

**MENGHUBUNGKAN TITIK DENGAN NILAI TERKECIL**

**Shorted Algorithem | algoritma dijkstra DI PHYTON**

**LAPORAN TUGAS**

Disusun Untuk Memenuhi Tugas

Mata Kuliah Algoritma dan Pemograman 2 Kelas F

**LINK YOUTUBE : https://youtu.be/7V8ohj0dGtM**

Oleh:

**Bagas Cahyo Purnomo 212410103041**

**Rohmatulloh Fadhilah 212410103026**

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS JEMBER

2022

1. **Coding**

import tkinter as tk

window = tk.Tk()

window.title("TUGAS ALGO")

canvas = tk.Canvas(window, width=1500, height=1500)

canvas.pack()

total\_node = int(input())

node\_list = []

for i in range(total\_node):

    nodes = input()

    v1 = int(nodes.split()[0])

    v2 = int(nodes.split()[1])

    node\_list.append([v1, v2])

    canvas.create\_oval(v1-10, v2-10, v1+10, v2+10, fill="green")

    canvas.create\_text(v1, v2, text=str(i), fill="white")

total\_edge = int(input())

edge\_list = {}

for i in range(total\_edge):

    edges = input()

    node\_x = int(edges.split()[0])

    node\_y = int(edges.split()[1])

    weight = int(edges.split()[2])

    if node\_x not in edge\_list:

        edge\_list[node\_x] = {}

    edge\_list[node\_x][node\_y] = weight

    if node\_y not in edge\_list:

        edge\_list[node\_y] = {}

    edge\_list[node\_y][node\_x] = weight

inputan = input()

start = int(inputan.split()[0])

finish = int(inputan.split()[1])

# print(edge\_list)

def cari\_jalur(graf, awal, akhir, jalur=[]):

    jalur = jalur + [awal]

    if awal == akhir:

        return [jalur]

    if awal not in graf:

        return []

    semua\_jalur = []

    for node in graf[awal]:

        if node not in jalur:

            jalur\_baru = cari\_jalur(graf, node, akhir, jalur)

            for jalur\_baru\_ in jalur\_baru:

                semua\_jalur.append(jalur\_baru\_)

    return semua\_jalur

jalur = cari\_jalur(edge\_list, start, finish)

for i in jalur:

    cost = 0

    for j in range(len(i)-1):

        cost += edge\_list[i[j]][i[j+1]]

    i.append(cost)

    # print(i)

# print(jalur)

jalur = sorted(jalur, key=lambda x: x[-1])

print("Jalur terpendek:", jalur[0][:-1], "dengan total cost:", jalur[0][-1])

for i in jalur:

    i.pop()

for i in range(len(jalur)):

    for j in range(len(jalur[i])):

        jalur[i][j] = node\_list[jalur[i][j]]

result = jalur[0]

for i in range(len(jalur)):

    for j in range(len(jalur[i])-1):

        canvas.create\_line(jalur[i][j][0], jalur[i][j][1], jalur[i][j+1][0], jalur[i][j+1][1], fill="red")

for i in range(len(result)-1):

    canvas.create\_line(result[i][0], result[i][1], result[i+1][0], result[i+1][1], fill="blue")

window.mainloop()

1. **Penjelasan Coding**
   1. **Import**

import tkinter as tk

Disini Kami mengunakan Import untuk mengimport tkinter yang nantinya banyak metod yang diambil dari dictionary ini. Seperti pembuatan canvas dan konfigurasinya.

* 1. **Mengatur Window**

window = tk.Tk()

window.title("TUGAS ALGO")

canvas = tk.Canvas(window, width=1500, height=1500)

canvas.pack()

Mengantur Lebar dan tinggi dari kanvas sekaligus memberi judul pada canvas sebenarnya bisa kita atur hal yang lain misal background namun dalam tugas ini kami tidak membutuhkannya.

* 1. **Memproses Inputan(1)**

total\_node = int(input())

node\_list = []

for i in range(total\_node):

    nodes = input()

    v1 = int(nodes.split()[0])

    v2 = int(nodes.split()[1])

    node\_list.append([v1, v2])

Disini inputan akan diproses agar dapat digunakan untuk proses2 selanjutnya. Seperti pemisahan dan memasukkan titik ke list kosong yang telah dibuat

* 1. **Menggambar Titik**

canvas.create\_oval(v1-10, v2-10, v1+10, v2+10, fill="green")

canvas.create\_text(v1, v2, text=str(i), fill="white")

Disini diatur titik nya bentuk apa, lokasinya dimana (didapat dari titik titik di testcase dengan pemrosesan dahulu) disini bentuknya oval dan warnanya hijau serta ada angkanya.

