Floyd-warshall algorithm to determine the shortest path based on android

Ramadiani*, D Bukhori, Azainil and N Degen

1 Faculty of Computer Science and Information Technology Mulawarman University, Samarinda, East Kalimantan, Indonesia
2 Faculty of Teacher Training and Education Mulawarman University, Samarinda, East Kalimantan, Indonesia

*Corresponding Author: ilkom.ramadiani@gmail.com

Abstract. The development of technology has made all areas of life easier now, one of which is the ease of obtaining geographic information. The use of geographic information may vary according to need, for example, the digital map learning, navigation systems, observations area, and much more. With the support of adequate infrastructure, almost no one will ever get lost to a destination even to foreign places or that have never been visited before. The reasons why many institutions and business entities use technology to improve services to consumers and to streamline the production process undertaken and so forth. Speaking of the efficient, there are many elements related to efficiency in navigation systems, and one of them is the efficiency in terms of distance. The shortest distance determination algorithm required in this research is used Floyd-Warshall Algorithm. Floyd-Warshall algorithm is the algorithm to find the fastest path and the shortest distance between 2 nodes, while the program is intended to find the path of more than 2 nodes.

1. Introduction

A donation program is one of the ideas developed and run by social institutions with the aim of collecting as much aid as possible to be distributed to refugees and victims of natural disasters. In order to be immediately channeled, the collection of donations must be carried out with a fast process that requires the mechanism as effectively and efficiently as possible. We need a medium that adapts to current trends, and one solution is to make an application based android donation pickup using the application of digital maps. Particularly in this research, the shortest route selection algorithm named Floyd-Warshall Algorithm is applied to a digital map of donation pickup program, which is expected to streamline the distance that must be passed by donation pickup institution to donor location.

1.1. Geographic Information System (GIS)

Technological advancements have given people more access to more difficult information. One of the ease that can be felt is access to a spatial and geographic area or geographic information that currently has been digitized or commonly called digital mapping.

Long ago, the use of maps is very important in many areas of life. For example, visiting a new place in the city or out of town is a very reasonable thing. The problem is sometimes some people difficult to recall the location of a place or direction of the route to the place. So, some people take a long time to rediscover [1]. In the event of a disaster, the inhabitants of the hospital are mostly running around with no direction or guidance. There are several empty areas that can be used as assembly points, ie in the north, west, and south. However, this area of utilization has not been maximized
because most hospital residents are unaware of the existence of the assembly point. With the evacuation map, expected during a disaster, residents can easily follow the direction of the evacuation arrow to the designated place. The simulation model will also be performed to evaluate the direction of the grooves in the evacuation map being applied [2].

For this reason, the importance of developing Geographic Information System (GIS) capable of storing or documenting spatial information in digital form, so that in various practice the use of the current map can be done more easily. Features not previously included in the mapping can be manifested such as navigation, GPS, more specific place/location search, and so on.

1.2. Android Operating System
Android is a popular and licensed free mobile platform (open source) that has taken over the mobile phone world on a large scale. Most of the smartphone industry chose to use Android as its operating system, so quickly android became the trend center among smartphone users. The developers of programs and applications are competing to create various android apps, so the development of Android in the world into a wave of great changes that have even changed the lifestyle of the crowd [3].

As an open platform, everyone is free to develop application and customization of the operating system from Android without worrying about licensing issues as Google releases Android codes under the Apache license, a software license and an open standard mobile device. Android Studio is one of the IDE for Android development which is a spin-off version of IntelliJ Java IDE. Android Studio is the latest solution from Google Inc. in the middle so crowded android development needs of late. Android Studio has a new unified build system called Gradle that is fully integrated to support the flexibility of the Android development process [4].

2. Method
Floyd-Warshall's algorithm is one of the variants of dynamic programming, a method that solves problems by looking at the solution to be obtained as an interrelated decision. This means that solutions are formed from solutions that come from the previous stage and there is the possibility of more than one solution [5].

