LAPORAN TUGAS KECIL 3

IF2211 – STRATEGI ALGORITME

SEMESTER II - TAHUN 2020/2021

Implementasi Algoritme A* untuk Menentukan Lintasan Terpendek

1

2

Daftar Anggota Kelompok:

1. **Andrew (13519036)**
2. **Leonardus Brandon Luwianto (13519102)**

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

2021

Kode Program

File **classes.py** berisi kelas Graph dan kelas Node.

```
import matplotlib.pyplot as plt

class Graph:
    def __init__(self):
        self.nodes = []
        self.nodesCount = 0
        self.aStarPath = []

    def addNode(self, nodes):
        self.nodes.append(nodes)
        self.nodesCount += 1

    def getNodeByName(self, name):
        for n in self.nodes:
            if n.name == name:
                return n
        return None

    def showGraph(self):
        print("Berikut list lokasi yang terdapat dalam peta:")
        for e in self.nodes:
            e.showNodeName()
        return

    def visualizeGraph(self):
        plt.axis([-10,10,-10,10])
        listName = []
        listX = []
        listY = []
        listEdges = []
        for node in self.nodes:
            listName.append(node.name)
            listX.append(node.x)
            listY.append(node.y)
            for adjNodes in node.adjacentNodes:
                listEdges.append((node.name,adjNodes))

        for i in range(self.nodesCount):
            plt.plot(listX[i],listY[i],"bo")
            plt.text(listX[i],listY[i],listName[i])

        for i in range(len(listEdges)):
            node1 = self.getNodeByName(listEdges[i][0])
            node2 = self.getNodeByName(listEdges[i][1])
            plt.plot([node1.x,node2.x],[node1.y,node2.y],color='b')
            plt.text((node1.x+node2.x)/2, (node1.y+node2.y)/2,
str(round(self.euclideanDistance(node1,node2),2)))
```

```
        plt.show()
        return

    def visualizePath(self):
        if self.aStarPath == None:
            print("Tidak terdapat jalan yang menghubungkan kedua lokasi.\n")
            return
        plt.axis([-10,10,-10,10])
        listName = []
        listX = []
        listY = []
        listNameInPath = []
        listEdges = []
        for node in self.nodes:
            listName.append(node.name)
            listX.append(node.x)
            listY.append(node.y)
            if node in self.aStarPath:
                listNameInPath.append(node.name)
            for adjNodes in node.adjacentNodes:
                listEdges.append((node.name,adjNodes))

        for i in range(self.nodesCount):
            plt.text(listX[i],listY[i],listName[i])
            if listName[i] in listNameInPath:
                plt.plot(listX[i],listY[i],"ro")
                continue
            plt.plot(listX[i],listY[i],"bo")

        for i in range(len(listEdges)):
            node1 = self.getNodeByName(listEdges[i][0])
            node2 = self.getNodeByName(listEdges[i][1])
            if node1.name in listNameInPath and node2.name in listNameInPath:
                plt.plot([node1.x,node2.x],[node1.y,node2.y],color='r')
                plt.text((node1.x+node2.x)/2, (node1.y+node2.y)/2,
str(round(self.euclideanDistance(node1,node2),2)))
                continue
            plt.plot([node1.x,node2.x],[node1.y,node2.y],color='b')
            plt.text((node1.x+node2.x)/2, (node1.y+node2.y)/2,
str(round(self.euclideanDistance(node1,node2),2)))

        plt.show()
        print("Jarak dari {0} menuju {1} adalah
{2}\n".format(self.aStarPath[0].name,self.aStarPath[-
1].name,str(round(self.aStarPath[-1].g,2))))
        return

    def euclideanDistance(self, node1, node2):
        return ((node1.x - node2.x)**2 + (node1.y - node2.y)**2)**(1/2)

    def aStar(self, start, end):
        self.aStarPath = []
```

```
        # define get node with minimum f value function
        def getNodeByF(openNodes):
            minNode = openNodes[0]
            for n in openNodes:
                if minNode.f > n.f:
                    minNode = n
            return minNode

        # initialize variables
        openNodes = []
        closedNodes = []

        # algorithm
        openNodes.append(self.getNodeByName(start))
        endNode = self.getNodeByName(end)
        while any(openNodes):
            current = getNodeByF(openNodes)

            # case 1 : path found
            if current == endNode:
                while current.previous != None:
                    self.aStarPath.append(current)
                    current = current.previous
                self.aStarPath.append(current)
                self.aStarPath = self.aStarPath[::-1]
                return

            # case 2 : path not yet found
            openNodes.remove(current)
            closedNodes.append(current)
            for adjacentNode in current.adjacentNodes:
                node = self.getNodeByName(adjacentNode)
                if node in closedNodes:
                    continue

                tempG = self.euclideanDistance(current,node) + current.g
                tempH = self.euclideanDistance(node,endNode)
                tempF = tempG + tempH

                if node in openNodes:
                    if tempG >= node.g:
                        continue

                node.f = tempF
                node.g = tempG
                node.h = tempH
                node.previous = current
                openNodes.append(node)

        # case 3 : path not found
        self.aStarPath = None
        return
```

```
class Node:
    def __init__(self, name, x, y, adjacentNodes):
        self.name = name
        self.x = x
        self.y = y
        self.adjacentNodes = adjacentNodes
        self.previous = None
        self.f = 0
        self.g = 0
        self.h = 0

    def showNodeName(self):
        print(self.name)
```

