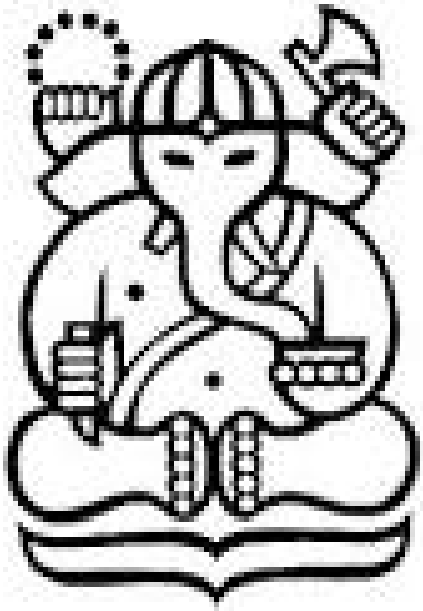
LAPORAN TUGAS KECIL 3

IF2211 - STRATEGI ALGORITMA

Pengaplikasian Algoritma UCS dan A* untuk Menentukan Lintasan Terpendek



Anggota Kelompok :

(13521097) Shidqi Indy Izhari

(13521159) Sulthan Dzaky Alfaro

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

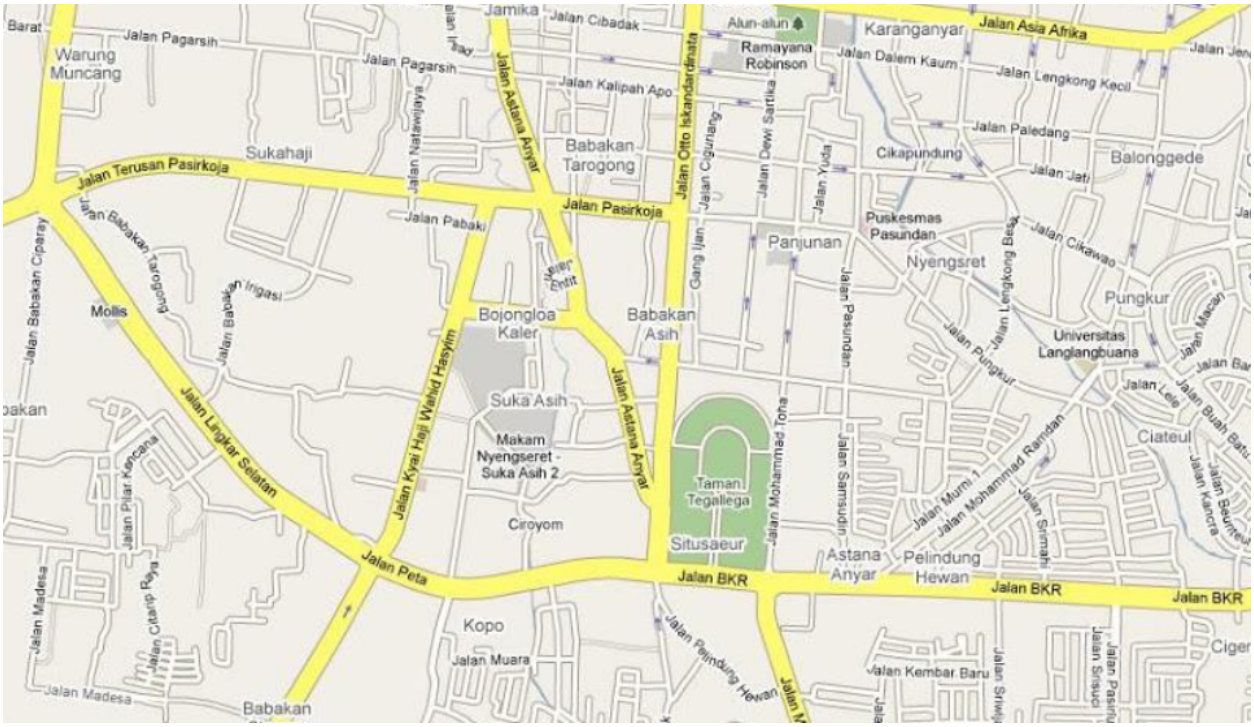
2022

Daftar Isi

Bab 1:
Deskripsi Tugas

1.1 Latar Belakang

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 jarakEuclidean (berdasarkan koordinat) atau dapat menggunakan ruler di Google Map, atau cara lainnya yang disediakan oleh Google Map.



Gambar 1. Contoh gambar peta

1.2 Deskripsi Tugas

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.

Bab 2:
Landasan Teori

2.1 Graf

Graf adalah salah satu struktur data yang digunakan untuk merepresentasikan kumpulan objek yang saling berkaitan atau terhubung. Objek-objek ini disebut simpul atau node, sedangkan hubungan antar simpul disebut sisi atau edge. Graf dapat digunakan untuk merepresentasikan berbagai macam informasi, seperti jaringan sosial, jaringan transportasi, dan jaringan komputer. Setiap graf memiliki properti khusus seperti jumlah simpul dan sisi, serta derajat simpul (yaitu jumlah sisi yang terhubung ke suatu simpul). Graf dapat digunakan untuk melakukan berbagai operasi, seperti mencari rute terpendek antara dua simpul, menemukan sirkuit tertentu dalam graf, atau memeriksa apakah suatu graf memiliki properti tertentu seperti sifat-sifat aliran. Pada kasus ini, graf akan digambarkan sebagai peta yang akan dicari rutenya.

