LAPORAN TUGAS KECIL 3
IF2211 STRATEGI ALGORITMA
"Implementasi Algoritma UCS dan A* untuk Menentukan Lintasan Terpendek"

Disusun oleh:
Moh. Aghna Maysan Abyan (13521076)
Althaaf Khasyi Atisomya (13521130)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN AJARAN 2022/2023

DAFTAR ISI

BAB I

DESKRIPSI MASALAH

Algoritma UCS (Uniform cost search) dan A* (atau A star) dapat digunakan untuk menentukan lintasan terpendek dari suatu titik ke titik lain. Pada tugas kecil 3 ini, anda diminta menentukan lintasan terpendek berdasarkan peta Google Map jalan-jalan di kota Bandung. Dari ruas-ruas jalan di peta dibentuk graf. Simpul menyatakan persilangan jalan (simpang 3, 4 atau 5) atau ujung jalan. Asumsikan jalan dapat dilalui dari dua arah. Bobot graf menyatakan jarak (m atau km) antar simpul. Jarak antar dua simpul dapat dihitung dari koordinat kedua simpul menggunakan rumus jarak Euclidean (berdasarkan koordinat) atau dapat menggunakan ruler di Google Map, atau cara lainnya yang disediakan oleh Google Map.

Langkah pertama di dalam program ini adalah membuat graf yang merepresentasikan peta (di area tertentu, misalnya di sekitar Bandung Utara/Dago). Berdasarkan graf yang dibentuk, lalu program menerima input simpul asal dan simpul tujuan, lalu menentukan lintasan terpendek antara keduanya menggunakan algoritma UCS dan A*. Lintasan terpendek dapat ditampilkan pada peta/graf (misalnya jalan-jalan yang menyatakan lintasan terpendek diberi warna merah). Nilai heuristik yang dipakai adalah jarak garis lurus dari suatu titik ke tujuan.

Spesifikasi program:

1. Program menerima input file graf (direpresentasikan sebagai matriks ketetanggaan berbobot), jumlah simpul minimal 8 buah.
2. Program dapat menampilkan peta/graf
3. Program menerima input simpul asal dan simpul tujuan.
4. Program dapat menampilkan lintasan terpendek beserta jaraknya antara simpul asal dan simpul tujuan.
5. Antarmuka program bebas, apakah pakai GUI atau command line saja.

BAB II

KODE PROGRAM

2.1 astar.py

src > astar.py > astar

```python
from node import Node

def calculateFValue(adj_m, heuristik, parent_node, node):
    g = parent_node.fValue - heuristik[parent_node.id] + adj_m[parent_node.id][node]
    h = heuristik[node]
    return g+h

def astar(adj_m,heuristik,s_node,g_node):
    # initiate start_node and goal_node
    start_node = Node(s_node,heuristik[s_node])
    goal_node = Node(g_node,0)

    # visited, open_nodes
    visited = []
    open_nodes = []

    # visit start node
    open_nodes.append(start_node)
    visited.append(start_node.id)

    while len(open_nodes) > 0 :
        # sort open nodes by f values
        open_nodes.sort(key=lambda x: x.fValue)

        # visit open_nodes with smallest fValue
        current_node = open_nodes.pop(0)
        visited.append(current_node.id)

        # current node is goal node
        if current_node.id==goal_node.id:
            current_node.parents.append(current_node.id)
            return current_node.parents, current_node.getAStarDistance(heuristik)

        # find neighbors of current node
        neighbors = []
        for i in range(len(adj_m[0])):
            if (adj_m[current_node.id][i]!=0 and (i not in visited)):
                neighbors.append(i)

```

Gambar 2.1 Cuplikan kode astar.py baris 1-39

```
40            # for each neighbors append to open nodes
41            for i in range(len(neighbors)):
42                newFvalue = calculateFValue(adj_m,heuristik,current_node,neighbors[i])
43                newNode = Node(neighbors[i],newFvalue)
44                newNode.setParent(current_node)
45                open_nodes.append(newNode)
46
47        return None,None
48
49    if __name__ == "__main__":
50        adj_m = [[0,1,0,0,0,10,0],
51                 [1,0,2,1,0,0,0],
52                 [0,2,0,0,5,0,0],
53                 [0,1,0,0,3,4,0],
54                 [0,0,5,3,0,2,0],
55                 [10,0,0,4,2,0,5],
56                 [0,0,0,0,5,0]]
57        heuristik = [5,3,4,2,6,1,0]
58        path,distance = astar(adj_m,heuristik,0,6)
59        print(path,distance)
```

Gambar 2.2 Cuplikan kode astar.py baris 40-59

2.2 filereader.py

src > filereader.py > ...

```
1   from math import *
2
3   def getLines(fileName):
4       file = open(fileName,'r')
5       lines = []
6       for line in file:
7           line = line.rstrip()
8           lines.append(line)
9       file.close()
10      return lines
11
12  def processNodes(lines):
13      nNodes = int(lines[0])
14      raw = []
15      raw = [[x for x in line.split(' ')] for line in lines]
16      nodes_raw = raw[1:nNodes+1]
17      adj_m_raw = raw[nNodes+1:]
18      return nodes_raw,adj_m_raw
19
20  def haversine(node1,node2):
21      r = 6371
22      dLat = pi/180 * (float(node1[1])-float(node2[1]))
23      dLon = pi/180 * (float(node1[2])-float(node2[2]))
24      akar =  sin(dLat/2)**2 + cos(float(node2[1])*(pi/180)) * cos(float(node1[1])*(pi/180)) * sin(dLon/2)**2
25      return 2*r*asin(sqrt(akar))
26
27  def euclidean(node1,node2):
28      d1 = (float(node1[1])-float(node2[1]))
29      d2 = (float(node1[2])-float(node2[2]))
30      return sqrt(d1**2+d2**2)
31
32  def getDistance(distance_method,node1,node2):
33      if (distance_method==1):
34          return euclidean(node1,node2)
35      else:
36          return haversine(node1,node2)
37
38  def getAdj_m(adj_m_raw,nodes_raw,distance_method):
39      n = len(adj_m_raw)
40      adj_m = [[0 for i in range (n)] for j in range (n)]
```

Gambar 2.3 Cuplikan kode filereader.py baris 1-40

```
41      for i in range(n):
42          for j in range(n):
43              if(int(adj_m_raw[i][j])!=0):
44                  adj_m[i][j] = getDistance(distance_method,nodes_raw[i],nodes_raw[j])
45      return adj_m
46
47  def getHeuristik(nodes_raw,goal_node,distance_method):
48      heuristik = [0 for i in range(len(nodes_raw))]
49      for i in range(len(nodes_raw)):
50          heuristik[i] = getDistance(distance_method,nodes_raw[i],nodes_raw[goal_node])
51      return heuristik
52
53  def fileReader(lines,goal_node,distance_method,path_finder_method):
54      nodes,adj_m_raw = processNodes(lines)
55      for i in range(len(nodes)):
56          nodes[i][1] = float(nodes[i][1])
57          nodes[i][2] = float(nodes[i][2])
58      adj_m = getAdj_m(adj_m_raw,nodes,distance_method)
59      if (path_finder_method==1):
60          return adj_m, nodes
61      else:
62          return adj_m, getHeuristik(nodes,goal_node,distance_method),nodes
63
64  if __name__ == "__main__":
65      lines = getLines("test/testcase1.txt")
66      nodes_raw,adj_m_raw = processNodes(lines)
67      print(nodes_raw)
68      print(adj_m_raw)
69      print(getAdj_m(adj_m_raw,nodes_raw))
70
```

