Nama : Dimas Nando Firzaki

NIM : G.211.22.0120

PENJELASAN

```
import heapq

class Graph:
    def __init__(self):
        self.vertices = {}

    def add_vertex(self, name):
        self.vertices[name] = []

    def add_edge(self, src, dest, weight):
        self.vertices[src].append((dest, weight))
        self.vertices[dest].append((src, weight))

    def dijkstra(self, start):
        distances = {vertex: float('infinity') for vertex in
self.vertices}
        distances[start] = 0
        priority_queue = [(0, start)]

        while priority_queue:
            current_distance, current_vertex =
heapq.heappop(priority_queue)

            if current_distance > distances[current_vertex]:
                continue

            for neighbor, weight in self.vertices[current_vertex]:
                distance = current_distance + weight

                if distance < distances[neighbor]:
                    distances[neighbor] = distance
                    heapq.heappush(priority_queue, (distance,
neighbor))

        return distances

# Contoh penggunaan
graph = Graph()
graph.add_vertex('A')
graph.add_vertex('B')
graph.add_vertex('C')
graph.add_vertex('D')
```

```
graph.add_vertex('E')

graph.add_edge('A', 'B', 4)
graph.add_edge('A', 'C', 2)
graph.add_edge('B', 'C', 5)
graph.add_edge('B', 'D', 10)
graph.add_edge('C', 'D', 3)
graph.add_edge('C', 'E', 1)
graph.add_edge('D', 'E', 7)

start_vertex = 'A'
shortest_distances = graph.dijkstra(start_vertex)
print(f"Shortest distances from node {start_vertex}:
{shortest_distances}")
```

1. vertices (self.vertices): Ini adalah sebuah dictionary yang merepresentasikan graf. Setiap kunci dalam dictionary adalah nama simpul, dan setiap nilai (dalam bentuk list) adalah daftar tetangga dari simpul tersebut beserta bobotnya.

2. distances: Ini adalah dictionary yang menyimpan jarak terpendek dari simpul awal ke semua simpul lainnya. Setiap kunci dalam dictionary adalah nama simpul, dan setiap nilai adalah jarak terpendek dari simpul awal ke simpul tersebut.

3. priority_queue: Ini adalah sebuah min-heap (heap terurut secara ascending) yang digunakan untuk memilih simpul berikutnya yang akan dieksplorasi berdasarkan jarak dari simpul awal. Setiap elemen dalam min-heap adalah tupel (jarak, simpul), yang mewakili jarak dari simpul awal ke simpul tersebut.

Penjelasan secara lebih rinci tentang setiap list/dictionary ini dalam kode:

- self.vertices: Dictionary ini merepresentasikan graf. Setiap kunci adalah nama simpul (misalnya, 'A', 'B', 'C', dst.) dan setiap nilai adalah daftar tetangga dari simpul tersebut beserta bobotnya. Misalnya, jika self.vertices['A'] = [('B', 4), ('C', 2)], itu berarti simpul 'A' terhubung dengan simpul 'B' dengan bobot 4 dan juga terhubung dengan simpul 'C' dengan bobot 2.
- distances: Dictionary ini awalnya diinisialisasi dengan jarak tak terhingga (float('infinity')) dari simpul awal ke semua simpul lainnya. Kemudian, saat algoritma Dijkstra berjalan, jarak terpendek dari simpul awal ke semua simpul lainnya akan diperbarui secara iteratif.
- priority_queue: Ini adalah min-heap yang digunakan untuk menjaga simpul yang akan dieksplorasi selanjutnya berdasarkan jarak terpendek dari simpul awal. Heap ini diisi dengan tupel (jarak, simpul), di mana jarak adalah jarak dari simpul awal ke simpul tersebut, dan simpul adalah simpul itu sendiri.