Terdapat beberapa jenis graf, seperti graf tak-berarah, graf berarah, dan graf campuran. Graf tak-berarah adalah graf yang tidak memiliki arah pada sisinya, sehingga sisi-sisi dapat dihubungkan ke simpul mana pun. Sedangkan graf berarah memiliki arah pada setiap sisinya, sehingga sisi hanya menghubungkan simpul pada arah tertentu. Graf campuran adalah gabungan dari kedua jenis graf tersebut, di mana sisi-sisi dapat memiliki arah atau tidak. Graf juga dapat direpresentasikan dalam bentuk matriks atau daftar ketetanggaan, tergantung pada kebutuhan aplikasi.

2.2 Traversal Graf

Traversal graf adalah proses mengunjungi setiap simpul atau titik dalam sebuah graf, dengan cara yang teratur dan sistematis. Proses traversal graf ini biasanya dilakukan untuk tujuan tertentu, seperti mencari rute terpendek antara dua simpul dalam graf, menemukan sirkuit atau lintasan tertentu dalam graf, atau memeriksa apakah suatu graf memiliki properti tertentu seperti sifat-sifat aliran.

Ada beberapa algoritma yang dapat digunakan untuk melakukan traversal graf, seperti algoritma Breadth-First Search (BFS), Depth-First Search (DFS), Uniform-cost Search (UCS), dan A* (A Star) . Dalam program ini, algoritma akan dikhususkan pada UCS dan A* untuk menjalankan mengunjungi setiap simpul pada graf agar mendapatkan jarak minimal antara dua simpul.

2.3 Penjelasan Algoritma UCS

Algoritma Uniform Cost Search (UCS) adalah salah satu metode pencarian jalur terpendek dalam graf berbobot yang menghitung biaya dari suatu rute dengan menjumlahkan bobot dari setiap edge yang dilewati. Algoritma ini bekerja dengan cara memperluas node yang memiliki biaya path terendah terlebih dahulu dan membandingkan biaya path dari setiap node yang dicapai dengan biaya path dari node sebelumnya. UCS akan terus memperluas node yang memiliki biaya path terendah hingga menemukan solusi terpendek. Dasar dari UCS adalah struktur data priority queue.

2.4 Penjelasan Algoritma A*

Algoritma A* merupakan algoritma pencarian untuk menemukan rute paling optimal dan terpendek. Algoritma ini biasa digunakan dalam pencarian rute pada peta untuk mencari rute terpendek. Algoritma A* mencari jalur yang lebih pendek terlebih dahulu, sehingga menjadikannya algoritma yang optimal dan lengkap. Algoritma ini menggunakan graph berbobot dalam pengolahan datanya. Ini berarti, algoritma ini mengambil jalur yang paling minimum dari segi jarak maupun waktu. Kelemahan algoritma ini adalah kompleksitas waktu yang cukup tinggi, dengan kompleksitas waktu O(b^d) dengan b adalah faktor percabangan dan d adalah jumlah node pada jalur yang dihasilkan. Cara kerja algoritma ini, yaitu hampir sama dengan UCS, algoritma akan mencari jarak yang paling minimum antar node yang akan dikunjungi, perbedaannya A* mempertimbangkan jarak heuristik yaitu jarak “euclidean” antara node dengan node tujuan.

Bab 3:

Kode Program

Program ditulis dalam bahasa pemrograman Python dan dibantu oleh beberapa library seperti tkinter, networkx, matplotlib dan os. Berikut adalah kode yang telah dibuat per-file:

3.1 UCS.py

```
import heapq

def ucs(graph, start, goal):
    explored = set()
    queue = [(0, start, [])]
    while queue:
        (cost, node, path) = heapq.heappop(queue)
        if node not in explored:
            explored.add(node)
            path = path + [node]
            if node == goal:
                return path, cost
            for neighbor in graph[node]:
                n_cost = graph[node][neighbor]
                # menggunakan prio queue sebagai dasar
                heapq.heappush(queue, (cost + n_cost, neighbor, path))
    return "No path found"

# membaca file graf
def read_graph(filename):
    with open(filename) as f:
        # membaca daftar simpul
        nodes = f.readline().strip().split()
        # membaca matriks ketetanggaan
        adjacency_matrix = []
        for line in f:
                                    adjacency_matrix.append(list(map(float,
line.strip().split())))
        # mmebuat graf
        graph = {}
        for i in range(len(nodes)):
            node = nodes[i]
            graph[node] = {}
```

```
            for j in range(len(nodes)):
                if adjacency_matrix[i][j] != 0:
                    neighbor = nodes[j]
                    cost = adjacency_matrix[i][j]
                    graph[node][neighbor] = cost
        return graph

# Menerima input simpul asal dan tujuan
def get_start_goal():
    start = input("Masukkan simpul asal: ")
    goal = input("Masukkan simpul tujuan: ")
    return start, goal

# Memanggil fungsi ucs dan menampilkan hasil, fungsi utamanya
def main():
    filename = input("Masukkan nama file graf: ")
    graph = read_graph(filename)
    start, goal = get_start_goal()
    print(graph)
    path, cost = ucs(graph, start, goal)
    if path == "No path found":
        print("Tidak ditemukan jalur antara", start, "dan", goal)
    else:
        print("Jalur terpendek:", "->".join(path))
        print("Biaya jalur terpendek:", cost)

if __name__ == "__main__":
    main()
```

