Available online at http://www.iorajournal.org/index.php/ijgor/index

International Journal of Global Operations Research

e-ISSN: 2722-1016
p-ISSN: 2723-1739

Vol. 3, No. 3, pp. 86-95, 2022

Determining The Shortest Path of Car Parking Layout in FMIPA UNPAD Using Floyd-Warshall Algorithm

Mochamad Suyudi [1*], Asma Ainun Mardiyah [2]

[1,2] *Department of Mathematics, Faculty of Mathematics and Natural Sciences, University of Padjadjaran*
Jl. Ir. Soekarno KM 21 Jatinangor, Sumedang Indonesia

**Corresponding author email: moch.suyudi@gmail.com*

Abstract

Vehicles that are increasingly needed by the community increase the volume of vehicle traffic, resulting in a high demand for parking spaces. Especially in public places such as campuses, offices, shopping centers, and other places. A parking lot is also needed that has a maximum capacity by determining the layout of the vehicle mileage in finding a parking location. In this paper, we are looking for the shortest path in the car park layout at FMIPA UNPAD using the Floyd-Warshall algorithm.

Keywords: Graph; Parking lot; Floyd-Warshall Algorithm; shortest path

1. Introduction

Vehicle is a means of transportation to transport people or goods. The size or capacity of the vehicles also varies from those that only carry two people to those that can accommodate hundreds of people. With increasing years and the human population, the need to use or own a vehicle is increasing. Congestion on the road is getting longer due to the increasing number of vehicles.

Increased traffic and vehicle volumes have also resulted in increased demand for parking spaces for certain areas, such as business areas/or areas that have activities that make parking a major problem (Fortuna, Sandra, Civil, & Trisakti, 2020). Parking lots are increasingly needed, especially in public places such as shopping centers, hospitals, offices, campuses, schools, places of worship, and other places. A parking lot is needed that has a maximum capacity and pays attention to comfort for its users by determining the layout and travel time of the vehicle in finding a parking location.

The shortest path problem is a problem to find a path between two vertices such that the sum of the weights from the edge of the arrangement can be minimized (Kumar, R.., & Kumar, M. 2010). Several algorithms that have been developed to solve the shortest path problem include the Dijkstra algorithm and the Floyd-Warshall algorithm. Dijkstra's algorithm is an algorithm for determining the shortest path from one vertex to another in a weighted graph, the distance between vertices has a weight on each edge of the graph (Vasudev, C. 2006). In this paper, we use one of the methods to determine the shortest path, namely the Floyd-Warshall method. Floyd-Warshall is an algorithm that can be used in calculating the shortest path, and can compare all possible paths in the graph for each edge of all existing vertices. Floyd-Warshall algorithm can be used in parking layouts to calculate the shortest path. The most efficient algorithm in determining the shortest path is Dijkstra's algorithm, however, in the problem of the shortest path where parking is located, the Floyd-Warshall algorithm is more effective because it is able to compare all possible paths on each edge of the vertices in the graph (Jayanti, 2017).

In this paper, we will look for the shortest path to the car parking at the Faculty of Mathematics and Natural Sciences (FMIPA) Universitas Padjadjaran using the Floyd-Warshall Algorithm, because the Floyd-Warshall Algorithm can solve parking layout problems effectively and optimally by finding a path. closest from the parking entrance to the parking lot.

2. Method

The shortest path problem discusses determining the path with the lowest total weight from the initial vertex to the destination vertex in a weighted graph. The shortest path can be interpreted as the smallest path between the two vertices obtained. There are several algorithms that can find the shortest path in a weighted graph. In this study using

the Floyd-Warshall algorithm. The algorithm that is often used in the shortest path problem is the Dijkstra algorithm, but for the parking lot layout problem the Floyd-Warshall algorithm is more effective.

The Floyd-Warshall algorithm was invented in 1967 by Robert W. Floyd. Floyd-Warshall algorithm can calculate the smallest weight of all paths that connect between a pair of vertices and is carried out at the same time. Comparing all possible paths on each edge of all vertices in the graph (Ramadhan, Siahaan, & Mesran, 2018).

The Floyd-Warshall algorithm is one of dynamic programming that performs problem solving by looking at the solution that will be obtained as an interconnected result. The solution is obtained from the previous stage solution and can allow more than one solution (Fatmala, Yudatama, & Burhanuddin, 2019). The Floyd-Warshall algorithm uses the following procedure (Rosen, K. H. 2012):

For $i := 1$ to n
  For $j := 1$ to n
    $d(v_i, v_j) := w(v_i, v_j)$
For $i := 1$ to n
  For $j := 1$ to n
    For $k := 1$ to n
      If $w(v_i, v_j) > w(v_i, v_k) + w(v_k, v_j)$
        then $w(v_i, v_j)$ replace with $w(v_i, v_k) + w(v_k, v_j)$
repeat$[w(v_i, v_j)]$, $\{w(v_i, v_j)$ is the length between v_i and v_j; For $1 \le i \le n, 1 \le j \le n\}$.

