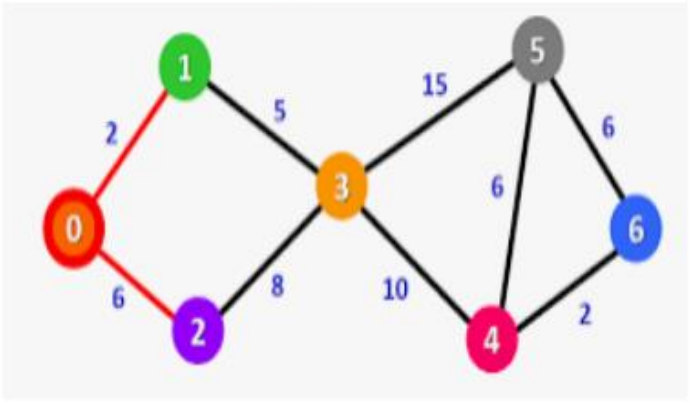
NAMA : VIJJA SETIADI

NIM : G.211.22.0098

PENJELASAN PROGRAM

C. Dijkstra shortest path 2



1. Algoritma Jalur Terpendek Dijkstra

Algoritma ini digunakan untuk menghitung dan menemukan jalur terpendek antara node menggunakan bobot yang diberikan dalam suatu grafik. (Dalam suatu jaringan, bobot diberikan oleh paket status tautan dan berisi informasi seperti kesehatan router, biaya lalu lintas, dll.).

2. Ringkasan Cara Kerja

Algoritma ini dimulai dengan node sumber dan menemukan sisa jarak dari node sumber. Algoritma Dijkstra melacak jarak yang diketahui saat ini dari node sumber ke sisa node dan secara dinamis memperbarui nilai-nilai ini jika ditemukan jalur yang lebih pendek.

Sebuah node kemudian ditandai sebagai telah dikunjungi dan ditambahkan ke jalur jika jarak antara node tersebut dan node sumber adalah yang terpendek. Ini terus berlanjut sampai semua node telah ditambahkan ke jalur, dan akhirnya, kita mendapatkan jalur terpendek dari node sumber ke semua node lainnya, yang dapat diikuti paket dalam suatu jaringan untuk mencapai tujuannya.

Kita memerlukan bobot yang positif karena bobot tersebut harus ditambahkan ke perhitungan untuk mencapai tujuan kita. Bobot negatif akan membuat algoritma tidak memberikan hasil yang diinginkan.

3. Penjelasan Program

Ada sebuah konstruktor untuk memberikan nilai awal init dan tiga fungsi yang ditentukan pengguna:

- printSolution()
- minDistance()
- dijkstra()

Konstruktor mengambil parameter nodes, yang merupakan jumlah node yang akan dianalisis.

```
def __init__(self, nodes):
    # Inisialisasi jarak array
    self.distArray = [0 for i in range(nodes)]
    # Inisialisasi set node yang telah lewati
    self.vistSet = [0 for i in range(nodes)]
    # Inisialisasi jumlah node
    self.V = nodes
    # Inisialisasi nilai tak terhingga
    self.INF = 1000000
    # Inisialisasi graf matriks
    self.graph = [[0 for kolom in range(nodes)]
                  for baris in range(nodes)]
```

Fungsi dijkstra() mengambil parameter, yaitu node sumber (srcNode). Pertama-tama, ia menginisialisasi setiap jarak menjadi tak terhingga dan status kunjungan menjadi false untuk menunjukkan bahwa node belum dikunjungi menggunakan loop for dan jarak awal dari node sumber menjadi 0.

Pada loop berikutnya, pertama-tama memilih node dengan jarak minimum dari himpunan node yang belum diproses. u selalu sama dengan srcNode pada iterasi pertama.

Kemudian menambahkan node dengan jarak minimum ke dalam himpunan node yang telah dikunjungi dengan mengatur nilai menjadi True. Pada loop terakhir, yang berada di dalam loop kedua, kode memperbarui jarak node dari node 0.

dist[v] hanya jika tidak ada di dalam vistSet, ada tepi dari u ke v, dan total jarak dari srcNode ke v melalui u lebih kecil dari nilai dist[v] saat ini.