3.2 Astar.py

```
import networkx as nx
import matplotlib.pyplot as plt
def addTitik(graph, titik):
    if titik not in graph:
        graph[titik] = []
def addEdge(graph, titik1, titik2, berat):
    simpan = [titik2, berat]
```

```
    graph[titik1].append(simpan)

def matrixToGraph(matrix):
    graph = {}
    for i in range(len(matrix)):
        addTitik(graph, i)
        for j in range(len(matrix[i])):
            if matrix[i][j] != 0:
                addEdge(graph, i, j, matrix[i][j])
    return graph

def printGraph(graph, namakota):
    for titik in graph:
        for j in range(len(graph[titik])):
                print(namakota[titik], "-", namakota[graph[titik][j][0]],
"berat:", graph[titik][j][1])


def printMatrix(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            print(matrix[i][j], end=" ")
        print()
def cekMatrix(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            # if matrix[i][j] != matrix[j][i]:
            #     return False
            # if(matrix[i][j] < 0):
            #     return False
            if i==j and matrix[i][j] != 0:
                return False
    return True

def getIDXName(list,nama):
    for i in range(len(list)):
        if(list[i]==nama):
            return i
            break
    return -1
```

```
def neighbour(node,graph):
    return graph[node]
def Astar(graph, awal, akhir, listkoor):
    jrkheur = jarakheuristik(listkoor,akhir)
    blm_semua = set([awal])
    udah_kunjungi = set([])
    parent = {}
    parent[awal] = awal
    simpan_length = {}
    simpan_length[awal] = 0

    while(len(blm_semua)>0):
        test = None
        for i in blm_semua:
                        if(test==None or simpan_length[i] + jrkheur[i] <
simpan_length[test] + jrkheur[test]):
                test = i
        if(test==None):
            print("Tidak menemukan rute")
            return None
        if(test==akhir):
            rute = []
            while(parent[test]!=test):
                rute.append(test)
                test = parent[test]
            rute.append(awal)
            rute.reverse()
            return rute
        for (tetangga,berat) in neighbour(test,graph):
                        if(tetangga not in blm_semua and tetangga not in
udah_kunjungi):
                blm_semua.add(tetangga)
                parent[tetangga] = test
                simpan_length[tetangga] = simpan_length[test] + berat
            else:
                if(simpan_length[tetangga]>simpan_length[test]+berat):
                    simpan_length[tetangga] = simpan_length[test] + berat
                    parent[tetangga] = test
                    if(tetangga in udah_kunjungi):
                        udah_kunjungi.remove(tetangga)
```

```
                        blm_semua.add(tetangga)
        blm_semua.remove(test)
        udah_kunjungi.add(test)
    print("Tidak menemukan rute")
    return None
def printRute(list,nama):
    if(list!=None):
        ngeprin = "Rute : "
        for i in range(len(list)):
            if(i==len(list)-1):
                ngeprin+=nama[list[i]]
            else:
                ngeprin += nama[list[i]]+"->"
    return ngeprin

def read_file(filename):
    with open(filename) as f:
        nama = f.readline().strip().split()
        banyak = len(nama)
        matriks = []
        koordinat = []
        for i in range(banyak):
            line = f.readline().strip().split()
            matriks.append(line)
        for j in range(banyak):
            koor = f.readline().strip().split()
            koordinat.append(koor)
        matstringtoint(matriks)
        koorstrtoint(koordinat)
        return nama,matriks,koordinat

def ecluidian(x1,x2,y1,y2):
    return (((x1-x2)**2)+((y1-y2)**2))**(1/2)
def jarakheuristik(listkoor,tujuan):
    listjawaban = []
    for i in range(len(listkoor)):
                                                        hasil        =
ecluidian(int(listkoor[i][0]),int(listkoor[tujuan][0]),int(listkoor[i][1])
,int(listkoor[tujuan][1]))
        listjawaban.append(hasil)
```

```
    return listjawaban
def matstringtoint(matriks):
    for i in range(len(matriks)):
        for j in range(len(matriks[0])):
            matriks[i][j] = float(matriks[i][j])
def koorstrtoint(matriks):
    for i in range(len(matriks)):
        for j in range(len(matriks[0])):
            matriks[i][j] = float(matriks[i][j])
def jarak(graph,list):
    hasil = 0
    for i in range(len(list)-1):
        for j in range(len(graph[list[i]])):
            if(graph[list[i]][j][0]==list[i+1]):
                hasil+=graph[list[i]][j][1]
    return hasil

def visualgrafkoor(nama,matriks,koor):
    graph = nx.Graph()
    for i in range(len(nama)):
        graph.add_node(nama[i],pos=(koor[i][0],koor[i][1]))
    for j in range(len(nama)):
        for k in range(len(nama)):
            if(matriks[j][k]!=0):
                graph.add_edge(nama[j],nama[k],weight = int(matriks[j][k])
)
    return graph
def draw_graph_koor(graph):
    pos=nx.get_node_attributes(graph,'pos')
    nx.draw(graph,pos,with_labels=True, font_weight='bold')
    labels = nx.get_edge_attributes(graph, 'weight')
    nx.draw_networkx_edge_labels(graph, pos, edge_labels=labels)
def draw_graph_koor_color(graph,hasil,nama):
    pos=nx.get_node_attributes(graph,'pos')
    nx.draw(graph,pos,with_labels=True, font_weight='bold')
    labels = nx.get_edge_attributes(graph, 'weight')
    nx.draw_networkx_edge_labels(graph, pos, edge_labels=labels)
    listedge = []
    for i in range(len(hasil)-1):
        edge = (nama[hasil[i]],nama[hasil[i+1]])
```

```
        listedge.append(edge)
                        nx.draw_networkx_edges(graph,pos,edgelist      =
listedge,edge_color="tab:blue")

# namafile = input("Masukan nama file:")
# n,m,k = read_file(namafile)
# print(m)
# # graph = visualgrafkoor(n,m,k)
# # #draw_graph_koor(graph)
# mtog = matrixToGraph(m)
# # hasilastar = Astar(mtog,0,3,k)
# # draw_graph_koor_color(graph,hasilastar,n)
# # print(mtog)
# # print(jarak(mtog,hasilastar))
# # nyoba = printRute(hasilastar,n)
# # print(nyoba)
# print(mtog)
# jrk = jarak(mtog,[0,3,4,3])
# print(jrk)
```

