Tugas Kecil 3 IF2211 Strategi Algoritma

Implementasi Algoritma UCS dan A* untuk Menentukan Lintasan Terpendek

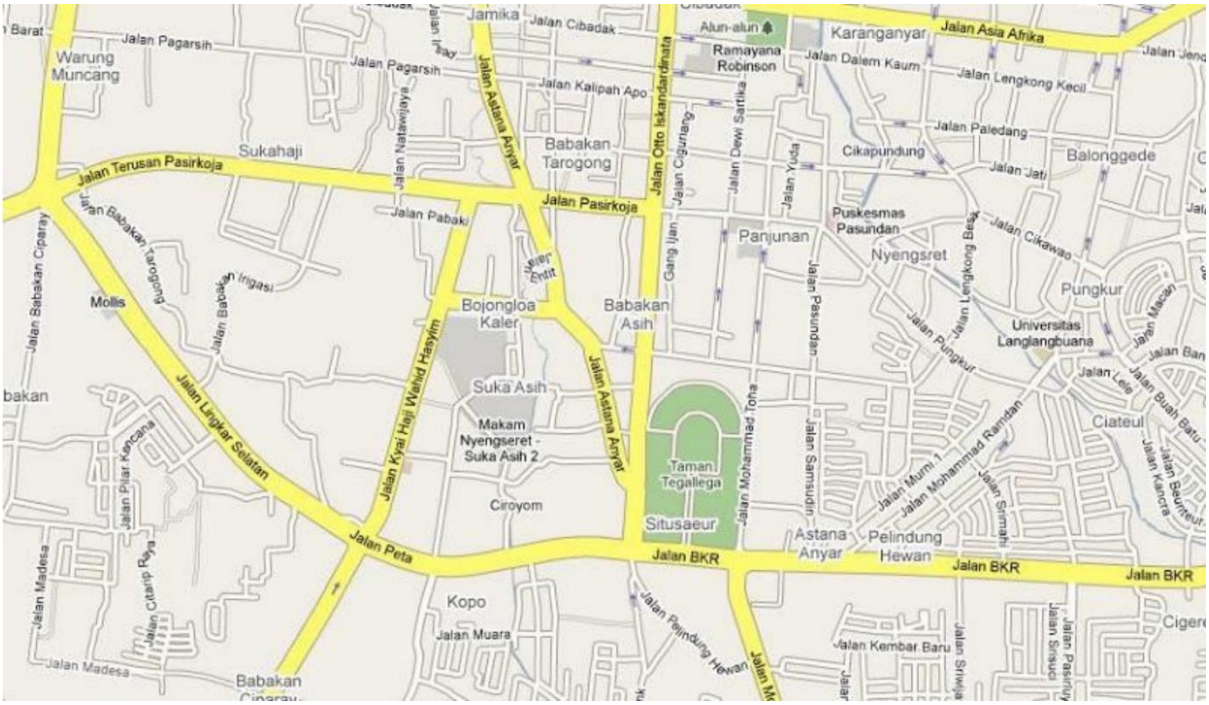
Ditulis oleh:

Muhammad Rifko Favian 13521075

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022/2023

BAGIAN I
DESKRIPSI PERSOALAN

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, kami 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.

BAGIAN II
KODE PROGRAM

1. graph.py

```python
class Graph(object):
    def __init__(self, type, numOfNode):
        self.type = type # Tipe tergantung antara untuk map atau graph biasa
        self.numOfNode = numOfNode
        self.nodeList = []

    def addNode(self, Node):
        self.nodeList.append(Node)

    def addNodeNeighbours(self, matrix):
        for i in range(self.numOfNode):
            for j in range(self.numOfNode):
                if matrix[i][j] != 0:
                    self.nodeList[i].addNeighbour(self.nodeList[j], matrix[i][j])

    def printAllNode(self):
        print("Daftar Node: ")
        for node in self.nodeList:
            node.print()
            print()

    def findNode(self, nodeName):
        for node in self.nodeList:
            if node.name == nodeName:
                return node
        return None

class Node(object):
    def __init__(self, name, x, y):
        self.name = name
        self.x = x # Latitude untuk map
        self.y = y # Longitude untuk map
        self.neighbour = [] # neighbour berisi objek Node dan jaraknya

    # Untuk sorting apabila terdapat cost yang memiliki nilai yang sama
    def __ls__(self, other):
        return self.name < other.name

    def addNeighbour(self, node, distance):
        self.neighbour.append([node, distance])

    def print(self):
        print("Nama Node:", self.name)
        print("Koordinat:", self.x, self.y)
        print("Tetangga: ", end="")
        for i in range(len(self.neighbour)):
            if i != len(self.neighbour) - 1:
                print(self.neighbour[i][0].name + " (" + str(self.neighbour[i][1]) + ")", end=", ")
            else:
                print(self.neighbour[i][0].name + " (" + str(self.neighbour[i][1]) + ")")

    def getDistanceNeighbour(self, neighbour):
        for i in range(len(self.neighbour)):
            if self.neighbour[i][0] == neighbour:
                return self.neighbour[i][1]
```