The above algorithm only calculates the total shortest distance to all vertices but does not show the path traversed which results in the shortest path. To be able to find out the path traversed so that the shortest path is found, a square matrix *Z* of size *n x n* must be added, with initialization (Azis, Mallongi, Lantara, & Salim, 2018):

$$Z_{ij}^0 = \begin{cases} j & \text{if } W_{ij}^0 \neq \infty \\ 0 & \text{if } W_{ij}^0 = \infty \end{cases} \quad (1)$$

If in the *k*-th iteration there is an exchange between $w(v_i, v_j)$ with $w(v_i, v_k) + w(v_k, v_j)$, then the value must be changed $Z[i, j]$ with value $Z[i, k]$. Thus the Floyd-Warshall algorithm procedure becomes:

$W = W_0; Z = Z_0$
For $i = 1$ to n,
For $j = 1$ to n,
For $k = 1$ to n,
If $W[i, j] > W[i, k] + W[k, j]$, then:
Replace $W[i, j]$ with $W[i, k] + W[k, j]$
Replace $Z[i, j]$ with $Z[i, k]$
$W^* = W; Z^* = Z$

The matrix W^* is the shortest adjacency matrix and W_{ij} is the shortest route from vertex v_i to v_j. Z^* is a matrix that shows the path that must be passed by the origin vertex to the end vertex.

3. Results and Discussion

The FMIPA UNPAD parking lot has 3 rows of car parking lots with 46 parking lots. The area of each parking lot is 2.5 x 5 meters. The FMIPA UNPAD car park has different entrances and exits, but does not have a unidirectional route rule so it can be reversed or passed in two directions. Here only calculates the closest distance from the entrance to the parking lot not to the parking exit. The Figure 1 is a picture of a floor plan for the FMIPA UNPAD car parking.

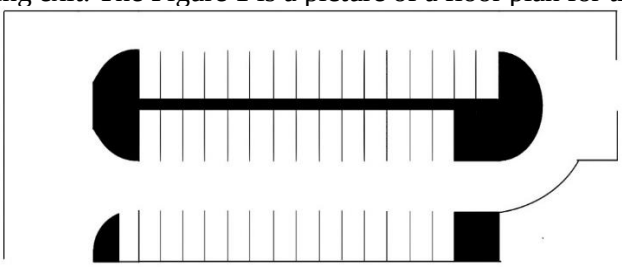

Figure 1. Parking Plan FMIPA UNPAD

From the floor plan of the FMIPA UNPAD car park, a graph can be formed by assigning a weight to each edges of the graph.

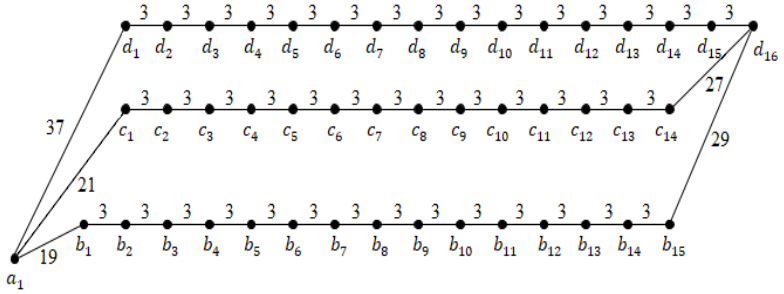

Figure 2. Parking Graph FMIPA UNPAD

Floyd-Warshall algorithm calculates all possible shortest paths to all vertices. In this study will also display the path traversed as to obtain the shortest path. Then the Floyd-Warshall algorithm procedure used is as follows:

$W = W_0; Z = Z_0$
For $i = 1$ to n,
For $j = 1$ to n,
For $k = 1$ to n,
If $W[i,j] > W[i,k] + W[k,j]$, then:
Replace $W[i,j]$ with $W[i,k] + W[k,j]$
Replace $Z[i,j]$ with $Z[i,k]$
$W^* = W; Z^* = Z$

The matrix W^* is the shortest adjacency matrix and W_{ij} is the shortest route from vertex v_i to v_j. Z^* matrix is a matrix that shows the path that must be passed by the origin vertex to the end vertex.

Change the weighted graph in Figure 4.2 into a matrix W_0 with dimensions of *n x n*, where *n* is the number of vertices, which is 46 vertices. Vertex a_1 is the entrance to the parking lot and vertex $b_1$to vertex d_{16} is the parking lot. The problem in this study is to find the shortest path from the entrance to the parking lot using the Floyd-Warshall algorithm. W_0 matrix as follows.

