Menentukan Jarak Tercepat Menuju UPJ Menggunakan Sepeda Motor

1st Anayah Hanifah Elsyaf, 2nd Anjaysnita Ibanezia Satria, 3rd Ari Gunawan,
4th Muhammad Bagoes Wicaksono, 5th Triana Micku Kuswara Putri.
Department of Informatics, Pembangunan Jaya University

***Abstract*—*Determine and identify the fastest route to the place that we wanted to go, is the most important thing to optimize our time when we want to travel somewhere. Especially for every UPJ (Pembangunan Jaya University) student, of course they come from various regions. In this research, we build a system to determine the fastest route to UPJ from the place where they live with Greedy Algorithm. This research was conducted in several stage, that is data grouping and application of data to find the total number of paths that will be taken by students. With the determination of the shortest operating path from the first node to the destination node. Because each nodes has their own distance value, which has been specified and used to give the best fastest route to optimize student time on the way to UPJ.*

Keywords—Greedy Algorithm, Shortest Path, Time Optimization.

I. Introduction

When we determining the fastest route to reach the place that we wanted to go, it is becoming the most important thing that we need the most, because there is a connection between those things with time optimizing. The fastest route can be identified as the minimum value in a track, because it is a value that come from the entire form of the existing track. Each person has their own different portion of time, even more, time becomes the most important thing in their daily of life. With the fastest route, it can help you a lot, if you work on something, or a project, it will be effective in optimizing time when we headed from initial distance, into the place we wanted to go.

When we choose the destination by selecting the right track, and we go through some of the existing track options, the distance of each track will be different. We use Greedy Algorithm to make it easier when we want to select the existing track options. It can helps every student who have to travel quite a long distance and take a longer time too.

This research make a simulation of a shortest path search with Greedy Algorithm, and with a scheme that has been defined and described. By applying Greedy Algorithm, we can find the fastest track to get to UPJ from the place where those student live, and we can go to the college more faster.

II. Theoretical Basis

A. What is Graph?

Graph is a discrete structure consisting of a knot (Vertex) and a side (Edge). Graph is a set pairs of (V,E) where v is a non-empty set from a vertex, and E is a set that connects a pair of knots inside the graph [2].

Graph can be divided into 2 :

1) Directed Graph

Directed Graph is a graph that whose side has a clear directional orientation and a directed side, and its called an (arc). Look at the figure 1.1 below :

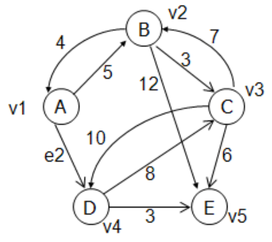


Figure 1.1 Directed Graph
Source : https://aerfanpratomo95.wordpress.com

2) Undirected Graph

Undirected Graph is a graph that whose side doesn't have any direction. Undirected Graph ignores the order of pairs of vertices that has been connected by sides. Look at the figure 1.2 below:

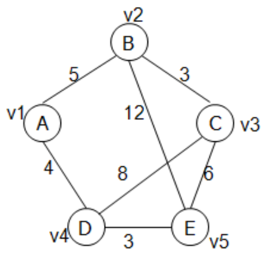


Figure 1. 2 Undirected Graph
Source : https://aerfanpratomo95.wordpress.com

B. Shortest Path

Shortest path is the path that passing around from one node, into another node who has a bigger value, or the value on the side whose their final value comes from the first node, into the last node which is the smallest. Shortest Path is a minimum path that we need for reach a place from another place. What is meant by Shortest Path is can be searched by using graph. A graph that we used, is a graph that has a weight value, which is every side of those graph has a weight value. [1].

C. Greedy Algorithm

Greedy Algorithm has an approach to build a solution incrementally through an ever-expanding sequence until the solution to the problem that has been reached [3]. Greedy Algorithn can be defined as implementation to optimize the distance, for instance like determine the closest distance.