3.3 GUI.py

```
import tkinter as tk
from tkinter import ttk
from tkinter import filedialog
from matplotlib.figure import Figure
import matplotlib.pyplot as plt
import os
from matplotlib.backends.backend_tkagg import (
    FigureCanvasTkAgg,
    NavigationToolbar2Tk
)
import Astar
import UCS

window = tk.Tk()
window.title("Find Route")
```

```
imageframe = tk.LabelFrame(window,text="Map")
imageframe.grid(column=2,row=1,rowspan=10,columnspan=2)
Labelimage = tk.Label(imageframe,width=100,height=30)
Labelimage.pack()

labelinput = tk.Label(window,text="Input File")
labelinput.grid(column=0,row=2)
butcari = tk.Button(window,text="Select",command=lambda:cari())
butcari.grid(column=0,row=3)
cekinput = tk.Label(window,text="Belum ada file terpilih")
cekinput.grid(column=0,row=4)

pilihmetod = tk.Label(window,text="Pilih Algoritma")
pilihmetod.grid(column=0,row=10)

pilihtitik = tk.Label(window,text="Pilih titik")
pilihtitik.grid(column=0,row=6)

ucs = tk.Button(window,text="UCS",command=lambda:UCSalgo())
ucs.grid(column=0,row=11)
astar = tk.Button(window,text="A*",command=lambda:AstarAlgo())
astar.grid(column=0,row=12)

filedirect = ''
namamatrks = ["kosong"]
matrksglob = []
koorglob = []

def cari():
    global filedirect
    ftypes = [('Text','*.txt')]
    filedirect = filedialog.askopenfilename(filetypes=ftypes)
    head, tail = os.path.split(filedirect)
    if(filedirect==''):
        cekinput.config(text='Belum ada file')
    else:
        cekinput.config(text=tail)
        nama,matriks,koor = Astar.read_file(filedirect)
```

```
        if(Astar.cekMatrix(matriks)):
            graf = Astar.visualgrafkoor(nama,matriks,koor)
            global matrksglob
            global namamatrks
            global koorglob
            namamatrks = nama
            matrksglob = matriks
            koorglob = koor
            #-------visualisasi-----------
            f = plt.figure(figsize=(6.5, 4.45), dpi=100)
            ax = f.add_subplot(111)
            Astar.draw_graph_koor(graf)
            canvas = FigureCanvasTkAgg(f, master=window)
            canvas.draw()
            canvas.get_tk_widget().grid(row=1, column=2,rowspan=10)
            #-----------------------------------------
            global opsiawal,opsiakhir,click2,click
            opsiawal.destroy()
            opsiawal = tk.OptionMenu(window,click,*nama)
            opsiawal.grid(column=0,row=7)
            opsiakhir.destroy()
            opsiakhir = tk.OptionMenu(window,click2,*nama)
            opsiakhir.grid(column=0,row=8)
        else:
            HasilResult.config(text="Input masih ada yang salah")

def AstarAlgo():
    if(filedirect!=''):
        if(namamatrks[0]!="kosong" and matrksglob!=[] and
koorglob!=[]):
            graph = Astar.matrixToGraph(matrksglob)
            hasil =
Astar.Astar(graph,Astar.getIDXName(namamatrks,click.get()),Astar.getI
DXName(namamatrks,click2.get()),koorglob)
            if(hasil!=None):
                print(graph)
                print(hasil)
```

```
                rute = Astar.printRute(hasil,namamatrks)
                jrk = Astar.jarak(graph,hasil)
                print(jrk)
                HasilResult.config(text=rute)
                hasiljarak.config(text=jrk)
                graphvis =
Astar.visualgrafkoor(namamatrks,matrksglob,koorglob)
                f = plt.figure(figsize=(6.5, 4.45), dpi=100)
                ax = f.add_subplot(111)

Astar.draw_graph_koor_color(graphvis,hasil,namamatrks)
                canvas = FigureCanvasTkAgg(f, master=window)
                canvas.draw()
                canvas.get_tk_widget().grid(row=1,
column=2,rowspan=10)
            else:
                HasilResult.config(text="Tidak menemukan rute")
                hasiljarak.config(text="")


def UCSalgo():
    if(filedirect!=''):
        graph = UCS.read_graph(filedirect)
        a = click.get()
        b = click2.get()
        if(len(UCS.ucs(graph,a,b)) > 2):
            hasil = "No path found"
        else:
            hasil, jarak = UCS.ucs(graph,a,b)
        if(hasil=="No path found"):
            HasilResult.config(text="Tidak menemukan rute")
            hasiljarak.config(text="")
        else:
            hasiljarak.config(text=jarak)
            ha = []
            rute = "Rute : "
            for j in range(len(hasil)):
                if(j==len(hasil)-1):
```