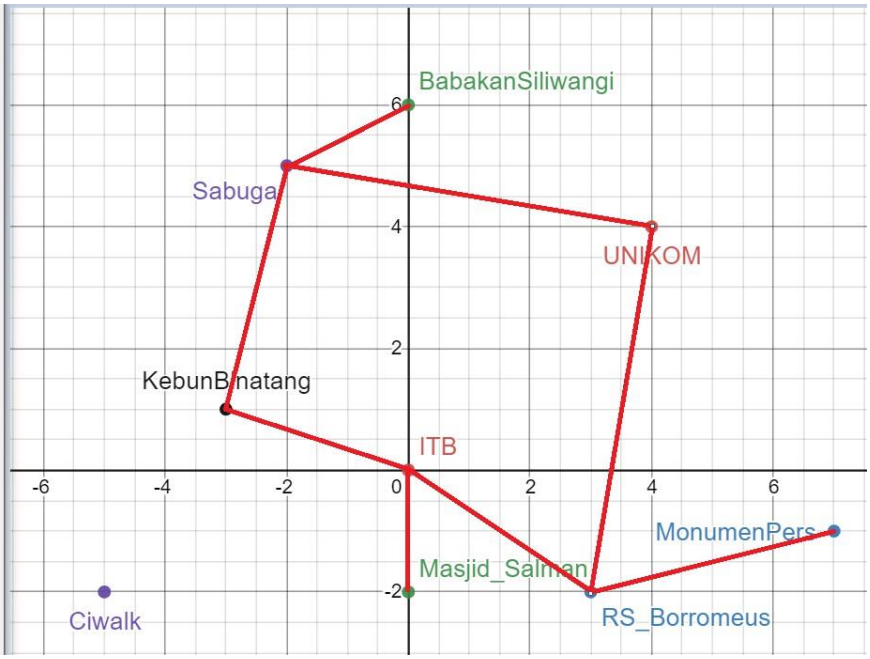
File **main.py** berisi program utama.

```
from classes import Graph, Node


# File Handling
def createGraphFromFile(filename):

    # open input file
    f = open("test/"+filename, "r")
    tempGraph = Graph()
    filelines = [line.strip() for line in f]

    # initialize vertex count
    num = int(filelines[0])
    filelines.pop(0)

    # initialize list of nodes
    nodesList = []
    for _ in range(num):
        nodesList.append(filelines[0].split())
        filelines.pop(0)

    # initialize adjacency matrix
    adjacencyMatrix = []
    for _ in range(num):
        adjacencyMatrix.append(''.join(filelines[0].split()))
        filelines.pop(0)

    # initialize graph
    for i in range(num):
        tempName = nodesList[i][2]
        tempX = int(nodesList[i][0])
        tempY = int(nodesList[i][1])
        tempAdj = []
```

```
        for j in range(num):
            if adjacencyMatrix[i][j] == "1":
                tempAdj.append(nodesList[j][2])

        tempNode = Node(tempName, tempX, tempY, tempAdj)
        tempGraph.addNode(tempNode)

    return tempGraph

# Main Function
if __name__ == "__main__":
    print("Input \"#\" untuk keluar.")

    # file input handler
    filename = input("Enter file name: ")
    while (filename != "#"):
        try:
            txtMap = createGraphFromFile(filename)
            txtMap.visualizeGraph()
            txtMap.showGraph()
            print("\n")

            # node input handler
            print("Silakan masukkan lokasi asal dan tujuan (Format:
<lokasi_asal><spasi><lokasi_tujuan>, contoh: ITB Sabuga) : ")
            start, end = input().split()

            # validate node input
            while not txtMap.getNodeByName(start) or not txtMap.getNodeByName(end):
                if not txtMap.getNodeByName(start):
                    print("Tidak ada lokasi dengan nama {0}".format(start))
                if not txtMap.getNodeByName(end):
                    print("Tidak ada lokasi dengan nama {0}".format(end))
                start, end = input("Silakan masukkan lokasi asal dan tujuan (Format:
<lokasi_asal><spasi><lokasi_tujuan>, contoh: ITB Sabuga) : ").split()

            txtMap.aStar(start,end)
            txtMap.visualizePath()
            filename = input("Enter file name: ")

        except Exception:
            print("File tidak ditemukan")
            filename = input("Enter file name: ")
```

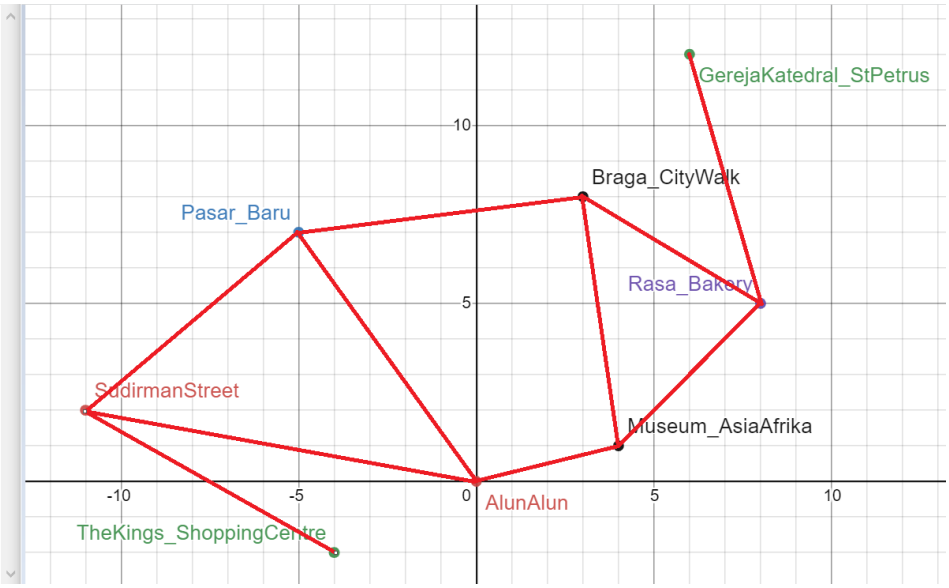
Peta/Graf Input

```
9
0 0 ITB
3 -2 RS_Borromeus
0 -2 Masjid_Salman
-5 -2 Ciwalk
-3 1 KebunBinatang
4 4 UNIKOM
7 -1 MonumenPers
0 6 BabakanSiliwangi
-2 5 Sabuga
0 1 1 0 1 0 0 0 0
1 0 0 0 0 1 1 0 0
1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1
0 0 0 0 1 1 0 1 0
```