2. ucs.py

```python
import heapq

class UCS:
    def __init__(self, graph, start, goal):
        self.graph = graph
        self.start = start
        self.goal = goal
        self.frontier = [(0, start, [start])] # cost, current, path
        self.explored = []
        self.final_path = []
        self.total_cost = 0

    # Mencari path dari start ke goal dan return True jika path ditemukan dan False jika tidak
    def search(self):
        while self.frontier:
            current_cost, current, path = heapq.heappop(self.frontier)
            self.explored.append(current)

            if current == self.goal:
                self.final_path = path
                self.total_cost = current_cost
                return True

            for i in range(len(current.neighbour)):
                neighbour = current.neighbour[i][0]
                if neighbour not in self.explored:
                    new_cost = current_cost + current.neighbour[i][1]
                    heapq.heappush(self.frontier, (new_cost, neighbour, path + [neighbour]))

        return False

    # Mencetak path dan jaraknya
    def printAnswer(self):
        print("Path: ", end="")
        for i in range(len(self.final_path)):
            if i != len(self.final_path) - 1:
                print(self.final_path[i].name, end=" -> ")
            else:
                print(self.final_path[i].name)
        if self.graph.type == "map":
            print("Distance: " + str(self.total_cost) + " m") # Penambahan satuan meter
        elif self.graph.type == "normal":
            print("Distance: " + str(self.total_cost))

    # Mengembalikan path akhir (mungkin kosong)
    def getPath(self):
        return self.final_path
```

3. astar.py

```python
import heapq
import distance

class Astar:
    def __init__(self, graph, start, goal):
        self.graph = graph
        self.start = start
        self.goal = goal
        self.frontier = [(0, 0, start, [start])] # fn, gn, current, path
        self.explored = []
        self.final_path = []
        self.total_cost = 0

    # Mencari path dari start ke goal dan return True jika path ditemukan dan False jika tidak
    def search(self):
        while self.frontier:
            fn, gn, current, path = heapq.heappop(self.frontier)
            self.explored.append(current)

            if current == self.goal:
                self.final_path = path
                self.total_cost = fn  # or gn
                return True

            for i in range(len(current.neighbour)):
                neighbour = current.neighbour[i][0]
                if neighbour not in self.explored:
                    new_gn = gn + current.neighbour[i][1]
                    if self.graph.type == "map":
                        new_fn = new_gn + distance.haversine(neighbour, self.goal)
                        heapq.heappush(self.frontier, (new_fn, new_gn, neighbour, path + [neighbour]))
                    elif self.graph.type == "normal":
                        new_fn = new_gn + distance.euclidean(neighbour, self.goal)
                        heapq.heappush(self.frontier, (new_fn, new_gn, neighbour, path + [neighbour]))

        return False

    # Mencetak path dan jaraknya
    def printAnswer(self):
        print("Path: ", end="")
        for i in range(len(self.final_path)):
            if i != len(self.final_path) - 1:
                print(self.final_path[i].name, end=" -> ")
            else:
                print(self.final_path[i].name)
        if self.graph.type == "map":
            print("Distance: " + str(self.total_cost) + " m") # Penambahan satuan meter
        elif self.graph.type == "normal":
            print("Distance: " + str(self.total_cost))

    # Mengembalikan path akhir (mungkin kosong)
    def getPath(self):
        return self.final_path
```

4. distance.py

```python
import math

# Menghitung jarak antara dua titik menggunakan rumus Haversine dalam satuan meter
def haversine(node1, node2):
    R = 6371000 # Earth's radius in meters
    lat1_rad = math.radians(node1.x)
    lat2_rad = math.radians(node2.x)
    dLat = math.radians(node2.x - node1.x)
    dLon = math.radians(node2.y - node2.y)

    a = math.sin(dLat / 2) * math.sin(dLat / 2) + math.cos(lat1_rad) * math.cos(lat2_rad) * math.sin(dLon / 2) * math.sin(dLon / 2)
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    d = R * c

    return round(d, 4)

# Menghitung jarak antara dua titik menggunakan rumus Euclidean
def euclidean(node1, node2):
    d = math.sqrt((node1.x - node2.x) ** 2 + (node1.y - node2.y) ** 2)
    return round(d, 4)
```