Gambar 2.4 Cuplikan kode filereader.py baris 41-70

2.3 main.py

```
src > main.py > ...
from astar import astar
from ucs import ucs
from filereader import getLines,fileReader
import visualization as vis
from flask import Flask, render_template
import googlemaps
from colorama import Fore, Style

def printNodesName(path,lines):
    n = (len(lines)-1)//2
    nodes_name = [el[0] for el in [[x for x in line.split(' ')] for line in lines][1:n+1]]
    for i in range(len(path)):
        print(str(i+1) + ". "+ nodes_name[path[i]])
    print()


# welcome message
print("Welcome to: \n")
print(Fore.YELLOW + "      ___    ___    _____   _ _           ___    ___    _ _    ___     ___     ___    ")
print("    | _ \  /   \  |_   _|  | || |     o O O  | __|   |_ _|   | \| |    |   \   | __|   | _ \   ")
print("    |  _/  | - |    | |    | __ |    o       | _|     | |    | .` |    | |) |  | _|    |   /   ")
print("   _|_|_   |_|_|   _|_|_   |_||_|   TS__[O]  _|_|_   |___|   |_|\_|    |___/   |___|   |_|_\   ")
print(Fore.BLUE + ' _| """ | _|"""""| _|"""""| _|"""""|  {======| _| """ | _|"""""| _|"""""| _|"""""| _|"""""| _|"""""|  ')
print(Fore.BLACK + " ''-0-0-' ''-0-0-' ''-0-0-' ''-0-0-' ./o--000' ''-0-0-' ''-0-0-' ''-0-0-' ''-0-0-' ''-0-0-' ''-0-0-'  ")
print(Fore.WHITE + "                                    Created by " + Fore.GREEN + "ABYAN" + Fore.WHITE + " and " + Fore.MAGENTA + "ALTHAA")
print(Style.RESET_ALL)

while (True):
    # home
    print("Please choose an option:")
    print("1. Run Path Finder\n2. Exit")
    while (True):
        option = input(">> ")
        print()
        if (option=='1' or option=='2'):
            option = int(option)
            break
        else:
            print("Invalid option, please try again\n")
    if (option==2):
```

Gambar 2.5 Cuplikan kode main.py baris 1-40

```
41            print(Fore.CYAN + "               ___       ___                            ___          ")
42            print("              (   )     (   )                          (   )         ")
43            print("               | |_      | | .-.     .---.   ___ .-.    | |   ___    ")
44            print("              (   __)    | |/   \   / .-, \ (   )   \   | |  (   )   ")
45            print("               | |       |  .-. .  (__) ; |  |  .-. .   | |   ' /    ")
46            print("               | | ___   | |  | |    .'`  |  | |  | |   | |,' /      ")
47            print("               | |(   )  | |  | |   / .'| |  | |  | |   | .  '.      ")
48            print("               | | | |   | |  | |  | /  | |  | |  | |   | | `. \     ")
49            print("               | ' | |   | |  | |  ; |  ; |  | |  | |   | |   \ \    ")
50            print("               ' `-' ;   | |  | |  ' `-'  |  | |  | |   | |    \ .   ")
51            print("                `.__.   (___)(___) `.__.'_. (___)(___) (___ ) (___)  ")
52            print("                           ___  ___    .--.    ___  ___    ")
53            print("                          (   )(   )  /    \  (   )(   )   ")
54            print("                           | |  | |  |  .-. ;  | |  | |    ")
55            print("                           | |  | |  | |  | |  | |  | |    ")
56            print("                           | '  | |  | |  | |  | |  | |    ")
57            print("                           '  `-' |  | |  | |  | |  | |    ")
58            print("                            `.__. |  | '  | |  | |  ; '    ")
59            print("                            ___ | |  '  `-' /  ' `-'  /    ")
60            print("                           (   )' |   `.__.'    '.__.'     ")
61            print("                            ; `-' '                        ")
62            print("                             .__.'                        \n")
63            print(Style.RESET_ALL)
64            break
65
66        # choose file
67        print("Enter file name:")
68        while (True):
69            fileName = input(">> ")
70            print()
71            try:
72                lines = getLines("../test/"+fileName)
73                break
74            except:
75                print("File not found, please try again\n")
76
77        # choose distance method
78        print("Please choose the distance method calculation:")
79        print("1. Euclidean, use this if your nodes position is in cartesian coordinate")
80        print("2. Haversine, use this if your nodes position is in latitude and longitude")
```

Gambar 2.6 Cuplikan kode main.py baris 41-80

```
81      while (True):
82          distance_method = input(">> ")
83          print()
84          if (distance_method=='1' or distance_method=='2'):
85              distance_method = int(distance_method)
86              break
87          else:
88              print("Invalid option, please try again\n")
89
90      # choose path finder method
91      print("Please choose the path finder method:")
92      print("1. UCS\n2. A*")
93      while (True):
94          path_finder_method = input(">> ")
95          print()
96          if (path_finder_method=='1' or path_finder_method=='2'):
97              path_finder_method = int(path_finder_method)
98              break
99          else:
100             print("Invalid option, please try again\n")
101
102     # choose start node
103     print("The nodes are as follows:")
104     p = [i for i in range((len(lines)-1)//2)]
105     printNodesName(p,lines)
106     print("Please choose the start node:")
107     while (True):
108         start_node = input(">> ")
109         print()
110         if (int(start_node)-1 >= 0 and int(start_node)-1 < int(lines[0])):
111             start_node = int(start_node)-1
112             break
113         else:
114             print("Invalid start node, please try again\n")
115
116     # choose goal node
117     print("Please choose the goal node:")
118     while (True):
119         goal_node = input(">> ")
120         print()
```

Gambar 2.7 Cuplikan kode main.py baris 81-120

```
121             if (int(goal_node)-1 >= 0 and int(goal_node)-1 < int(lines[0])):
122                 goal_node = int(goal_node)-1
123                 break
124             else:
125                 print("Invalid goal node, please try again\n")
126 
127         # path finding
128         isFound = True
129         if (path_finder_method == 1):
130             adj_m,nodes = fileReader(lines,goal_node,distance_method,path_finder_method)
131             path,distance = ucs(adj_m,start_node,goal_node)
132             if (path==None):
133                 print("No path found, please make sure that both start node and goal node are connected\n")
134                 isFound = False
135             else:
136                 if (distance_method==1): unit = "unit"
137                 else: unit = "km"
138                 print("Result: ")
139                 printNodesName(path,lines)
140                 print("Distance:",round(distance,2),"%s\n"%(unit))
141         else:
142             adj_m,heuristik,nodes = fileReader(lines,goal_node,distance_method,path_finder_method)
143             path,distance = astar(adj_m,heuristik,start_node,goal_node)
144             if (path==None):
145                 print("No path found, please make sure that both start node and goal node are connected\n")
146                 isFound = False
147             else:
148                 if (distance_method==1): unit = "unit"
149                 else: unit = "km"
150                 print("Result: ")
151                 printNodesName(path,lines)
152                 print("Distance:",round(distance,2),"%s\n"%(unit))
153 
154         # visualization
155         if(isFound):
156             print("How do you want to visualize the path?")
157             print("1. Graph\n2. Map\n3. No visualization")
158             while (True):
159                 visualization = input(">> ")
160                 print()
```

Gambar 2.8 Cuplikan kode main.py baris 121-160

```
161             if (visualization=='1' or visualization=='2' or visualization<='3'):
162                 visualization = int(visualization)
163                 break
164             else:
165                 print("Invalid option, please try again\n")
166 
167         if (visualization==1 or visualization==2):
168             print("Exit visualization to continue\n")
169             if (visualization==1):
170                 vis.drawgraph(nodes,adj_m,path)
171             else:
172                 path = [[nodes[path[i]][1],nodes[path[i]][2]] for i in range(len(path))]
173                 # print(nodes)
174                 app = Flask(__name__)
175                 @app.route('/')
176                 def map():
177                     gmaps = googlemaps.Client(key='AIzaSyBysSbgLtum7i-eQ3_qMqiMMsVlNHe88Yw')
178                     return render_template('map.html',gmaps=gmaps,nodes=nodes,path=path)
179                 app.run(debug=False)
```