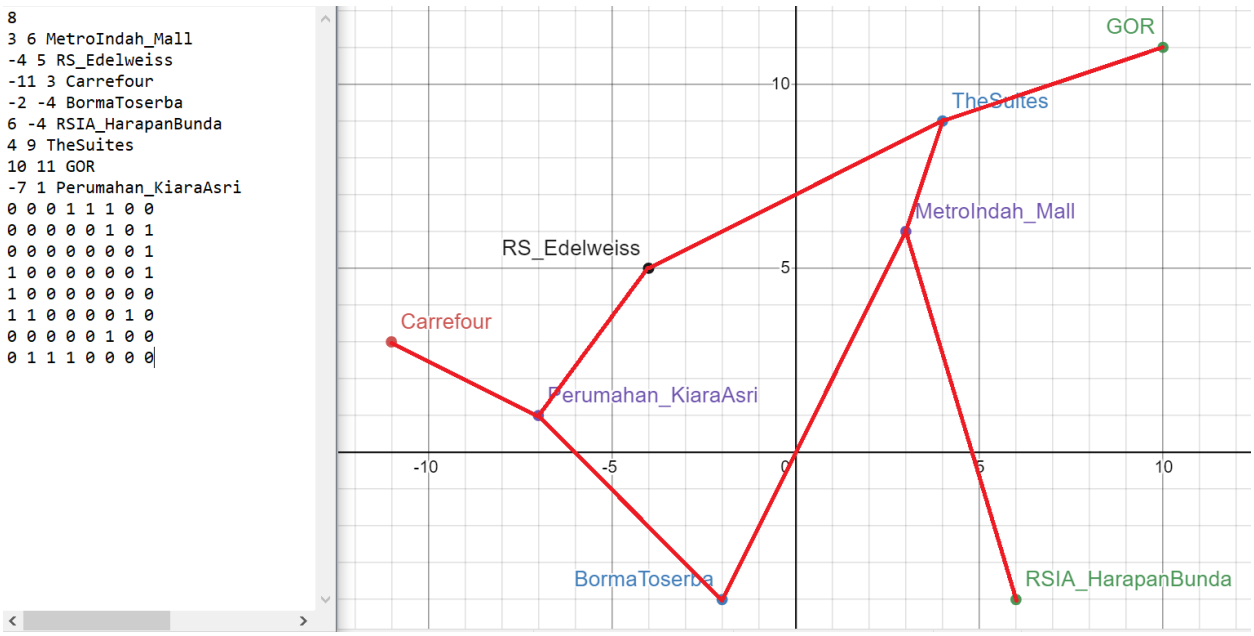


Gambar 1: File *input1.txt* merepresentasikan peta jalan sekitar kampus ITB/Dago.

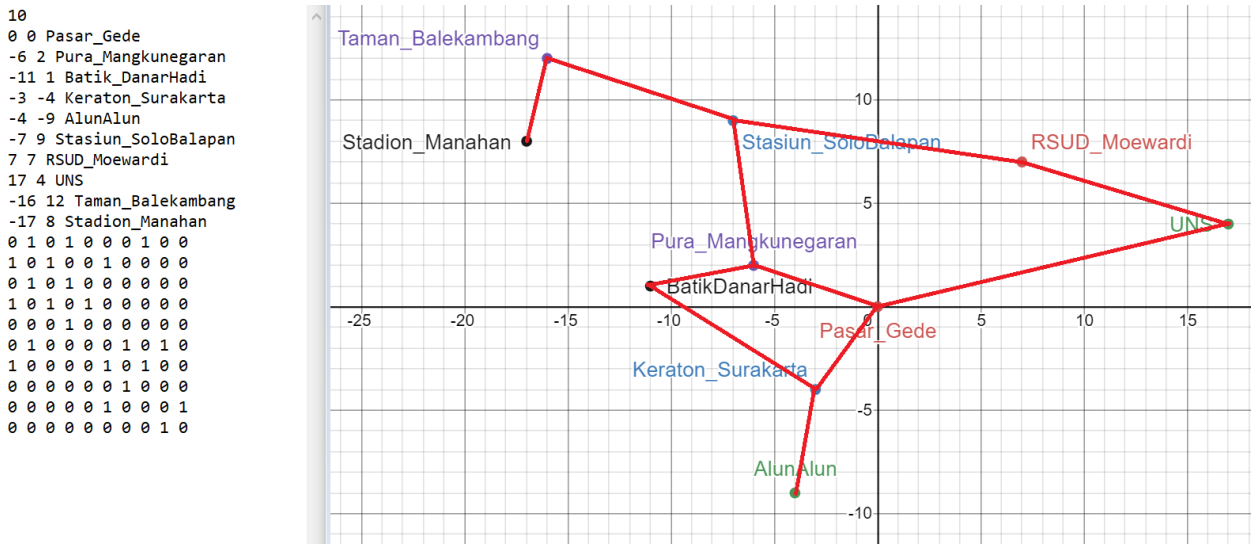
```
8
0 0 AlunAlun
4 1 Museum_AsiaAfrika
-4 -2 TheKings_ShoppingCentre
-11 2 SudirmanStreet
8 5 Rasa_Bakery
-5 7 Pasar_Baru
3 8 Braga_CityWalk
6 12 GerejaKatedral_StPetrus
0 1 0 1 0 1 0 0
1 0 0 0 1 0 1 0
0 0 0 1 0 0 0 0
1 0 1 0 0 1 0 0
0 1 0 0 0 0 1 1
1 0 0 1 0 0 1 0
0 1 0 0 1 1 0 0
0 0 0 0 1 0 0 0
```