The optimization problem in the context of the Greedy Algorithm is composed of components, as follows [4] :

a) Candidate Set (C) : The set that contains elements forming a solution. At each step, one candidate is taken from the set.
b) Solution Set (S) : Solution set is a set that contains elements of problem solving solutions.
c) Selection Function : A function that at each step, selects the most likely candidate to reach the optimal solution. Candidates who have been selected in a step, are never considered again in the next step.
d) Eligibility Function : A function that checks whether a selected candidate can provide a viable solution, that is the candidate together with the set of solutions that have been formed to not to violate the existing constraints.
e) Objective Function : Objective function is a function that maximizes or minimizes the value of the solution.

III. METHODOLOGY

In determining the fastest path, the autors searched the path coordinate data through the help of Google Maps, in order to find out the path to be traversed and determine the fastest path precisely using the Greedy Algorithm. In addition, the autors use several stages such as data grouping and data application to determine the total path that will be traversed.

The coordinate data of the fastest path that has been obtained from the data collection stage, will act as a vertex, while the distance to be traversed is taken from Google Maps, which will act as an edge. The solution in this study is described by graph and solved by calculating the Greedy Algorithm, which will produce the shortest total path from the initial position to the destination.

From the results of data calculations using the Greedy Algorithm, it can be seen which path will provide the shortest path to be passed by UPJ students. The result of this stage, is that, the data obtained can be applied to the Greedy Algorithm and can create solutions to research problems that can be implented. The flowchart of the Greedy Algorithm method (Figure 1.3) and the user process algorithm selects the path (Figure 1.4), as follows :

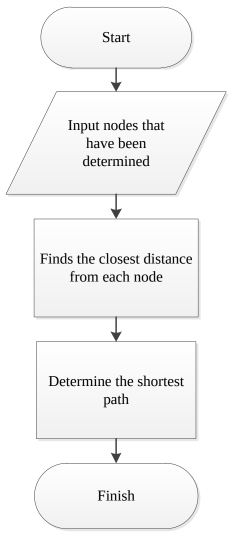

Start
Input nodes that have been determined
Finds the closest distance from each node
Determine the shortest path
Finish

Figure 1.3 Greedy Algorithm Flowchart

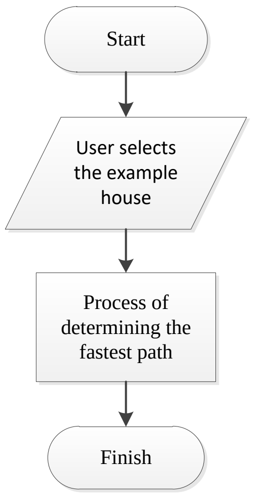

Start
User selects the example house
Process of determining the fastest path
Finish

Figure 1.4 Flowchart Of The Shortest Path Selection Process

IV. RESULTS AND DISCUSSION

There are several examples of the results of the Greedy Algorithm in determining the fastest path which is divided into several parts, according to different starting positions and the same ending position, but with different amounts of distance, as follows :

a) Example 1 : Anayah's House → UPJ
b) Example 2 : Ari's House → UPJ
c) Example 3 : Banez House → UPJ
d) Example 4 : Bagoes's House → UPJ
e) Example 5 : Triana's House → UPJ

To implement the calculation of the Greedy Algorithm with the data that has been obtained, figure 1.3 is the optimal trip sequence based on the Greedy Algorithm in accordance with the examples above. In that way, the author can find out whose house has the fastest route to get to UPJ first, as follows :

TABLE I. SELECTION OF EXAMPLE 1

	A	B	C	D	E	F	G	H
A	-	-	2.6	5.0	-	-	-	-
B	-	-	-	0.7	-	-	-	2.5
C	2.6	-	-	-	2.8	-	-	-
D	5.0	0.7	-	-	-	3.3	-	-
E	-	-	2.8	-	-	4.4	-	-
F	-	-	-	3.3	4.4	-	-	2.7
G	-	-	-	-	-	-	-	2.5
H	-	2.5	-	-	-	2.7	2.5	-