Table 1. Matrix of parking lot graph of FMIPA UNPAD

$i\backslash j$	a_1	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	⋯	d_{12}	d_{13}	d_{14}	d_{15}	d_{16}
a_1	0	19	∞	∞	∞	∞	∞	∞	∞	∞	⋯	∞	∞	∞	∞	∞
b_1	19	0	3	∞	∞	∞	∞	∞	∞	∞	⋯	∞	∞	∞	∞	∞
b_2	∞	3	0	3	∞	∞	∞	∞	∞	∞	⋯	∞	∞	∞	∞	∞
b_3	∞	∞	3	0	3	∞	∞	∞	∞	∞	⋯	∞	∞	∞	∞	∞
b_4	∞	∞	∞	3	0	3	∞	∞	∞	∞	⋯	∞	∞	∞	∞	∞
b_5	∞	∞	∞	∞	3	0	3	∞	∞	∞	⋯	∞	∞	∞	∞	∞
b_6	∞	∞	∞	∞	∞	3	0	3	∞	∞	⋯	∞	∞	∞	∞	∞
b_7	∞	∞	∞	∞	∞	∞	3	0	3	∞	⋯	∞	∞	∞	∞	∞
b_8	∞	∞	∞	∞	∞	∞	∞	3	0	3	⋯	∞	∞	∞	∞	∞
b_9	∞	∞	∞	∞	∞	∞	∞	∞	3	0	⋯	∞	∞	∞	∞	∞
...	...	...	...	...	...	...	...	...	...	...	⋯	...	...	...	...	...
d_{12}	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	⋯	0	3	∞	∞	∞
d_{13}	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	⋯	3	0	3	∞	∞
d_{14}	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	⋯	∞	3	0	3	∞
d_{15}	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	⋯	∞	∞	3	0	3
d_{16}	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	⋯	∞	∞	∞	3	0

At this stage only displays the cells in the matrix that change, for example:

Vertex a_1 = 1	Vertex c_1 = 17	Vertex d_1 = 31
Vertex b_1 = 2	Vertex c_2 = 18	Vertex d_2 = 32
Vertex b_2 = 3	Vertex c_3 = 19	Vertex d_3 = 33
Vertex b_3 = 4	Vertex c_4 = 20	Vertex d_4 = 34
Vertex b_4 = 5	Vertex c_5 = 21	Vertex d_5 = 35
Vertex b_5 = 6	Vertex c_6 = 22	Vertex d_6 = 36
Vertex b_6 = 7	Vertex c_7 = 23	Vertex d_7 = 37
Vertex b_7 = 8	Vertex c_8 = 24	Vertex d_8 = 38
Vertex b_8 = 9	Vertex c_9 = 25	Vertex d_9 = 39
Vertex b_9 = 10	Vertex c_{10} = 26	Vertex d_{10} = 40
Vertex b_{10} = 11	Vertex c_{11} = 27	Vertex d_{11} = 41
Vertex b_{11} = 12	Vertex c_{12} = 28	Vertex d_{12} = 42
Vertex b_{12} = 13	Vertex c_{13} = 29	Vertex d_{13} = 43
Vertex b_{13} = 14	Vertex c_{14} = 30	Vertex d_{14} = 44
Vertex b_{14} = 15		Vertex d_{15} = 45
Vertex b_{15} = 16		Vertex d_{16} = 46

If $W[i,j] > W[i,k] + W[k,j]$, then replace $W[i,j]$ with $W[i,k] + W[k,j]$. $Z[i,j]$ is also replace for $Z[i,k]$. The following is the result of a matrix with only $W[i,j]$] replace for $W[i,k] + W[k,j]$ and $Z[i,j]$ replace for $Z[i,k]$, where $W[i,j] = W[j,i]$ or vice versa.

Iteration 1:
$W[2,17] = 40$ $W[2,31] = 56$ $W[17,31] = 58$
$Z[2,17], Z[2,31], Z[17,31] = Z[i,1] = 1$

Iteration 2:
$W[1,3] = 22$ $W[3,17] = 43$ $W[3,31] = 59$
In the matrix Z_2 replace $Z[i,j]$] with $Z[i,2] = 2$.

Iteration 3:
$W[1,4] = 25$ $W[2,4] = 6$ $W[4,17] = 46$ $W[4,31] = 62$
In the matrix Z_3 replace $Z[i,j]$ with $Z[i,3] = 3$.

Iteration 4:
$W[1,5] = 28$ $W[2,5] = 9$ $W[3,5] = 6$ $W[5,17] = 49$
$W[5,31] = 65$
In the matrix Z_4 replace $Z[i,j]$ with $Z[i,4] = 4$.

Iteration 5:
$W[1,6] = 31$ $W[2,6] = 12$ $W[3,6] = 9$ $W[4,6] = 6$
$W[6,17] = 52$ $W[6,31] = 68$
In the matrix Z_5 replace $Z[i,j]$ with $Z[i,5] = 5$.

Iteration 6:
$W[1,7] = 34$ $W[2,7] = 15$ $W[3,7] = 12$ $W[4,7] = 9$
$W[5,7] = 6$ $W[7,17] = 55$ $W[7,31] = 71$
In the matrix Z_6 replace $Z[i,j]$ with $Z[i,6] = 6$.

Iteration 7:
$W[1,8] = 37$ $W[2,8] = 18$ $W[3,8] = 15$ $W[4,8] = 12$
$W[5,8] = 9$ $W[6,8] = 6$ $W[8,17] = 58$ $W[8,31] = 74$
In the matrix Z_7 replace $Z[i,j]$ with $Z[i,7] = 7$.

