Acara 2
Pokok Bahasan : Masalah, Ruang Masalah dan Pencarian Solusi
Minggu/Acara Praktikum : 2 / 5
Tempat : Lab. Rekayasa Sistem Informasi
Alokasi Waktu : 1 x 110 menit

a. Indikator

1. Mahasiswa mampu menyelesaikan studi kasus graph dengan algorithm Djiktra dalam mencari rute terpendek

b. Dasar Teori

Dijkstra's Algorithm: Definisi Dasar dan Rumus:

Definisi Dasar:

Dijkstra's Algorithm adalah algoritma pencarian jalur terpendek dalam graf berbobot positif dari simpul asal ke semua simpul lainnya. Algoritma ini menggunakan pendekatan berbasis greedy yang selalu memilih simpul dengan jarak kumulatif terkecil yang belum diproses.

Rumus Dasar:

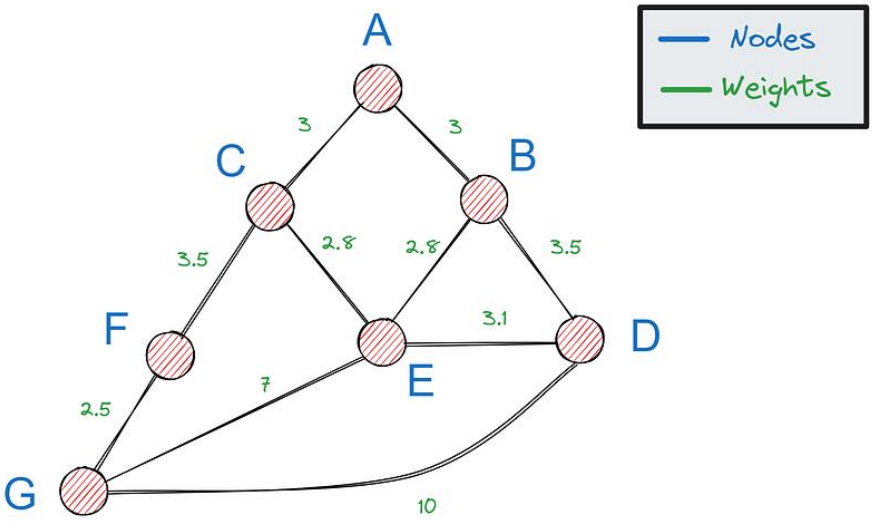
1. Inisialisasi jarak dari simpul awal ke dirinya sendiri dengan nilai 0, dan jarak ke semua simpul lain dengan ∞ (tak terhingga).
2. Setel simpul awal sebagai simpul saat ini dan tandai sebagai dikunjungi.
3. Untuk setiap simpul yang terhubung langsung ke simpul saat ini (tetangga):
    - Hitung jarak sementara dari simpul awal ke simpul tetangga melalui simpul saat ini.
    - Jika jarak sementara lebih kecil daripada jarak yang telah disetel sebelumnya untuk simpul tetangga, perbarui jarak tersebut.
4. Pilih simpul yang belum dikunjungi dengan jarak terkecil sebagai simpul saat ini dan ulangi langkah 3.
5. Ulangi proses hingga semua simpul telah dikunjungi atau jarak terpendek ke simpul tujuan ditemukan.

Studi Kasus Sederhana

c. Alat dan Bahan

1. BKPM
2. Bahasa Pemograman Python
3. Bolpoin
4. Folio Bergaris
5. HVS A4

d. Prosedur Kerja



Berdasarkan Graph di atas Mari kita coba mencari jalur terpendek antara titik **B** dan **F** menggunakan algoritma Dijkstra dari sedikitnya **tujuh jalur yang mungkin**.

Pengerjaan hitung manual:

Inisialisasi:

- Simpul awal: B
- Jarak awal:
  - B ke B = 0 (karena titik awal)
  - Semua simpul lainnya diatur ke ∞ (tak terhingga).
- Tabel Jarak Awal:
  - distances = {'A': ∞, 'B': 0, 'C': ∞, 'D': ∞, 'E': ∞, 'F': ∞, 'G': ∞}

Iterasi 1: Kunjungi Simpul B

- Dari B ke A: jarak = 0 (B) + 3 (B ke A) = 3
- Dari B ke D: jarak = 0 (B) + 3.5 (B ke D) = 3.5
- Dari B ke E: jarak = 0 (B) + 2.8 (B ke E) = 2.8
- Update Tabel Jarak:
  - distances = {'A': 3, 'B': 0, 'C': ∞, 'D': 3.5, 'E': 2.8, 'F': ∞, 'G': ∞}