* 1. **Memproses Inputan(2)**

total\_edge = int(input())

edge\_list = {}

Selain inputan titik kita disini juga perlu mengolah inputan edgesm, karena dalam testcase tugas kali ini tidak hanya ada koorfinat titik namun njuga ada edges.

* 1. **Mengelola dictionary**

for i in range(total\_edge):

    edges = input()

    node\_x = int(edges.split()[0])

    node\_y = int(edges.split()[1])

    weight = int(edges.split()[2])

    if node\_x not in edge\_list:

        edge\_list[node\_x] = {}

    edge\_list[node\_x][node\_y] = weight

    if node\_y not in edge\_list:

        edge\_list[node\_y] = {}

    edge\_list[node\_y][node\_x] = weight

Disini kami mengelola dictionary yang terdapat key awal berupa titik 0 lalu valuenya adalah key pasangan node yang valuenya adalah bobot.

* 1. **Mendapatkan titik awal dan akhir**

inputan = input()

start = int(inputan.split()[0])

finish = int(inputan.split()[1])

Dari testcase akan diambil titik yang sudah ditentukan (oleh dosen) sebagai titik awal dan titik akhir dari tugas ini.

* 1. **Debugging**

print(edge\_list)

print(i)

print(jalur)

Disinikita memprint list edges, jalur dan i yang mana hal ini untuk mengetahui apakah algoritma dan source code yang kami kembangkan sudah benar atau belum.

* 1. **Mencari semua jalur yang mungkin**

def cari\_jalur(graf, awal, akhir, jalur=[]):

    jalur = jalur + [awal]

    if awal == akhir:

        return [jalur]

    if awal not in graf:

        return []

    semua\_jalur = []

    for node in graf[awal]:

        if node not in jalur:

            jalur\_baru = cari\_jalur(graf, node, akhir, jalur)

            for jalur\_baru\_ in jalur\_baru:

                semua\_jalur.append(jalur\_baru\_)

    return semua\_jalur

Di algoritma ini kami mencari semua jalur yang mun gkin dari titik awal menuju titik akhir diman nanti jalur jalur baru tersebut akan dimasukkan dalam list semua\_jalutr yang nantinya akan digambarkan dalam proses selanjutnya.

* 1. **Menghitung Jarak semua jalur**

jalur = cari\_jalur(edge\_list, start, finish)

Jika sebelumnya kita mencari jalur maka sekarang kita mencari beratnya yang mana berat disini menjadi poin penting karena kita nantinya akan membedakan jarak terpendek(jyang dicari).

* 1. **Menghitung total bobot**

for i in jalur:

    cost = 0

    for j in range(len(i)-1):

        cost += edge\_list[i[j]][i[j+1]]

    i.append(cost)

 Menghitung total bobot dari semua nilai pada array di jalur tersebut dan nilainya ditambahkan pada masing-masing array.

* 1. **Mengurutkan Jalur dan memprinbt jawaban**

jalur = sorted(jalur, key=lambda x: x[-1])

print("Jalur terpendek:", jalur[0][:-1], "dengan total cost:", jalur[0][-1])

Disini akan diurutrkan dari jalur terpendek atau terringan menggunakan fungsi sorted setelah itu akan memprint jawaban dari soal yakni jalurnya dan nilai beratnya.

* 1. **Menghapus costnya**

for i in jalur:

    i.pop()

Menghapus costnya sehingga tidak mempengaruhi proses menggambar.

* 1. **Merubah indeks ke node**

for i in range(len(jalur)):

    for j in range(len(jalur[i])):

        jalur[i][j] = node\_list[jalur[i][j]]

result = jalur[0]

Disini kami mengkonfersi indeks ke node atau kedalam bentuk list lagi agar dalam proses selanjutnya ( pengambaran) kesalahan yang terjadi menjadi semakin kecil.

* 1. **Menggambar semua jalur**

for i in range(len(jalur)):

    for j in range(len(jalur[i])-1):

        canvas.create\_line(jalur[i][j][0], jalur[i][j][1], jalur[i][j+1][0], jalur[i][j+1][1], fill="red")

Setelah ini adalah proses pengambaran semua jalur menggunakan create\_line yang mana yang digambarkan berada pada list semua\_jalur dan memberkan warna padanya agar nanti dapat dibedakan.