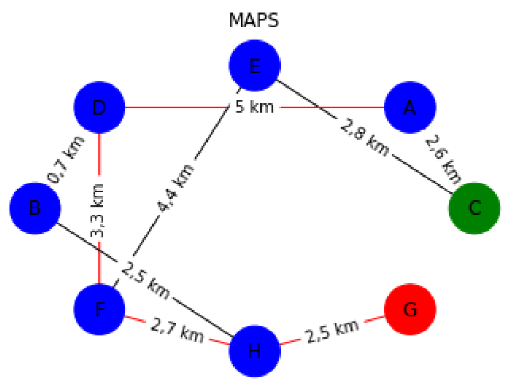


Figure 1.5 Graph Example 1

Based on the table above, the shortest travel path that must be taken from node C (Anayah's House) to node G (UPJ) is through node C-A-D-B-H-G. The calculation of the total edge that has been optimized with the Greedy Algorithm is a follows :

Next node : C-A | Distance : 2.6 KM
Next node : A-D | Distance : 5 KM
Next node : D-B | Distance : 0.7 KM
Next node : H-G | Distance : 2.4 KM
Mileage : 13.2 KM

TABLE II. SELECTION OF EXAMPLE 2

	A	B	C	D	E	F	G
A	-	1.4	-	-	-	8.5	-
B	1.4	-	2.9	2.1	-	-	-
C	-	2.9	-	-	1.4	-	-
D	-	2.1	-	-	2.3	-	-
E	-	-	1.4	2.3	-	-	1.3
F	8.5	-	-	-	-	-	7.3
G	-	-	-	-	1.3	7.3	-

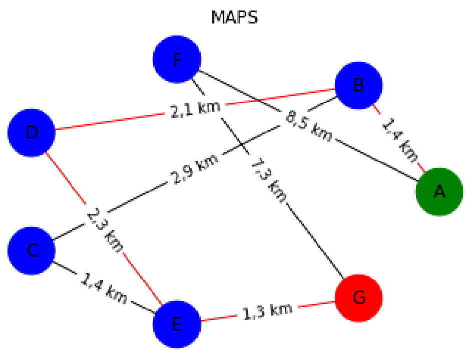


Figure 1.6 Graph Example 2

Based on the table above, the shortest travel path that must be taken from node A (Ari's House) to node G (UPJ) is through node A-B-D-E-G. The calculation of the total edge that has been optimized with the Greedy Algorithm is a follows :

Next node : A-B | Distance : 1.4 KM
Next node : B-D | Distance : 2.1 KM
Next node : D-E | Distance : 2.3 KM
Next node : E-G | Distance : 1.3 KM
Mileage : 7.1 KM

TABLE III. SELECTION OF EXAMPLE 3

	K	L	M	N	O	P	Q
K	-	0.7	-	-	-	-	-
L	0.7	-	1.4	-	-	2.5	-
M	-	1.4	-	2.6	-	-	-
N	-	-	2.6	-	1.9	-	1.5
O	-	-	-	1.9	-	-	-
P	-	2.5	-	-	-	-	1.4
Q	-	-	-	1.5	-	1.4	-

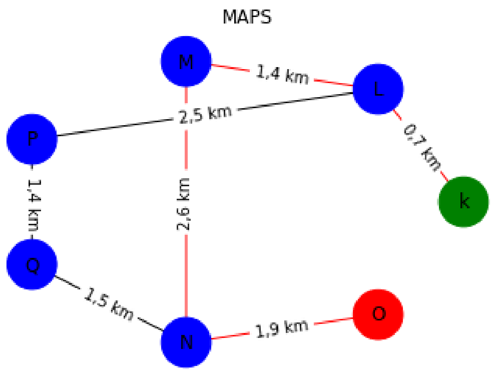


Figure 1.7 Graph Example 3

Based on the table above, the shortest travel path that must be taken from node K (Banez's House) to node O (UPJ) is through node K-L-M-N-O. The calculation of the total edge that has been optimized with the Greedy Algorithm is a follows :