Selanjutnya, ia memanggil printSolution() untuk menampilkan tabel setelah melewatkan array jarak ke fungsi tersebut.

```
def dijkstra(self, srcNode):
    for i in range(self.V):
        # Inisialisasi jarak menjadi tak terhingga terlebih dahulu
        self.distArray[i] = self.INF
        # Atur set node yang telah dilewati menjadi False untuk setiap node
        self.vistSet[i] = False
    # Inisialisasi jarak pertama menjadi 0
    self.distArray[srcNode] = 0
    for i in range(self.V):

        # Pilih node dengan jarak minimum dari
        # himpunan node yang belum diproses.
        # 'u' selalu sama dengan srcNode pada iterasi pertama
        u = self.minDistance(self.distArray, self.vistSet)

        # Masukkan node dengan jarak minimum ke dalam
        # himpunan node yang telah dikunjungi
        self.vistSet[u] = True

        # Perbarui dist[v] hanya jika v tidak ada di dalam vistSet, ada tepi dari
        # 'u' ke 'v', dan total bobot dari src ke 'v' melalui 'u'
        # lebih kecil dari nilai dist[v] saat ini
        for v in range(self.V):
            if self.graph[u][v] > 0 and self.vistSet[v] == False and self.distArray[v] > self.distArray[u] + self.graph[u][v]:
                self.distArray[v] = self.distArray[u] + self.graph[u][v]

    self.printSolution(self.distArray)
```

Fungsi minDistance() memeriksa node terdekat dalam distArray yang belum termasuk dalam nodes yang belum dikunjungi dalam array vistSet[v]. Kemudian mengembalikan indeks node tersebut. Ia mengambil dua array sebagai parameter, yaitu distArray dan vistSet[v].

```
# Fungsi bantu untuk menemukan node dengan nilai jarak minimum, dari
# himpunan node yang belum termasuk dalam pohon jalur terpendek
def minDistance(self, distArray, vistSet):

    # Inisialisasi jarak minimum untuk node selanjutnya
    min = self.INF

    # Cari node terdekat yang belum ada di
    # node yang belum dikunjungi
    for v in range(self.V):
        if distArray[v] < min and vistSet[v] == False:
            min = distArray[v]
            min_index = v

    return min_index
```

Fungsi printSolution() digunakan untuk menampilkan hasil akhir, yaitu node dan tabel mereka yang disimpan dalam array distArray, yang diambil sebagai parameter.

```
def printSolution(self, distArray):
    print("Node \tJarak dari 0")
    for i in range(self.V):
        print(i, "\t", distArray[i])
```

Selanjutnya, kita membuat objek menampilkanGraf dari kelas Graph() dan melewatkan jumlah node kepadanya.

```
menampilkanGraf = Graf(7)
```

Kemudian, membuat matriks untuk menyimpan jarak.

```
menampilkanGraf.graph = [[0, 2, 6, 0, 0, 0, 0],
                         [2, 0, 0, 5, 0, 0, 0],
                         [6, 6, 0, 8, 0, 0, 0],
                         [0, 0, 8, 0, 10, 15, 0],
                         [0, 0, 0, 10, 0, 6, 2],
                         [0, 0, 0, 15, 6, 0, 6],
                         [0, 0, 0, 0, 2, 6, 0],
                         ]
```

Matriks ini sama dengan tabel yang ditunjukkan di atas:

	0	1	2	3	4	5	6
0	0	2	6	0	0	0	0
1	2	0	0	5	0	0	0
2	6	6	0	8	0	0	0
3	0	0	8	0	10	15	0
4	0	0	0	10	0	6	2
5	0	0	0	15	6	0	6
6	0	0	0	0	2	6	0

Baris teratas dan kolom paling kiri mewakili node. Kita membaca node dari kolom kiri dan memeriksa jaraknya dengan baris teratas. Titik pertemuan menunjukkan jarak. Jaraknya adalah 0 jika node tidak bertetangga.

Sebagai contoh:

Jarak dari 0 ke 0 adalah 0.

	0	1	2	3	4	5	6
0	*0*	2	6	0	0	0	0
1	2	0	0	5	0	0	0
2	6	6	0	8	0	0	0
3	0	0	8	0	10	15	0
4	0	0	0	10	0	6	2
5	0	0	0	15	6	0	6
6	0	0	0	0	2	6	0

Jarak dari 5 ke 3 adalah 15.

	0	1	2	3	4	5	6
0	*0*	2	6	0	0	0	0
1	2	0	0	5	0	0	0
2	6	6	0	8	0	0	0
3	0	0	8	0	10	15	0
4	0	0	0	10	0	6	2
5	0	0	0	15	6	0	6
6	0	0	0	0	2	6	0

Terakhir, kita menampilkan hasilnya.

```
menampilkanGraf.dijkstra(0)
```

Outputnya akan menjadi:

```
Node    Jarak dari 0
0        0
1        2
2        6
3        7
4        17
5        22
6        19
```