Gambar 2.9 Cuplikan kode main.py baris 161-179

2.4 node.py

src > node.py > Node

```
1  class Node:
2      def __init__(self,id,fValue):
3          self.id = id
4          self.fValue = fValue
5          self.parents = []
6  
7      def setParent(self,parent_node):
8          for i in parent_node.parents:
9              self.parents.append(i)
10         self.parents.append(parent_node.id)
11 
12     def getAStarDistance(self,heuristik):
13         return self.fValue - heuristik[self.id]
14 
15     def getUCSDistance(self):
16         return self.fValue
```

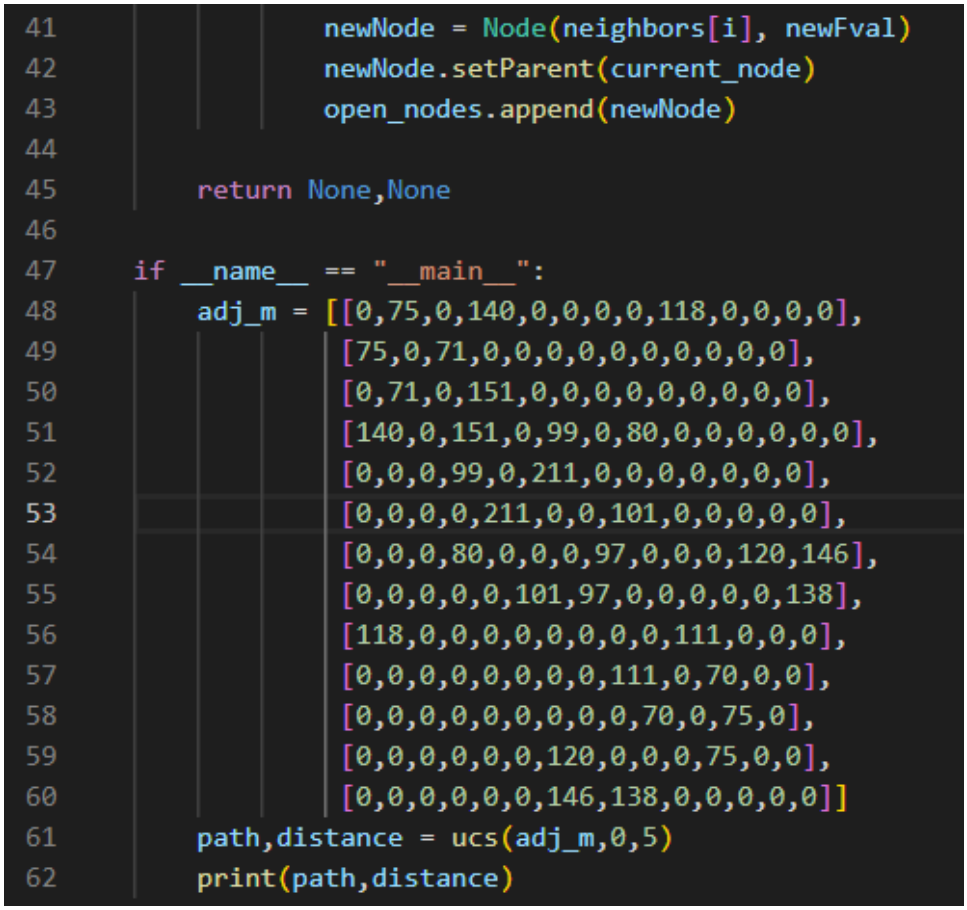
Gambar 2.10 Cuplikan kode node.py baris 1-16

2.5 ucs.py

src > ucs.py > ...

```python
from node import Node

def calculateFVal(adj_m, parent, node):
    return (parent.fValue + adj_m[parent.id][node])

def ucs(adj_m, s_node, g_node):
    # initiate start_node and goal_node
    start_node = Node(s_node, 0)
    goal_node = Node(g_node, 0)

    # visited, open_nodes
    visited = []
    open_nodes = []

    # visit start node
    open_nodes.append(start_node)
    visited.append(start_node.id)

    while len(open_nodes) > 0:
        # sort open nodes by f values
        open_nodes.sort(key = lambda x: x.fValue)

        # visit open_nodes with smallest fValue
        current_node = open_nodes.pop(0)
        visited.append(current_node.id)

        # current node is goal node
        if (current_node.id == goal_node.id):
            current_node.parents.append(current_node.id)
            return current_node.parents, current_node.getUCSDistance()

        # find neighbors of current node
        neighbors = []
        for i in range(len(adj_m[0])):
            if (adj_m[current_node.id][i] != 0 and (i not in visited)):
                neighbors.append(i)

        # for each neighbors append to open nodes
        for i in range(len(neighbors)):
            newFval = calculateFVal(adj_m, current_node, neighbors[i])
```

Gambar 2.11 Cuplikan kode node.py baris 1-40



```
41                  newNode = Node(neighbors[i], newFval)
42                  newNode.setParent(current_node)
43                  open_nodes.append(newNode)
44
45          return None,None
46
47      if __name__ == "__main__":
48          adj_m = [[0,75,0,140,0,0,0,0,118,0,0,0,0],
49                   [75,0,71,0,0,0,0,0,0,0,0,0,0],
50                   [0,71,0,151,0,0,0,0,0,0,0,0,0],
51                   [140,0,151,0,99,0,80,0,0,0,0,0,0],
52                   [0,0,0,99,0,211,0,0,0,0,0,0,0],
53                   [0,0,0,0,211,0,0,101,0,0,0,0,0],
54                   [0,0,0,80,0,0,0,97,0,0,0,120,146],
55                   [0,0,0,0,0,101,97,0,0,0,0,0,138],
56                   [118,0,0,0,0,0,0,0,0,111,0,0,0],
57                   [0,0,0,0,0,0,0,0,111,0,70,0,0],
58                   [0,0,0,0,0,0,0,0,0,70,0,75,0],
59                   [0,0,0,0,0,0,120,0,0,0,75,0,0],
60                   [0,0,0,0,0,0,146,138,0,0,0,0,0]]
61          path,distance = ucs(adj_m,0,5)
62          print(path,distance)
```

Gambar 2.12 Cuplikan kode node.py baris 41-62

2.6 visualization.py

src > visualization.py > ...

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

def adj_mToEdges(adj_m,nodes):
    # convert adjacency matrix to edges
    edges = []
    for i in range(len(adj_m)):
        for j in range(len(adj_m)):
            if (adj_m[i][j] != 0 and i<j):
                edges.append((nodes[i][0],nodes[j][0]))
    return edges

def getWeight(adj_m,nodes_name,edge):
    # get weight of an edge
    for x in range(len(nodes_name)):
        if (nodes_name[x][0] == edge[0]):
            i = x
        if (nodes_name[x][0] == edge[1]):
            j = x
    return adj_m[i][j]

def isPath(edge,path,nodes_name):
    # check if an edge is in a path
    pathEdges1 = []
    pathEdges2 = []
    for i in range(len(path)-1):
        pathEdges1.append((nodes_name[path[i]][0],nodes_name[path[i+1]][0]))
        pathEdges2.append((nodes_name[path[i+1]][0],nodes_name[path[i]][0]))
    if (edge in pathEdges1 or edge in pathEdges2):
            return True
    return False

def drawgraph(nodes,adj_m,path):
    G = nx.Graph()
    # add nodes
    for i in range(len(nodes)):
        G.add_node(nodes[i][0],pos=(nodes[i][2],nodes[i][1]))