Iteration 8:
$W[1,9] = 40$ $W[2,9] = 21$ $W[3,9] = 18$ $W[4,9] = 15$
$W[5,9] = 12$ $W[6,9] = 9$ $W[7,9] = 6$ $W[9,17] = 61$
$W[9,31] = 77$

In the matrix Z_8 replace $Z[i,j]$ with $Z[i,8]=8$.

Iteration 9:

$W[1,10]=43$ $W[2,10]=24$ $W[3,10]=21$ $W[4,10]=18$
$W[5,10]=15$ $W[6,10]=12$ $W[7,10]=9$ $W[8,10]=6$
$W[10,17]=64$ $W[10,31]=80$

In the matrix Z_9 replace $Z[i,j]$ with $Z[i,9]=9$.

Iteration 10:

$W[1,11]=46$ $W[2,11]=27$ $W[3,11]=24$ $W[4,11]=21$
$W[5,11]=18$ $W[6,11]=15$ $W[7,11]=12$ $W[8,11]=9$
$W[9,11]=6$ $W[11,17]=67$ $W[11,31]=83$

In the matrix Z_{10} replace $Z[i,j]$ with $Z[i,10]=10$.

Iteration 11:

$W[1,12]=49$ $W[2,12]=30$ $W[3,12]=27$ $W[4,12]=24$
$W[5,12]=21$ $W[6,12]=18$ $W[7,12]=15$ $W[8,12]=12$
$W[9,12]=9$ $W[10,11]=6$ $W[12,17]=70$ $W[12,31]=86$

In the matrix Z_{11} replace $Z[i,j]$ with $Z[i,11]=11$.

Iteration 12:

$W[1,13]=52$ $W[2,13]=33$ $W[3,13]=30$ $W[4,13]=27$
$W[5,13]=24$ $W[6,13]=21$ $W[7,13]=18$ $W[8,13]=15$
$W[9,13]=12$ $W[10,13]=9$ $W[11,13]=6$ $W[13,17]=73$
$W[13,31]=89$

In the matrix Z_{12} replace $Z[i,j]$ with $Z[i,12]=12$.

Iteration 13:

$W[1,14]=55$ $W[2,14]=36$ $W[3,14]=33$ $W[4,14]=30$
$W[5,14]=27$ $W[6,14]=24$ $W[7,14]=21$ $W[8,14]=18$
$W[9,14]=15$ $W[10,14]=12$ $W[11,14]=9$ $W[12,14]=6$
$W[14,17]=76$ $W[14,31]=92$

In the matrix Z_{13} replace $Z[i,j]$ with $Z[i,13]=13$.

Iteration 14:

$W[1,15]=58$ $W[2,15]=39$ $W[3,15]=36$ $W[4,15]=33$
$W[5,15]=30$ $W[6,15]=27$ $W[7,15]=24$ $W[8,15]=21$
$W[9,15]=18$ $W[10,15]=15$ $W[11,15]=12$ $W[12,15]=9$
$W[13,15]=6$ $W[15,17]=79$ $W[15,31]=95$

In the matrix Z_{14} replace $Z[i,j]$ with $Z[i,14]=14$.

Iteration 15:

$W[1,16]=61$ $W[2,16]=42$ $W[3,16]=39$ $W[4,16]=36$
$W[5,16]=33$ $W[6,16]=30$ $W[7,16]=27$ $W[8,16]=24$
$W[9,16]=21$ $W[10,16]=18$ $W[11,16]=15$ $W[12,16]=12$
$W[13,16]=9$ $W[14,16]=6$ $W[16,17]=82$ $W[16,31]=98$

In the matrix Z_{15} replace $Z[i,j]$ with $Z[i,15]=15$.

⋮

Iteration 46:

$W[2,42]=83$ $W[2,43]=80$ $W[2,44]=77$ $W[2,45]=74$
$W[3,41]=83$ $W[3,42]=80$ $W[3,43]=77$ $W[3,44]=74$
$W[3,45]=71$ $W[4,40]=83$ $W[4,41]=80$ $W[4,42]=77$
$W[4,43]=74$ $W[4,44]=71$ $W[4,45]=68$ $W[5,39]=83$
$W[5,40]=80$ $W[5,41]=77$ $W[5,42]=74$ $W[5,43]=71$
$W[5,44]=68$ $W[5,45]=65$ $W[6,30]=86$ $W[6,38]=83$
$W[6,39]=80$ $W[6,40]=77$ $W[6,41]=74$ $W[6,42]=71$
$W[6,43]=68$ $W[6,44]=65$ $W[6,45]=62$ $W[7,29]=86$