5. Beberapa fungsi pada main

```python
# Membuat graph dari matriks berbobot
def makeGraphFromFile(file, type):
    # Membaca file dan inisiasi graph kosong
    fileLines = file.read().splitlines()
    numOfPlace = int(fileLines[0])
    locationList = fileLines[1:numOfPlace+1]
    readMatrix = fileLines[numOfPlace+1:]
    adjacencyMatrix = [[0 for i in range(numOfPlace)] for j in range(numOfPlace)]

    graph = Graph(type, numOfPlace)

    # Membuat objek Node dan menambahkan ke graph
    for i in range(numOfPlace):
        splitString = locationList[i].split(", ")
        nameOfNode = splitString[0]
        xOfNode = float(splitString[1])
        yOfNode = float(splitString[2])
        newNode = Node(nameOfNode, xOfNode, yOfNode)
        graph.addNode(newNode)
        adjacencyMatrix[i] = [float(x) for x in readMatrix[i].split(" ")]

    graph.addNodeNeighbours(adjacencyMatrix)

    return graph

def visualizeGraph(graph, path): # Hanya untuk graf biasa
    G = nx.Graph()
    for node in graph.nodeList:
        G.add_node(node.name, pos=(node.x, node.y))
        for neighbour in node.neighbour:
            G.add_edge(node.name, neighbour[0].name, weight=neighbour[1], color='black')


    # Path yang dilewati akan berwarna merah
    for i in range(len(path) - 1):
        G.remove_edge(path[i].name, path[i+1].name)
        G.add_edge(path[i].name, path[i+1].name, weight=path[i].getDistanceNeighbour(path[i+1]), color='red')

    pos = nx.get_node_attributes(G, 'pos')
    edge_colors = nx.get_edge_attributes(G, 'color').values()
    nx.draw(G, pos,with_labels=True,edge_color=edge_colors,
            node_size=1000,font_color="white",font_size=20,
            font_family="Times New Roman", font_weight="bold",width=5)
    edge_labels = nx.get_edge_attributes(G,'weight')
    nx.draw_networkx_edge_labels(G,pos,edge_labels=edge_labels,font_size=10)
    plt.show()
```

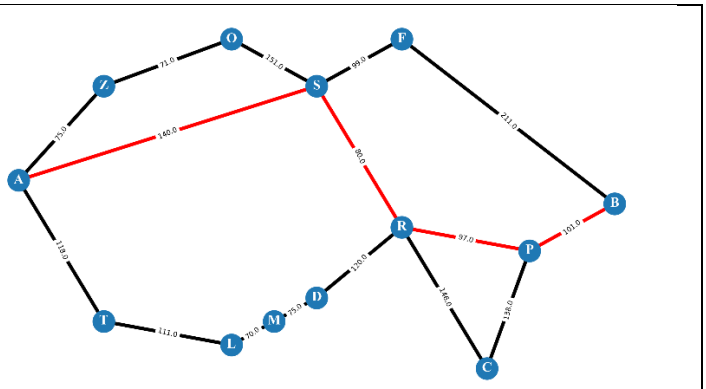
6. Gambaran isi dari main program

```python
if inputType == "Biasa" or inputType == "biasa":
    if algorithm.search():
        algorithm.printAnswer()
        visualizeGraph(graph, algorithm.getPath())
    else:
        print("Path tidak ditemukan")

# Menampilkan hasil pencarian untuk graf map
elif inputType == "Map" or inputType == "map":
    # Initialize Flask and render map
    app = Flask(__name__, template_folder="visual")

    @app.route('/')
    def map():
        return render_template('map.html')

    if not algorithm.search():
        print("\nPath tidak ditemukan")

    # Menampilkan koordinat awal
    start_coords = [graph.findNode(start).x, graph.findNode(start).y]
    map = folium.Map(location=start_coords, zoom_start=500)

    # Menambahkan marker dan garis hitam antar semua Location
    for location in graph.nodeList:
        folium.Marker(location=[location.x, location.y], popup=location.name, icon=folium.Icon(color='blue')).add_to(map)

    # Membedakan warna marker titik awal dan titik akhir
    folium.Marker(start_coords, popup=graph.findNode(start).name, icon=folium.Icon(color='green')).add_to(map)
    folium.Marker([graph.findNode(end).x, graph.findNode(end).y], popup=graph.findNode(end).name, icon=folium.Icon(color='orange')).add_to(map)

    for location in graph.nodeList:
        for neighbour in location.neighbour:
            folium.PolyLine(locations=[[location.x, location.y], [neighbour[0].x, neighbour[0].y]], color='black').add_to(map)

    path = algorithm.getPath()

    # Menambahkan garis merah antar Location yang ada di path
    if path != None:
        print()
        algorithm.printAnswer()
        for i in range(len(path)-1):
            folium.PolyLine(locations=[[path[i].x, path[i].y], [path[i+1].x, path[i+1].y]], color='red').add_to(map)

    if not os.path.exists('src/visual'):
        os.makedirs('src/visual')
    map.save('src/visual/map.html')

    if path == None:
        ans = input("Apakah Anda tetap ingin melihat visualisasi graph-Nya? (Y/n): ")
    elif path != None:
        ans = input("Apakah Anda ingin melihat visualisasi path-Nya? (Y/n): ")

    while ans != "Y" and ans != "y" and ans != "N" and ans != "n":
        print("Masukan salah, coba lagi\n")
        if path == None:
            ans = input("Apakah Anda tetap ingin melihat visualisasi graph-Nya? (Y/n): ")
        elif path != None:
            ans = input("Apakah Anda ingin melihat visualisasi path-Nya? (Y/n): ")

    if ans == "Y" or ans == "y":
        app.run(debug=False)

    os.remove('src/visual/map.html')
```

