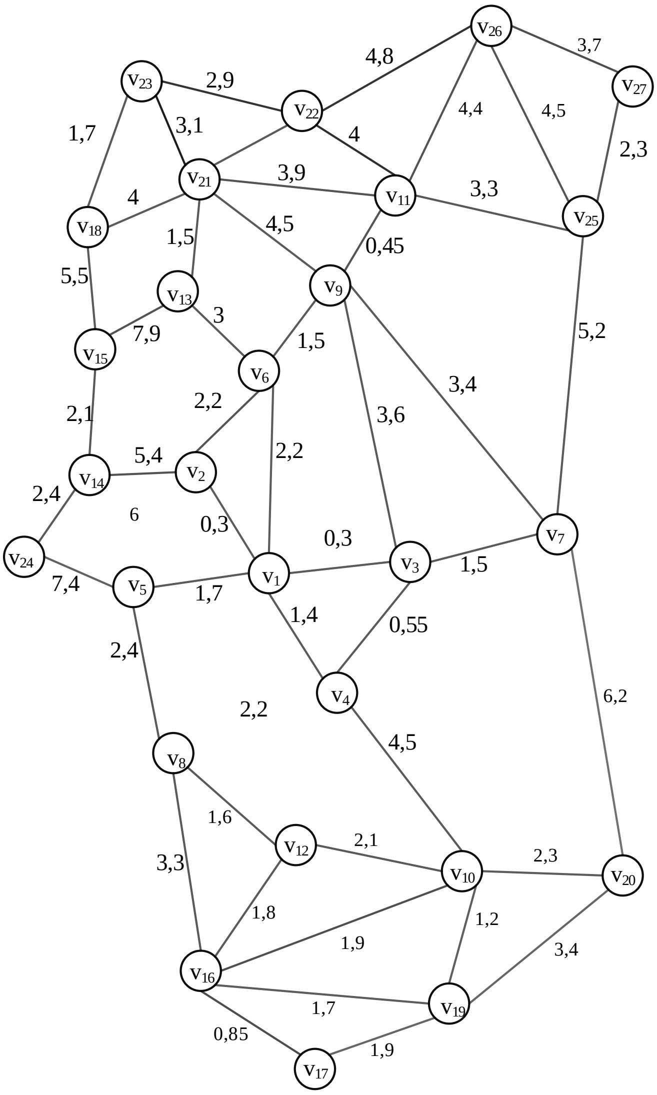


Figure 1. The graph of the location of 27 traditional markets in Bandar Lampung

Table 2 shows the distance among 27 traditional markets in Bandar Lampung. The notation ∞ indicates that there is no direct distance from those two locations, i.e. there is an intermediary location between those two locations.

Table 2 The data of distance of 27 traditional markets in Bandar Lampung (the distance is in km)

	Distance (km)																										
Vertex	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27
v1	0	0.3	0.3	1.4	1.7	2.2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
v2	0.3	0	∞	∞	∞	2.2	∞	∞	∞	∞	∞	∞	∞	5.4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
v3	0.3	∞	0	0.55	∞	∞	1.5	∞	3.6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
v4	1.4	∞	0.55	0	∞	∞	∞	2.2	∞	4.5	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
v5	1.7	∞	∞	∞	0	∞	∞	2.4	∞	∞	∞	∞	∞	6	∞	∞	∞	∞	∞	∞	∞	∞	∞	7.4	∞	∞	∞
v6	2.2	2.2	∞	∞	∞	0	∞	∞	1.5	∞	∞	∞	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
v7	∞	∞	1.5	∞	∞	∞	0	∞	3.4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	6.2	∞	∞	∞	∞	5.2	∞	∞
v8	∞	∞	∞	2.2	2.4	∞	∞	0	∞	∞	∞	1.6	∞	∞	∞	3.3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
v9	∞	∞	3.6	∞	∞	1.5	3.4	∞	0	∞	0.45	∞	∞	∞	∞	∞	∞	∞	∞	∞	4.5	∞	∞	∞	∞	∞	∞
v10	∞	∞	∞	4.5	∞	∞	∞	∞	∞	0	∞	2.1	∞	∞	∞	1.9	∞	∞	1.2	2.3	∞	∞	∞	∞	∞	∞	∞
v11	∞	∞	∞	∞	∞	∞	∞	∞	0.45	∞	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	3.9	4	∞	∞	3.3	4.4	∞
v12	∞	∞	∞	∞	∞	∞	∞	1.6	∞	2.1	∞	0	∞	∞	∞	1.8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
v13	∞	∞	∞	∞	∞	3	∞	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	5	∞	∞	1.5	∞	∞	∞	∞	∞	∞
v14	∞	5.4	∞	∞	6	∞	∞	∞	∞	∞	∞	∞	∞	0	2.1	∞	∞	∞	∞	∞	∞	∞	∞	2.4	∞	∞	∞
v15	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	2.1	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
v16	∞	∞	∞	∞	∞	∞	∞	3.3	∞	1.9	∞	1.8	∞	∞	∞	0	0.85	∞	1.7	∞	∞	∞	∞	∞	∞	∞	∞
v17	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0.85	0	∞	1.9	∞	∞	∞	∞	∞	∞	∞	∞
v18	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	5	∞	∞	∞	∞	0	∞	∞	4	∞	1.7	∞	∞	∞	∞
v19	∞	∞	∞	∞	∞	∞	∞	∞	∞	1.2	∞	∞	∞	∞	∞	1.7	1.9	∞	0	3.4	∞	∞	∞	∞	∞	∞	∞
v20	∞	∞	∞	∞	∞	∞	6.2	∞	∞	2.3	∞	∞	∞	∞	∞	∞	∞	∞	3.4	0	∞	∞	∞	∞	∞	∞	∞
v21	∞	∞	∞	∞	∞	∞	∞	∞	4.5	∞	3.9	∞	1.5	∞	∞	∞	∞	4	∞	∞	0	0.5	3.1	∞	∞	∞	∞
v22	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	4	∞	∞	∞	∞	∞	∞	∞	∞	∞	0.5	0	2.9	∞	∞	4.8	∞
v23	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1.7	∞	∞	3.1	2.9	0	∞	∞	∞	∞
v24	∞	∞	∞	∞	7.4	∞	∞	∞	∞	∞	∞	∞	∞	2.4	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	∞	∞	∞
v25	∞	∞	∞	∞	∞	∞	5.2	∞	∞	∞	3.3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	4.5	2.3
v26	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	4.4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	4.8	∞	∞	.5	0	3.7
v27	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	.3	3.7	0