$W[7,30] = 83$ $W[7,37] = 83$ $W[7,38] = 80$ $W[7,39] = 77$
$W[7,40] = 74$ $W[7,41] = 71$ $W[7,42] = 68$ $W[7,43] = 65$
$W[7,44] = 62$ $W[7,45] = 59$ $W[8,28] = 86$ $W[8,29] = 83$
$W[8,30] = 80$ $W[8,36] = 83$ $W[8,37] = 80$ $W[8,38] = 77$
$W[8,39] = 74$ $W[8,40] = 71$ $W[8,41] = 68$ $W[8,42] = 65$
$W[8,43] = 62$ $W[8,44] = 59$ $W[8,45] = 56$ $W[9,27] = 86$
$W[9,28] = 83$ $W[9,29] = 80$ $W[9,30] = 77$ $W[9,35] = 83$
$W[9,36] = 80$ $W[9,37] = 77$ $W[9,38] = 74$ $W[9,39] = 71$
$W[9,40] = 68$ $W[9,41] = 65$ $W[9,42] = 62$ $W[9,43] = 59$
$W[9,44] = 56$ $W[9,45] = 53$ $W[10,26] = 86$ $W[10,27] = 83$
$W[10,28] = 80$ $W[10,29] = 77$ $W[10,30] = 74$ $W[10,34] = 83$
$W[10,35] = 80$ $W[10,36] = 77$ $W[10,37] = 74$ $W[10,38] = 71$
$W[10,39] = 68$ $W[10,40] = 65$ $W[10,41] = 62$ $W[10,42] = 59$
$W[10,43] = 56$ $W[10,44] = 53$ $W[10,45] = 50$ $W[11,25] = 86$
$W[11,26] = 83$ $W[11,27] = 80$ $W[11,28] = 77$ $W[11,29] = 74$
$W[11,30] = 71$ $W[11,33] = 83$ $W[11,34] = 80$ $W[11,35] = 77$
$W[11,36] = 74$ $W[11,37] = 71$ $W[11,38] = 68$ $W[11,39] = 65$
$W[11,40] = 62$ $W[11,41] = 59$ $W[11,42] = 56$ $W[11,43] = 53$
$W[11,44] = 50$ $W[11,45] = 47$ $W[12,24] = 86$ $W[12,25] = 83$
$W[12,26] = 80$ $W[12,27] = 77$ $W[12,28] = 74$ $W[12,29] = 71$
$W[12,30] = 68$ $W[12,32] = 83$ $W[12,33] = 80$ $W[12,34] = 77$
$W[12,35] = 74$ $W[12,36] = 71$ $W[12,37] = 68$ $W[12,38] = 65$
$W[12,39] = 62$ $W[12,40] = 59$ $W[12,41] = 56$ $W[12,42] = 53$
$W[12,43] = 50$ $W[12,44] = 47$ $W[12,45] = 44$ $W[13,23] = 86$
$W[13,24] = 83$ $W[13,25] = 80$ $W[13,26] = 77$ $W[13,27] = 74$
$W[13,28] = 71$ $W[13,29] = 68$ $W[13,30] = 65$ $W[13,31] = 83$
$W[13,32] = 80$ $W[13,33] = 77$ $W[13,34] = 74$ $W[13,35] = 71$
$W[13,36] = 68$ $W[13,37] = 65$ $W[13,38] = 62$ $W[13,39] = 59$
$W[13,40] = 56$ $W[13,41] = 53$ $W[13,42] = 50$ $W[13,43] = 47$
$W[13,44] = 44$ $W[13,45] = 41$ $W[14,22] = 86$ $W[14,23] = 83$
$W[14,24] = 80$ $W[14,25] = 77$ $W[14,26] = 74$ $W[14,27] = 71$
$W[14,28] = 68$ $W[14,29] = 65$ $W[14,30] = 62$ $W[14,31] = 80$
$W[14,32] = 77$ $W[14,33] = 74$ $W[14,34] = 71$ $W[14,35] = 68$
$W[14,36] = 65$ $W[14,37] = 62$ $W[14,38] = 59$ $W[14,39] = 56$
$W[14,40] = 53$ $W[14,41] = 50$ $W[14,42] = 47$ $W[14,43] = 44$
$W[14,44] = 41$ $W[14,45] = 38$ $W[15,21] = 86$ $W[15,22] = 83$
$W[15,23] = 80$ $W[15,24] = 77$ $W[15,25] = 74$ $W[15,26] = 71$
$W[15,27] = 68$ $W[15,28] = 65$ $W[15,29] = 62$ $W[15,30] = 59$
$W[15,31] = 77$ $W[15,32] = 74$ $W[15,33] = 71$ $W[15,34] = 68$
$W[15,35] = 65$ $W[15,36] = 62$ $W[15,37] = 59$ $W[15,38] = 56$
$W[15,39] = 53$ $W[15,40] = 50$ $W[15,41] = 47$ $W[15,42] = 44$
$W[15,43] = 41$ $W[15,44] = 38$ $W[15,45] = 35$ $W[16,20] = 86$
$W[16,21] = 83$ $W[16,22] = 80$ $W[16,23] = 77$ $W[16,24] = 74$
$W[16,25] = 71$ $W[16,26] = 68$ $W[16,27] = 65$ $W[16,28] = 62$
$W[16,29] = 59$ $W[16,30] = 56$ $W[16,31] = 74$ $W[16,32] = 71$
$W[16,33] = 68$ $W[16,34] = 65$ $W[16,35] = 62$ $W[16,36] = 59$
$W[16,37] = 56$ $W[16,38] = 53$ $W[16,39] = 50$ $W[16,40] = 47$
$W[16,41] = 44$ $W[16,42] = 41$ $W[16,43] = 38$ $W[16,44] = 35$
$W[16,45] = 32$ $W[17,40] = 84$ $W[17,41] = 81$ $W[17,42] = 78$
$W[17,43] = 75$ $W[17,44] = 72$ $W[17,45] = 69$ $W[18,39] = 84$
$W[18,40] = 81$ $W[18,41] = 78$ $W[18,42] = 75$ $W[18,43] = 72$
$W[18,44] = 69$ $W[18,45] = 66$ $W[19,38] = 84$ $W[19,39] = 81$
$W[19,40] = 78$ $W[19,41] = 75$ $W[19,42] = 72$ $W[19,43] = 69$
$W[19,44] = 66$ $W[19,45] = 63$ $W[20,37] = 84$ $W[20,38] = 81$
$W[20,39] = 78$ $W[20,40] = 75$ $W[20,41] = 72$ $W[20,42] = 69$
$W[20,43] = 66$ $W[20,44] = 63$ $W[20,45] = 60$ $W[21,36] = 84$
$W[21,37] = 81$ $W[21,38] = 78$ $W[21,39] = 75$ $W[21,40] = 72$