Gambar 2: File *input2.txt* merepresentasikan peta jalan sekitar alun-alun Bandung.

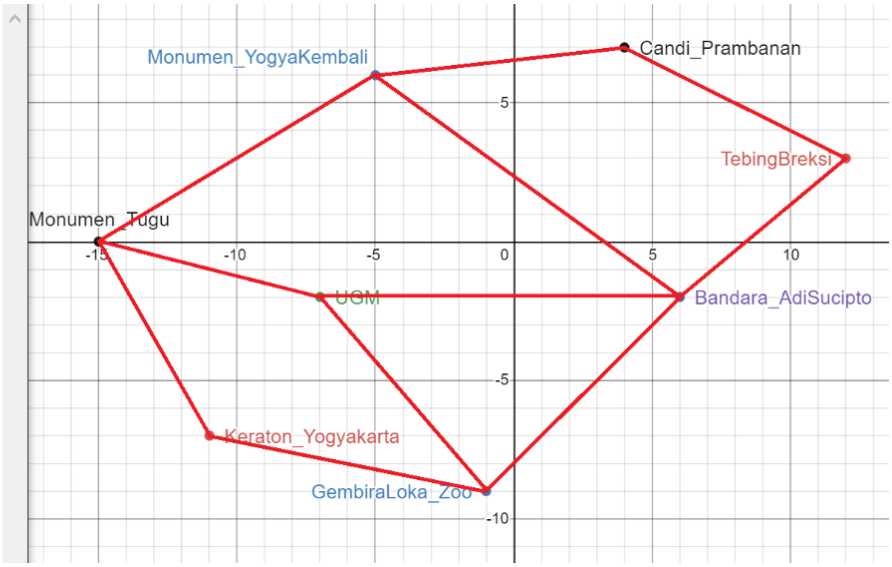


Gambar 3: File *input3.txt* merepresentasikan peta jalan sekitar Buahbatu.



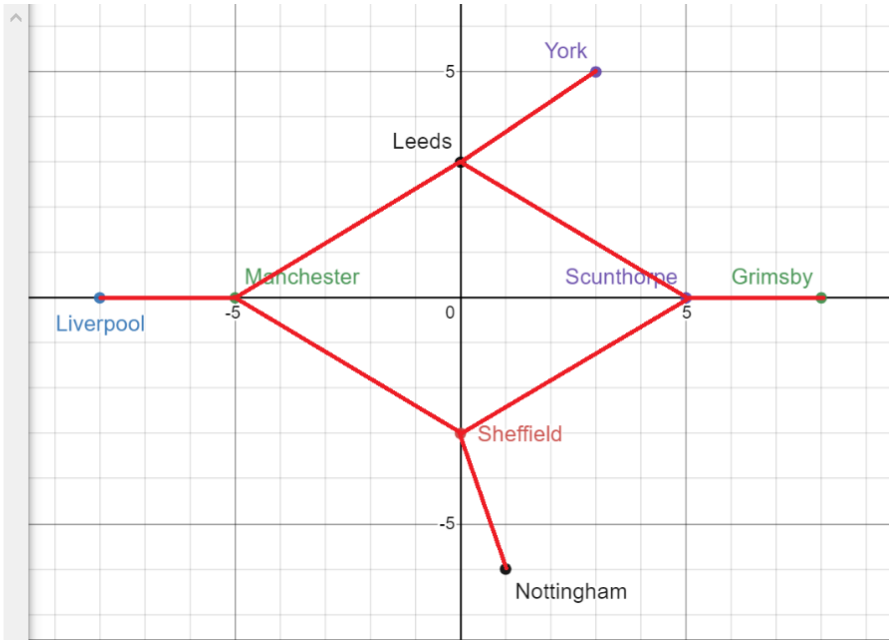
Gambar 4: File *input4.txt* merepresentasikan peta jalan sekitar Kota Surakarta.

```
8
-15 0 Monumen_Tugu
-11 -7 Keraton_Yogyakarta
-7 -2 UGM
-1 -9 GembiraLoka_Zoo
-5 6 Monumen_YogyaKembali
6 -2 Bandara_AdiSucipto
4 7 Candi_Prambanan
12 3 TebingBreksi
0 1 1 0 1 0 0 0
1 0 0 1 0 0 0 0
1 0 0 1 0 1 0 0
0 1 1 0 0 1 0 0
1 0 0 0 0 1 1 0
0 0 1 1 1 0 0 1
0 0 0 0 1 0 0 1
0 0 0 0 0 1 1 0
```



Gambar 5: File *input5.txt* merepresentasikan peta jalan sekitar Kota Yogyakarta.