    # add edges
    edges = adj_mToEdges(adj_m,nodes)
```

Gambar 2.13 Cuplikan kode visualization.py baris 1-40

```
41        for i in range(len(edges)):
42            c = '#93B8C5'
43            if (isPath(edges[i],path,nodes)):
44                c = 'r'
45            G.add_edge(edges[i][0],edges[i][1],weight=round(getWeight(adj_m,nodes,edges[i]),2),color=c)
46
47        # draw graph
48        pos = nx.get_node_attributes(G,'pos')
49        labels = nx.get_edge_attributes(G,'weight')
50        edges,colors = zip(*nx.get_edge_attributes(G, 'color').items())
51        nx.draw_networkx(G,pos)
52        nx.draw_networkx_edges(G,pos,edgelist=edges,edge_color=colors)
53        nx.draw_networkx_edge_labels(G,pos,edge_labels=labels)
54        ax = plt.gca()
55        ax.margins(0.10)
56        plt.axis("off")
57        plt.show()
58
59    if __name__ == "__main__":
60        adj_m = [[0,1,0,0,0,10,0],
61                 [1,0,2,1,0,0,0],
62                 [0,2,0,0,5,0,0],
63                 [0,1,0,0,3,4,0],
64                 [0,0,5,3,0,2,0],
65                 [10,0,0,4,2,0,5],
66                 [0,0,0,0,5,0,0]]
67        nodes = [["bonbin",0,1],["kota",1,5],["kabupaten",2,7],["provinsi",3,8],["negara",4,9],["benua",5,5],["planet",8,-1]]
68        path = [0,1]
69        drawgraph(nodes,adj_m,path)
```

Gambar 2.14 Cuplikan kode visualization.py baris 41-69

BAB III

PETA/GRAF INPUT

3.1 testcase1.txt (Peta jalan sekitar kampus ITB/Dago/Bandung Utara)

testcase1.txt

```
15
Tamgan -6.893259051558486 107.61002547087841
Besthal -6.894779700886209 107.61014979910928
Tamansari -6.894873507021786 107.60882694669466
BNI -6.893717608739038 107.6084482586054
Wisma -6.894732142111585 107.61170909154976
SR -6.893588173925802 107.61194734205844
Juanda -6.8937229052181905 107.61294579765874
Dayang -6.887380626959768 107.61346186576496
AlfaX -6.8872263037003805 107.6114834753278
SBM -6.887829567094039 107.60826859091632
McD -6.885223632863306 107.613641884003
Panera -6.887648107858758 107.60990500494593
Sabuga -6.886240152379595 107.60835853849272
TeukuUmar -6.891496969853349 107.61314046939485
DAMRI -6.89279552798831 107.61810327432394
0 1 0 1 0 1 0 0 0 0 0 0 0 0 0
1 0 1 0 1 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 1 0 0 0 0 0
0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
1 0 0 0 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 1 0 1 0 0 1 0
0 0 0 0 0 0 0 1 0 0 1 1 0 0 0
0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 1
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 1 0 0 1 0
```

Gambar 3.1 Isi file testcase1.txt

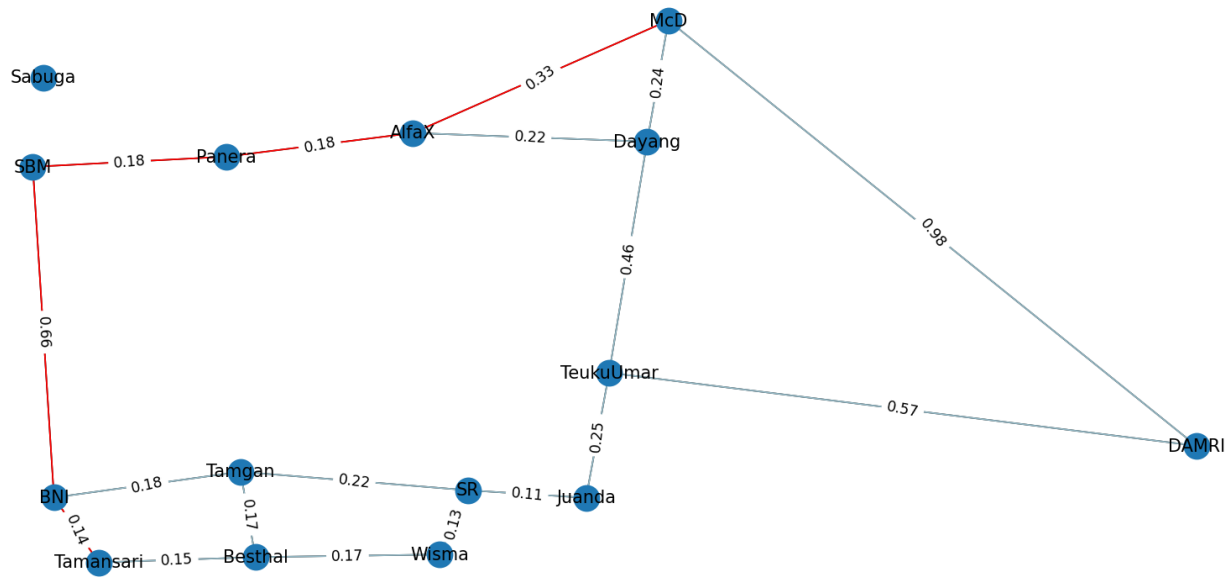
```
Please choose an option:
1. Run Path Finder
2. Exit
>> 1

Enter file name:
>> testcase1.txt

Please choose the distance method calculation:
1. Euclidean, use this if your nodes position is in cartesian coordinate
2. Haversine, use this if your nodes position is in latitude and longitude
>> 2

Please choose the path finder method:
1. UCS
2. A*
>> 1

The nodes are as follows:
1. Tamgan
2. Besthal
3. Tamansari
4. BNI
5. Wisma
6. SR
7. Juanda
8. Dayang
9. AlfaX
10. SBM
11. McD
12. Panera
13. Sabuga
14. TeukuUmar
15. DAMRI

Please choose the start node:
>> 3

Please choose the goal node:
>> 13

No path found, please make sure that both start node and goal node are connected
```

Gambar 3.2 Hasil & Input apabila start node dan goal node tidak terhubung

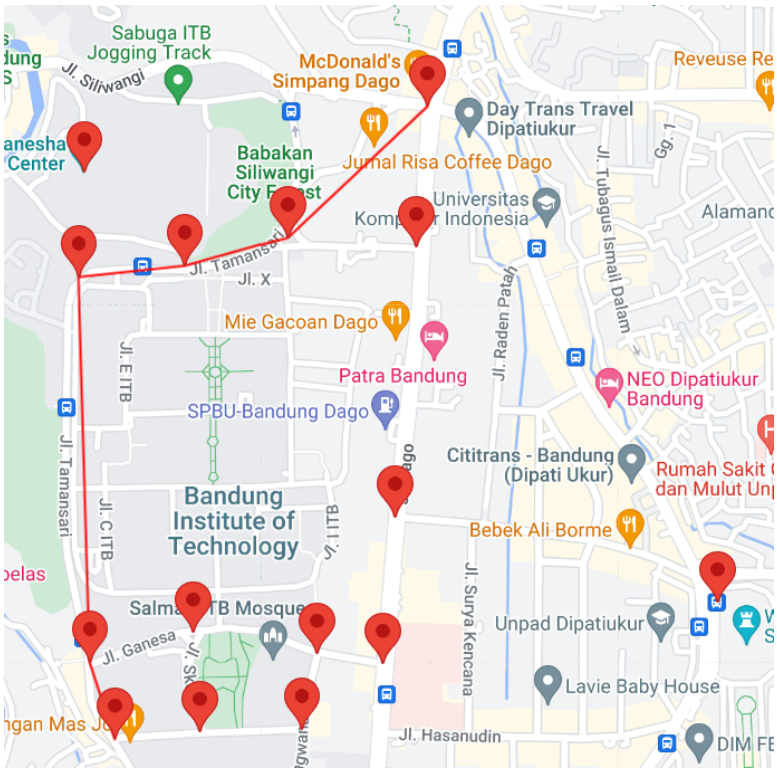
```
Please choose an option:
1. Run Path Finder
2. Exit
>> 1