$W[21,41] = 69$ $W[21,42] = 66$ $W[21,43] = 63$ $W[21,44] = 60$
$W[21,45] = 57$ $W[22,35] = 84$ $W[22,36] = 81$ $W[22,37] = 78$
$W[22,38] = 75$ $W[22,39] = 72$ $W[22,40] = 69$ $W[22,41] = 66$
$W[22,42] = 63$ $W[22,43] = 60$ $W[22,44] = 57$ $W[22,45] = 54$
$W[23,34] = 84$ $W[23,35] = 81$ $W[23,36] = 78$ $W[23,37] = 75$
$W[23,38] = 72$ $W[23,39] = 68$ $W[23,40] = 66$ $W[23,41] = 63$
$W[23,42] = 60$ $W[23,43] = 57$ $W[23,44] = 54$ $W[23,45] = 51$
$W[24,33] = 84$ $W[24,34] = 81$ $W[24,35] = 78$ $W[24,36] = 75$
$W[24,37] = 72$ $W[24,38] = 69$ $W[24,39] = 66$ $W[24,40] = 63$
$W[24,41] = 60$ $W[24,42] = 57$ $W[24,43] = 54$ $W[24,44] = 51$
$W[24,45] = 48$ $W[25,32] = 84$ $W[25,33] = 81$ $W[25,34] = 78$
$W[25,35] = 75$ $W[25,36] = 72$ $W[25,37] = 68$ $W[25,38] = 66$
$W[25,39] = 63$ $W[25,40] = 60$ $W[25,41] = 57$ $W[25,42] = 54$
$W[25,43] = 51$ $W[25,44] = 48$ $W[25,45] = 45$ $W[26,31] = 84$
$W[26,32] = 81$ $W[26,33] = 78$ $W[26,34] = 75$ $W[26,35] = 72$
$W[26,36] = 69$ $W[26,37] = 66$ $W[26,38] = 63$ $W[26,39] = 60$
$W[26,40] = 57$ $W[26,41] = 54$ $W[26,42] = 51$ $W[26,43] = 48$
$W[26,44] = 45$ $W[26,45] = 42$ $W[27,31] = 81$ $W[27,32] = 78$
$W[27,33] = 75$ $W[27,34] = 72$ $W[27,35] = 69$ $W[27,36] = 66$
$W[27,37] = 63$ $W[27,38] = 60$ $W[27,39] = 57$ $W[27,40] = 54$
$W[27,41] = 51$ $W[27,42] = 48$ $W[27,43] = 45$ $W[27,44] = 42$
$W[27,45] = 39$ $W[28,31] = 78$ $W[28,32] = 75$ $W[28,33] = 72$
$W[28,34] = 69$ $W[28,35] = 66$ $W[28,36] = 63$ $W[28,37] = 60$
$W[28,38] = 57$ $W[28,39] = 54$ $W[28,40] = 51$ $W[28,41] = 48$
$W[28,42] = 45$ $W[28,43] = 42$ $W[28,44] = 39$ $W[28,45] = 36$
$W[29,31] = 75$ $W[29,32] = 72$ $W[29,33] = 69$ $W[29,34] = 66$
$W[29,35] = 63$ $W[29,36] = 60$ $W[29,37] = 57$ $W[29,38] = 54$
$W[29,39] = 51$ $W[29,40] = 48$ $W[29,41] = 45$ $W[29,42] = 42$
$W[29,43] = 39$ $W[29,44] = 36$ $W[29,45] = 33$ $W[30,31] = 72$
$W[30,32] = 69$ $W[30,33] = 66$ $W[30,34] = 63$ $W[30,35] = 60$
$W[30,36] = 57$ $W[30,37] = 54$ $W[30,38] = 51$ $W[30,39] = 48$
$W[30,40] = 45$ $W[30,41] = 42$ $W[30,42] = 39$ $W[30,43] = 36$
$W[30,44] = 33$ $W[30,45] = 30$

In the matrix Z_{46} replace $Z[i,j]$with $Z[i,46] = 46$.