3. Results and Discussion

A. Implementation of Dijkstra's Algorithm

Dijkstra's Algorithm is implemented on data on Table 2. Because of the space limitation, we only present the first seven iterations out of 26 iterations.

EXPLORE : ISSN: 2087-2062, Online ISSN: 2686-181X / DOI: http://dx.doi.org/10.36448/jsit.v15i1.3714

Table 3 The first seven iteration of Dijkstra's Algorithm

Iteration	Vertices on N	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}	v_{21}	v_{22}	v_{23}	v_{24}	v_{25}	v_{26}	v_{27}
0	$\{V_1\}$	0	0,3	0,3	1,4	1,7	2,2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
1	$\{V_{1,} V_2\}$	0	0,3	0,3	1,4	1,7	2,2	∞	∞	∞	∞	∞	∞	∞	5,7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	$\{V_{1,} V_2, V_3\}$	0	0,3	0,3	0,85	1,7	2,2	1,8	∞	3,9	∞	∞	∞	∞	5,7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	$\{V_{1,} V_2, V_{3,} V_4\}$	0	0,3	0,3	0,85	1,7	2,2	1,8	3,05	3,9	5,35	∞	∞	∞	5,7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
4	$\{V_{1,} V_2, V_{3,} V_{4,} V_5\}$	0	0,3	0,3	0,85	1,7	2,2	1,8	3,05	3,9	5,35	∞	∞	∞	5,7	∞	∞	∞	∞	∞	∞	∞	∞	∞	9,1	∞	∞	∞
5	$\{V_{1,} V_2, V_{3,} V_{4,} V_5, V_7\}$	0	0,3	0,3	0,85	1,7	2,2	1,8	3,05	3,9	5,35	∞	∞	∞	5,7	∞	∞	∞	∞	∞	∞	∞	∞	∞	9,1	7	∞	∞
6	$\{V_{1,} V_2, V_{3,} V_{4,} V_5, V_{6,} V_7\}$	0	0,3	0,3	0,85	1,7	2,2	1,8	3,05	3,7	5,35	∞	∞	5,2	5,7	∞	∞	∞	∞	∞	∞	∞	∞	∞	9,1	7	∞	∞
7	$\{V_{1,} V_2, V_{3,} V_4 , V_5, V_{6,} V_{7,} V_8\}$	0	0,3	0,3	0,85	1,7	2,2	1,8	3,05	3,7	5,35	∞	4,65	5,2	5,7	∞	6,35	∞	∞	∞	∞	∞	∞	∞	9,1	7	∞	∞

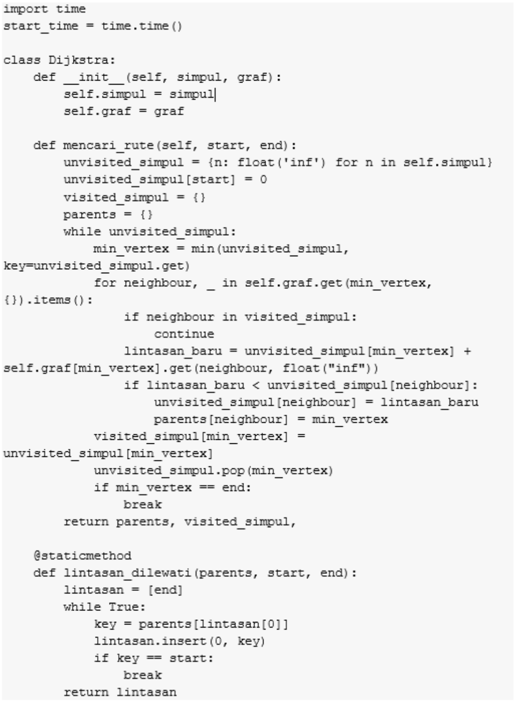
🟨 : The shortest path from v_1 to v_j in j^{th} column.

After running the last iteration, we get the following result: The second row of Table 4 shows the shortest path from v_1 (Bambu Kuning traditional market) to other 26 traditional markets in Bandar Lampung.

Table 4 The last iteration of Dijkstra's Algorithm

Iteration	Vertices on N	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}	v_{21}	v_{22}	v_{23}	v_{24}	v_{25}	v_{26}	v_{27}
26	$\{V_1, V_2, V_{3,}, V_4, V_5$ $V_6, V_7, V_8, V_9, V_{10}, V_{11}$ $, V_{12}, V_{13},$ $V_{14}, V_{15}, V_{16}, V_{17}, V_{18}$ $, V_{19}, V_{20},$ $V_{21}, V_{22}, V_{23}, V_{24}, V_{25}, V_{26}, V_{27},\}$	0	0,3	0,3	0,85	1,7	2,2	1,8	3,05	3,7	5,35	4,15	4,65	5,2	5,7	7,8	6,35	7,2	10,2	6,55	7,65	6,7	7,2	9,8	8,1	7	8,55	9,3

Figure 2 shows screenshot of part of the source code, including some input for Dijkstra's Algorithm, while Figure 4 shows screenshot of part of the source code for Floyd Warshall's Algorithm. Both source codes are written in Phyton programming language.