BAGIAN III
CONTOH PROGRAM

Contoh 1 bukan peta (dengan UCS)

Input	Output
13 A, 2, 0 (titik mulai) Z, 3, 1 T, 3, -1.5 O, 4.5, 1.5 L, 4.5, -1.75 M, 5, -1.5 S, 5.5, 1 D, 5.5, -1.25 R, 6.5, -0.5 F, 6.5, 1.5 C, 7.5, -2 P, 8, -0.75 B, 9, -0.25 (titik akhir) 0 75 118 0 0 0 140 0 0 0 0 0 0 75 0 0 71 0 0 0 0 0 0 0 0 0 118 0 0 0 111 0 0 0 0 0 0 0 0 0 71 0 0 0 0 151 0 0 0 0 0 0 0 0 111 0 0 70 0 0 0 0 0 0 0 0 0 0 0 70 0 0 75 0 0 0 0 0 140 0 0 151 0 0 0 0 80 99 0 0 0 0 0 0 0 0 75 0 0 120 0 0 0 0 0 0 0 0 0 0 80 120 0 0 146 97 0 0 0 0 0 0 0 99 0 0 0 0 0 211 0 0 0 0 0 0 0 0 146 0 0 138 0 0 0 0 0 0 0 0 0 97 0 138 0 101 0 0 0 0 0 0 0 0 0 211 0 101 0	Masukan berupa graph biasa atau map (Biasa/Map)? biasa Masukkan nama file: test Daftar Node: Nama Node: A Koordinat: 2.0 0.0 Tetangga: Z (75), T (118), S (140) Nama Node: Z Koordinat: 3.0 1.0 Tetangga: A (75), O (71) Nama Node: T Koordinat: 3.0 -1.5 Tetangga: A (118), L (111) Nama Node: O Koordinat: 4.5 1.5 Tetangga: Z (71), S (151) Nama Node: L Koordinat: 4.5 -1.75 Tetangga: T (111), M (70) Nama Node: M Koordinat: 5.0 -1.5 Tetangga: L (70), D (75) Nama Node: S Koordinat: 5.5 1.0 Tetangga: A (140), O (151), R (80), F (99) Nama Node: D Koordinat: 5.5 -1.25 Tetangga: M (75), R (120) Nama Node: R Koordinat: 6.5 -0.5 Tetangga: S (80), D (120), C (146), P (97) Nama Node: F Koordinat: 6.5 1.5 Tetangga: S (99), B (211) Nama Node: C Nama Node: C Koordinat: 7.5 -2.0 Tetangga: R (146), P (138) Nama Node: P Koordinat: 8.0 -0.75 Tetangga: R (97), C (138), B (101) Nama Node: B Koordinat: 9.0 -0.25 Tetangga: F (211), P (101) Masukkan nama titik awal: A Masukkan nama titik akhir: B Masukkan algoritma yang ingin digunakan (UCS/A*): ucs Path: A -> S -> R -> P -> B Distance: 418

Contoh 2 Area ITB (dengan A*)

