Nama : Rayhan Prasetya N
Nim : G.211.22.0076
TI/B2
Praktikum 8 Struktur Data

```
class Graph:

    def __init__(self, vertices):
        self.V = vertices
        self.graph = []

    def add_edge(self, u, v, w):
        self.graph.append([u, v, w])

    def bellman_ford(self, src):
        # Inisialisasi jarak dari src ke semua titik lainnya dengan nilai tak
terhingga
        dist = [float("inf")] * self.V
        dist[src] = 0

        # Lakukan relaksasi untuk semua sisi V-1 kali
        for _ in range(self.V - 1):
            for u, v, w in self.graph:
                if dist[u] != float("inf") and dist[u] + w < dist[v]:
                    dist[v] = dist[u] + w

        # Periksa apakah ada siklus berbobot negatif
        for u, v, w in self.graph:
            if dist[u] != float("inf") and dist[u] + w < dist[v]:
                print("Graf berisi siklus berbobot negatif")
                return

        # Tampilkan jarak dari src ke semua titik lainnya
        print("Jarak dari source node:", src)
        for i in range(self.V):
            print(f"Node {i} -> Jarak: {dist[i]}")


# Contoh penggunaan
g = Graph(5)
g.add_edge(0, 1, -1)
g.add_edge(0, 2, 4)
g.add_edge(1, 2, 3)
g.add_edge(1, 3, 2)
g.add_edge(1, 4, 2)
g.add_edge(3, 2, 5)
g.add_edge(3, 1, 1)
g.add_edge(4, 3, -3)
```

```
Jarak dari source node: 0
Node 0 -> Jarak: 0
```

```
Node 1 -> Jarak: -1
Node 2 -> Jarak: 2
Node 3 -> Jarak: -2
Node 4 -> Jarak: 1
```

Penjelasan :

Algoritma Bellman-Ford digunakan untuk menemukan jalur terpendek dari satu titik ke semua titik lainnya dalam graf berarah dengan bobot lintasan yang dapat bernilai negatif. Algoritma ini memperbarui estimasi jarak dari titik awal ke semua titik lainnya secara berulang sebanyak V-1 kali, di mana V adalah jumlah node dalam graf. Pada setiap iterasi, algoritma melakukan relaksasi pada setiap sisi, yaitu memperbarui estimasi jarak jika ditemukan jalur yang lebih pendek.

Selain itu, algoritma Bellman-Ford melakukan langkah tambahan untuk mendeteksi siklus berbobot negatif. Jika pada iterasi ke-V (iterasi terakhir) masih terdapat perubahan, maka graf tersebut berisi siklus berbobot negatif.

Penjelasan Kode Python:

1. `class Graph:`: Ini adalah definisi kelas untuk representasi graf. Kode ini mencakup metode untuk inisialisasi graf, menambahkan sisi, dan menjalankan algoritma Bellman-Ford.
2. `__init__(self, vertices)`: Metode ini inisialisasi objek graf dengan jumlah node (vertices) dan array untuk menyimpan sisi-sisi graf.
3. `add_edge(self, u, v, w)`: Metode ini digunakan untuk menambahkan sisi ke graf dengan node awal `u`, node tujuan `v`, dan bobot lintasan `w`.
4. `bellman_ford(self, src)`: Metode ini menjalankan algoritma Bellman-Ford. Pertama-tama, ia menginisialisasi array `dist` yang menyimpan estimasi jarak dari node awal (`src`) ke setiap node lainnya dengan nilai tak terhingga. Kemudian, ia melakukan relaksasi pada setiap sisi sebanyak V-1 kali. Setelah itu, dilakukan pengecekan untuk mendeteksi siklus berbobot negatif.
5. `for _ in range(self.V - 1):`: Ini adalah loop untuk melakukan V-1 iterasi relaksasi pada setiap sisi.
6. `for u, v, w in self.graph:`: Loop ini mengiterasi melalui setiap sisi graf dan melakukan relaksasi jika ditemukan jalur yang lebih pendek.
7. `for u, v, w in self.graph:` (bagian kedua): Loop ini digunakan untuk mengecek apakah ada siklus berbobot negatif setelah V-1 iterasi. Jika masih terdapat perubahan pada iterasi ke-V, graf berisi siklus berbobot negatif.
8. Output menampilkan jarak dari node awal (`src`) ke semua node lainnya.