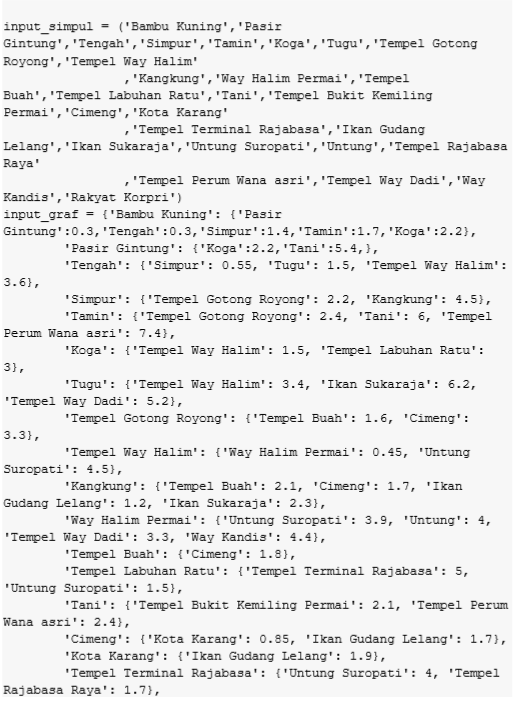


```
import time
start_time = time.time()

class Dijkstra:
    def __init__(self, simpul, graf):
        self.simpul = simpul
        self.graf = graf

    def mencari_rute(self, start, end):
        unvisited_simpul = {n: float('inf') for n in self.simpul}
        unvisited_simpul[start] = 0
        visited_simpul = {}
        parents = {}
        while unvisited_simpul:
            min_vertex = min(unvisited_simpul,
key=unvisited_simpul.get)
            for neighbour, _ in self.graf.get(min_vertex,
{}).items():
                if neighbour in visited_simpul:
                    continue
                lintasan_baru = unvisited_simpul[min_vertex] +
self.graf[min_vertex].get(neighbour, float("inf"))
                if lintasan_baru < unvisited_simpul[neighbour]:
                    unvisited_simpul[neighbour] = lintasan_baru
                    parents[neighbour] = min_vertex
            visited_simpul[min_vertex] =
unvisited_simpul[min_vertex]
            unvisited_simpul.pop(min_vertex)
            if min_vertex == end:
                break
        return parents, visited_simpul,

    @staticmethod
    def lintasan_dilewati(parents, start, end):
        lintasan = [end]
        while True:
            key = parents[lintasan[0]]
            lintasan.insert(0, key)
            if key == start:
                break
        return lintasan
```

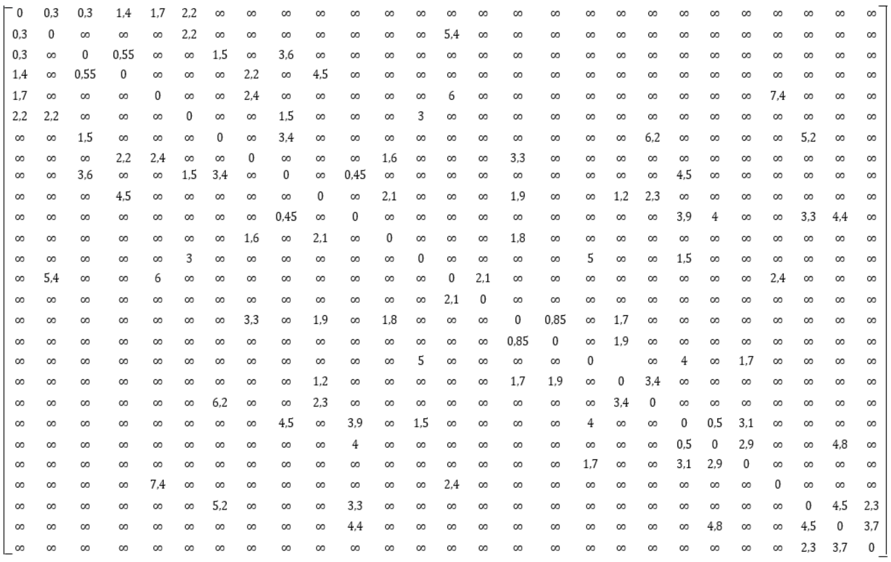


```
input_simpul = ('Bambu Kuning','Pasir
Gintung','Tengah','Simpur','Tamin','Koga','Tugu','Tempel Gotong
Royong','Tempel Way Halim'
                ,'Kangkung','Way Halim Permai','Tempel
Buah','Tempel Labuhan Ratu','Tani','Tempel Bukit Kemiling
Permai','Cimeng','Kota Karang'
                ,'Tempel Terminal Rajabasa','Ikan Gudang
Lelang','Ikan Sukaraja','Untung Suropati','Untung','Tempel Rajabasa
Raya'
                ,'Tempel Perum Wana asri','Tempel Way Dadi','Way
Kandis','Rakyat Korpri')
input_graf = {'Bambu Kuning': {'Pasir
Gintung':0.3,'Tengah':0.3,'Simpur':1.4,'Tamin':1.7,'Koga':2.2},
        'Pasir Gintung': {'Koga':2.2,'Tani':5.4,},
        'Tengah': {'Simpur': 0.55, 'Tugu': 1.5, 'Tempel Way Halim':
3.6},
        'Simpur': {'Tempel Gotong Royong': 2.2, 'Kangkung': 4.5},
        'Tamin': {'Tempel Gotong Royong': 2.4, 'Tani': 6, 'Tempel
Perum Wana asri': 7.4},
        'Koga': {'Tempel Way Halim': 1.5, 'Tempel Labuhan Ratu':
3},
        'Tugu': {'Tempel Way Halim': 3.4, 'Ikan Sukaraja': 6.2,
'Tempel Way Dadi': 5.2},
        'Tempel Gotong Royong': {'Tempel Buah': 1.6, 'Cimeng':
3.3},
        'Tempel Way Halim': {'Way Halim Permai': 0.45, 'Untung
Suropati': 4.5},
        'Kangkung': {'Tempel Buah': 2.1, 'Cimeng': 1.7, 'Ikan
Gudang Lelang': 1.2, 'Ikan Sukaraja': 2.3},
        'Way Halim Permai': {'Untung Suropati': 3.9, 'Untung': 4,
'Tempel Way Dadi': 3.3, 'Way Kandis': 4.4},
        'Tempel Buah': {'Cimeng': 1.8},
        'Tempel Labuhan Ratu': {'Tempel Terminal Rajabasa': 5,
'Untung Suropati': 1.5},
        'Tani': {'Tempel Bukit Kemiling Permai': 2.1, 'Tempel Perum
Wana asri': 2.4},
        'Cimeng': {'Kota Karang': 0.85, 'Ikan Gudang Lelang': 1.7},
        'Kota Karang': {'Ikan Gudang Lelang': 1.9},
        'Tempel Terminal Rajabasa': {'Untung Suropati': 4, 'Tempel
Rajabasa Raya': 1.7},
```