Next node : K-L | Distance : 0.7 KM
Next node : L-M | Distance : 1.4 KM
Next node : M-N | Distance : 2.6 KM
Next node : N-O | Distance : 1.9 KM
Mileage : 6.6 KM

TABLE IV. SELECTION OF EXAMPLE 4

	A	B	C	D	E	F	G	I	J
A	-	3.7	-	-	-	-	-	-	-
B	3.7	-	3.5	-	-	-	-	-	4.7
C	-	3.5	-	1.4	-	1.8	-	-	-
D	-	-	1.4	-	1.6	-	-	-	-
E	-	-	-	1.6	-	1.2	2.6	1.4	-
F	-	-	1.8	-	1.2	-	-	-	-
G	-	-	-	-	2.6	-	-	-	3.2
I	-	-	-	-	1.4	-	-	-	-
J	-	4.7	-	-	-	-	3.2	-	-

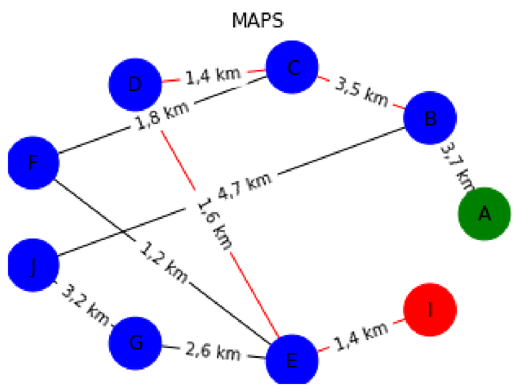


Figure 1.8 Graph Example 4

Based on the table above, the shortest travel path that must be taken from node A (Bagoes's House) to node I (UPJ) is through node A-B-C-D-E-I. The calculation of the total edge that has been optimized with the Greedy Algorithm is a follows :

Next node : A-B | Distance : 3.7 KM
Next node : B-C | Distance : 3.5 KM
Next node : C-D | Distance : 1.4 KM
Next node : D-E | Distance : 1.6 KM
Next node : E-I | Distance : 1.4
Mileage : 11.6 KM

TABLE V. SELECTION OF EXAMPLE 5

	R	S	T	U	V	W	X
R	-	1.9	-	-	2.1	2.6	-
S	1.9	-	1.9	-	-	-	-
T	-	1.9	-	2.3	-	-	-
U	-	-	2.3	-	3.1	2.35	2.1
V	2.1	-	-	3.1	-	-	-
W	2.6	-	-	2.35	-	-	-
X	-	-	-	2.1	-	-	-

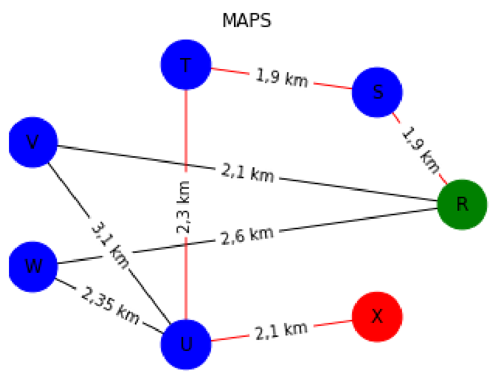


Figure 1.9 Graph Example 5

Based on the table above, the shortest travel path that must be taken from node R (Triana's House) to node X (UPJ) is through node R-S-T-U-X. The calculation of the total edge that has been optimized with the Greedy Algorithm is a follows :

Next node : R-S | Distance : 1.9 KM
Next node : S-T | Distance : 1.9 KM
Next node : T-U | Distance : 2.3 KM
Next node : U-X | Distance : 2.1 KM
Mileage : 8.2 KM