The problem solving that is the focus of the discussion in this study using the Floyd-Warshall algorithm is the shortest path selection between 2 or more marker positions or location markers in a digital map. The Floyd-Warshall algorithm compares each route variation with the distance from each point to the destination point and accumulates the distance between the nodes (intersection points) that are passed. The line that connects between nodes has a value in the magnitude of the number indicating the distance weight. The Floyd-Warshall algorithm will take each line as a calculation material in determining the shortest route. The solution of the shortest route selection problem using the Floyd-Warshall Algorithm will be described in following steps [6-7]:

Step 1: Delete all loop paths
Delete all loop paths in the node set. Each path that starts and ends on the same node is a loop as in node A. A loop path is not needed in route selection because it does not connect more than one node.

• Step 2: Erase the parallel path
Remove parallel paths or paths more than one between two nodes and leave only one path with the smallest weights. Removing the parallel path and making it a single path will streamline route selection.

• Step 3: create a matrix with the Distance Table and Sequence Table
To find the shortest path between two nodes, there are several requirements in the form of a matrix with two tables called Distance Table (D) and Sequence Table (S). Distance Table (D) contains the value of the distance between two nodes, and the Sequence Table (S) contains the name of the node that serves to indicate which nodes are passed and included in the route. If a routing graph has an
number of ns then table D and table S will have n rows and n columns (table n x n). To do the problem solving used iteration method or looping k as much as n times.

K = iteration
n = number of iterations
Dk = Table D on the k-iteration
Sk = Table S on the k-iteration
dij = distance between node i and node j
Cij = cell holding d value between node i and j

There is a 0 iterations condition or k = 0 is the state before any change of value or initial condition of the table based on the distance weight on the original route graph. Then in Table D0 inserted a value of the weighted distance between every two nodes in accordance with the route graph. In cells that have the same row names and column names, a zero value (0) indicates the absence of a distance weight. Whereas between 2 nodes that do not have a direct connecting line (inserted value) is inserted value infinity (∞).

For Table S0 we include values in the name of the node in which the node must be passed or is in the middle of node i and node j called pit stop node. However, on the 0th iteration, the value in table S0 is still adjusted to table D0 which uses the direct line so it does not have a pit stop node. The value entered into table S0 is the value of the original node. Especially in cells that have value ∞ or 0 in table D0 then adjust to the condition value entered in table S0 is strip (-).

• Step 4: Iterate
The iteration is done as many times in sequence and each subsequent matrix of the input value is heavily dependent on the previous matrix. In iteration 1 and so on begin to enter value pit stop node to find the combination of routes along with the distance variation. More detailed information regarding the use of pit stop nodes on each iteration can be seen in Table 1.

Table 1. PitstopNode at Each Iterations

| Iterations Number | Table  | Pitstop (p) |
|-------------------|--------|-------------|
| 0                 | D0, S0 | -           |
| 1                 | D1, S1 | A           |
| 2                 | D2, S2 | B           |
| 3                 | D3, S3 | C           |
| 4                 | D4, S4 | D           |
| N                 | Dn, Sn | pn          |

The purpose of using a pit stop node is to compare the distance (dij) from node i to node j. The smallest value of the result of the comparison is what will be the value in the table in the next iteration. Mathematically filling the value in the 1st iteration table and so on using the formula as follows:

\[ d_{ij} = \min \{ (d_{ij})^{k-1}, (d_{ip})^{k-1} + (d_{pj})^{k-1} \} \]  

(1)

The 1st iteration begins by creating D1and S1 tables and entering values based on the comparison results using the above formula. Similar to Table D0 and S0, in cells with the initial node and the same destination node (Cij | i = j) it loads zero (0) and strip (-) values. Next, each cell of the row and column A in Tables D1 and S1 is filled with the jo from Table D0 and S0. This is because the 1st iteration uses node A as a pit stop (p) in which the accumulated distance will not gain an addition so that the value will be equal to d_{ip} + d_{jp}. As for the remaining cells are filled with values in accordance with the formula, namely by comparing the value of the dip with d_{ip} + d_{jp} and find the minimum value.