Figure 2. Screenshot of part of the source code for Dijkstra's Algorithm

B. Implementation of Floyd Warshall's Algorithm

The first step of Floyd Warshall's Algorithm is to create a 26 x 26 matrix, which is shown on Figure 3 as follows:

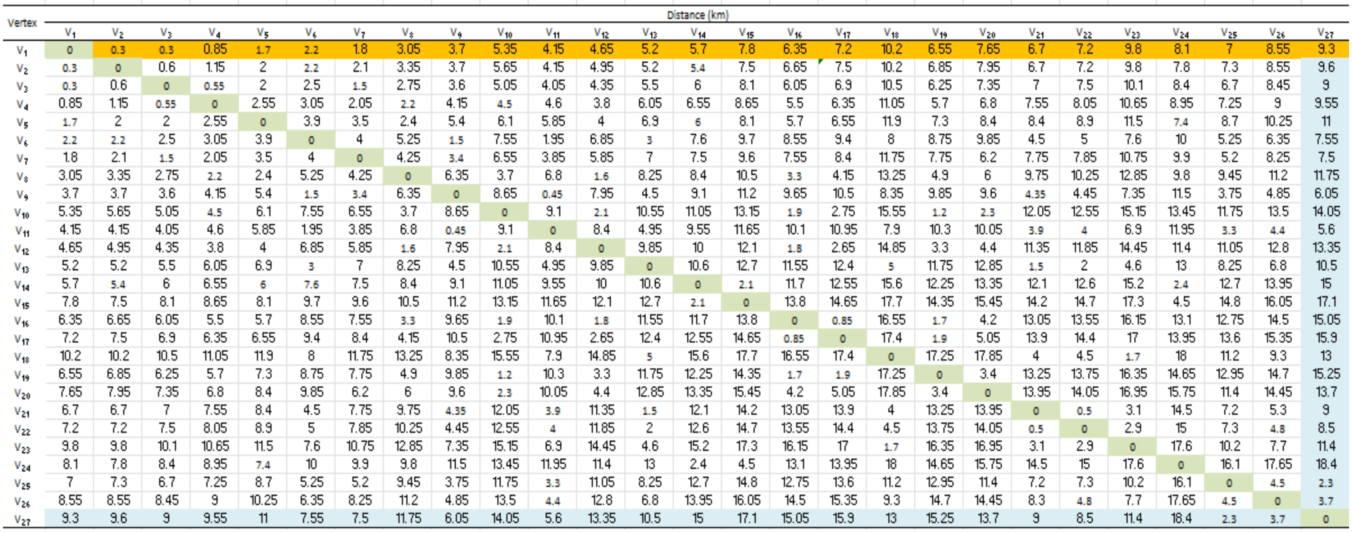


$$D_0 = \begin{bmatrix}
0 & 0{,}3 & 0{,}3 & 1{,}4 & 1{,}7 & 2{,}2 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
0{,}3 & 0 & \infty & \infty & \infty & 2{,}2 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 5{,}4 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
0{,}3 & \infty & 0 & 0{,}55 & \infty & \infty & 1{,}5 & \infty & 3{,}6 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
1{,}4 & \infty & 0{,}55 & 0 & \infty & \infty & \infty & 2{,}2 & \infty & 4{,}5 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
1{,}7 & \infty & \infty & \infty & 0 & \infty & \infty & 2{,}4 & \infty & \infty & \infty & \infty & \infty & 6 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 7{,}4 & \infty & \infty & \infty \\
2{,}2 & 2{,}2 & \infty & \infty & \infty & 0 & \infty & \infty & 1{,}5 & \infty & \infty & \infty & 3 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & 1{,}5 & \infty & \infty & \infty & 0 & \infty & 3{,}4 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 6{,}2 & \infty & \infty & \infty & \infty & 5{,}2 & \infty & \infty \\
\infty & \infty & \infty & 2{,}2 & 2{,}4 & \infty & \infty & 0 & \infty & \infty & \infty & 1{,}6 & \infty & \infty & \infty & 3{,}3 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & 3{,}6 & \infty & \infty & 1{,}5 & 3{,}4 & \infty & 0 & \infty & 0{,}45 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 4{,}5 & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & 4{,}5 & \infty & \infty & \infty & \infty & \infty & 0 & \infty & 2{,}1 & \infty & \infty & \infty & 1{,}9 & \infty & \infty & 1{,}2 & 2{,}3 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0{,}45 & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 3{,}9 & 4 & \infty & \infty & 3{,}3 & 4{,}4 & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 1{,}6 & \infty & 2{,}1 & \infty & 0 & \infty & \infty & \infty & 1{,}8 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & 3 & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & 5 & \infty & \infty & 1{,}5 & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & 5{,}4 & \infty & \infty & 6 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 2{,}1 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 2{,}4 & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 2{,}1 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 3{,}3 & \infty & 1{,}9 & \infty & 1{,}8 & \infty & \infty & \infty & 0 & 0{,}85 & \infty & 1{,}7 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0{,}85 & 0 & \infty & 1{,}9 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 5 & \infty & \infty & \infty & \infty & 0 & & \infty & 4 & \infty & 1{,}7 & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 1{,}2 & \infty & \infty & \infty & \infty & \infty & 1{,}7 & 1{,}9 & \infty & 0 & 3{,}4 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & 6{,}2 & \infty & \infty & 2{,}3 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 3{,}4 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 4{,}5 & \infty & 3{,}9 & \infty & 1{,}5 & \infty & \infty & \infty & \infty & 4 & \infty & \infty & 0 & 0{,}5 & 3{,}1 & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 4 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0{,}5 & 0 & 2{,}9 & \infty & \infty & 4{,}8 & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 1{,}7 & \infty & \infty & 3{,}1 & 2{,}9 & 0 & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & 7{,}4 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 2{,}4 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & 5{,}2 & \infty & \infty & \infty & 3{,}3 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 4{,}5 & 2{,}3 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 4{,}4 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 4{,}8 & \infty & \infty & 4{,}5 & 0 & 3{,}7 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 2{,}3 & 3{,}7 & 0
\end{bmatrix}$$