* 1. **Menggambar jalur Terpendek**

for i in range(len(result)-1):

    canvas.create\_line(result[i][0], result[i][1], result[i+1][0], result[i+1][1], fill="blue")

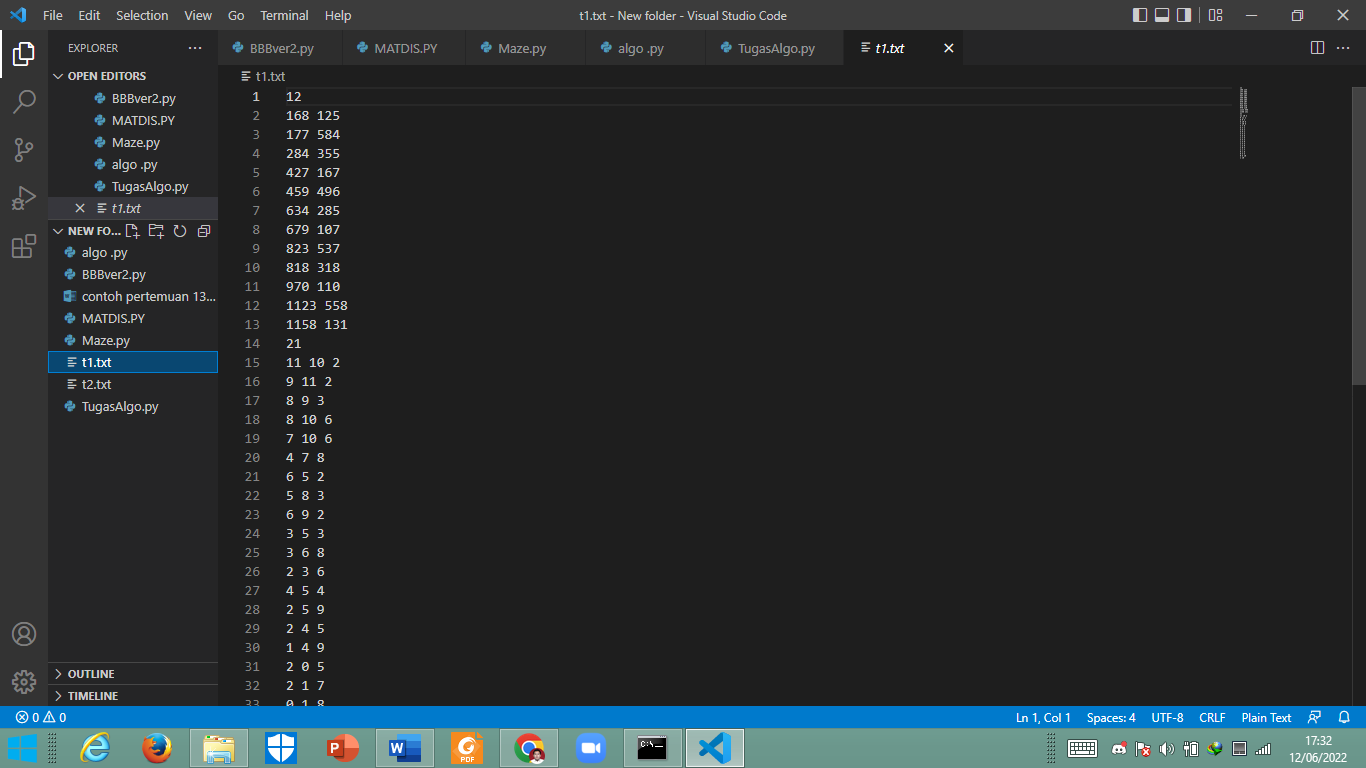