In the Floyd-Warshall Algorithm, if a numeric value is confronted with an infinite value (∞) then the minimum value of a number is determined because it is assumed that the infinity value (∞) is always more likely. The same is done for other remaining cells, all looking for the least possible distance if the node travel route is always through the pit stop A (p = A) node. Any change of value is
found in table $D_1$, in table $S_1$ is also given a change of value by filling the value according to the pit stop node in the iteration. The iteration is stopped at the $n$th iteration as the limit of the number of iterations to be performed in accordance with the number of nodes present. Back to step making of matrix which explains that Table $D$ is Distance Table (distance) and Table $S$ is Sequence Table (node name), then Table $D$ and Table $S$ which is requirement to find the shortest path between nodes is Table $D_n$ and Table $S_n$.

- Step 5: check the shortest path using Table $D$ and Table $S$
In the last step, it is about how to use Table $D$ and Table $S$ to find the shortest path between nodes. Each value in the rows and columns connected in Table $D$ is the value of the shortest distance between the two connected nodes so that it is obtained from table $D$ the value of the $dij$. After knowing the shortest distance, the next thing to know is which line is used. For that, we will switch to Table $S$ with respect to the value of $C_{ij}$ in it. With regard to the shortest distance ($d$), it will be known the route of the node to be passed.

3. Result and Discussion
The use of Floyd-Warshall algorithm in this research will use case study of donation picking from donors located in Samarinda City area of East Kalimantan.

3.1. Sample of Research
Used 20 respondents as the research sample. This number is an estimate of the number of requests for picking up daily donations.

| Table 2. The Nodes of Research Sample |
|---------------------------------------|
| **Node** | **Nama** | **Latitude** | **Longitude** |
|----------|----------|--------------|---------------|
| 0 | Kantor KNRP Kaltim | -0.5046652 | 117.1082298 |
| 1 | M. Sabransyah | -0.478543 | 117.151212 |
| 2 | Richa Rosmita | -0.4722673 | 117.1508941 |
| 3 | E. Mutmayanti | -0.4512754 | 117.1675302 |
| 4 | Lasiyo | -0.4979957 | 117.1207656 |
| 5 | MegaPutri | -0.4536098 | 117.1585432 |
| 6 | Suhartanto | -0.4623663 | 117.1164232 |
| 7 | HadiMulyadi | -0.4772829 | 117.1333146 |
| 8 | Irvana | -0.458934 | 117.1593112 |
| 9 | Windy | -0.4618985 | 117.1839463 |
| 10 | Thamrin | -0.4527096 | 117.132746 |
| 11 | Siti Fatimah | -0.4944913 | 117.1545706 |
| 12 | M. Ubaid | -0.4743301 | 117.1802164 |
| 13 | Aditya H. | -0.4778609 | 117.127978 |
| 14 | AyuFadhilah | -0.4631318 | 117.1499509 |
| 15 | TopanYudha | -0.473763 | 117.163867 |
| 16 | Firza | -0.5011503 | 117.1417186 |
| 17 | Asmiliyani | -0.47919 | 117.124319 |
| 18 | AzrinaNindia | -0.489814 | 117.107383 |
| 19 | Robby A. | -0.4895625 | 117.1407127 |
| 20 | Nur Wahidah | -0.464537 | 117.153216 |
3.2. Route Graph

After the location or coordinates of the donors obtained then will form a routing graph that shows the connection between the nodes (the position of each donor). With Google Maps API and Google service, each node symbolized with a marker will appear in the fragment folder. Each marker has a weighted distance to another marker in a different location. Visually the graph of the formed route can be seen in Figure 1.

