The Comparison of Dijsktra’s Algorithm and Floyd 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, wamiliana1963@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 Djikstra’s Algorithm and Floyd Warshall’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 Djikstra’s Algorithm (0.1063 seconds) is faster than Floyd Warshall’s algorithm (0.1691 seconds).

Keywords: Shortest Path, Djikstra’s Algorithm, Floyd Warshall’s Algorithm, traditional market

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 Djikstra'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 Dijsktra’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 Djikstra’s Algorithm was discovered by Edger Wybe Dijskstra in 1956 and published in 1959, three years later.
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 \( c \), 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 path. 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 Floyd 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.

Initialisation: \( N = \{ s \} \), \( d_{ij} \) = 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 \( d_{ij} = 0 \) because the distance of vertex \( s \) to itself is zero, \( d_{ij} = c_{ij} \) for every \( j \neq i \), \( c_{ij} \) is the weight of edge \( e_{ij} \).

The Dijkstra’s Algorithm is divided into two stages:

I. Determine \( i \notin N \) from so that \( d_{ij} = \min(d_{ij}, d_{ik} + c_{kj}) \).

II. For each node \( j \notin N \)

\( d_{ij} = \min(d_{ij}, d_{iN} + c_{Nj}) \)

Go to I. Note that \( d_{iN} + c_{Nj} \) 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 as 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 \( d_{ij} = \min(d_{ik}, d_{kj}, c_{ik} + c_{kj}) \).
3. Two possibilities are conceivable for any pair \( (ij) \) 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 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>
<thead>
<tr>
<th>No</th>
<th>Name of Traditional Market</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bambu</td>
<td>Jl. Imam Bonjol No.1, Kec. Tanjung Karang Pusat</td>
</tr>
<tr>
<td>2</td>
<td>Pasir Gintung</td>
<td>Jl. Pisang, Kec. Tanjung Karang Pusat</td>
</tr>
<tr>
<td>3</td>
<td>Pasar Tengah</td>
<td>Jl. Padang, Kec. Tj. Karang Pusat</td>
</tr>
<tr>
<td>4</td>
<td>Pasar Simpur</td>
<td>Jl. Raden Intan No.32, Enggal</td>
</tr>
<tr>
<td>5</td>
<td>Pasar Tamin</td>
<td>Jl. Antara, Kelapa Tiga, Kec. Tanjung Karang Barat</td>
</tr>
<tr>
<td>6</td>
<td>Pasar Koga</td>
<td>Jl. Teuku Umar, Sidodadi, Kec. Kedaton</td>
</tr>
</tbody>
</table>
Based on the data in Table 2, we constructs the graph of those 27 traditional markets in Bandar Lampung as shown in Figure 1.

Table 2 shows the distance among 27 traditional markets in Bandar Lampung. The notation $\infty$ 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)

<table>
<thead>
<tr>
<th>Distance (km)</th>
</tr>
</thead>
<tbody>
<tr>
<td>---------</td>
</tr>
<tr>
<td>V1</td>
</tr>
<tr>
<td>V2</td>
</tr>
<tr>
<td>V3</td>
</tr>
<tr>
<td>V4</td>
</tr>
<tr>
<td>V5</td>
</tr>
<tr>
<td>V6</td>
</tr>
<tr>
<td>V7</td>
</tr>
<tr>
<td>V8</td>
</tr>
<tr>
<td>V9</td>
</tr>
<tr>
<td>V10</td>
</tr>
<tr>
<td>V11</td>
</tr>
<tr>
<td>V12</td>
</tr>
<tr>
<td>V13</td>
</tr>
<tr>
<td>V14</td>
</tr>
</tbody>
</table>

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.
### Table 3: The first seven iterations of Dijkstra’s Algorithm