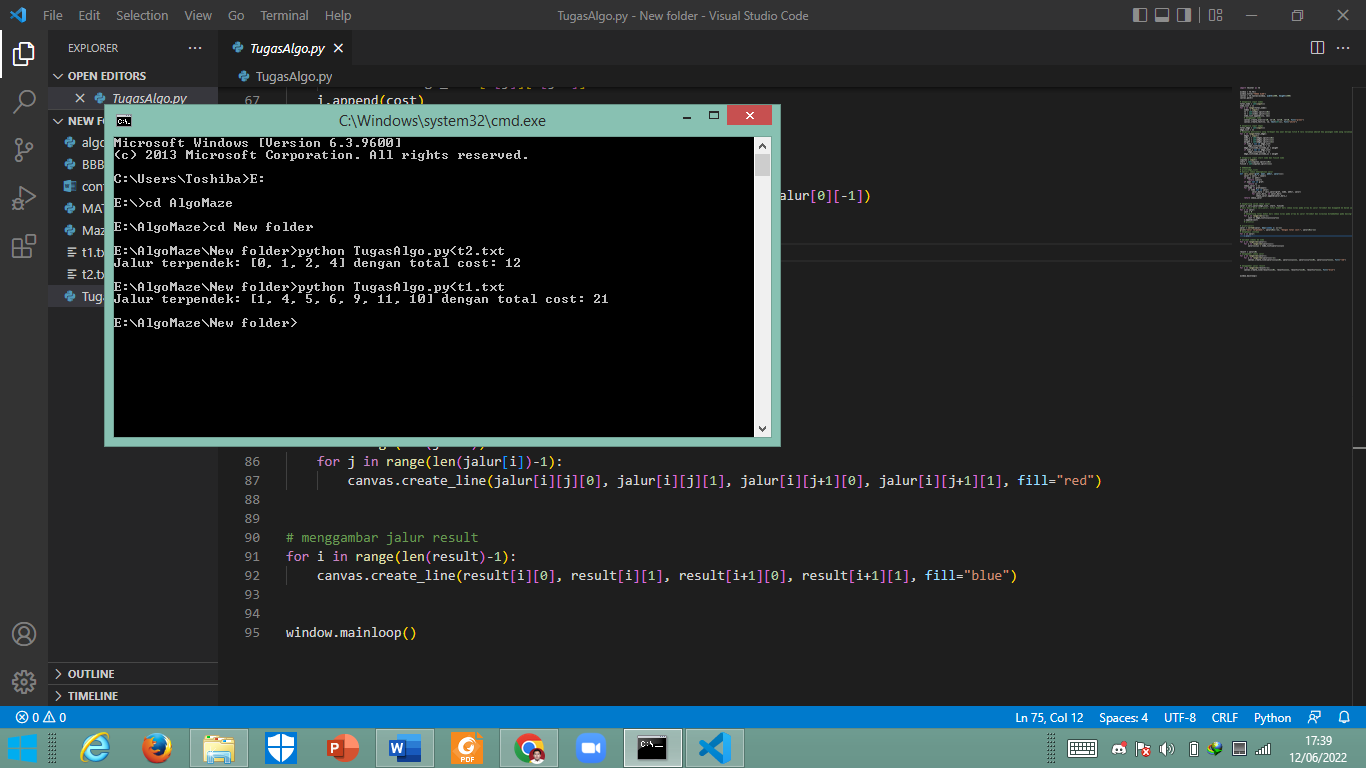
Dari hasil pengurutan yang diletakkan di list result. Kita menggambarkan hasilnya , untuk membuatnya lebih terlihat berbeda dengan jalur umum kami memberi variasi warna biru untuk garisnya.