Iterasi 2: Kunjungi Simpul E (Jarak Terdekat Berikutnya)

- Dari E ke C: jarak = 2.8 (E) + 2.8 (E ke C) = 5.6
- Dari E ke D: jarak = 2.8 (E) + 3.1 (E ke D) = 5.9 (tidak perlu update karena 5.9 > 3.5)
- Dari E ke G: jarak = 2.8 (E) + 7 (E ke G) = 9.8
- Update Tabel Jarak:
  - distances = {'A': 3, 'B': 0, 'C': 5.6, 'D': 3.5, 'E': 2.8, 'F': ∞, 'G': 9.8}

Iterasi 3: Kunjungi Simpul A (Jarak Terdekat Berikutnya)

- Dari A ke C: jarak = 3 (A) + 3 (A ke C) = 6 (tidak perlu update karena 6 > 5.6)

- Tabel Jarak Tetap:
  - distances = {'A': 3, 'B': 0, 'C': 5.6, 'D': 3.5, 'E': 2.8, 'F': ∞, 'G': 9.8}

Iterasi 4: Kunjungi Simpul D (Jarak Terdekat Berikutnya)

- Dari D ke G: jarak = 3.5 (D) + 10 (D ke G) = 13.5 (tidak perlu update karena 13.5 > 9.8)
- Tabel Jarak Tetap:
  - distances = {'A': 3, 'B': 0, 'C': 5.6, 'D': 3.5, 'E': 2.8, 'F': ∞, 'G': 9.8}

Iterasi 5: Kunjungi Simpul C (Jarak Terdekat Berikutnya)

- Dari C ke F: jarak = 5.6 (C) + 3.5 (C ke F) = 9.1
- Update Tabel Jarak:
  - distances = {'A': 3, 'B': 0, 'C': 5.6, 'D': 3.5, 'E': 2.8, 'F': 9.1, 'G': 9.8}

Iterasi 6: Kunjungi Simpul F

- Simpul F adalah simpul tujuan, proses selesai.

Hasil Perhitungan:

Jalur terpendek dari B ke F melalui algoritma Dijkstra adalah **B → E → C → F** dengan **total jarak 9.1**. Jalur ini dipilih melalui iterasi dengan pembaruan jarak di setiap langkah berdasarkan simpul yang belum dikunjungi dengan jarak minimum.

Contoh Program Python

```
graph = {
  "A": {"B": 3, "C": 3},
  "B": {"A": 3, "D": 3.5, "E": 2.8},
  "C": {"A": 3, "E": 2.8, "F": 3.5},
  "D": {"B": 3.5, "E": 3.1, "G": 10},
  "E": {"B": 2.8, "C": 2.8, "D": 3.1, "G": 7},
  "F": {"G": 2.5, "C": 3.5},
  "G": {"F": 2.5, "E": 7, "D": 10},
}

def dijkstra(graph, start, end):
  distances = {node: float('inf') for node in graph}
  distances[start] = 0
  visited = set()
  previous_nodes = {node: None for node in graph}
  while len(visited) < len(graph):
    current_node = None
    current_distance = float('inf')
    for node in distances:
      if node not in visited and distances[node] < current_distance:
        current_distance = distances[node]
        current_node = node
    if current_node is None:
      break
    visited.add(current_node)

    for neighbor, weight in graph[current_node].items():
```

```
            distance = current_distance + weight

            if distance < distances[neighbor]:
                distances[neighbor] = distance
                previous_nodes[neighbor] = current_node
    path = []
    current = end
    while current is not None:
        path.append(current)
        current = previous_nodes[current]
    path = path[::-1]
    return distances[end], path

shortest_distance, shortest_path = dijkstra(graph, "B", "F")

# Output hasil
print(f"Jarak terpendek dari B ke F: {shortest_distance}")
print(f"Jalur terpendek dari B ke F: {' -> '.join(shortest_path)}")
```

Hasil running program:

```
PS C:\Users\faith> & C:/Users/faith/AppData/Local/Programs/Python/Python310/python.exe c:/weight_cow/YOLO_weightCOW/jarak7simpul.py
Jarak terpendek dari B ke F: 9.1
Jalur terpendek dari B ke F: B -> E -> C -> F
```

Tugas Masing-masing individu

Soal:

1. Temukan jalur terpendek dari titik B ke G menggunakan algoritma Dijkstra dengan graf yang sama di atas.
2. Temukan **semua jalur terpendek** dari titik B menggunakan algoritma Dijkstra dengan graf yang sama di atas.

Catatan:

3. Kerjakan perhitungan manual di kertas
4. Buat program untuk di demokan