| Table 3. Distance Table from Node 0 to 20 |
|--------------------------------------------|
| Number | Distances |
|-------|-----------|
| 0     | 0         |
| 1     | 5588      |
| 2     | 3945      |
| 3     | 3606      |
| 4     | 4118      |
| 5     | 4765      |
| 6     | 7609      |
| 7     | 9664      |
| 8     | 6385      |
| 9     | 5279      |
| 10    | 6531      |
| 11    | 7073      |
| 12    | 3748      |
| 13    | 3338      |
| 14    | 3338      |
| 15    | 3338      |
| 16    | 3338      |
| 17    | 3338      |
| 18    | 3338      |
| 19    | 3338      |
| 20    | 3338      |

With the use of Google Maps API then the creation of Sequence tables becomes very simple because basically, the weight of the distance drawn from the line between the marker is already the shortest distance selection.

3.3 Route Selection Algorithm

In order for the purpose of research and the making of this donation pickup program can be achieved, it takes a method or a path or route determination algorithm. Basically, the Floyd-Warshall algorithm is the algorithm for finding the fastest path and the shortest distance between 2 nodes, while the donation pickup program is intended to find paths between more than 2 nodes [6][7].

Theoretically, determining the route can be done by finding the shortest distance from one node to another node in the sequence. Because the actor who runs the pick-up system is KNRP Kaltim, the initial node of the route starts from the KNRP office in East Kalimantan. In Table D, the location or coordinates of the KNRP East Kalimantan office marker are located at node 0. The problem is on the determination of which next node will be the second, third, and so on markers. For that, it takes a special algorithm as follows:

- An asted loop is performed to check the smallest value in each row of Table D.
  
\[ K_1, K_2, K_3, K_4, \ldots K_n \]
b. For each c where i = j changed its value to a high value, in this study used the number 99999 meters or 99 km, assuming there is no distance between 2 nodes in the city of Samarinda (Area 712km$^2$) that exceeds 99 km. $C_{i j} = 99999$

c. Perform a value comparison of each cell within each row of Table D with the formula:
\[ d_{ij} = \min \{ C_{i0}, C_{i1}, C_{i2}, \ldots C_{in} \} \]

d. Each value of the column with the smallest distance in the previous iteration is replaced with a value of 99999. $C_{j=(j)^{k-1}, \ldots, C_{ij} = 99999}$.

The creation of distance tables will be based on the weighted distance information between the nodes of the route graph formed using the Google Maps API in Figure 1 and Figure 2.

![Figure 1. The Route Graph Between the Nodes](image1)

![Figure 2. Application Output](image2)

### 3.4 Testing Program

To test whether the program has been created and all the output generated from the program is appropriate, then required comparative data. Using the formula contained in the route selection algorithm, we get:

| Iteration | Distance value between node i and j |
|-----------|-----------------------------------|
| $K_1$     | $d_{0j} = \min \{ 99999, 5588, 5948, 8855, 1578, 7952, 4765, 4118, 7609, 9664, 6385, 5279, 8686, 3690, 6531, 7073, 3748, 3338, 1644, 3982, 6690 \} = 1578$ |
| $K_2$     | $C_{00} = 99999$ $d_{ij} = \min \{ 99999, 4014, 4397, 7333, 99999, 6463, 3969, 2682, 6088, 8086, 5197, 378, 2, 7116, 2366, 5041, 5495, 2358, 2116, 1742, 2408, 5170 \} = 1742$ |
| $K_3$     | $C_{180} = 99999$ $C_{184} = 99999$ $d_{18j} = \min \{ 99999, 5035, 5217, 7936, 99999, 5691, 3197, 3201, 6713, 9064, 5013, 527, 8, 8286, 2646, 5581, 6533, 4022, 2221, 99999, 3710, 5817 \} = 2221$ |