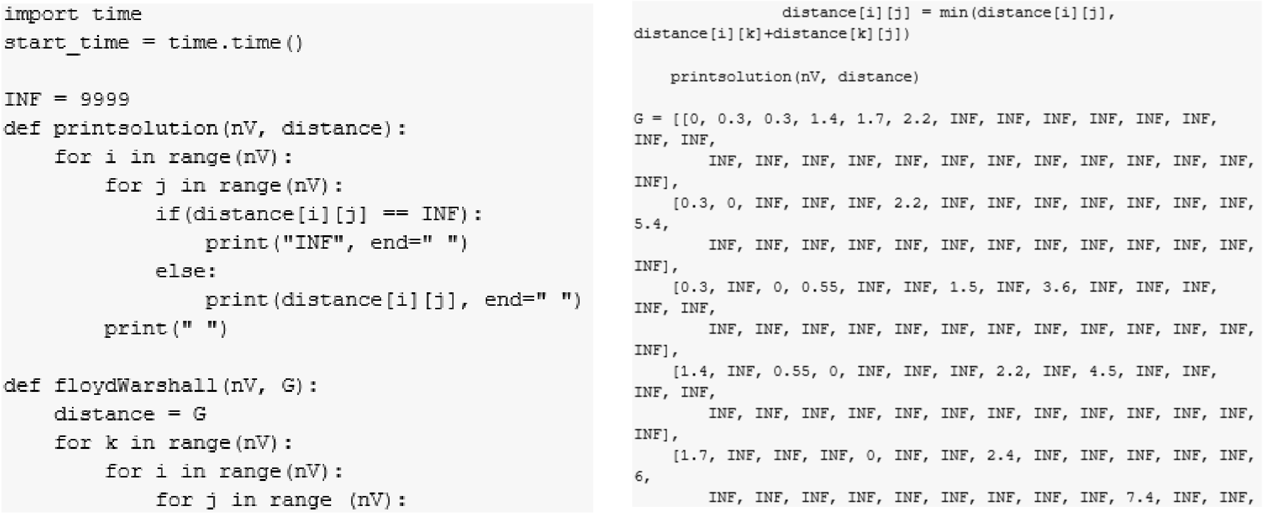
Figure 3. The 27 x 27 matrix for first step of Floyd Warshall' Algorithm

Table 5 shows the result after running the last iteration for Floyd Warshall's Algorithm. The table is formed from the last matrix in the last iteration.