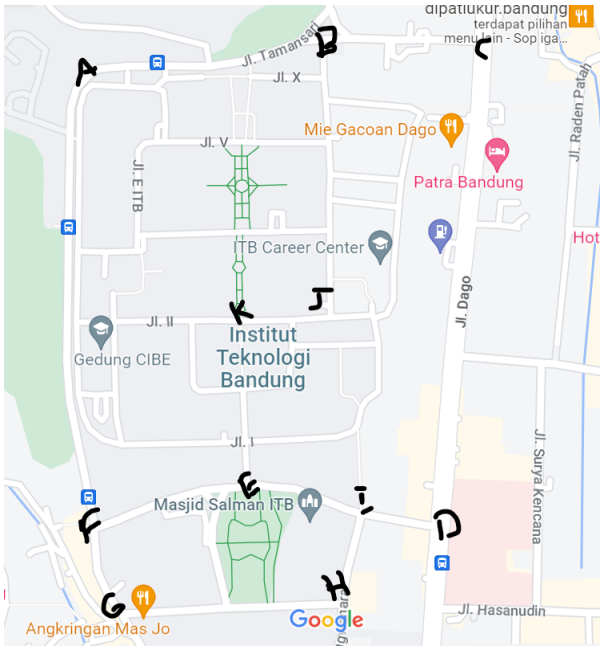
```
                    rute+=hasil[j]
                else:
                    rute+=hasil[j]+"->"
            HasilResult.config(text=rute)
            for i in range(len(hasil)):
                ha.append(Astar.getIDXName(namamatrks,hasil[i]))
            graphvis =
Astar.visualgrafkoor(namamatrks,matrksglob,koorglob)
            f = plt.figure(figsize=(6.5, 4.45), dpi=100)
            ax = f.add_subplot(111)
            Astar.draw_graph_koor_color(graphvis,ha,namamatrks)
            canvas = FigureCanvasTkAgg(f, master=window)
            canvas.draw()
            canvas.get_tk_widget().grid(row=1, column=2,rowspan=10)




click = tk.StringVar()
click.set("Pilih Titik Awal")
opsiawal = tk.OptionMenu(window,click,*namamatrks)
opsiawal.grid(column=0,row=7)

click2 = tk.StringVar()
click2.set("Pilih Titik Akhir")
opsiakhir = tk.OptionMenu(window,click2,*namamatrks)
opsiakhir.grid(column=0,row=8)

Result = tk.Label(window,text="Result:")
Result.grid(column=0,row=13)
HasilResult = tk.Label(window,text="")
HasilResult.grid(column=1,row=13,columnspan=2)

jarak = tk.Label(window,text="Jarak:")
jarak.grid(column=0,row=15)
hasiljarak = tk.Label(window,text="")
hasiljarak.grid(column=1,row=15,columnspan=2)
```

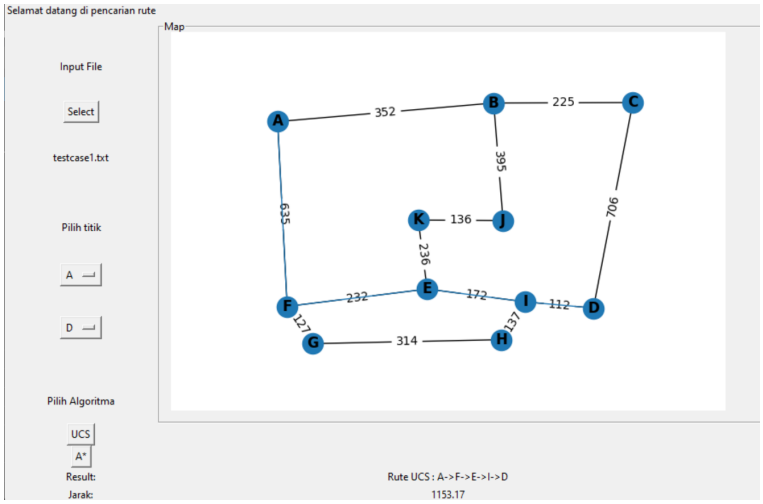
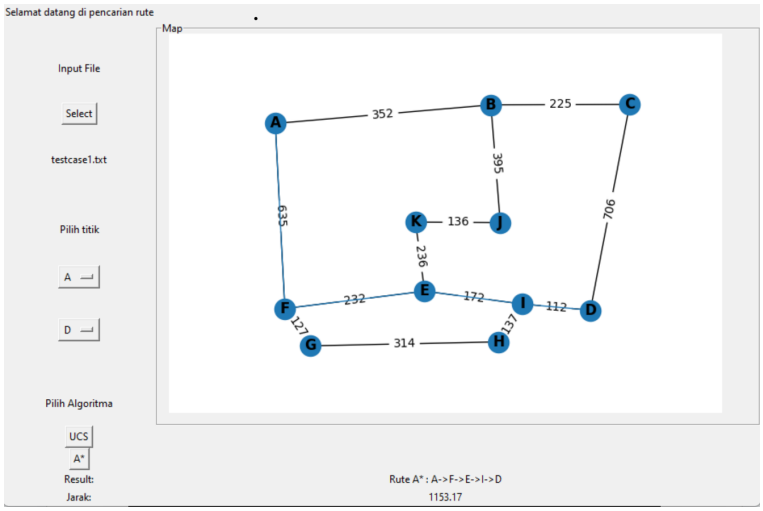
```
judul = tk.Label(window,text="Selamat datang di pencarian rute")
judul.grid(column=0,row=0)


window.mainloop()
```

Bab 4:

Test Input/Output

4.1 Peta jalan sekitar kampus ITB/Dago/Bandung Utara

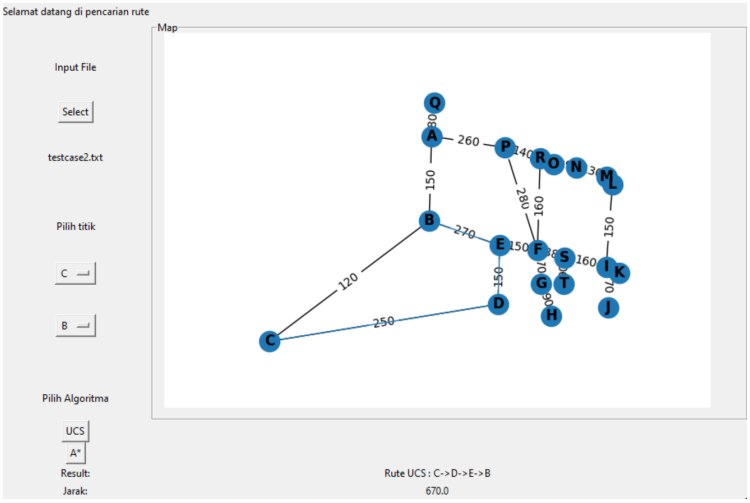
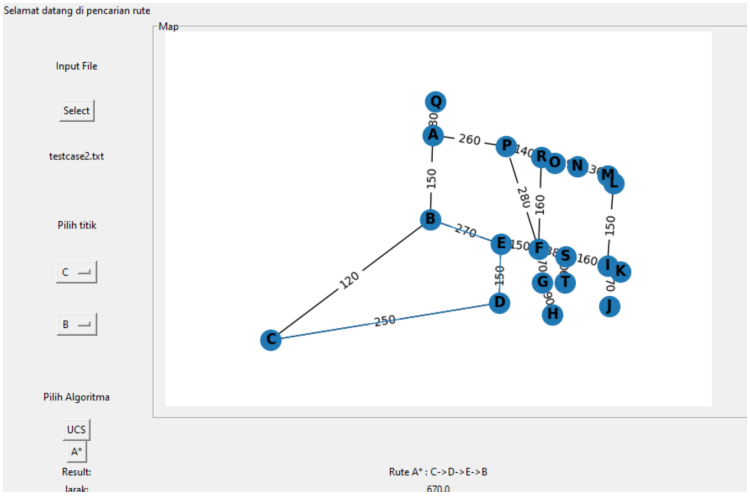
Nama/keterangan	Foto
Matriks ketetanggan	A B C D I E F J K H G 0 352.71 0 0 0 0 635.95 0 0 0 0 352.71 0 225.19 0 0 0 0 395.18 0 0 0 0 225.19 0 706.01 0 0 0 0 0 0 0 0 0 706.01 0 112.46 0 0 0 0 0 0 0 0 0 112.46 0 172.72 0 0 0 137.71 0 0 0 0 0 172.72 0 232.04 0 236.86 0 0 635.95 0 0 0 0 232.04 0 0 0 0 127.77 0 395.18 0 0 0 0 0 0 136.30 0 0 0 0 0 0 0 236.86 0 136.30 0 0 0 0 0 0 0 137.71 0 0 0 0 0 314.45 0 0 0 0 0 0 127.77 0 0 314.45 0 -6.887972114252407 107.60825419780714 -6.887393820540502 107.61145146197863 -6.887380969561116 107.61350961988661 -6.8937678634459685 107.61292712236548 -6.893575101310124 107.61191745999554 -6.8931767259809105 107.6104676883874 -6.893716460217411 107.60839658609008 -6.891046535984688 107.61159033521389 -6.891033685104295 107.61033472944611 -6.894750313922001 107.61156699283828 -6.894874350099037 107.60877146668979
Gambar peta	