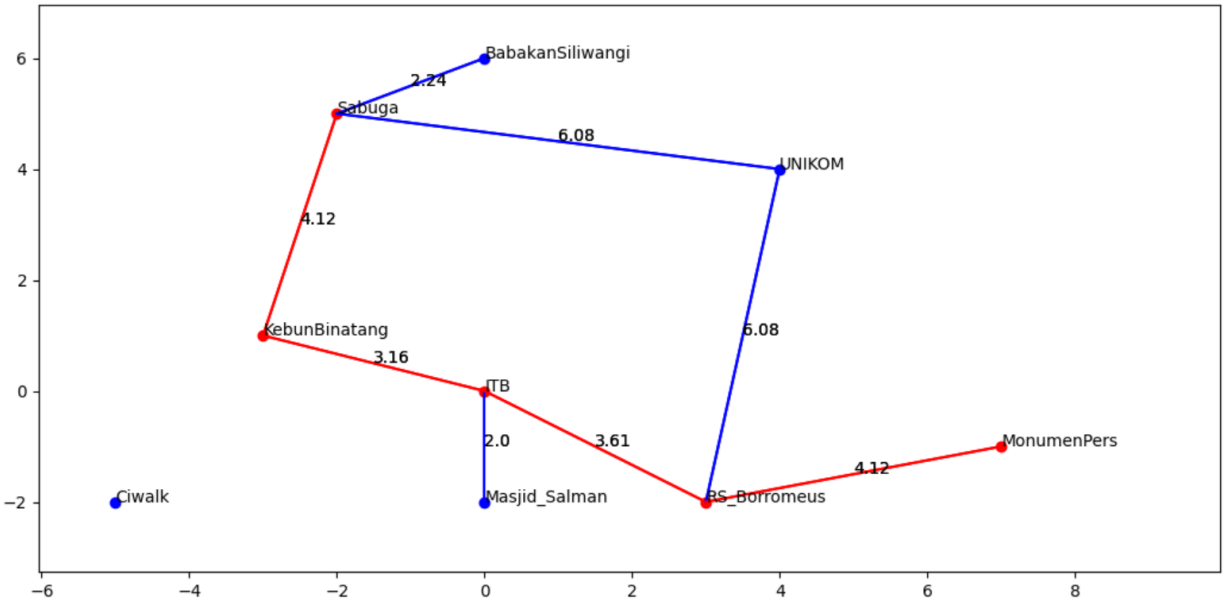
```
8
-8 0 Liverpool
-5 0 Manchester
0 3 Leeds
3 5 York
0 -3 Sheffield
1 -6 Nottingham
5 0 Scunthorpe
8 0 Grimsby
0 1 0 0 0 0 0 0
1 0 1 0 1 0 0 0
0 1 0 1 0 0 1 0
0 0 1 0 0 0 0 0
0 1 0 0 0 1 1 0
0 0 0 0 1 0 0 0
0 0 1 0 1 0 0 1
0 0 0 0 0 0 1 0
```



Gambar 6: File *input6.txt* merepresentasikan peta jalan antarkota di Inggris.

Hasil Tangkapan Layar Peta

```
Enter file name: input1.txt
Berikut list lokasi yang terdapat dalam peta:
ITB
RS_Borromeus
Masjid_Salman
Ciwalk
KebunBinatang
UNIKOM
MonumenPers
BabakanSiliwangi
Sabuga


Silakan masukkan lokasi asal dan tujuan (Format: <lokasi_asal><spasi><lokasi_tujuan>, contoh: ITB Sabuga) :
Sabuga MonumenPers
Jarak dari Sabuga menuju MonumenPers adalah 15.01
```



Gambar 7: Lintasan terpendek dari MonumenPers ke Sabuga.

Pada peta di gambar 7, terdapat dua kemungkinan lintasan dari Sabuga ke MonumenPers: lintasan 1 adalah Sabuga–KebunBinatang–ITB–RS_Borromeus–MonumenPers dan lintasan 2 adalah Sabuga–UNIKOM–RS_Borromeus–MonumenPers. Mari kita buktikan menggunakan rumus jarak Euclidean bahwa lintasan 1 adalah lintasan terpendek.

Lintasan 1

Sabuga – KebunBinatang $= \sqrt{(-3-(-2))^2 + (1-5)^2} = 4{,}1231$

KebunBinatang – ITB $= \sqrt{(0-(-3))^2 + (0-1)^2} = 3{,}1623$

ITB – RS_Borromeus $= \sqrt{(3-0)^2 + (-2-0)^2} = 3{,}6056$

$$\text{RS\_Borromeus} - \text{MonumenPers} = \sqrt{(7-3)^2 + (-1-(-2))^2} = 4{,}1231$$

$$+$$

$$\mathbf{15{,}0141}$$

Lintasan 2

$$\text{Sabuga} - \text{UNIKOM} = \sqrt{(4-(-2))^2 + (4-5)^2} = 6{,}0828$$

$$\text{UNIKOM} - \text{RS\_Borromeus} = \sqrt{(3-4)^2 + (-2-4)^2} = 6{,}0828$$

$$\text{RS\_Borromeus} - \text{MonumenPers} = \sqrt{(7-3)^2 + (-1-(-2))^2} = 4{,}1231$$