From the calculation of the Greedy Algorithm, it can be written in the Phyton programming language, which will later be applied to the system as follows :

```
import networkx as nx
import matplotlib.pyplot as plt


def shortest_path(graph, source, dest):
  result = []
  result.append(source)
  jumlah = 0
  while dest not in result:
    current_node = result[-1]


    local_max = min(graph[current_node].values())
    for node, weight in graph[current_node].items():
      if weight == local_max:
        print ("Node Selanjutnya :",node,"| Jarak :",
weight)
        jumlah = jumlah + weight
        result.append(node)
  print ("Jarak Tempuh :", jumlah,"KM")


return result
```

```
rumah_anayah = {'C': {'A': 2.6, 'E': 2.8},
    'A': {'D': 5},
    'D': {'B': 0.7, 'F': 3.3},
    'E': {'F': 4.4},
    'F': {'H': 2.7},
    'B': {'H': 2.5},
    'H': {'G': 2.4}}
rumah_ari = {'A': {'B': 1.4, 'F': 8.5},
    'B': {'C': 2.9, 'D': 2.1},
    'D': {'E': 2.3},
    'C': {'E': 1.4},
    'E': {'G': 1.3},
    'F': {'G': 7.3}}
rumah_banez = {'K': {'L': 0.7},
    'L': {'M': 1.4, 'P': 2.5},
    'M': {'N': 2.6},
        'P': {'Q': 1.4},
        'Q': {'N': 1.5},
        'N': {'O': 1.9}}
    rumah_bagoes = {'A': {'B': 3.7},
        'B': {'C': 3.5, 'J': 4.7},
        'J': {'G': 3.2},
        'E': {'I': 1.4},
        'G': {'E': 2.6},
        'D': {'E': 1.6},
        'F': {'E': 1.2},
        'C': {'D': 1.4, 'F': 1.8}}
    rumah_triana = {'R': {'S': 1.9, 'V' : 2.1, 'W' : 2.6},
        'S': {'T' : 1.9},
        'T': {'U': 2.3},
        'V': {'U': 3.1},
        'W': {'U': 2.35},
    'U': {'X': 2.1}}
```

```
    def draw(maps):
      g = nx.DiGraph()
      color = []
      edgeColor = []
      if(maps == int (1)):
        g.add_edge("C", "A", weight="2,6 km")
        g.add_edge("C", "E", weight="2,8 km")
        g.add_edge("A", "D", weight="5 km")
        g.add_edge("D", "B", weight="0,7 km")
        g.add_edge("D", "F", weight="3,3 km")
        g.add_edge("E", "F", weight="4,4 km")
        g.add_edge("F", "H", weight="2,7 km")
        g.add_edge("B", "H", weight="2,5 km")
        g.add_edge("H", "G", weight="2,5 km")
        color = ['g','b','b','b','b','b','b','r']
        edgeColor =['r','k','r','k','r','r','k','r']
      elif(maps == int (2)):
        g.add_edge("A", "B", weight="1,4 km")
        g.add_edge("A", "F", weight="8,5 km")
        g.add_edge("B", "D", weight="2,1 km")
        g.add_edge("B", "C", weight="2,9 km")
        g.add_edge("D", "E", weight="2,3 km")
        g.add_edge("C", "E", weight="1,4 km")
        g.add_edge("E", "G", weight="1,3 km")
        g.add_edge("F", "G", weight="7,3 km")
```