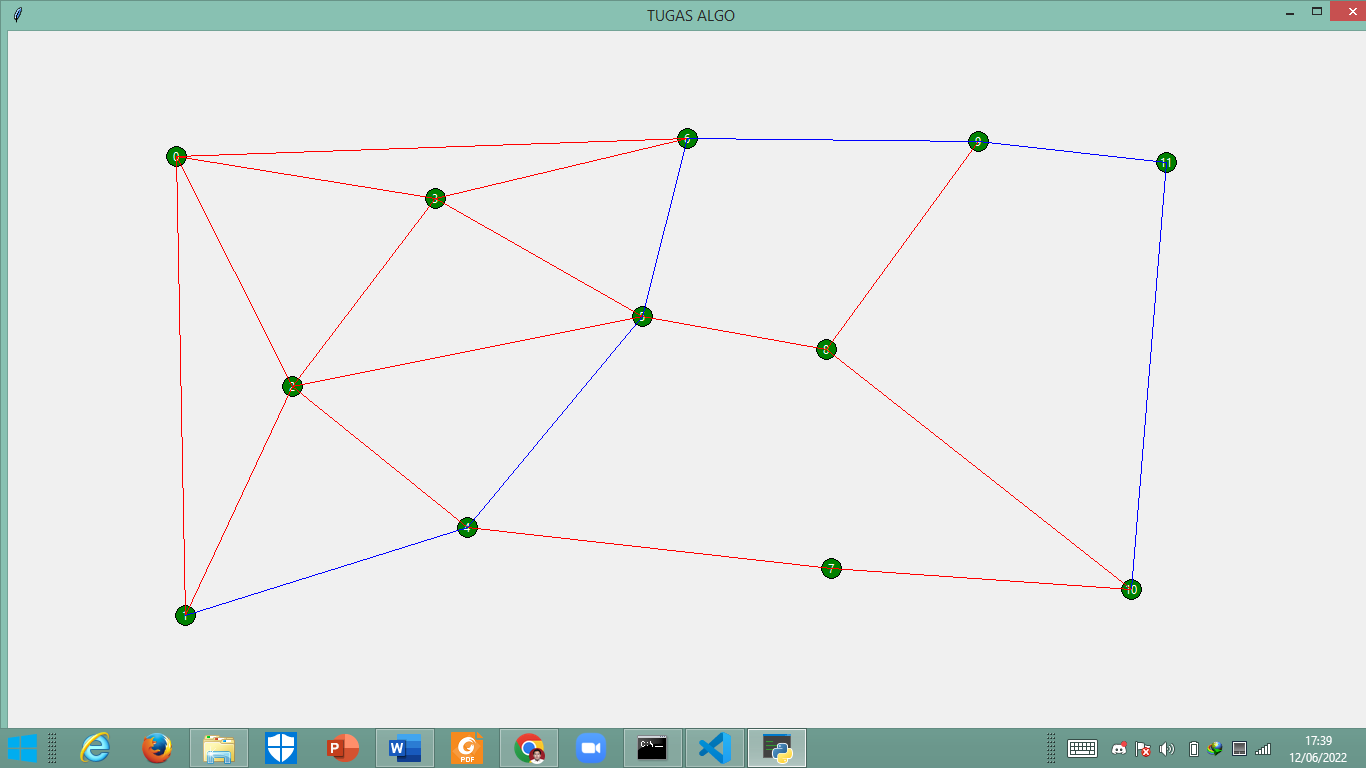
* 1. **Penutup**

window.mainloop()

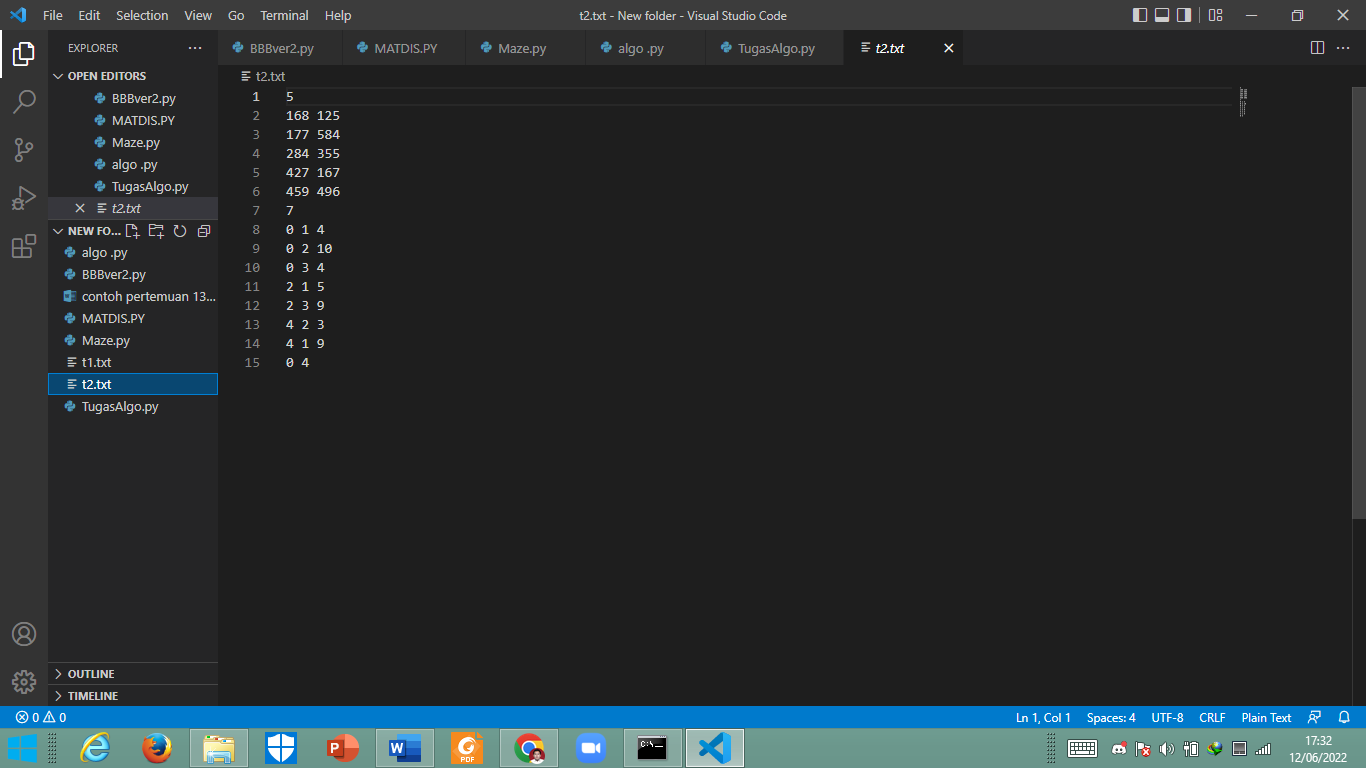
window.mainloop() digunakan di line terakhir sebagai penutup karena kalau tidak menggunakan ini maka program kaan eror.