Enter file name:
>> testcase1.txt

Please choose the distance method calculation:
1. Euclidean, use this if your nodes position is in cartesian coordinate
2. Haversine, use this if your nodes position is in latitude and longitude
>> 2

Please choose the path finder method:
1. UCS
2. A*
>> 1

The nodes are as follows:
1. Tamgan
2. Besthal
3. Tamansari
4. BNI
5. Wisma
6. SR
7. Juanda
8. Dayang
9. AlfaX
10. SBM
11. McD
12. Panera
13. Sabuga
14. TeukuUmar
15. DAMRI
```

```
Please choose the start node:
>> 3

Please choose the goal node:
>> 11

Result:
1. Tamansari
2. BNI
3. SBM
4. Panera
5. AlfaX
6. McD

Distance: 1.48 km
```

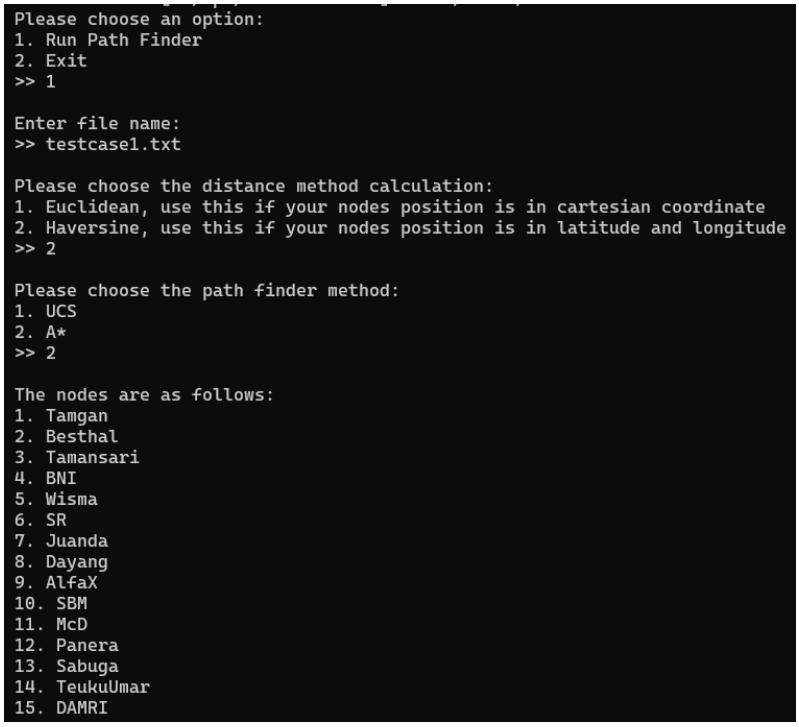
Gambar 3.3 & 3.4 Input & Hasil pecarian path pada testcase1 dengan start node TamanSari dan goal node McD menggunakan metode UCS

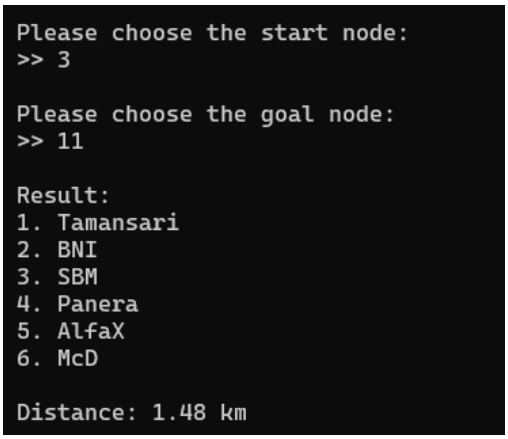


Gambar 3.5 Visualisasi graph testcase1 dengan start node dengan start node TamanSari dan goal node McD



Gambar 3.6 Visualisasi map testcase1 dengan start node dengan start node TamanSari dan goal node McD





Gambar 3.7 & 3.8 Input & Hasil pecarian path pada testcase1 dengan start node TamanSari dan goal node McD menggunakan metode A*

3.2 testcase2.txt (Peta jalan sekitar Alun-alun Bandung)

testcase2.txt

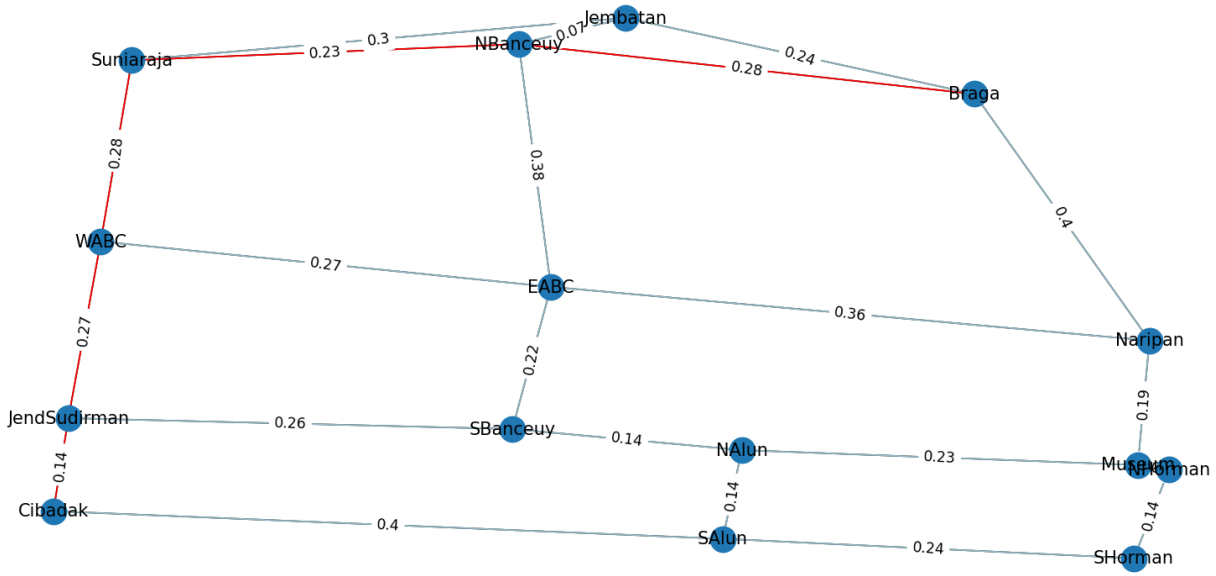
```
15
Cibadak -6.922095797275739 107.60400865707209
JendSudirman -6.9208014172204955 107.60408901536037
NAlun -6.921246376560327 107.60768713716887
SAlun -6.922480508562946 107.60758565201992
NHorman -6.921519434359909 107.60997047992721
SHorman -6.922751964071557 107.60978236178545
Museum -6.921453378579499 107.60980370975157
Naripan -6.919722210249814 107.60986597907178
Braga -6.91629083203957 107.60893014634982
Jembatan -6.915231500614575 107.60706655196725
Suniaraja -6.915821769671452 107.60442803748106
NBanceuy -6.915596901176499 107.60649549656402
SBanceuy -6.92095300561848 107.60646113103361
WABC -6.918343191616551 107.60426173708734
EABC -6.918974324429761 107.60666732421606
0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 1 1 0
0 0 0 1 0 0 1 0 0 0 0 0 1 0 0
1 0 1 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 0 0 0 0 1
0 0 0 0 0 0 0 1 0 1 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0 1 1 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 1 0
0 0 0 0 0 0 0 0 1 1 1 0 0 0 1
0 1 1 0 0 0 0 0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 0 1 0 0 0 1
0 0 0 0 0 0 0 1 0 0 0 1 1 1 0
```

Gambar 3.9 Isi file testcase2.txt

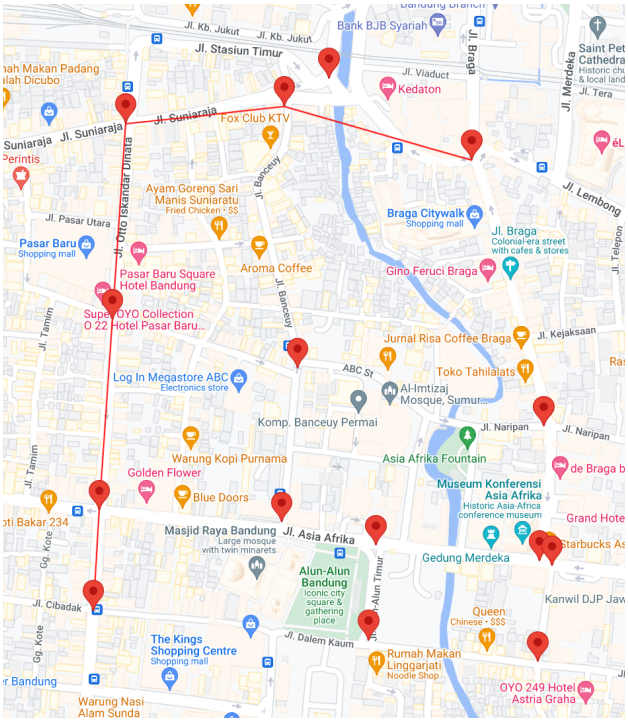
```
Please choose an option:
1. Run Path Finder
2. Exit
>> 1