Hasil pencarian dengan UCS	
Hasil pencarian dengan A*	

4.2 Peta jalan sekitar alun alun Bandung

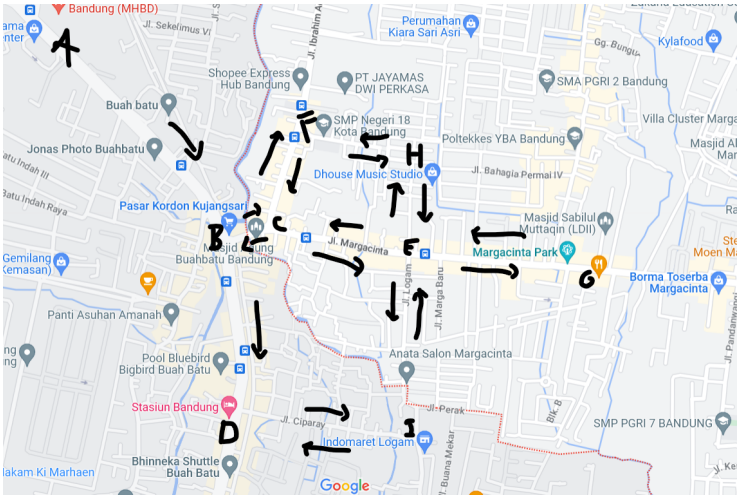
Nama/keterangan	Foto

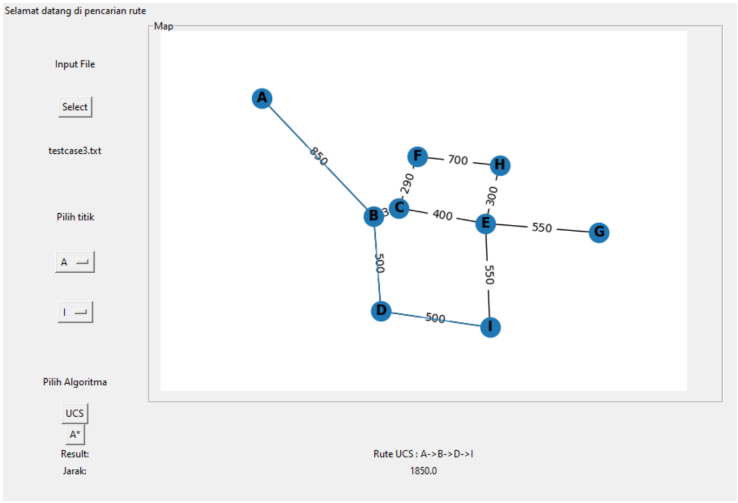
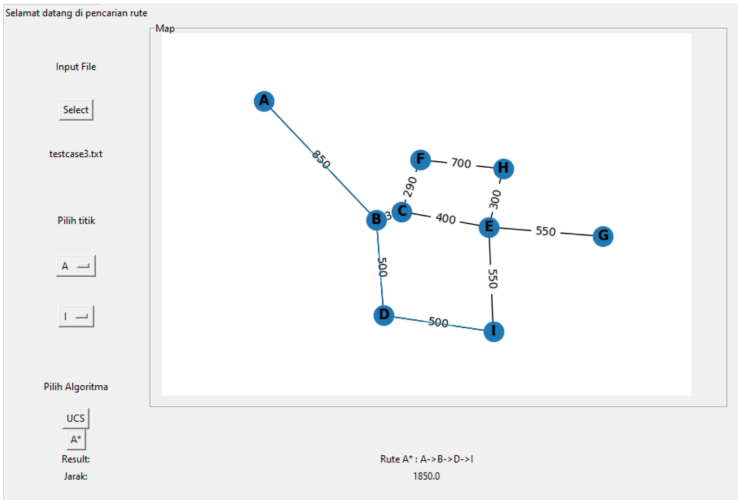
Matriks ketetanggan	A B C D E F G H I J K L M N O P Q R S T 0 150 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 120 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 250 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 150 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 270 0 0 0 150 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 70 0 0 0 0 0 0 0 0 280 0 0 88 0 0 0 0 0 0 0 0 90 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 70 50 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 70 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 50 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 150 0 0 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 130 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 89 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 48 0 0 260 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 80 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 160 0 0 0 0 0 0 0 0 0 140 0 0 0 0 0 0 0 0 0 0 0 0 160 0 0 0 0 0 0 0 0 0 90 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 90 0 -6.9207892890974, 107.60405843357132 -6.922088258421074, 107.60397972648667 -6.923944995506452, 107.5987550671152 -6.9233726194073135, 107.6062285344252 -6.922467317208778, 107.60628217860406 -6.92254187157311, 107.60751599471817 -6.9230550325704785, 107.60762423465427 -6.923551147399013, 107.60796781887286 -6.922795418724843, 107.60976126351315 -6.9234167983639034, 107.60981976290623 -6.92288833535842, 107.61019415902199 -6.921535235075423, 107.60996601138895 -6.9214148969156515, 107.60977406106005 -6.921269239882861, 107.60877568430539 -6.921215410092754, 107.60803248370122 -6.920964170465973, 107.60646098091036 -6.9202756195987005, 107.60413538546972 -6.921132197573589, 107.60760636699621 -6.922655429707178, 107.60840676209392 -6.923058523383053, 107.60837943153109
Gambar peta	

Hasil pencarian dengan UCS	
Hasil pencarian dengan A*	

4.3 Peta jalan sekitar Buahbatu atau Bandung Selatan

Nama/keterangan	Foto

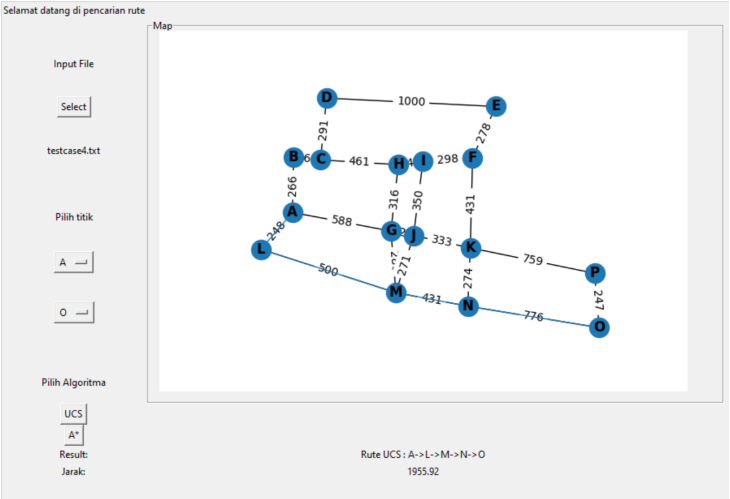
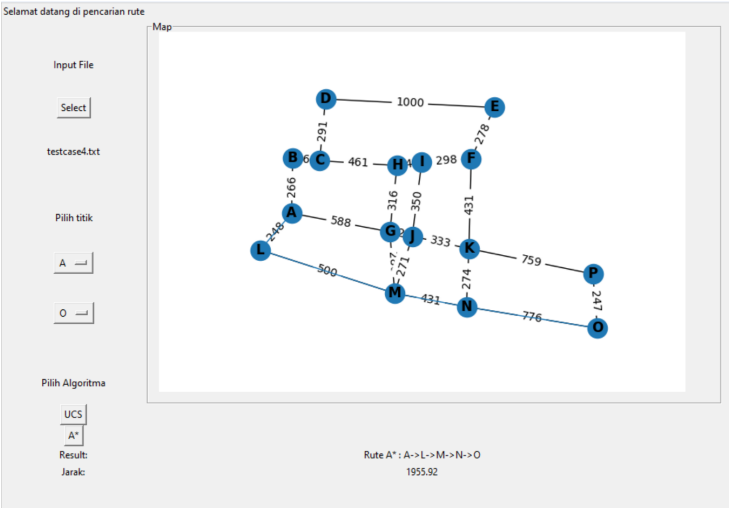
Matriks ketetanggan	A B C D E F G H I 0 850 0 0 0 0 0 0 0 0 0 130 500 0 0 0 0 0 0 0 130 0 0 400 290 0 0 0 0 0 0 0 0 0 0 0 500 0 0 400 0 0 0 550 300 550 0 0 290 0 0 0 0 700 0 0 0 0 0 550 0 0 0 0 0 0 0 0 300 700 0 0 0 0 0 0 500 550 0 0 0 0 -6.9489804771617365, 107.63445320433523 -6.954409525925381, 107.63914126200241 -6.954028089625104, 107.64018868922288 -6.958771412628588, 107.63944495394013 -6.954735863616492, 107.64381978740178 -6.951663175810896, 107.64096473742293 -6.955162551657583, 107.64856578651116 -6.952073913386774, 107.64440874425028 -6.959512529519497, 107.64400180719937
Gambar peta	