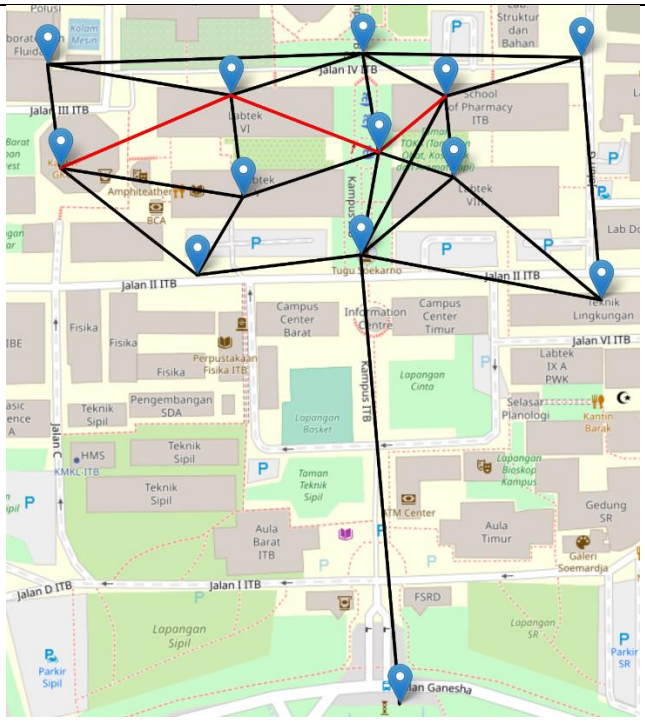
Input di txt:
13 Kubus, -6.893311689500024, 107.61054852083436 Titik Tengah, -6.890915377731855, 107.61034091465416 Parkiran Labtek V, -6.891023877300416, 107.60945879583674 GKUB, -6.890455545882821, 107.60872239575617 Intel, -6.89036884274947, 107.61043670639772 Gedung LFM, -6.891156756723175, 107.61162847754282 Kantin Bengkok, -6.889862510450611, 107.61152439273246 Gedung FTMD, -6.889898445152209, 107.6086521160611 Labtek V, -6.89060627683087, 107.6097059748216 Labtek VI, -6.890066361526841, 107.60963831969185 Labtek VII, -6.890071528190981, 107.6107962632752 Labtek VIII, -6.890500360487576, 107.61083529508082 Pusat Gema, -6.889841034853903, 107.6103502670937 0 254.11 0 0 0 0 0 0 0 0 0 0 254.11 0 100.6 0 66.49 133.73 0 0 0 0 56.75 55.57 0 0 100.6 0 95.23 0 0 0 0 42.27 0 0 0 0 0 0 95.23 0 0 0 0 62.75 96.84 96.02 0 0 0 0 66.49 0 0 0 0 0 0 89.42 88.77 63.34 44.98 55.69 0 133.73 0 0 0 0 124.66 0 0 0 0 104.23 0 0 0 0 0 0 124.66 0 0 0 0 82.97 0 133.54 0 0 0 62.75 0 0 0 0 0 80.83 0 0 169.74 0 0 42.27 96.84 89.42 0 0 0 0 42.05 0 0 0 0 0 0 96.02 88.77 0 0 80.83 42.05 0 0 0 102.4 0 56.75 0 0 63.34 0 82.97 0 0 0 0 54.33 56.84 0 55.57 0 0 44.98 104.23 0 0 0 0 54.33 0 0 0 0 0 0 55.69 0 133.54 169.74 0 102.4 56.84 0 0
Output:

```
Masukan berupa graph biasa atau map (Biasa/Map)? map
Masukkan nama file: itb
Daftar Node:
Nama Node: Kubus
Koordinat: -6.893311689500024 107.61054852083436
Tetangga: Titik Tengah (254.11)

Nama Node: Titik Tengah
Koordinat: -6.890915377731855 107.61034091465416
Tetangga: Kubus (254.11), Parkiran Labtek V (100.6), Intel (66.49), Gedung LFM (133.73), Labtek VII (56.75), Labtek VIII (55.57)

Nama Node: Parkiran Labtek V
Koordinat: -6.891023877300416 107.60945879583674
Tetangga: Titik Tengah (100.6), GKUB (95.23), Labtek V (42.27)

Nama Node: GKUB
Koordinat: -6.890455545882821 107.60872239575617
Tetangga: Parkiran Labtek V (95.23), Gedung FTMD (62.75), Labtek V (96.84), Labtek VI (96.02)

Nama Node: Intel
Koordinat: -6.89036884274947 107.61043670639772
Tetangga: Titik Tengah (66.49), Labtek V (89.42), Labtek VI (88.77), Labtek VII (63.34), Labtek VIII (44.98), Pusat Gema (55.69)

Nama Node: Gedung LFM
Koordinat: -6.891156756723175 107.61162847754282
Tetangga: Titik Tengah (133.73), Kantin Bengkok (124.66), Labtek VIII (104.23)

Nama Node: Kantin Bengkok
Koordinat: -6.889862510450611 107.61152439273246
Tetangga: Gedung LFM (124.66), Labtek VII (82.97), Pusat Gema (133.54)

Nama Node: Gedung FTMD
Koordinat: -6.889898445152209 107.6086521160611
Tetangga: GKUB (62.75), Labtek VI (80.83), Pusat Gema (169.74)

Nama Node: Labtek V
Koordinat: -6.89060627683087 107.6097059748216
Tetangga: Parkiran Labtek V (42.27), GKUB (96.84), Intel (89.42), Labtek VI (42.05)

Nama Node: Labtek VI
Koordinat: -6.890066361526841 107.60963831969185
Tetangga: GKUB (96.02), Intel (88.77), Gedung FTMD (80.83), Labtek V (42.05), Pusat Gema (102.4)
```

```
Nama Node: Labtek VII
Koordinat: -6.890071528190981 107.6107962632752
Tetangga: Titik Tengah (56.75), Intel (63.34), Kantin Bengkok (82.97), Labtek VIII (54.33), Pusat Gema (56.84)

Nama Node: Labtek VIII
Koordinat: -6.890500360487576 107.61083529508082
Tetangga: Titik Tengah (55.57), Intel (44.98), Gedung LFM (104.23), Labtek VII (54.33)

Nama Node: Pusat Gema
Koordinat: -6.889841034853903 107.6103502670937
Tetangga: Intel (55.69), Kantin Bengkok (133.54), Gedung FTMD (169.74), Labtek VI (102.4), Labtek VII (56.84)

Masukkan nama titik awal: GKUB
Masukkan nama titik akhir: Labtek VII
Masukkan algoritma yang ingin digunakan (UCS/A*): A*

Path: GKUB -> Labtek VI -> Intel -> Labtek VII
Distance: 248.13 m
```