Table 5. The results of the last iteration of Floyd Warshall's Algorithm



Vertex	Distance (km)																									
	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}	V_{11}	V_{12}	V_{13}	V_{14}	V_{15}	V_{16}	V_{17}	V_{18}	V_{19}	V_{20}	V_{21}	V_{22}	V_{23}	V_{24}	V_{25}	V_{26}	V_{27}
V_1	0	0.3	0.3	0.85	1.7	2.2	1.8	3.05	3.7	5.35	4.15	4.65	5.2	5.7	7.8	6.35	7.2	10.2	6.55	7.65	6.7	7.2	9.8	8.1	7	8.55	9.3
V_2	0.3	0	0.6	1.15	2	2.2	2.1	3.35	3.7	5.65	4.15	4.95	5.2	5.4	7.5	6.65	7.5	10.2	6.85	7.95	6.7	7.2	9.8	7.8	7.3	8.55	9.6
V_3	0.3	0.6	0	0.55	2	2.5	1.5	2.75	3.6	5.05	4.05	4.35	5.5	6	8.1	6.05	6.9	10.5	6.25	7.35	7	7.5	10.1	8.4	6.7	8.45	9
V_4	0.85	1.15	0.55	0	2.55	3.05	2.05	2.2	4.15	4.5	4.6	3.8	6.05	6.55	8.65	5.5	6.35	11.05	5.7	6.8	7.55	8.05	10.65	8.95	7.25	9	9.55
V_5	1.7	2	2	2.55	0	3.9	3.5	2.4	5.4	6.1	5.85	4	6.9	6	8.1	5.7	6.55	11.9	7.3	8.4	8.4	8.9	11.5	7.4	8.7	10.25	11
V_6	2.2	2.2	2.5	3.05	3.9	0	4	5.25	1.5	7.55	1.95	6.85	3	7.6	9.7	8.55	9.4	8	8.75	9.85	4.5	5	7.6	10	5.25	6.35	7.55
V_7	1.8	2.1	1.5	2.05	3.5	4	0	4.25	3.4	6.55	3.85	5.85	7	7.5	9.6	7.55	8.4	11.75	7.75	6.2	7.75	7.85	10.75	9.9	5.2	8.25	7.5
V_8	3.05	3.35	2.75	2.2	2.4	5.25	4.25	0	6.35	3.7	6.8	1.6	8.25	8.4	10.5	3.3	4.15	13.25	4.9	6	9.75	10.25	12.85	9.8	9.45	11.2	11.75
V_9	3.7	3.7	3.6	4.15	5.4	1.5	3.4	6.35	0	8.65	0.45	7.95	4.5	9.1	11.2	9.65	10.5	8.35	9.85	9.6	4.35	4.45	7.35	11.5	3.75	4.85	6.05
V_{10}	5.35	5.65	5.05	4.5	6.1	7.55	6.55	3.7	8.65	0	9.1	2.1	10.55	11.05	13.15	1.9	2.75	15.55	1.2	2.3	12.05	12.55	15.15	13.45	11.75	13.5	14.05
V_{11}	4.15	4.15	4.05	4.6	5.85	1.95	3.85	6.8	0.45	9.1	0	8.4	4.95	9.55	11.65	10.1	10.95	7.9	10.3	10.05	3.9	4	6.9	11.95	3.3	4.4	5.6
V_{12}	4.65	4.95	4.35	3.8	4	6.85	5.85	1.6	7.95	2.1	8.4	0	9.85	10	12.1	1.8	2.65	14.85	3.3	4.4	11.35	11.85	14.45	11.4	11.05	12.8	13.35
V_{13}	5.2	5.2	5.5	6.05	6.9	3	7	8.25	4.5	10.55	4.95	9.85	0	10.6	12.7	11.55	12.4	5	11.75	12.85	1.5	2	4.6	13	8.25	6.8	10.5
V_{14}	5.7	5.4	6	6.55	6	7.6	7.5	8.4	9.1	11.05	9.55	10	10.6	0	2.1	11.7	12.55	15.6	12.25	13.35	12.1	12.6	15.2	2.4	12.7	13.95	15
V_{15}	7.8	7.5	8.1	8.65	8.1	9.7	9.6	10.5	11.2	13.15	11.65	12.1	12.7	2.1	0	13.8	14.65	17.7	14.35	15.45	14.2	14.7	17.3	4.5	14.8	16.05	17.1
V_{16}	6.35	6.65	6.05	5.5	5.7	8.55	7.55	3.3	9.65	1.9	10.1	1.8	11.55	11.7	13.8	0	0.85	16.55	1.7	4.2	13.05	13.55	16.15	13.1	12.75	14.5	15.05
V_{17}	7.2	7.5	6.9	6.35	6.55	9.4	8.4	4.15	10.5	2.75	10.95	2.65	12.4	12.55	14.65	0.85	0	17.4	1.9	5.05	13.9	14.4	17	13.95	13.6	15.35	15.9
V_{18}	10.2	10.2	10.5	11.05	11.9	8	11.75	13.25	8.35	15.55	7.9	14.85	5	15.6	17.7	16.55	17.4	0	17.25	17.85	4	4.5	1.7	18	11.2	9.3	13
V_{19}	6.55	6.85	6.25	5.7	7.3	8.75	7.75	4.9	9.85	1.2	10.3	3.3	11.75	12.25	14.35	1.7	1.9	17.25	0	3.4	13.25	13.75	16.35	14.65	12.95	14.7	15.25
V_{20}	7.65	7.95	7.35	6.8	8.4	9.85	6.2	6	9.6	2.3	10.05	4.4	12.85	13.35	15.45	4.2	5.05	17.85	3.4	0	13.95	14.05	16.95	15.75	11.4	14.45	13.7
V_{21}	6.7	6.7	7	7.55	8.4	4.5	7.75	9.75	4.35	12.05	3.9	11.35	1.5	12.1	14.2	13.05	13.9	4	13.25	13.95	0	0.5	3.1	14.5	7.2	5.3	9
V_{22}	7.2	7.2	7.5	8.05	8.9	5	7.85	10.25	4.45	12.55	4	11.85	2	12.6	14.7	13.55	14.4	4.5	13.75	14.05	0.5	0	2.9	15	7.3	4.8	8.5
V_{23}	9.8	9.8	10.1	10.65	11.5	7.6	10.75	12.85	7.35	15.15	6.9	14.45	4.6	15.2	17.3	16.15	17	1.7	16.35	16.95	3.1	2.9	0	17.6	10.2	7.7	11.4
V_{24}	8.1	7.8	8.4	8.95	7.4	10	9.9	9.8	11.5	13.45	11.95	11.4	13	2.4	4.5	13.1	13.95	18	14.65	15.75	14.5	15	17.6	0	16.1	17.65	18.4
V_{25}	7	7.3	6.7	7.25	8.7	5.25	5.2	9.45	3.75	11.75	3.3	11.05	8.25	12.7	14.8	12.75	13.6	11.2	12.95	11.4	7.2	7.3	10.2	16.1	0	4.5	2.3
V_{26}	8.55	8.55	8.45	9	10.25	6.35	8.25	11.2	4.85	13.5	4.4	12.8	6.8	13.95	16.05	14.5	15.35	9.3	14.7	14.45	8.3	4.8	7.7	17.65	4.5	0	3.7
V_{27}	9.3	9.6	9	9.55	11	7.55	7.5	11.75	6.05	14.05	5.6	13.35	10.5	15	17.1	15.05	15.9	13	15.25	13.7	9	8.5	11.4	18.4	2.3	3.7	0



```
import time
start_time = time.time()

INF = 9999
def printsolution(nV, distance):
    for i in range(nV):
        for j in range(nV):
            if(distance[i][j] == INF):
                print("INF", end=" ")
            else:
                print(distance[i][j], end=" ")
        print(" ")

def floydWarshall(nV, G):
    distance = G
    for k in range(nV):
        for i in range(nV):
            for j in range (nV):
```

```
                distance[i][j] = min(distance[i][j],
distance[i][k]+distance[k][j])

    printsolution(nV, distance)

G = [[0, 0.3, 0.3, 1.4, 1.7, 2.2, INF, INF, INF, INF, INF, INF,
INF, INF,
        INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
INF],
    [0.3, 0, INF, INF, INF, 2.2, INF, INF, INF, INF, INF, INF, INF,
5.4,
        INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
INF],
    [0.3, INF, 0, 0.55, INF, INF, 1.5, INF, 3.6, INF, INF, INF,
INF, INF,
        INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
INF],
    [1.4, INF, 0.55, 0, INF, INF, INF, 2.2, INF, 4.5, INF, INF,
INF, INF,
        INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF,
INF],
    [1.7, INF, INF, INF, 0, INF, INF, 2.4, INF, INF, INF, INF, INF,
6,
        INF, INF, INF, INF, INF, INF, INF, INF, INF, 7.4, INF, INF,
```