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Vertices on N</th>
<th>v₁</th>
<th>v₂</th>
<th>v₃</th>
<th>v₄</th>
<th>v₅</th>
<th>v₆</th>
<th>v₇</th>
<th>v₈</th>
<th>v₉</th>
<th>v₁₀</th>
<th>v₁₁</th>
<th>v₁₂</th>
<th>v₁₃</th>
<th>v₁₄</th>
<th>v₁₅</th>
<th>v₁₆</th>
<th>v₁₇</th>
<th>v₁₈</th>
<th>v₁₉</th>
<th>v₂₀</th>
<th>v₂₁</th>
<th>v₂₂</th>
<th>v₂₃</th>
<th>v₂₄</th>
<th>v₂₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{v₁}</td>
<td>0</td>
<td>0</td>
<td>1.4</td>
<td>1.7</td>
<td>2.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>{v₁, v₂}</td>
<td>0</td>
<td>0</td>
<td>1.3</td>
<td>1.4</td>
<td>1.7</td>
<td>2.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>{v₁, v₂, v₃}</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.8</td>
<td>1.7</td>
<td>2.2</td>
<td>1.8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5.7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>{v₁, v₂, v₃, v₄}</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.8</td>
<td>1.7</td>
<td>2.2</td>
<td>1.8</td>
<td>3.0</td>
<td>3.9</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
</tr>
<tr>
<td>4</td>
<td>{v₁, v₂, v₃, v₄, v₅}</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.8</td>
<td>1.7</td>
<td>2.2</td>
<td>1.8</td>
<td>3.0</td>
<td>3.9</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
</tr>
<tr>
<td>5</td>
<td>{v₁, v₂, v₃, v₄, v₅, v₆}</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.8</td>
<td>1.7</td>
<td>2.2</td>
<td>1.8</td>
<td>3.0</td>
<td>3.9</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
</tr>
<tr>
<td>6</td>
<td>{v₁, v₂, v₃, v₄, v₅, v₆, v₇}</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.8</td>
<td>1.7</td>
<td>2.2</td>
<td>1.8</td>
<td>3.0</td>
<td>3.9</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
</tr>
<tr>
<td>7</td>
<td>{v₁, v₂, v₃, v₄, v₅, v₆, v₇, v₈}</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.8</td>
<td>1.7</td>
<td>2.2</td>
<td>1.8</td>
<td>3.0</td>
<td>3.9</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>4.65</td>
<td>5.2</td>
<td>5.7</td>
<td>6.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
<td>5.35</td>
</tr>
</tbody>
</table>

After running the last iteration, we get the following result: The second row of Table 4 shows the shortest path from v₁ (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 | 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 |
|-----------|--------------|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 26        | {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} | 0  | 0  | 0.8| 0.8| 1.7| 2.2| 1.8| 3.0| 3.7| 3.75| 4.15| 4.65| 5.2 | 5.7 | 5.7 | 7.8 | 6.85| 7.2 | 10.2| 6.55| 7.65| 6.7 | 7.2 | 9.8 | 8.1 | 7  | 8.55| 9.3|

The shortest path from v₁ to v₇ in jᵗʰ column.

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’s Algorithm. Both source codes are written in Python programming language.
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:

\[
D_{ij} =
\]

Figure 2. Screenshot of part of the source code for Dijkstra’s 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
import time
start_time = time.time()

INF = 9999
def print_solution(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])

    print_solution(nV, distance)

G = [(0, 0.3, 0.5, 1.6, 1.7, 2.2, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF),
     (0.3, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF),
     (0.5, INF, 0.55, INF, 1.5, INF, 3.6, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF),
     (1.6, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF),
     (1.7, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF),
     (1.5, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF),
     (3.6, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF),
     (INF, INF, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF),
     (INF, INF, INF, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF),
     (INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF),
     (INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF, INF),
     (INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF, INF),
     (INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF, INF),
     (INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF, INF),
     (INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF, INF),
     (INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, INF, INF, INF, INF),
     (INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, INF, INF, INF),
     (INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, INF, INF),
     (INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, INF),
     (INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0)]

floydWarshall(15, G)

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

Figure 4. The screenshot of part of the source code for Floyd's 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 (v1) 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 to vertex v; i, j =1, 2, 3, ..., 27. In other word, using Floyd 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 traditional markets 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 (v1 – v3 – v4).
4. The shortest route from Bambu to Pasar T'amin 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 (v1 – v3 – v7).
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 (v1 – v6 – v9).
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 (v1 – v6 – v9 – v13).
13. The shortest route from Bambu Kuning to Pasar Tani is 5,7 km using path Pasar Bambu Kuning – Pasar Tani (v1 – v2 – v14).
14. The shortest route from Bambu Kuning to Pasar Bukit Kemiling Pemai is 7,8 km using path Pasar Bambu Kuning – Pasar Bukit Kemiling Pemai – Pasar Tani –Pemai (v1 – v2 – v14 – v15)


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