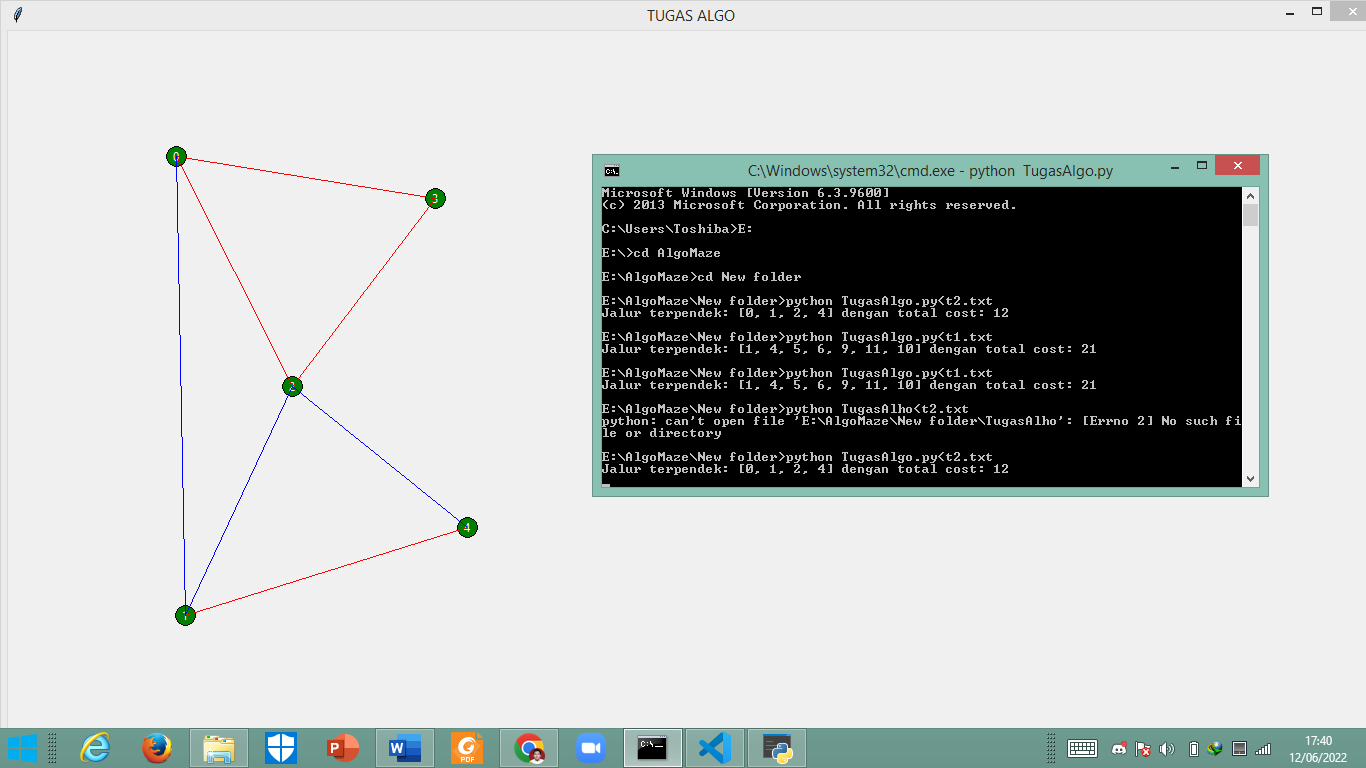
1. **Dokumentasi Hasil**
   1. **Testcase 1**

** **

****

* 1. **Testcase 2**

****

****