Figure 4. The screenshot of part of the source code for Flyod Warshall's Algorithm

The last row on Table 4 (indicated in orange colour), shows the last iteration of Dijkstra's Algorithm. The value in that row shows the shortest distance from Bambu Kuning traditional market (v_1) to other 26 locations of traditional markets. For Floyd Warshall's Algorithm, On Table 5, the grey colour indicates the distance of one location to itself (wich is zero), and the row in orange colour shows the shortest distance from the origin (Bambu Kuning) to others. There are more information given by Table 5. The last row and the last column (indicate in light blue colour) show the shortest distance from vertex v_i to vertex v_j ; $i, j = 1, 2, 3, ..., 27$. In other word, using Flyod Warshall's Algorithm, we not only determine the shortest distance from Bambu Kuning traditional market to other 26 locations, but also the shortest distance from one location of taditional martkets to the other 26 locations.

Based on Table 4 and table 5, we can see that both algorithms give the the same solution, which are:

1. The shortest route from Bambu Kuning to Pasar Pasir Gintung is 0,3 km.
2. The shortest route from Bambu Kuning to Pasar Tengah is 0,3 km.
3. The shortest route from Bambu Kuning to Pasar Simpur is 0,85 km using path Pasar Bambu Kuning – Pasar Tengah – Pasar Simpur ($v_1 - v_3 - v_4$).
4. The shortest route from Bambu to Pasar Tamin is 1,7 km.
5. The shortest route from Bambu Kuning to Pasar Koga is 2,2 km.
6. The shortest route from Bambu Kuning to Pasar Tugu is 1,8 km using path Pasar Bambu Kuning- Pasar Tengah – Pasar Tugu ($v_1 - v_3 - v_7$).
7. The shortest route from Bambu Kuning to Pasar Gotong Royong is 3,05 km using path Pasar Bambu Kuning – Pasar Tengah – Pasar Simpur -Pasar Gotong Royong ($v_1 - v_3 - v_4 - v_8$).
8. The shortest route from Bambu Kuning to Pasar Way Halim is 3,7 km using path Pasar Bambu Kuning – Pasar Koga – Pasar Way Halim ($v_1 - v_6 - v_9$).
9. The shortest route from Bambu Kuning to Pasar Kangkung is 5,35 km using path Pasar Bambu Kuning – Pasar Tengah – Pasar Simpur – Pasar Kangkung ($v_1 - v_3 - v_4 - v_{10}$).
10. The shortest route from Bambu Kuning to Pasar Way Halim Permai is 4,15 km using path Pasar Bambu Kuning – Pasar Koga – Pasar Way Halim – Pasar Way HalimPermai ($v_1 - v_6 - v_9 - v_{11}$).
11. The shortest route from Bambu Kuning to Pasar Buah is 4,65 km using path Pasar Bambu Kuning – Pasar Tengah – Pasar Simpur – Pasar Gotong Royong – Pasar buah ($v_1 - v_3 - v_4 - v_8 - v_{12}$).
12. The shortest route from Bambu Kuning to Pasar Labuhan Ratu is 5,2 km using path Pasar Bambu Kuning – Pasar Koga Pasar Labuhan Ratu ($v_1 - v_6 - v_{13}$).
13. The shortest route from Bambu Kuning to Pasar Tani is 5,7 km using path Pasar Bambu Kuning – Pasar Pasir Gintung – Pasar Tani ($v_1 - v_2 - v_{14}$).
14. The shortest route from Bambu Kuning to Pasar Bukit Kemiling Permai is 7,8 km using path Pasar Bambu Kuning – Pasar Pasir Gintung – Pasar Tani –Pasar Bukit Kemiling Permai ($v_1 - v_2 - v_{14} - v_{15}$).