Enter file name:
>> testcase2.txt

Please choose the distance method calculation:
1. Euclidean, use this if your nodes position is in cartesian coordinate
2. Haversine, use this if your nodes position is in latitude and longitude
>> 2

Please choose the path finder method:
1. UCS
2. A*
>> 2

The nodes are as follows:
1. Cibadak
2. JendSudirman
3. NAlun
4. SAlun
5. NHorman
6. SHorman
7. Museum
8. Naripan
9. Braga
10. Jembatan
11. Suniaraja
12. NBanceuy
13. SBanceuy
14. WABC
15. EABC
```

```
Please choose the start node:
>> 1

Please choose the goal node:
>> 9

Result:
1. Cibadak
2. JendSudirman
3. WABC
4. Suniaraja
5. NBanceuy
6. Braga

Distance: 1.21 km
```

Gambar 3.10 & 3.11 Input & Hasil pecarian path pada testcase1 dengan start node Cibadak dan goal node Braga menggunakan metode A*



Gambar 3.12 Visualisasi graph testcase2 dengan start node Cibadak dan goal node Braga



Gambar 3.13 Visualisasi map testcase2 dengan start node Cibadak dan goal node Braga

```
Please choose an option:
1. Run Path Finder
2. Exit
>> 1

Enter file name:
>> testcase2.txt

Please choose the distance method calculation:
1. Euclidean, use this if your nodes position is in cartesian coordinate
2. Haversine, use this if your nodes position is in latitude and longitude
>> 2

Please choose the path finder method:
1. UCS
2. A*
>> 1

The nodes are as follows:
1. Cibadak
2. JendSudirman
3. NAlun
4. SAlun
5. NHorman
6. SHorman
7. Museum
8. Naripan
9. Braga
10. Jembatan
11. Suniaraja
12. NBanceuy
13. SBanceuy
14. WABC
15. EABC
```

```
Please choose the start node:
>> 1

Please choose the goal node:
>> 9

Result:
1. Cibadak
2. JendSudirman
3. WABC
4. Suniaraja
5. NBanceuy
6. Braga

Distance: 1.21 km
```

Gambar 3.14 & 3.15 Input & Hasil pecarian path pada testcase1 dengan start node Cibadak dan goal node Braga menggunakan metode UCS

3.3 testcase3.txt (Peta jalan sekitar Buahbatu atau Bandung Selatan)

testcase3.txt

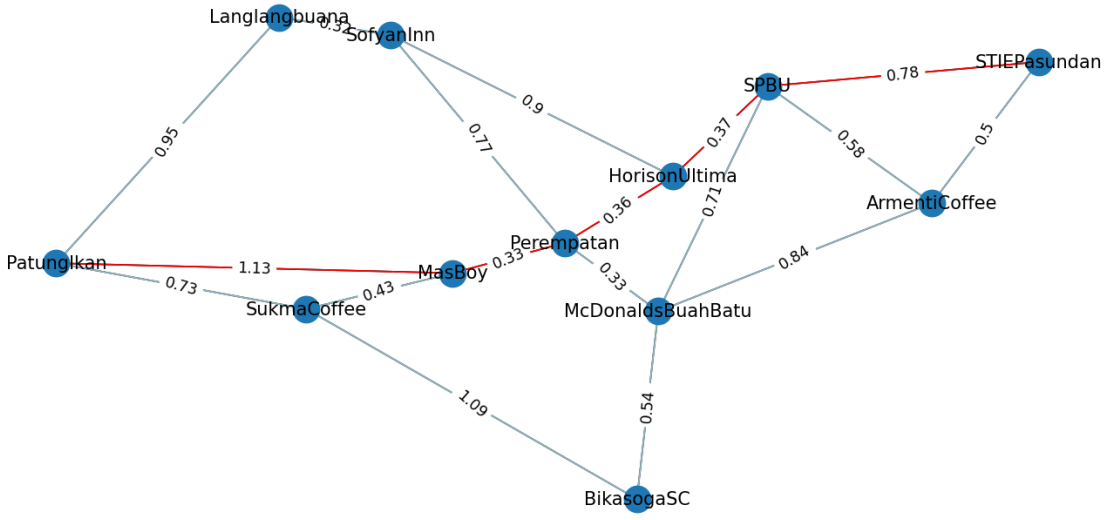
```
12
Perempatan -6.936932465407775 107.62271698513482
SofyanInn -6.9316172902297835 107.61820760681194
Langlangbuana -6.931164980074095 107.61531844273748
PatungIkan -6.9374531350083055 107.60953762493916
MasBoy -6.937707340465891 107.61981024328794
McDonaldsBuahBatu -6.938683213873038 107.6251415203825
HorisonUltima -6.935230114386316 107.62551962517628
SPBU -6.932924722333966 107.62797546679072
BikasogaSC -6.943484020437823 107.62458527472594
SukmaCoffee -6.9386246333087325 107.61601954715178
ArmentiCoffee -6.935919122549465 107.63223225166969
STIEPasundan -6.932305828335907 107.63499858927968
0 1 0 0 1 1 1 0 0 0 0 0
1 0 1 0 0 0 1 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 1 0 0
1 0 0 1 0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 1 1 0 1 0
1 1 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 1 0 0 0 1 1
0 0 0 0 0 1 0 0 0 1 0 0
0 0 0 1 1 0 0 0 1 0 0 0
0 0 0 0 0 1 0 1 0 0 0 1
0 0 0 0 0 0 0 1 0 0 1 0
```

Gambar 3.16 Isi file testcase3.txt

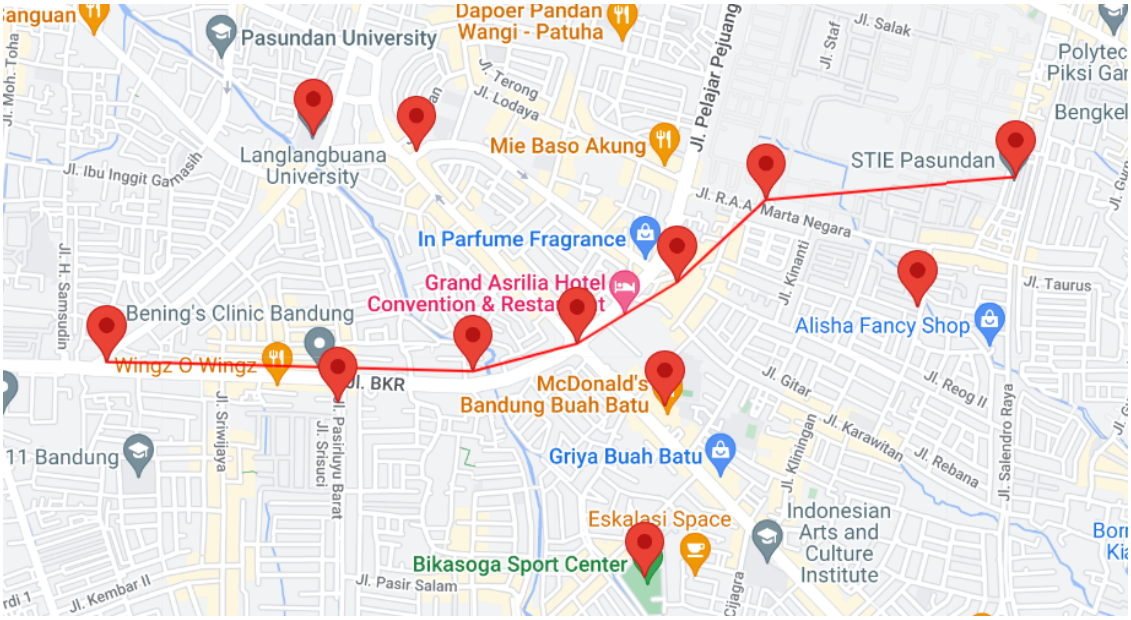
```
Please choose an option:
1. Run Path Finder
2. Exit
>> 1