$$+$$

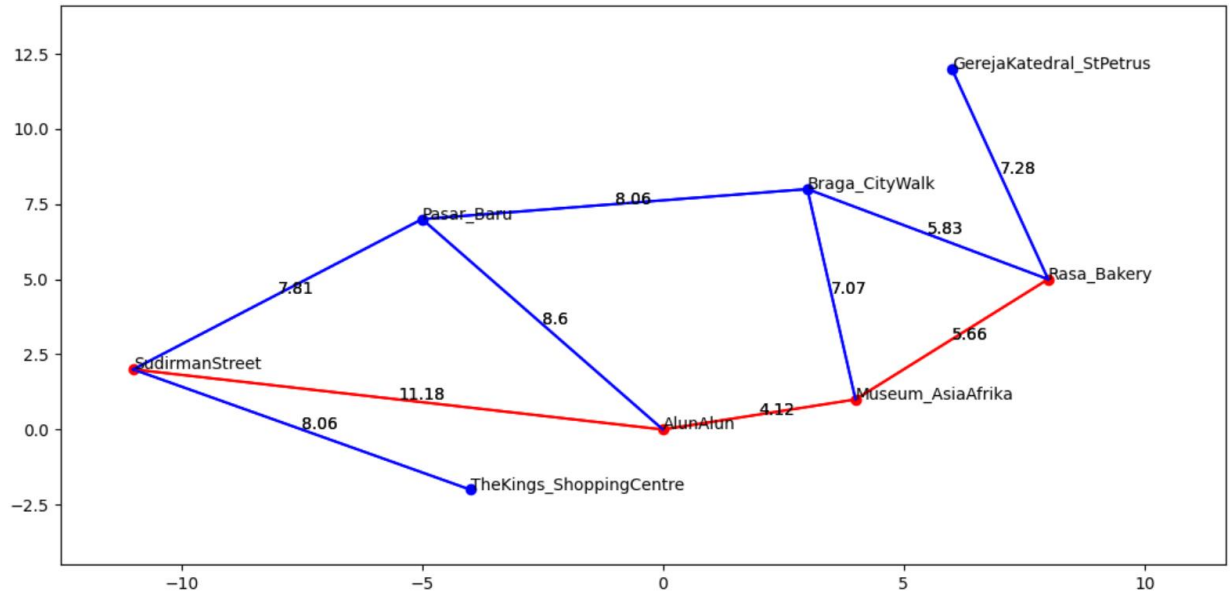
$$\mathbf{16{,}2887}$$

Jadi, terbukti bahwa program benar memilih lintasan 1 sebagai lintasan terpendek dengan jarak total 15,0141 satuan.

```
Enter file name: input1.txt
Silakan masukkan lokasi asal dan tujuan (Format: <lokasi_asal><spasi><lokasi_tujuan>, contoh: ITB Sabuga) :
Ciwalk ITB
Tidak terdapat jalan yang menghubungkan kedua lokasi.
```

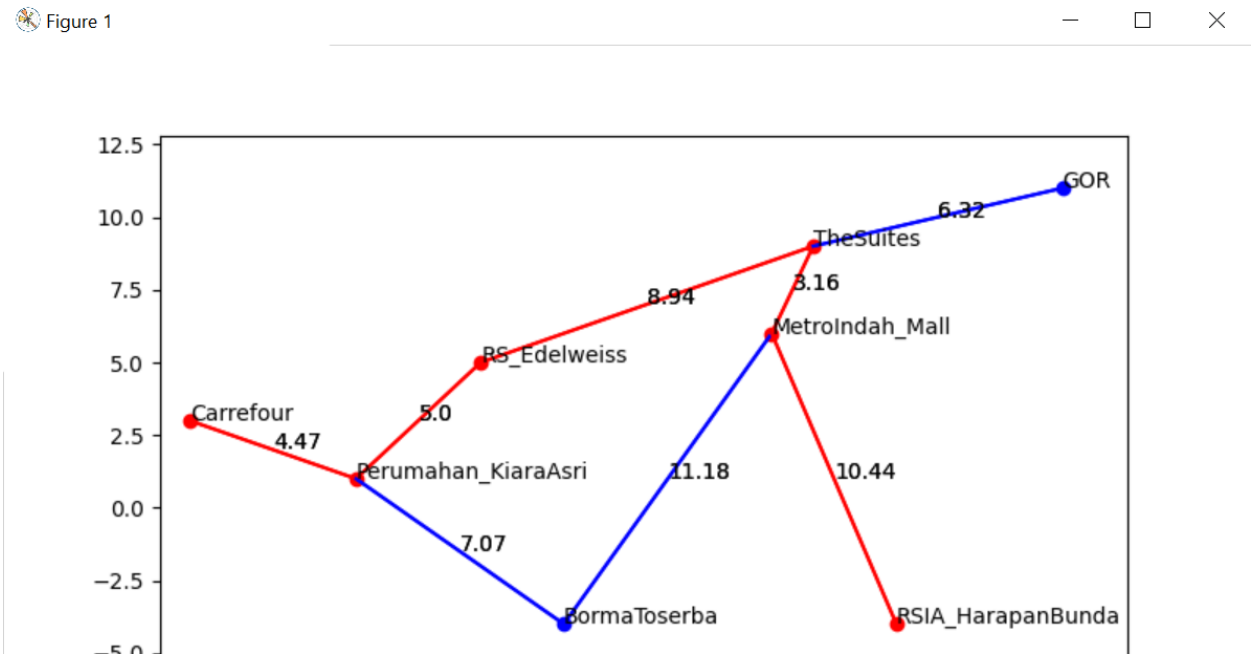
Gambar 8: Keluaran program apabila tidak ada jalur yang menghubungkan lokasi asal dan lokasi tujuan.
Lihat Gambar 7, tidak ada jalur yang menghubungkan Ciwalk dengan ITB.

```
Enter file name: input2.txt
Berikut list lokasi yang terdapat dalam peta:
AlunAlun
Museum_AsiaAfrika
TheKings_ShoppingCentre
SudirmanStreet
Rasa_Bakery
Pasar_Baru
Braga_CityWalk
GerejaKatedral_StPetrus


Silakan masukkan lokasi asal dan tujuan (Format: <lokasi_asal><spasi><lokasi_tujuan>, contoh: ITB Sabuga) :
SudirmanStreet Rasa_Bakery
Jarak dari SudirmanStreet menuju Rasa_Bakery adalah 20.96
```