Tambahan untuk visualisasi map dengan path pada link yang terbentuk

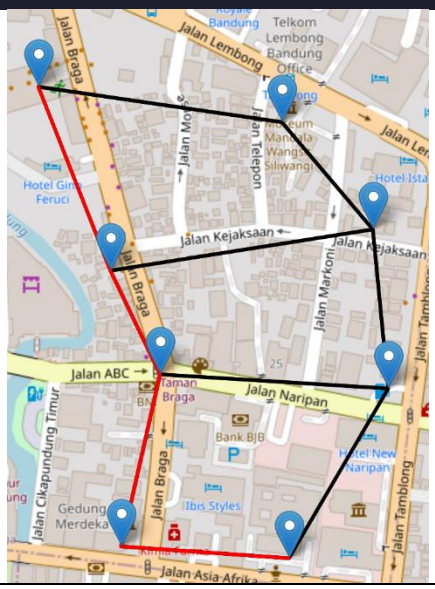
```
Apakah Anda ingin melihat visualisasi path-Nya? (Y/n): y
 * Serving Flask app 'main'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```



Contoh 3 Alun-alun Bandung (dengan A*)

Input txt:
8 Museum KAA, -6.921284918454274, 107.60957943240983 Simpang Braga, -6.919723089454119, 107.609940342634 Jurnal Risa, -6.918789028521304, 107.60948363279562 Braga City Walk, -6.91713043077999, 107.60883021614896 Museum Mandala, -6.9174506555519075, 107.61103859891946 Bakmi Rica Kejaksaan, -6.918415434001528, 107.61186984411852 SPBU, -6.919859329250897, 107.61201040242935 The Centre of Bandung, -6.921387017201472, 107.611106436847 0 176.98 0 0 0 0 0 162.5 176.98 0 120.86 0 0 0 229.44 0 0 120.86 0 185.92 0 268.73 0 0 0 0 185.92 0 232.2 0 0 0 0 0 0 232.2 0 166.93 0 0 0 0 268.73 0 166.93 0 160.47 0 0 229.44 0 0 0 160.47 0 202.82 162.5 0 0 0 0 0 202.82 0
Output:

```
Masukan berupa graph biasa atau map (Biasa/Map)? map
Masukkan nama file: alunalunbandung
Daftar Node:
Nama Node: Museum KAA
Koordinat: -6.921284918454274 107.60957943240983
Tetangga: Simpang Braga (176.98), The Centre of Bandung (162.5)

Nama Node: Simpang Braga
Koordinat: -6.919723089454119 107.609940342634
Tetangga: Museum KAA (176.98), Jurnal Risa (120.86), SPBU (229.44)

Nama Node: Jurnal Risa
Koordinat: -6.918789028521304 107.60948363279562
Tetangga: Simpang Braga (120.86), Braga City Walk (185.92), Bakmi Rica Kejaksaan (268.73)

Nama Node: Braga City Walk
Koordinat: -6.91713043077999 107.60883021614896
Tetangga: Jurnal Risa (185.92), Museum Mandala (232.2)

Nama Node: Museum Mandala
Koordinat: -6.9174506555519075 107.61103859891946
Tetangga: Braga City Walk (232.2), Bakmi Rica Kejaksaan (166.93)

Nama Node: Bakmi Rica Kejaksaan
Koordinat: -6.918415434001528 107.61186984411852
Tetangga: Jurnal Risa (268.73), Museum Mandala (166.93), SPBU (160.47)

Nama Node: SPBU
Koordinat: -6.919859329250897 107.61201040242935
Tetangga: Simpang Braga (229.44), Bakmi Rica Kejaksaan (160.47), The Centre of Bandung (202.82)

Nama Node: The Centre of Bandung
Koordinat: -6.921387017201472 107.611106436847
Tetangga: Museum KAA (162.5), SPBU (202.82)

Masukkan nama titik awal: The Centre of Bandung
Masukkan nama titik akhir: Braga City Walk
Masukkan algoritma yang ingin digunakan (UCS/A*): A*