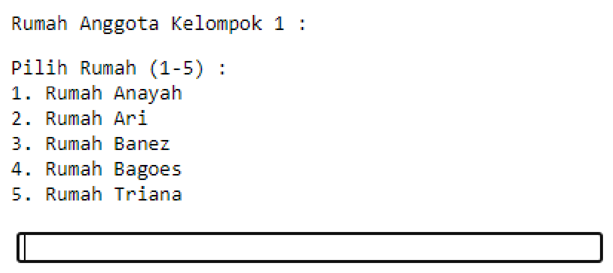
```
        color = ['g','b','b','b','b','b','r']
        edgeColor = ['r','k','r','k','k','r','k']
      elif(maps == int (3)):
        g.add_edge("k", "L", weight="0,7 km")
        g.add_edge("L", "M", weight="1,4 km")
        g.add_edge("L", "P", weight="2,5 km")
        g.add_edge("P", "Q", weight="1,4 km")
        g.add_edge("Q", "N", weight="1,5 km")
        g.add_edge("M", "N", weight="2,6 km")
        g.add_edge("N", "O", weight="1,9 km")
        color = ['g','b','b','b','b','b','r']
        edgeColor = ['r','r','k','r','k','k','r']
      elif(maps == int (4)):
        g.add_edge("A", "B", weight="3,7 km")
        g.add_edge("B", "C", weight="3,5 km")
        g.add_edge("C", "D", weight="1,4 km")
        g.add_edge("C", "F", weight="1,8 km")
        g.add_edge("J", "G", weight="3,2 km")
        g.add_edge("B", "J", weight="    4,7 km")
        g.add_edge("G", "E", weight="2,6 km")
        g.add_edge("F", "E", weight="1,2 km")
        g.add_edge("D", "E", weight="1,6 km")
        g.add_edge("E", "I", weight="1,4 km")
        color = ['g','b','b','b','b','b','b','b','r']
        edgeColor = ['r','r','k','r','k','r','k','k','k']
      elif(maps == int (5)):
        g.add_edge("R", "S", weight="1,9 km")
        g.add_edge("S", "T", weight="1,9 km")
        g.add_edge("R", "V", weight="2,1 km")
        g.add_edge("R", "W", weight="2,6 km")
        g.add_edge("T", "U", weight="2,3 km")
        g.add_edge("V", "U", weight="3,1 km")
        g.add_edge("W", "U", weight="2,35 km")
        g.add_edge("U", "X", weight="2,1 km")
        color = ['g','b','b','b','b','b','r']
        edgeColor = ['r','k','k','r','r','k','k']
      else :
        print ("Pilihan tidak tersedia")


      pos = nx.shell_layout(g)
      edge_labels = { (u,v): d['weight'] for u,v,d in
g.edges(data=True) }
```

```
        nx.draw_networkx_nodes(g,pos,node_size=1000,
node_color=color)

        nx.draw_networkx_edges(g,pos,
edge_color=edgeColor)

        nx.draw_networkx_labels(g,pos)


nx.draw_networkx_edge_labels(g,pos,edge_labels=edge_lab
els)

        plt.title("MAPS")

        plt.axis("off")

        plt.show()


     def _main():


        print ("Rumah Anggota Kelompok 1 :")

        choose = int(input("Pilih Rumah (1-5) :\n1. Rumah
Anayah\n2. Rumah Ari\n3. Rumah Banez\n4. Rumah
Bagoes\n5. Rumah Triana\n\n"))

        if (choose == 1):

           print ("\nRumah Anayah :")

           draw(choose)

           maps = shortest_path(rumah_anayah,"C","G")

           print ("\nJalur Dari 'C' ke 'G' :", maps)

        elif (choose == 2):

           print ("\nRumah Ari :")

           draw(choose)

           maps = shortest_path(rumah_ari,"A","G")

           print ("\nJalur Dari 'A' ke 'G' :", maps)

        elif (choose == 3):

           print ("\nRumah Banez :")

           draw(choose)
```

```
           maps = shortest_path(rumah_banez,"K","O")

           print ("\nJalur Dari 'K' ke 'O' :", maps)

        elif (choose == 4):

           print ("\nRumah Bagoes :")

           draw(choose)

           maps = shortest_path(rumah_bagoes,"A","I")

           print ("\nJalur Dari 'A' ke 'I' :", maps)

        elif (choose == 5):

           print ("\nRumah Triana :")

           draw(choose)

           maps = shortest_path(rumah_triana,"R","X")

           print ("\nJalur Dari 'R' ke 'X' :", maps)

        else :
```

```
           print ("Pilihan tidak tersedia")


        jarak = [13.2, 7.1, 6.6, 11.6, 8.2]

        terkecil = min(jarak)

        nama =""

        if terkecil == 13.2:

          nama = "Rumah Anayah"

        elif terkecil == 7.1:

          nama = "Rumah Ari"

        elif terkecil == 6.6:

          nama = "Rumah Banez"

        elif terkecil == 11.6:

          nama = "Rumah Bagoes"

        elif terkecil == 8.2:

          nama = "Rumah Triana"


        print ("Silahkan pergi ke UPJ dari ", nama, "karena
hanya berjarak", terkecil, "dari UPJ")


     if __name__ == '__main__':

   _main()
```