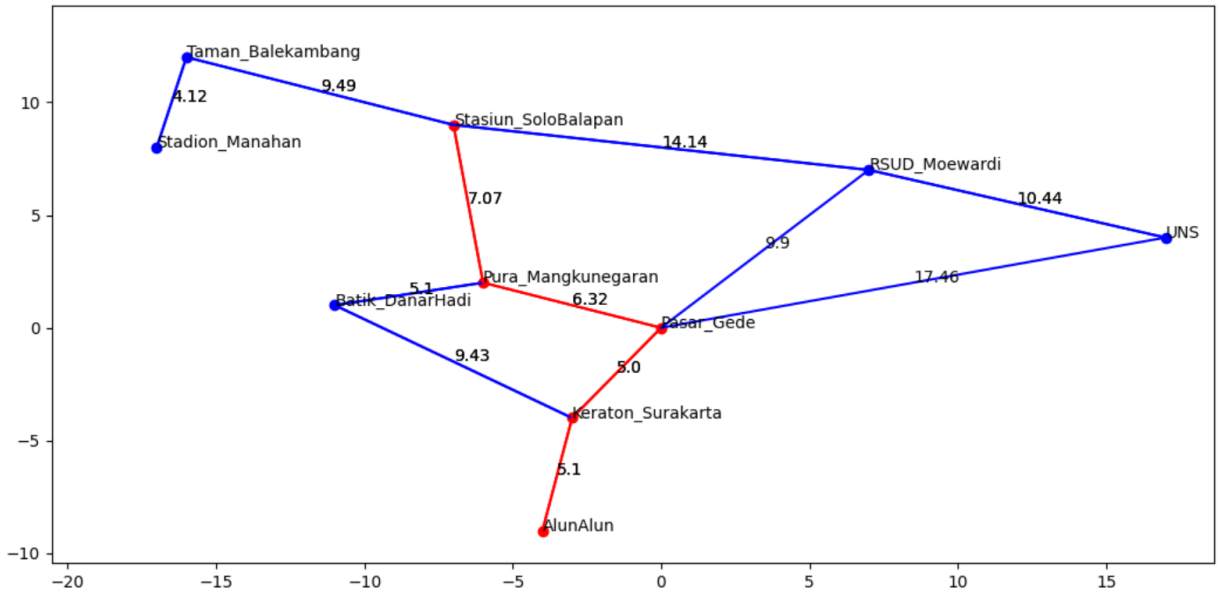
Gambar 9: Lintasan terpendek dari SudirmanStreet ke Rasa_Bakery.

```
Enter file name: input3.txt
Berikut list lokasi yang terdapat dalam peta:
MetroIndah_Mall
RS_Edelweiss
Carrefour
BormaToserba
RSIA_HarapanBunda
TheSuites
GOR
Perumahan_KiaraAsri


Silakan masukkan lokasi asal dan tujuan (Format: <lokasi_asal><spasi><lokasi_tujuan>, contoh: ITB Sabuga) :
Carrefour RSIA_HarapanBunda
```