Path: The Centre of Bandung -> Museum KAA -> Simpang Braga -> Jurnal Risa -> Braga City Walk
Distance: 646.26 m
```



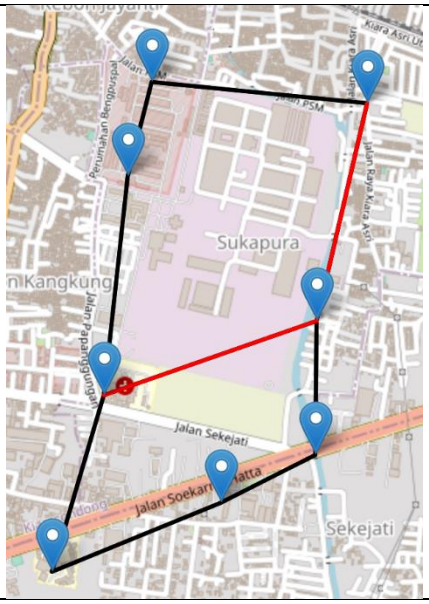
Contoh 3 Buah Batu (dengan UCS)

Input txt:

```
8
Universitas Islam Nusantara, -6.945832500507639, 107.64434603645425
Edelweiss Hospital, -6.943617699023188, 107.64979863586177
Ciduran Selatan Bypass, -6.9421024125955135, 107.652860318267
Masjid Miftahul Jannah, -6.937807631379689, 107.65290393209608
RSU Pindad, -6.9401974740772765, 107.64600424305948
Gedung Graha SembadhaVicaya, -6.933174808590936, 107.6467978495106
Direktorat Perhub AD, -6.930144163663747, 107.64755672810774
Pertigaan Terusan PSM, -6.930812727620998, 107.6545234905284
0 601.97 0 0 625.17 0 0 0
601.97 0 386.76 0 0 0 0 0
0 386.76 0 460.15 0 0 0 0
0 0 460.15 0 780.39 0 0 776.34
625.17 0 0 780.39 0 753.74 0 0
0 0 0 0 753.74 0 375.02 0
0 0 0 0 0 375.02 0 756.08
0 0 0 776.34 0 0 756.08 0
```

Output:

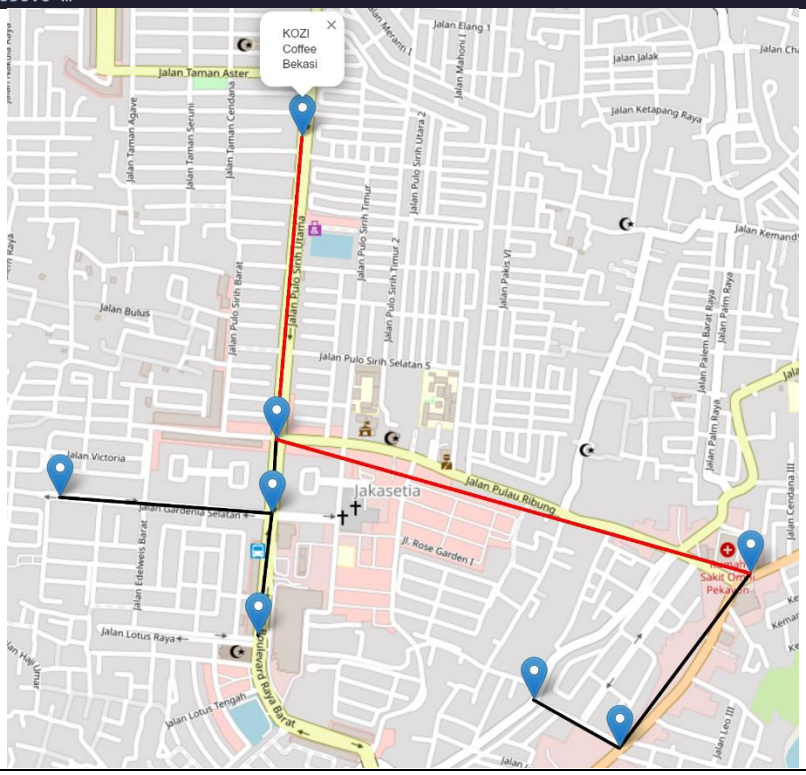
```
Masukan berupa graph biasa atau map (Biasa/Map)? map
Masukkan nama file: BuahBatu
Daftar Node:
Nama Node: Universitas Islam Nusantara
Koordinat: -6.945832500507639 107.64434603645425
Tetangga: Edelweiss Hospital (601.97), RSU Pindad (625.17)

Nama Node: Edelweiss Hospital
Koordinat: -6.943617699023188 107.64979863586177
Tetangga: Universitas Islam Nusantara (601.97), Ciduran Selatan Bypass (386.76)

Nama Node: Ciduran Selatan Bypass
Koordinat: -6.9421024125955135 107.652860318267
Tetangga: Edelweiss Hospital (386.76), Masjid Miftahul Jannah (460.15)

Nama Node: Masjid Miftahul Jannah
Koordinat: -6.937807631379689 107.65290393209608
Tetangga: Ciduran Selatan Bypass (460.15), RSU Pindad (780.39), Pertigaan Terusan PSM (776.34)