The following is the output of the code above :



```
Rumah Anggota Kelompok 1 :

Pilih Rumah (1-5) :
1. Rumah Anayah
2. Rumah Ari
3. Rumah Banez
4. Rumah Bagoes
5. Rumah Triana
```

V. *CONCLUSION*

Based on the results of the research, it can be concluded that the Greedy Algorithm :

a) *In determining the shortest and the fastest path using the Greedy Algorithm, it operates from the initial node to the destination node. Where each node has a predetermined distance value.*

b) *Greedy Algorithm can be used to provide the fastest path to optimize student time on their way to UPJ.*

REFERENCES

[1] Nur H., Enty., & Yohanes, Antoni. (2014). Pencarian Rute Terpendek Menggunakan Algoritma Greedy. Seminar Nasional IENACO-2014, 391-397. (Diakses : 18 Desember 2020).

[2] J. Daud. “Studi Efektifitas Penggunaan jalan Kota Medan”. Jurnal Sistem Teknik Industri, No. 3 Vol. 6. 2005.

[3] *Levitin, A., & Mukherjee, S. (2007). Introduction to the design & analysis of algorithms. Vol. 2. Pearson Addison-Wesley.*

[4] Efendi, F. S., Pinto, M., & Tempake, H. S. (2012). Implementasi Algoritma Greedy Untuk Melakukan Graph Coloring: Studi Kasus Peta Propinsi Jawa Timur. Jurnal Informatika vol 4 edisi 2 , 440-448.

[5] Rachmawati, Dian., & Candra, Ade. (2013). Implementasi Algoritma Greedy Untuk Menyelesaikan Masalah Knapsack Problem. Jurnal SAINTIKOM Vol. 12, 185-192. (Diakses : 18 Desember 2020).

[6] Khoiruddin H., Muhammad., & Khairina, Nurul. (2017). Pencarian Jalur Terpendek dengan Algoritma Dijkstra, Jurnal & Penelitian Teknik Informatika Volume 2 Nomor 2, 18-23. (Diakses : 18 Desember 2020).

[7] Pratomo, Erfan. (2014). *Graph* (Graf). Diakses : 18 Desember 2020. https://aerfanpratomo95.wordpress.com/2014/06/26/graph/ .

[8] *Maximillian H. Audrey*., Kharisma R. Indra., & Purbandini. (2015). SISTEM PENCARIAN HOTEL BERDASARKAN RUTE PERJALANAN TERPENDEK DENGAN MEMPERTIMBANGKAN DAYA TARIK WISATA MENGGUNAKAN ALGORITMA GREEDY. *Journal of Information Systems Engineering and Business Intelligence* Vol. 1, No. 1, 9-16. . (Diakses : 15 Desember 2020).

[9] Danies M. Yonny., Nuryanto., & Burhanuddin. Auliya. (2019). SISTEM PENENTUAN JARAK TERDEKAT DALAM PENGIRIMAN DARAH DI PMI KOTA SEMARANG DENGAN METODE ALGORITMA GREEDY. Jurnal Komtika – Komputasi dan Informatika Vol. 2 No. 2. (Diakses : 15 Desember 2020).