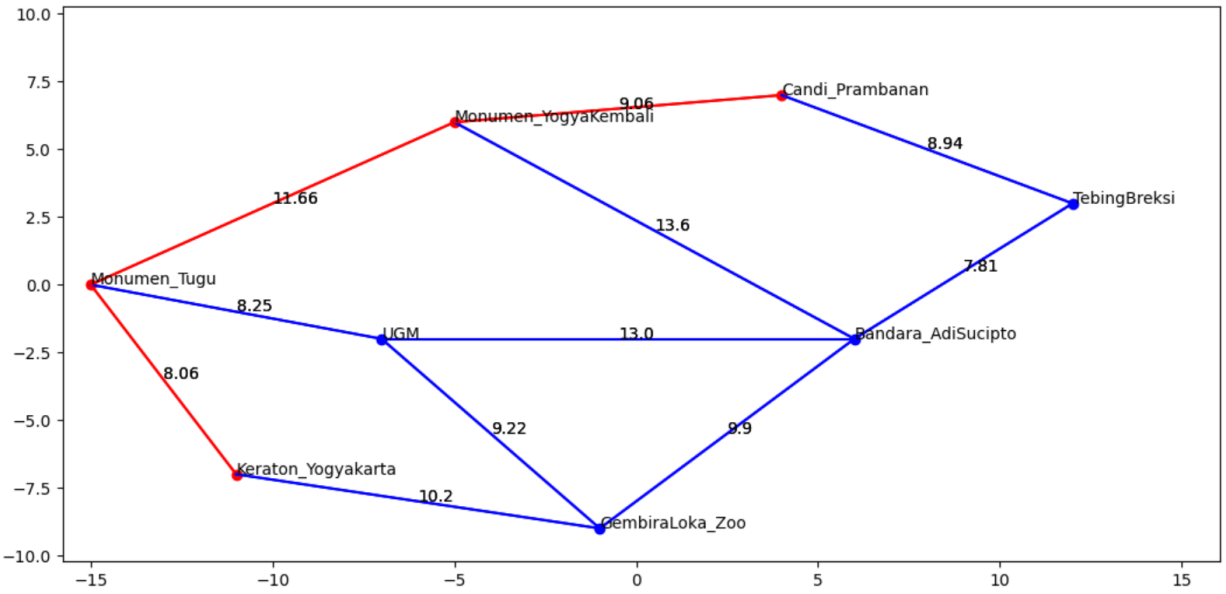
Gambar 10: Lintasan terpendek dari Carrefour ke RSIA_HarapanBunda.

```
Enter file name: input4.txt
Berikut list lokasi yang terdapat dalam peta:
Pasar_Gede
Pura_Mangkunegaran
Batik_DanarHadi
Keraton_Surakarta
AlunAlun
Stasiun_SoloBalapan
RSUD_Moewardi
UNS
Taman_Balekambang
Stadion_Manahan


Silakan masukkan lokasi asal dan tujuan (Format: <lokasi_asal><spasi><lokasi_tujuan>, contoh: ITB Sabuga) :
Stasiun_SoloBalapan AlunAlun
```



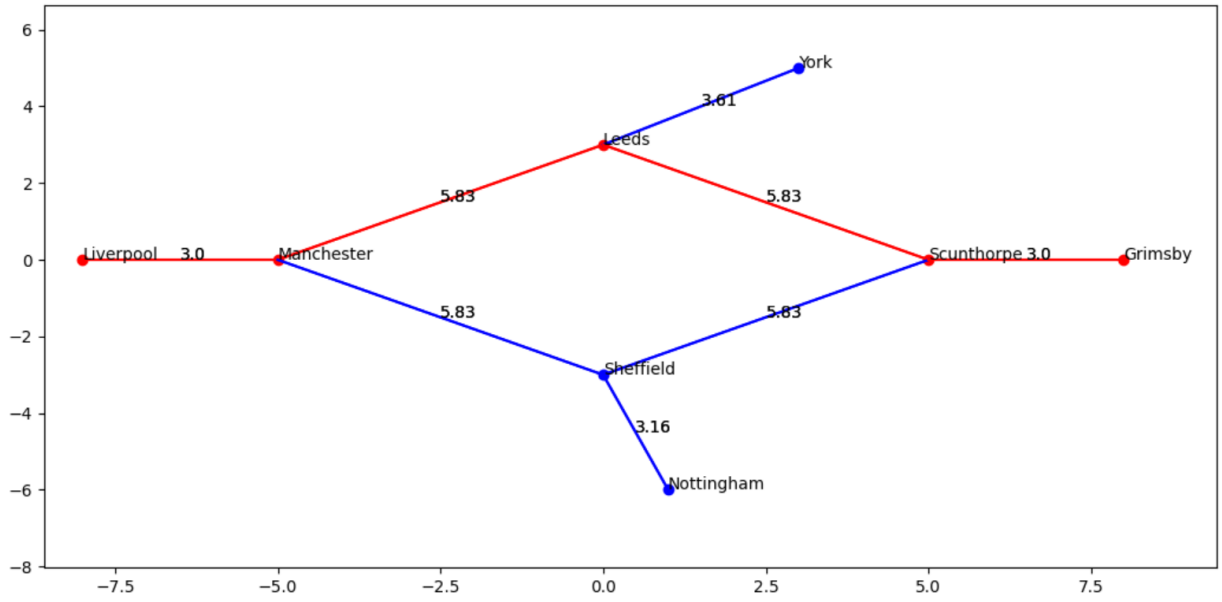
Gambar 11: Lintasan terpendek dari Stasiun_SoloBalapan ke AlunAlun

```
Enter file name: input5.txt
Berikut list lokasi yang terdapat dalam peta:
Monumen_Tugu
Keraton_Yogyakarta
UGM
GembiraLoka_Zoo
Monumen_YogyaKembali
Bandara_AdiSucipto
Candi_Prambanan
TebingBreksi


Silakan masukkan lokasi asal dan tujuan (Format: <lokasi_asal><spasi><lokasi_tujuan>, contoh: ITB Sabuga) :
Keraton_Yogyakarta Candi_Prambanan
```



Gambar 12: Lintasan terpendek dari Keraton_Yogyakarta ke Candi_Prambanan.

```
Enter file name: input6.txt
Berikut list lokasi yang terdapat dalam peta:
Liverpool
Manchester
Leeds
York
Sheffield
Nottingham
Scunthorpe
Grimsby


Silakan masukkan lokasi asal dan tujuan (Format: <lokasi_asal><spasi><lokasi_tujuan>, contoh: ITB Sabuga) :
Liverpool Grimsby
```



Gambar 13: Lintasan terpendek dari Liverpool ke Grimsby.

Pranal repo GitHub: https://github.com/andrcyes/Tucil3_13519036.git

1.	Program dapat menerima input graf.	✔
2.	Program dapat menghitung lintasan terpendek.	✔
3.	Program dapat menampilkan lintasan terpendek serta jaraknya.	✔
4.	Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta	—