Nama Node: RSU Pindad
Koordinat: -6.9401974740772765 107.64600424305948
Tetangga: Universitas Islam Nusantara (625.17), Masjid Miftahul Jannah (780.39), Gedung Graha SembadhaVicaya (753.74)

Nama Node: Gedung Graha SembadhaVicaya
Koordinat: -6.933174808590936 107.6467978495106
Tetangga: RSU Pindad (753.74), Direktorat Perhub AD (375.02)

Nama Node: Direktorat Perhub AD
Koordinat: -6.930144163663747 107.64755672810774
Tetangga: Gedung Graha SembadhaVicaya (375.02), Pertigaan Terusan PSM (756.08)

Nama Node: Pertigaan Terusan PSM
Koordinat: -6.930812727620998 107.6545234905284
Tetangga: Masjid Miftahul Jannah (776.34), Direktorat Perhub AD (756.08)

Masukkan nama titik awal: RSU Pindad
Masukkan nama titik akhir: Pertigaan Terusan PSM
Masukkan algoritma yang ingin digunakan (UCS/A*): ucs

Path: RSU Pindad -> Masjid Miftahul Jannah -> Pertigaan Terusan PSM
Distance: 1556.73 m
```



Contoh 4 Bekasi (dengan UCS)

Input txt:
8 Grand Galaxy Park, -6.270771, 106.971625 Kedai Maniac, -6.266856, 106.971992 McDonald's Pekayon, -6.269521, 106.981468 Alfamart Pekayon, -6.273015, 106.978860 Jaka Setia, -6.272035, 106.977134 KOZI Coffee Bekasi, -6.260849, 106.972500 Warung Makan Belut Galaxy, -6.268023, 106.967645 Rumah Sakit Medika Galaxi, -6.268332, 106.971896 0 0 0 0 0 0 0 320 0 0 1160 0 0 890 0 180 0 1160 0 690 0 0 0 0 0 0 690 0 520 0 0 0 0 0 0 520 0 0 0 0 0 890 0 0 0 0 0 0 0 0 0 0 0 0 0 630 320 180 0 0 0 0 630 0
Output:

```
Masukan berupa graph biasa atau map (Biasa/Map)? map
Masukkan nama file: bekasi
Daftar Node:
Nama Node: Grand Galaxy Park
Koordinat: -6.270771 106.971625
Tetangga: Rumah Sakit Medika Galaxi (320.0)

Nama Node: Kedai Maniac
Koordinat: -6.266856 106.971992
Tetangga: McDonald's Pekayon (1160.0), KOZI Coffee Bekasi (890.0), Rumah Sakit Medika Galaxi (180.0)

Nama Node: McDonald's Pekayon
Koordinat: -6.269521 106.981468
Tetangga: Kedai Maniac (1160.0), Alfamart Pekayon (690.0)

Nama Node: Alfamart Pekayon
Koordinat: -6.273015 106.97886
Tetangga: McDonald's Pekayon (690.0), Jaka Setia (520.0)

Nama Node: Jaka Setia
Koordinat: -6.272035 106.977134
Tetangga: Alfamart Pekayon (520.0)

Nama Node: KOZI Coffee Bekasi
Koordinat: -6.260849 106.9725
Tetangga: Kedai Maniac (890.0)

Nama Node: Warung Makan Belut Galaxy
Koordinat: -6.268023 106.967645
Tetangga: Rumah Sakit Medika Galaxi (630.0)

Nama Node: Rumah Sakit Medika Galaxi
Koordinat: -6.268332 106.971896
Tetangga: Grand Galaxy Park (320.0), Kedai Maniac (180.0), Warung Makan Belut Galaxy (630.0)

Masukkan nama titik awal: KOZI Coffee Bekasi
Masukkan nama titik akhir: McDonald's Pekayon
Masukkan algoritma yang ingin digunakan (UCS/A*): ucs

Path: KOZI Coffee Bekasi -> Kedai Maniac -> McDonald's Pekayon
Distance: 2050.0 m
```



BAGIAN IV
LINK REPOSITORY

Link repository GitHub : https://github.com/Mifkiyan/Tucil3_13521075

BAGIAN V
KESIMPULAN

Dari hasil pengerjaan Tugas Kecil ini, saya memiliki kesimpulan bahwa Algoritma UCS dan A* adalah algoritma yang sangat baik untuk digunakan dalam mencari jarak path antara dua simpul terpendek dari suatu graf. Hal ini dikarenakan kedua algoritma sudah memberikan solusi path terpendek yang optimal berdasarkan uji coba yang sudah dilakukan

Dari proses dan hasil pengerjaan ini saya juga memiliki komentar kalua Tugas Kecil ini mencakup spesifikasi yang cukup menarik, karena dengan mengerjakan tugas ini saya menjadi lebih mendalami materi *path finding*.

BAGIAN VI
CHECK LIST

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	✓	