Enter file name:
>> testcase3.txt

Please choose the distance method calculation:
1. Euclidean, use this if your nodes position is in cartesian coordinate
2. Haversine, use this if your nodes position is in latitude and longitude
>> 2

Please choose the path finder method:
1. UCS
2. A*
>> 1

The nodes are as follows:
1. Perempatan
2. SofyanInn
3. Langlangbuana
4. PatungIkan
5. MasBoy
6. McDonaldsBuahBatu
7. HorisonUltima
8. SPBU
9. BikasogaSC
10. SukmaCoffee
11. ArmentiCoffee
12. STIEPasundan
```

```
Please choose the start node:
>> 4

Please choose the goal node:
>> 12

Result:
1. PatungIkan
2. MasBoy
3. Perempatan
4. HorisonUltima
5. SPBU
6. STIEPasundan

Distance: 2.98 km
```

Gambar 3.17 & 3.18 Input & Hasil pecarian path pada testcase3 dengan start node PatungIkan dan goal node STIEPasundan menggunakan metode UCS



Gambar 3.19 Visualisasi graph testcase3 dengan start node PatungIkan dan goal node STIEPasundan



Gambar 3.20 Visualisasi map testcase3 dengan start node PatungIkan dan goal node STIEPasundan

```
Please choose an option:
1. Run Path Finder
2. Exit
>> 1

Enter file name:
>> testcase3.txt

Please choose the distance method calculation:
1. Euclidean, use this if your nodes position is in cartesian coordinate
2. Haversine, use this if your nodes position is in latitude and longitude
>> 2

Please choose the path finder method:
1. UCS
2. A*
>> 2

The nodes are as follows:
1. Perempatan
2. SofyanInn
3. Langlangbuana
4. PatungIkan
5. MasBoy
6. McDonaldsBuahBatu
7. HorisonUltima
8. SPBU
9. BikasogaSC
10. SukmaCoffee
11. ArmentiCoffee
12. STIEPasundan
```

```
Please choose the start node:
>> 4

Please choose the goal node:
>> 12

Result:
1. PatungIkan
2. MasBoy
3. Perempatan
4. HorisonUltima
5. SPBU
6. STIEPasundan

Distance: 2.98 km
```

Gambar 3.21 & 3.22 Input & Hasil pecarian path pada testcase3 dengan start node PatungIkan dan goal node STIEPasundan menggunakan metode A*

3.4 testcase4.txt (Peta jalan sebuah kawasan di kota asalmu)

testcase4.txt

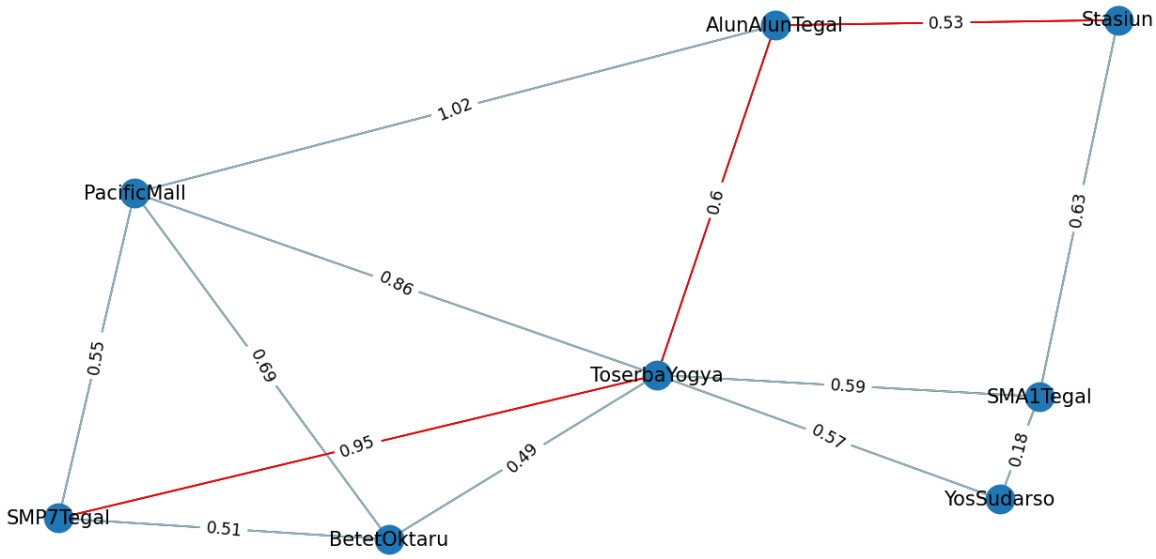
```
8
SMA1Tegal -6.872917465368176 109.14156414088383
AlunAlunTegal -6.867426507386413 109.13788358850171
Stasiun -6.867348343802633 109.14266633838582
YosSudarso -6.874431394764144 109.14101550671661
PacificMall -6.869903512722211 109.12894895189994
SMP7Tegal -6.8747135709853415 109.12789176964044
ToserbaYogya -6.872595647070554 109.13623759318712
BetetOktaru -6.875027103144559 109.1324974617358
0 0 1 1 0 0 1 0
0 0 1 0 1 0 1 0
1 1 0 0 0 0 0 0
1 0 0 0 0 0 1 0
0 1 0 0 0 1 1 1
0 0 0 0 1 0 1 1
1 1 0 1 1 1 0 1
0 0 0 0 1 1 1 0
```

Gambar 3.23 Isi file testcase3.txt

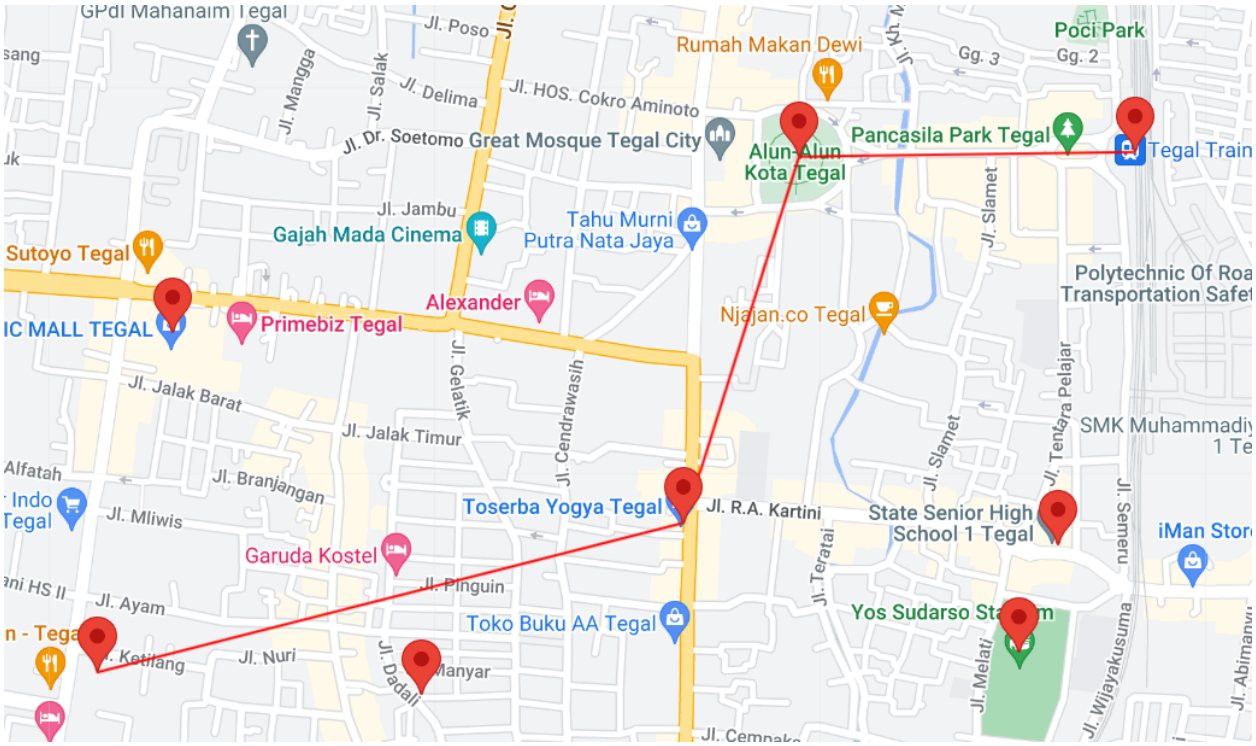
```
Please choose an option:
1. Run Path Finder
2. Exit
>> 1