15. The shortest route from Bambu Kuning traditional market to Pasar Cimeng is 6,35 km using path Pasar Bambu Kuning – Pasar Tengah – Pasar Simpur – Pasar Cimeng $(v_1 - v_3 - \ v_4 - v_8 - v_{16})$.
16. The shortest route from Bambu Kuning to Pasar Kota Karang is 7,2 km using path Pasar Bambu Kuning – Pasar Tengah Pasar Simpur Pasar Gotong Royong Pasar Buah Pasar Cimeng Pasar Kota Karang $(v_1 - v_3 - \ v_4 - v_8 - v_{12} - v_{16} - v_{17})$.
17. The shortest route from Bambu Kuning to Pasar Rajabasa is 10,2 km using path Pasar Bambu Kuning – Pasar Koga– Pasar Labuhan Ratu – Pasar Rajabasa $(v_1 - v_6 - \ v_{13} - v_{18})$.
18. The shortest route from Bambu Kuning to Pasar Gudang Lelang is 6,55 km using path Pasar Bambu Kuning – Pasar Tengah – Pasar Simpur – Pasar Kangkung – Pasar Gudang Lelang $(v_1 - v_3 - \ v_4 - v_{10} - v_{19})$.
19. The shortest route from Bambu Kuning to Pasar Sukaraja is 7,65 km using path Pasar Bambu Kuning – Pasar Tengah – Pasar Simpur – Pasar Kangkung – Pasar Sukaraja $(v_1 - v_3 - \ v_4 - v_{10} - v_{20})$.
20. The shortest route from Bambu Kuning to Pasar Untung Suropati is 6,7 km using path Pasar Bambu Kuning – Pasar Koga – Pasar Labuhan Ratu – Pasar Untung Suropati $(v_1 - v_6 - \ v_{13} - v_{21})$.
21. The shortest route from Bambu Kuning to Pasar Untung is 7,2 km using path Pasar Bambu Kuning – Pasar Koga – Pasar Labuhan Ratu – Pasar Untung Suropati – Pasar Untung $(v_1 - v_6 - \ v_{13} - v_{21} - v_{22})$.
22. The shortest route from Bambu Kuning to Pasar Rajabasa Raya is 9,8 km using path Pasar Bambu Kuning – Pasar Koga – Pasar Labuhan Ratu – Pasar Untung Suropati – Pasar Rajabasa Raya $(v_1 - v_6 - v_{13} - v_{21} - v_{23})$.
23. The shortest route from Bambu Kuning to Pasar Wana Asri is 8,1 km using path Pasar Bambu Kuning – Pasar Pasir Gintung – Pasar Tani –Pasar Wana Asri $(v_1 - v_2 - \ v_{14} - v_{24})$.
24. The shortest route from Bambu Kuning to Pasar Way Dadi is 7 km using path Pasar Bambu Kuning – Pasar Tengah – Pasar Tugu – Pasar Way Dadi $(v_1 - v_3 - \ v_7 - v_{25})$.
25. The shortest route from Bambu Kuning to Pasar Way Kandis is 8,55 km using path Pasar Bambu Kuning – Pasar Koga – Pasar Way Halim – Pasar Way Halim Permai – Pasar Way Kandis $(v_1 - v_6 - v_9 - v_{11} - v_{26})$.
26. The shortest route from Bambu to Pasar Korpri is 9,3 km using path Pasar Bambu Kuning – Pasar Tengah – Pasar Tugu – Pasar Way Dadi – Pasar Korpri $(v_1 - v_3 - \ v_7 - v_{25} - v_{27})$.

4. Conclusion

Based on the Results and Discussion above, we can conclude that Dijkstra's Algorithm and Floyd Warshall's Algorithm give the same solutions for determining the shortest path from Bambu Kuning traditional market to the 26 traditional markets in Bandar Lampung. Moreover, Floyd Warshall's Algorithm gives not only the shortest distance from Bambu Kuning traditional market to other 26 locations, but also the shortest path for every pair of locations.

5. References

[1] A. R. Aulia, Wamiliana, Asmiati, and Notiragayu. "Perbandingan Algoritme Dijkstra dan Algoritme A* (A-Star) dalam Penentuan Lintasan Terpendek dari Dinas Pendidikan Provinsi Lampung ke Beberapa Sekolah Menengah Atas (SMA) Negeri di Provinsi Lampung", *Jurnal Pepadun*, Vol. 1 No. 2, pp. 182-190, 2023.

[2] Deo, N. (2004).*Graph Theory With Application to Engineering and Computer Science*: Prentice Hall of India, Ltd.

[3] Schrijver, A. (2012). "On the history of the shortest path problem" *Documenta Mathematica*., *Documenta Mathematica Series:* 155–167. doi:10.4171/dms/6/19. ISBN 978-3-936609-58-5.

[4] Putriani, A.D., Wamiliana, and Chasanah S. L., " Comparison of the Dijkstra's Algorithm and the Floyd Warshall's Algorithm to Determine The Shortest Path between Hospitals in Several Cities in Lampung Province", *Journal of Mathematical Sciences and Optimization*, Vol. 1 No. 2, pp. 44-52, 2024.

[5] Cantona, A., Fauziah, F., and Winarsih, W. "Implementasi Algoritma Dijkstra Pada Pencarian Lintasan Terpendek ke Museum di Jakarta", *Jurnal Teknologi dan Manajemen Informatika*, *6*(1), 27-34, 2020.

[6] N. Kai, Z. Yao-Ting, and M. Yue-Peng." Shortest Path Analysis Based on Dijkstra's Algorithm in Emergency Response System", *Telkomnika Indonesian Journal of Electrical Engineering*, Vol. 12 No. 5, 3476-3482, 2014.

[7] Darnita, Y., Toyib, R., and Rinaldi, R. "Implementasi Algoritma Floyd Warshall Untuk Menentukan Letak Dan Lokasi Perusahaan Travel/Rental Mobil Di Kota Bengkulu", *Pseudocode*, *4*(2), 144-155, 2017.

[8] A. Fadillah, S. H. Mansyur, and Purnawansyah." Analisis Perbandingan Algoritma Floyd-Warshall Dan A Star (A*) Dalam Penentuan Lintasan Terpendek", *Jurnal Teknik Informatika dan Sistem Informasi*, Vol. 9 No. 4, p.2736-2751, 2022.

[9] T.S. Dermawan, "Comparison of Dijkstra Dan Floyd-Warshall Algorithm to Determine the Best Route of Train." *International Journal on Informatics for Development*, vol. 7, no. 2, pp. 54-58, 2018. doi:10.14421/ijid.2018.07202

[10] Risald, A. E. Mirino, and Suyoto, "Best routes selection using Dijkstra and Floyd-Warshall algorithm," *2017 11th International Conference on Information & Communication Technology and System (ICTS)*, Surabaya, Indonesia, pp. 155-158, 2017. doi: 10.1109/ICTS.2017.8265662