$K_2, K_3, \ldots, K_{20}$
In the same way, we get the d value from the next iteration up to the last iteration. Manual calculations of the overall iteration performed result in the selection of routes with the shortest weights in sequence as follows:

| Nodes i to j | Distance     |
|------------|--------------|
| 0 → 4      | 1578 meters  |
| 4 → 18     | 1742 meters  |
| 18 → 17    | 2221 meters  |
| 17 → 13    | 433 meters   |
| 13 → 7     | 597 meters   |
| 7 → 19     | 1588 meters  |
| 19 → 16    | 1286 meters  |
| 16 → 11    | 1608 meters  |
| 11 → 1     | 1802 meters  |
| 1 → 2      | 694 meters   |
| 2 → 20     | 892 meters   |
| 20 → 14    | 395 meters   |
| 14 → 8     | 1140 meters  |
| 8 → 5      | 594 meters   |
| 5 → 3      | 1033 meters  |
| 3 → 9      | 2172 meters  |
| 9 → 12     | 1435 meters  |
| 12 → 15    | 1821 meters  |
| 15 → 10    | 4125 meters  |

From the manual data calculation formed a route pattern in the sequence: 0-4-17-13-7-19-16-11-1-2-20-14-8-5-3-9-12-15-10-6. This pattern shows the route that must be taken and the order of donor locations that must be visited starting from node 0 to node 6. As for the total distance from node 0 to node 6 is: 0 → 6 = 1578 + 1742 + 2221 + 433 + 1802 + 694 + 892 + 395 + 1140 + 594 + 1033 + 2172 + 1435 + 1821 + 4125 + 2158 = 29314 meters

After the manual testing that has been done, it is found that conformity with the output of the donation pickup program as shown in figure 2 and the program making in this research has resulted in a donation pickup route to 20 donor markers scattered throughout the area in Samarinda with a total distance of 29,314 meters or about 29 km follows the line path of the route to be skipped shown in the map fragment view.

4. Conclusion
The design and implementation carried out to get some conclusions that can be submitted as a result of research, among others; A digital map from Google Maps can be used in making a donation pickup application. The donor location can be poured into a marker form on Google Maps so that the purpose of picking up donations becomes clearer and from the existing 20 donor nodes or markers can be found the best route that has the shortest distance using the Floyd-Warshall algorithm and the route-specific algorithm.

Acknowledgments
Authors wishing to acknowledge for Rector and the Dean of Computer Science and Information Technology Faculty, Mulawarman University through Vice Rector for Academic who has provided funding the research stimulant.
References

[1] Hati, GM, Suprayogi, Andri, S and Bandi 2013Aplikasi Penanda Lokasi Peta Digital Berbasis Mobile GIS pada Smartphone Android *Jurnal Geodesi Undip* 2(4)pp 26-40

[2] Iftadi, Irwan, Jauhari, Wahid A &Nugroho, Beny 2011Perancangan Peta Evakuasi Menggunakan Algoritma Floyd-Warshall untuk Penentuan Lintasan Terpendek *Performa* 10(2) pp 95-104

[3] Annuzi, J., Darcey, L., and Conder, S. (n.d.) *Introduction to android application development*.

[4] Grant, K. and Haseman, C. 2014 *Beginning Android programming*. [S.l.]: PeachPit.

[5] Novandi, Raden AD. 2017 Perbandingan Algoritma Dijkstra dan Algoritma Floyd-Warshall dalam Penentuan Lintasan Terpendek (Single Pair Shortest Path). *Makalah IF2251 Strategi Algoritmik*, 2, pp. 1-5

[6] Gutenschwager, Radtke, Völker, and Zeller 2012 The Shortest Path Comparison of Different Approaches and Implementations for the Automatic Routing of Vehicles. Proc., of the 2012 Winter Simulation Conference, pp 3312-3323.

[7] Kairanbay M, Hajar M J. 2013 A Review and Evaluations of Shortest Path Algorithms, *International Journal of Science & Technology Research (IJSTR)* 2(6) June, ISSN 2277-8616 pp 96-104