The following is the form of the W_{46} matrix from iteration 46 along with the Z_{46} matrix, complete in Table 2 and 3.

Table 2. Matrix W_{46} from 46 iteration

i\j	1	2	3	4	5	6	7	8	9	10	...	42	43	44	45	46
1	0	19	22	25	28	31	34	37	40	43	…	70	73	76	79	82
2	19	0	3	6	9	12	15	18	21	24	⋯	84	81	78	75	72
3	22	3	0	3	6	9	12	15	18	21	⋯	81	78	75	72	69
4	25	6	3	0	3	6	9	12	15	18	⋯	78	75	72	69	66
5	28	9	6	3	0	3	6	9	12	15	⋯	75	72	69	66	63
6	31	12	9	6	3	0	3	6	9	12	⋯	72	69	66	63	60
7	34	15	12	9	6	3	0	3	6	9	⋯	69	66	63	60	57
8	37	18	15	12	9	6	3	0	3	6	⋯	66	63	60	57	54
9	40	21	18	15	12	9	6	3	0	3	⋯	63	60	57	54	51
10	43	24	21	18	15	12	9	6	3	0	⋯	60	57	54	51	48
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
42	70	81	78	75	72	69	66	63	60	57	⋯	0	3	6	9	12
43	73	78	75	72	69	66	63	60	57	54	⋯	3	0	3	6	9
44	76	75	72	69	66	63	60	57	54	51	⋯	6	3	0	3	6
45	79	72	69	66	63	60	57	54	51	48	⋯	9	6	3	0	3
46	82	69	66	63	60	57	54	51	48	45	⋯	12	9	6	3	0

Table 3. Matrix Z_{46} from 46 iteration

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	. . .	42	43	44	45	46
1	1	2	2	3	4	5	6	7	8	9	…	41	42	43	44	45
2	1	2	3	3	4	5	6	7	8	9	...	46	46	46	46	16
3	2	2	3	4	4	5	6	7	8	9	...	46	46	46	46	16
4	3	3	3	4	5	5	6	7	8	9	...	46	46	46	46	16
5	4	4	4	4	5	6	6	7	8	9	...	46	46	46	46	16
6	5	5	5	5	5	6	7	7	8	9	...	46	46	46	46	16
7	6	6	6	6	6	6	7	8	8	9	...	46	46	46	46	16
8	7	7	7	7	7	7	7	8	9	9	...	46	46	46	46	16
9	8	8	8	8	8	8	8	8	9	10	...	46	46	46	46	16
10	9	9	9	9	9	9	9	9	9	10	...	46	46	46	46	16
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
42	41	46	46	46	46	46	46	46	46	46	···	42	43	43	44	45
43	42	46	46	46	46	46	46	46	46	46	···	42	43	44	44	45
44	43	46	46	46	46	46	46	46	46	46	···	43	43	44	45	45
45	44	46	46	46	46	46	46	46	46	46	···	44	44	44	45	46
46	45	16	16	16	16	16	16	16	16	16	···	45	45	45	45	46

To check whether the final result of the Floyd-Warshall algorithm is manually correct, Python is used to get all the possible shortest paths from the entrance a_1 to the parking lot b_1 to d_{16}, the syntax used is as follows (Bhavya, 2022)

```
INF = 9999
# Utility function to print solution
# W range
def printSolution(sumvertex, W):
    for i in range(sumvertex):
        for j in range(sumvertex):
            if(W[i][j] == INF):
                print("%7s" % ("INF"),end=" ")
            else:
                print("%1d\t" % (W[i][j]),end=' ')
        print(" ")
def floydWarshall(sumvertex, W):
    for k in range(Select all vertices as targets for selected origin):
      # select all vertices as origin one by one
       for i in range(sumvertex):
          # Select all vertices as targets for selected origin
            for j in range(sumvertex):
              # If vertex k is on the shortest path from i to j, then update
the value of W[i][j]
                W[i][j] = min(
                    W[i][j], W[i][k]+W[k][j])
printSolution(sumvertex, W)
W = [ #the contents of the cells in the matrix W0]
floydWarshall(46, W)
```

After the program is run it will get the same results as in Table 4.
The following table shows the shortest distance from the a_1 entrance to each parking lot with the shortest path.