Hasil pencarian dengan UCS	
Hasil pencarian dengan A*	

4.4 Peta jalan sebuah kawasan di kota asal

Nama/keterangan	Foto

Matriks ketetanggan	A B C D E F G H I J K L M N O P 0 266.01 0 0 0 0 588.2 0 0 0 0 248.44 0 0 0 0 266.01 0 161.22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 161.22 0 291.68 0 0 0 461.88 0 0 0 0 0 0 0 0 0 0 291.68 0 1000 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1000 0 278.64 0 0 0 0 0 0 0 0 0 0 0 0 0 0 278.64 0 0 0 298.24 0 431.71 0 0 0 0 0 588.2 0 0 0 0 0 0 316.03 0 129.94 0 0 286.70 0 0 0 0 0 461.88 0 0 0 316.03 0 144.33 0 0 0 0 0 0 0 0 0 0 0 0 298.24 0 144.33 0 350.8 0 0 0 0 0 0 0 0 0 0 0 0 129.94 0 350.8 0 333.54 0 271.69 0 0 0 0 0 0 0 0 431.71 0 0 0 333.54 0 0 0 274.72 0 759.24 248.44 0 0 0 0 0 0 0 0 0 0 500 0 0 0 0 0 0 0 0 0 286.70 0 0 271.69 0 500 0 431.09 0 0 0 0 0 0 0 0 0 0 0 0 274.72 0 431.09 0 776.39 0 0 0 0 0 0 0 0 0 0 0 0 0 0 776.39 0 247.52 0 0 0 0 0 0 0 0 0 0 759.24 0 0 0 247.52 0 -7.3885645095824986, 109.35736609162383 -7.386201735459852, 109.35742507303253 -7.386303646210506, 109.35888060548059 -7.3836477647819345, 109.35920078360658 -7.384008373982846, 109.36830533405373 -7.386220119988258, 109.36705697422968 -7.389366486379681, 109.3626672272046 -7.386511451186166, 109.36306171689823 -7.386378271252413, 109.36438787373778 -7.389558745754446, 109.36387899908313 -7.390089335244248, 109.36693824683134 -7.390146983225189, 109.35567400638381 -7.392015824435537, 109.36293484662492 -7.392595401473738, 109.36680091415644 -7.393520432307658, 109.37387315623161 -7.39118148806399, 109.37362256053753
Gambar peta	

Hasil pencarian dengan UCS	
Hasil pencarian dengan A*	

Bab 5:
Penutup

5.1 Kesimpulan

Kita dapat menggunakan bahasa pemrograman Python untuk mengimplementasikan strategi algoritma A* dan UCS dalam pencarian jarak terdekat antara dua simpul. Tujuan utama dari program ini adalah untuk menemukan jarak yang paling optimal antara kedua simpul tersebut. Melalui implementasi algoritma UCS dan A* yang telah diajarkan dalam mata kuliah IF2211 - Strategi Algoritma, program ini dapat mencapai tingkat efektivitas yang cukup tinggi dalam menemukan jarak terdekat. Oleh karena itu, pembuatan program ini merupakan contoh bagaimana strategi algoritma yang telah diajarkan dapat diterapkan dalam konteks nyata dan memberikan solusi yang efektif.

5.2 Saran

Kami sadar bahwasannya program kami bukanlah yang terbaik jika dibandingkan dengan banyak program lainnya di luar sana. Berikut adalah beberapa saran yang telah kami diskusikan dan kumpulkan untuk pembuatan program serupa kedepannya agar program dapat membuahkan hasil yang jauh lebih baik:

1. Menggunakan API Google Maps agar mendapatkan hasil yang lebih akurat
2. Memperbanyak simpul agar presisi hasil jarak yang diperoleh meningkat
3. Memperbarui GUI agar terlihat lebih menarik

5.3 Refleksi

Pembuatan tugas besar mata kuliah IF2211 Strategi Algoritma ini merupakan pengalaman yang sangat berharga bagi penulis. Kami belajar bagaimana suatu pengimplementasian strategi algoritma dapat sangat berguna pada kehidupan sehari-hari, khususnya pembuatan program pencarian jarak antara dua simpul pada peta. Meskipun demikian, penulis merasa masih terdapat beberapa kekurangan pada program sehingga penulis sangat berharap untuk pembuatan algoritma serupa kedepannya dapat dieksekusi dengan lebih baik.

Daftar Pustaka

1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf
2. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf
3. https://www.trivusi.web.id/2023/01/algoritma-a-star.html

Lampiran

Link Repository GitHub : https://github.com/SulthanDA28/Tucil3_13521097_13521159