Enter file name:
>> testcase4.txt

Please choose the distance method calculation:
1. Euclidean, use this if your nodes position is in cartesian coordinate
2. Haversine, use this if your nodes position is in latitude and longitude
>> 2

Please choose the path finder method:
1. UCS
2. A*
>> 2

The nodes are as follows:
1. SMA1Tegal
2. AlunAlunTegal
3. Stasiun
4. YosSudarso
5. PacificMall
6. SMP7Tegal
7. ToserbaYogya
8. BetetOktaru
```

```
Please choose the start node:
>> 6

Please choose the goal node:
>> 3

Result:
1. SMP7Tegal
2. ToserbaYogya
3. AlunAlunTegal
4. Stasiun

Distance: 2.08 km
```

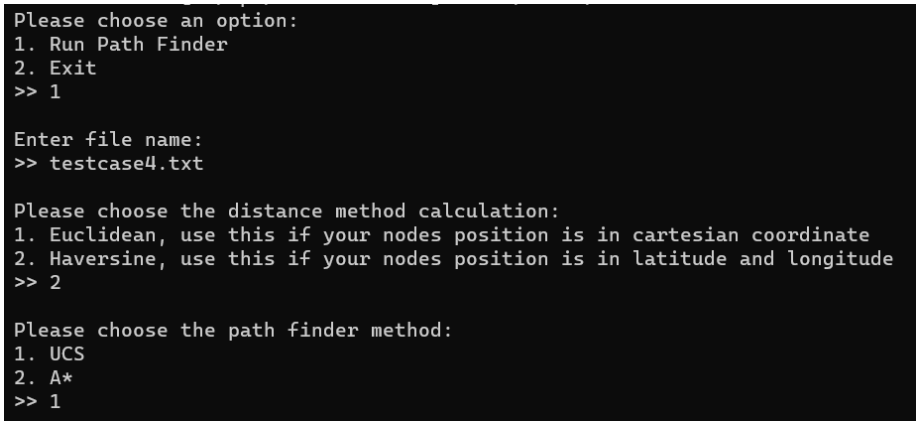
Gambar 3.24 & 3.25 Input & Hasil pecarian path pada testcase4 dengan menggunakan metode A*

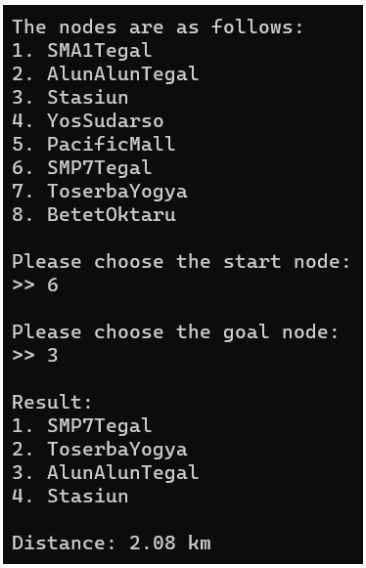


Gambar 3.26 Visualisasi graph testcase3 dengan start node SMP7Tegal dan goal node Stasiun



Gambar 3.27 Visualisasi map testcase3 dengan start node SMP7Tegal dan goal node Stasiun





Gambar 3.28 & 3.29 Input & Hasil pecarian path pada testcase4 dengan menggunakan metode UCS

BAB IV

PENUTUP

4.1 Kesimpulan

Algoritma UCS dan A* merupakan dua algoritma yang sama-sama berguna untuk mencari rute terpendek pada sebuah peta. Meskipun begitu, terdapat perbedaan yang mempengaruh efektivitas algoritma-algoritma tersebut.

Uniform Cost Search (UCS) menerapkan fungsi $f(n) = g(n)$, dimana $g(n)$ merupakan panjang rute dari node awal ke n. Sedangkan, A* menerapkan fungsi $f(n) = g(n) + h(n)$, dimana $g(n)$ merupakan panjang rute dari node awal ke n dan $h(n)$ merupakan fungsi heuristik, yaitu fungsi untuk mengestimasi jarak dari n ke node akhir.

Fungsi heuristik merupakan pembeda utama antara algoritma UCS dan A*. Algoritma UCS tidak memiliki fungsi heuristik, sehingga algoritma tidak mengetahui seberapa dekat posisi dirinya dengan node akhir, yang membuat UCS disebut sebagai *uninformed search* (pencarian kurang informasi). Sedangkan, A* memiliki fungsi heuristik, sehingga algoritma mengetahui seberapa dekat posisi dirinya dengan node akhir, yang membuat A* disebut sebagai *informed search* (pencarian dengan informasi).

4.2 Komentar

Setelah mengerjakan tugas kecil ini, kami mempelajari banyak hal. Antara lain seperti mengenal algoritma UCS, mengenal algoritma A*, memahami lebih lanjut tentang penggunaan API Google Maps, dan mengetahui cara melakukan visualisasi algoritma UCS dan A* pada python melalui input matriks ketetanggaan.

BAB V

LAMPIRAN

5.1 Pranala GitHub

https://github.com/AghnaAbyan/Tucil3_13521076_13521130

5.2 *Checklist*

Poin	Ya	Tidak
1. Program dapat menerima input graf	✓	
2. Program dapat menghitung lintasan terpendek dengan UCS	✓	
3. Program dapat menghitung lintasan terpendek dengan A*	✓	
4. Program dapat menampilkan lintasan terpendek serta jaraknya	✓	
5. Bonus: Program dapat menerima input peta dengan Google Map API		✓
6. Bonus: Program dapat menampilkan peta serta lintasan terpendek pada peta dengan Google Map API	✓	