Table 4. Results of the Shortest Distance and Path

Origin	Target	Distance(meters)	Path (a_1, …)
a_1	b_1	19	b_1
a_1	b_2	22	b_1, b_2

a_1	b_3	25	b_1, b_2, b_3
a_1	b_4	28	b_1, b_2, b_3, b_4
a_1	b_5	31	b_1, b_2, b_3, b_4, b_5
a_1	b_6	34	$b_1, b_2, b_3, b_4, b_5, b_6$
a_1	b_7	37	$b_1, b_2, b_3, b_4, b_5, b_6, b_7$
a_1	b_8	40	$b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8$
a_1	b_9	43	$b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9$
a_1	b_{10}	46	$b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}$
a_1	b_{11}	49	$b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11}$
a_1	b_{12}	52	$b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11}, b_{12}$
a_1	b_{13}	55	$b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11}, b_{12}, b_{13}$
a_1	b_{14}	58	$b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11}, b_{12}, b_{13}, b_{14}$
a_1	b_{15}	61	$b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11}, b_{12}, b_{13}, b_{14}, b_{15}$
a_1	c_1	21	c_1
a_1	c_2	24	c_1, c_2
a_1	c_3	27	c_1, c_2, c_3
a_1	c_4	30	c_1, c_2, c_3, c_4
a_1	c_5	33	c_1, c_2, c_3, c_4, c_5
a_1	c_6	36	$c_1, c_2, c_3, c_4, c_5, c_6$
a_1	c_7	39	$c_1, c_2, c_3, c_4, c_5, c_6, c_7$
a_1	c_8	42	$c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$
a_1	c_9	45	$c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9$
a_1	c_{10}	48	$c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}$
a_1	c_{11}	51	$c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11}$
a_1	c_{12}	54	$c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11}, c_{12}$
a_1	c_{13}	57	$c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11}, c_{12}, c_{13}$
a_1	c_{14}	60	$c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11}, c_{12}, c_{13}, c_{14}$
a_1	d_1	37	d_1
a_1	d_2	40	d_1, d_2
a_1	d_3	43	d_1, d_2, d_3
a_1	d_4	46	d_1, d_2, d_3, d_4
a_1	d_5	49	d_1, d_2, d_3, d_4, d_5
a_1	d_6	52	$d_1, d_2, d_3, d_4, d_5, d_6$
a_1	d_7	55	$d_1, d_2, d_3, d_4, d_5, d_6, d_7$
a_1	d_8	58	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8$
a_1	d_9	61	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9$
a_1	d_{10}	64	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}$
a_1	d_{11}	67	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}$
a_1	d_{12}	70	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}, d_{12}$
a_1	d_{13}	73	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}, d_{12}, d_{13}$
a_1	d_{14}	76	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}$
a_1	d_{15}	79	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}$
a_1	d_{16}	82	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}, d_{16}$

4. Conclussion

The use of the Floyd-Warshall Algorithm is more effective in determining the shortest path in the parking layout, because it gets the shortest distance that is more accurate and calculates all the shortest distances at the vertex and knows all the paths traversed by the shortest distance from the origin vertex to the target vertex. So as to get the shortest distance from the parking entrance (a_1) to all parking lots from (b_1) to (d_{16}). From the entrance (a_1) to the parking lot (b_1) is the shortest distance of all parking lots with a length of 19.

References

Azis, H., Mallongi, R., Lantara, D., & Salim, Y. (2018). Comparison of Floyd-Warshall Algorithm and Greedy Algorithm in Determining the Shortest Route. *2018 2nd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, 294–298.

Bhavya. (2022). All Pair Shortest Path Problem in Python. Retrieved June 15, 2022, from https://pythonwife.com/all-pair-shortest-path-problem-in-python/

Fatmala, F., Yudatama, U., & Burhanuddin, A. (2019). Panduan Jalur Angkutan Umum Menggunakan Algoritma Floyd Warshall. *Jurnal Komtika (Komputasi Dan Informatika)*, *3*, 1–9. https://doi.org/https://doi.org/10.31603/komtika.v3i1.3462

Fortuna, C., Sandra, K., Sipil, J. T., & Trisakti, U. (2020). Kebijakan Strategi Parkir (Studi Kasus : Ibu Kota Metropolitan Jakarta) Parking Strategy Policy (Case Study : Metropolitan Jakarta), (September), 103–108.

Jayanti, N. K. D. A. (2017). Penggunaan Algoritma Floyd Warshall Dalam Masalah Jalur Terpendek Pada Penentuan Tata Letak Parkir. *In Seminar Nasional Informatika (SNIf)*, *1*, 75–81.

Kumar, R.., & Kumar, M. (2010). Exploring Genetic Algorithm for Shortest Path Optimization in Data Networks, *10*(11), 8–12.

Ramadhan, Z., Siahaan, A. P. U., & Mesran, M. (2018). Prim and Floyd-Warshall Comparative Algorithms in Shortest Path Problem, 47–58.

Rosen, K. H. (2012). *Discrete Mathematics and Its Discrete and Its Seventh Edition*.

Vasudev, C. (2006). *Graph theory with applications*. New Age International.