

Nama : Afriyan Kuncoro Aji S

Nim : G.231.22.0110

Kelas : Teknik Informarika B sore

## ALGORITMA

### 1. Inisialisasi Kelas GraphVisualization:

- Buat objek grafik menggunakan library networkx.
- Simpan data grafik yang diberikan pada saat inisialisasi.

```
class GraphVisualisasi:  
    def __init__(self, graph_data):  
        self.G = nx.Graph()  
        self.graph_data = graph_data
```

### 2. Tambahkan Sisi Grafik:

- Iterasi melalui setiap node dan koneksi yang diberikan dalam data grafik.
- Tambahkan sisi grafik dengan bobot sesuai yang ditentukan.

```
def add_edges(self):  
    for node, connections in self.graph_data.items():  
        for edge in connections:  
            self.G.add_edge(node, edge['v'],  
weight=edge['w'])
```

### 3. Visualisasikan Grafik:

- Tentukan posisi node menggunakan layout Spring.
- Dapatkan atribut berat sisi dari grafik.
- Gambarkan grafik dengan menggunakan nx.draw() dan nx.draw\_networkx\_edge\_labels().
- Tampilkan gambar grafik.

```
def visualize(self):  
    pos = nx.spring_layout(self.G)  
    weights = nx.get_edge_attributes(self.G, "weight")  
  
    plt.figure()  
    nx.draw(  
        self.G, pos, edge_color='black', width=2,  
linewidths=2,  
        node_size=500, node_color='blue', alpha=0.9,  
        labels={node: node for node in self.G.nodes()}  
    )  
    nx.draw_networkx_edge_labels(self.G, pos,  
edge_labels=weights)  
    plt.axis('off')  
    plt.show()
```

#### 4. Fungsi Utama Visualisasi Grafik:

- Panggil metode `add_edges()` untuk menambahkan sisi pada grafik.
- Panggil metode `visualize()` untuk menampilkan grafik.

```
• def graph_visualization(self):  
•     self.add_edges()  
•     self.visualize()
```

#### 5. Definisikan Data Grafik dan Visualisasikan:

- Buat data grafik sesuai kebutuhan.
- Buat objek dari kelas `GraphVisualization` dengan data grafik tersebut.
- Panggil metode `graph_visualization()` untuk menampilkan grafik.

```
• graph_data = {  
•     '0': [{ 'v': '1', 'w': 2}, { 'v': '2', 'w': 6}],  
•     '1': [{ 'v': '0', 'w': 2}, { 'v': '3', 'w': 5}],  
•     '2': [{ 'v': '0', 'w': 6}, { 'v': '3', 'w': 8}],  
•     '3': [{ 'v': '1', 'w': 5}, { 'v': '2', 'w': 8}, { 'v': '5',  
•         'w': 15}, { 'v': '4', 'w': 10}],  
•     '4': [{ 'v': '3', 'w': 10}, { 'v': '5', 'w': 6}, { 'v':  
•         '6', 'w': 2}],  
•     '5': [{ 'v': '3', 'w': 15}, { 'v': '4', 'w': 6}, { 'v':  
•         '6', 'w': 6}],  
•     '6': [{ 'v': '5', 'w': 6}, { 'v': '4', 'w': 2}],  
• }  
•  
• # Visualize the graph  
• C_Dijkstra_shortest_path_2 =  
•     GraphVisualisasi(graph_data=graph_data)  
• C_Dijkstra_shortest_path_2.graph_visualization()
```

#### 6. Fungsi `get_path_weight` untuk Mendapatkan Bobot Jalur:

- Fungsi ini menerima jalur sebagai argumen dan menghitung bobot jalur tersebut berdasarkan data grafik yang diberikan.

```
• def get_path_weight(path):  
•     path_weight = 0  
•     for index, value in enumerate(path):  
•         try:  
•             for j in new_graph_data[value]:  
•                 if j['v'] == path[index + 1]:  
•                     path_weight += j['w']  
•             except IndexError:  
•                 break  
•     return path_weight
```

#### 7. Fungsi `find_shortest_path` untuk Menemukan Jalur Terpendek:

- Fungsi ini menggunakan pendekatan rekursif untuk mencari jalur terpendek dari titik awal ke titik akhir.

```

• def find_shortest_path(graph_data, start, end, path=[]):
•     path = path + [start]
•     shortest = None
•     weights = None
•
•     if start == end:
•         return path
•
•     for node in graph_data[start]:
•         if node['v'] not in path:
•             new_path = find_shortest_path(graph_data,
node['v'], end, path)
•             if new_path:
•                 new_weight = get_path_weight(new_path)
•                 if not weights or new_weight < weights:
•                     shortest = new_path
•                     weights = new_weight
•     return shortest

```

#### 8. Mencari Jalur Terpendek dan Bobotnya:

- Panggil fungsi find\_shortest\_path dengan parameter data grafik, titik awal, dan titik akhir.
- Hitung bobot jalur yang ditemukan menggunakan fungsi get\_path\_weight.
- Tampilkan jalur terpendek dan bobotnya.

```

• shortest_path = find_shortest_path(new_graph_data, '0', '6')
• path_length = get_path_weight(shortest_path)
•
• print('Shortest Path:', shortest_path)
• print('Path Length:', path_length)

```