



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()

```

2. Penjelasan Coding

2.1.Import

```
import tkinter as tk
```

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

2.2.Mengatur Window

```
window = tk.Tk()  
window.title("TUGAS ALGO")  
canvas = tk.Canvas(window, width=1500, height=1500)  
canvas.pack()
```

Mengatur 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.

2.3.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

2.4.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.

2.5.Memproses Inputan(2)

```
total_edge = int(input())
edge_list = {}
```

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

2.6.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.

2.7.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.

2.8.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.

2.9. 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 mungkin dari titik awal menuju titik akhir dimana nanti jalur jalur baru tersebut akan dimasukkan dalam list semua_jalur yang nantinya akan digambarkan dalam proses selanjutnya.

2.10. 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 (yang dicari).

2.11. 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.

2.12. Mengurutkan Jalur dan memprint jawaban

```
jalur = sorted(jalur, key=lambda x: x[-1])
print("Jalur terpendek:", jalur[0][: -1], "dengan total cost:",
      jalur[0][ -1])
```

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

2.13. Menghapus costnya

```
for i in jalur:  
    i.pop()
```

Menghapus costnya sehingga tidak mempengaruhi proses menggambar.

2.14. 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 mengkonversi indeks ke node atau kedalam bentuk list lagi agar dalam proses selanjutnya (penggambaran) kesalahan yang terjadi menjadi semakin kecil.

2.15. 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 penggambaran semua jalur menggunakan create_line yang mana yang digambarkan berada pada list semua_jalur dan memberikan warna padanya agar nanti dapat dibedakan.

2.16. 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.

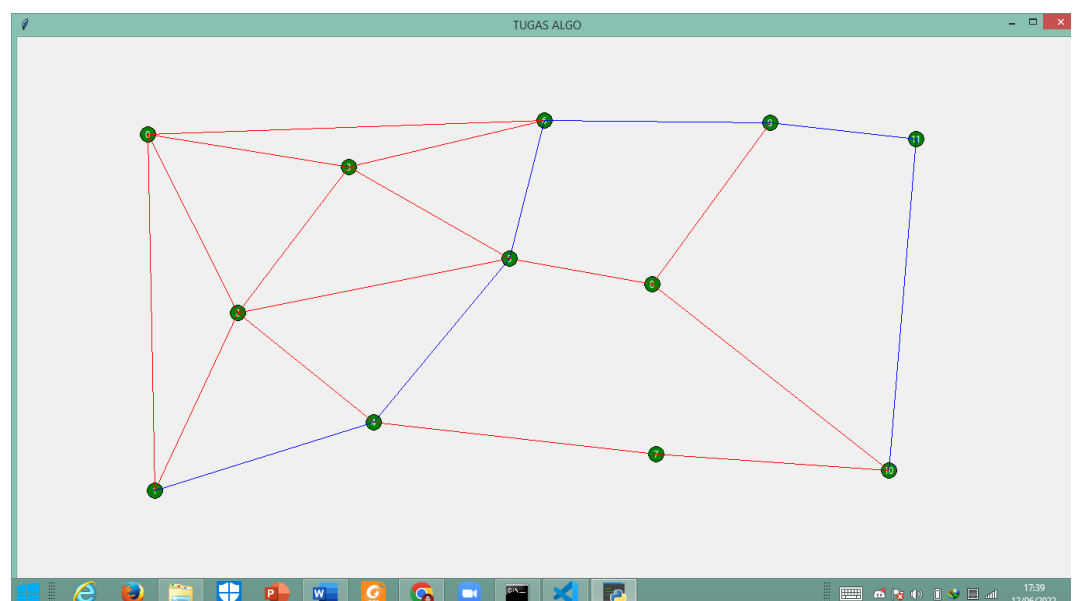
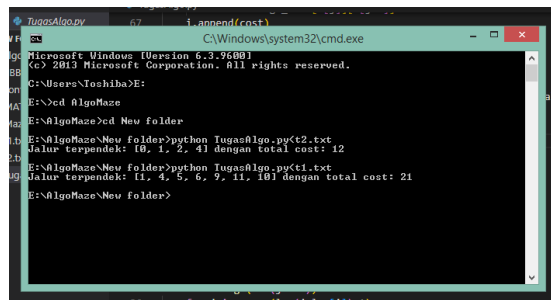
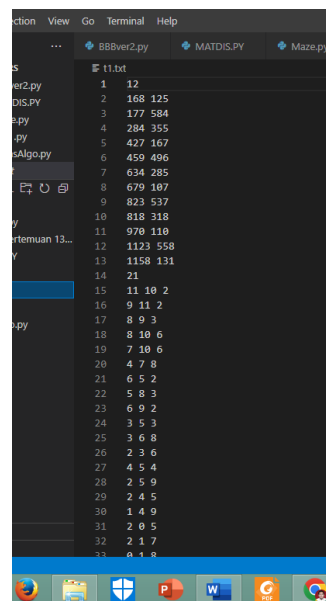
2.17. Penutup

```
window.mainloop()
```

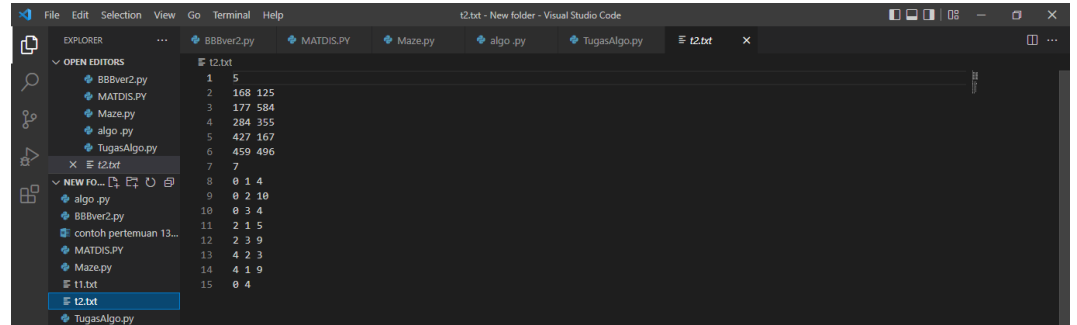
`window.mainloop()` digunakan di line terakhir sebagai penutup karena kalau tidak menggunakan ini maka program akan error.

3. Dokumentasi Hasil

3.1. Testcase 1



3.2. Testcase 2



```
1 5
2 168 125
3 177 584
4 284 355
5 427 167
6 459 496
7 7
8 0 1 4
9 0 2 10
10 0 3 4
11 2 1 5
12 2 3 9
13 4 2 3
14 4 1 9
15 0 4
```

