**Pseudocode untuk Program C++ DFS**

**Inisialisasi Graf**

**1. Inisialisasi Kelas Graph:**

- Buat `adjList` sebagai peta untuk menyimpan daftar adjacency.

**2. DFS Utilitas:**

- Fungsi rekursif untuk melakukan pencarian DFS dan menemukan jalur terpendek.

**Menambahkan Jalan Antar Kota**

**3. Menambahkan Jalan (Edge):**

- Tambahkan edge dari `src` ke `dest` dalam `adjList`.

- Tambahkan edge dari `dest` ke `src` (graf tak berarah).

**Menemukan Jalur Terpendek**

**4. Depth-First Search (DFS):**

- Inisialisasi `visited` sebagai peta untuk melacak vertex yang dikunjungi.

- Inisialisasi `path` sebagai vektor untuk menyimpan jalur saat ini.

- Inisialisasi `shortestPath` sebagai vektor untuk menyimpan jalur terpendek.

- Panggil fungsi DFS utilitas untuk mencari jalur terpendek dari `start` ke `goal`.

**Fungsi Utama**

**5. Fungsi utama:**

- Buat objek graf.

- Tambahkan semua edge dari data yang diberikan.

- Tentukan simpul awal dan simpul tujuan.

- Panggil fungsi DFS untuk menemukan jalur terpendek dari `start` ke `goal`.

- Cetak jalur terpendek atau pesan jika jalur tidak ditemukan.

**Pseudocode Detil**

Kelas Graph

Pribadi:

adjList: Peta untuk menyimpan daftar adjacency

Fungsi DFSUtil(current, goal, visited, path, shortestPath)

Tambahkan current ke dalam path

Set visited[current] ke true

Jika current sama dengan goal

Jika shortestPath kosong atau ukuran path lebih kecil dari shortestPath

Set shortestPath ke path

Jika tidak

Untuk setiap neighbor dalam adjList[current]

Jika neighbor belum dikunjungi

Panggil DFSUtil(neighbor, goal, visited, path, shortestPath)

Hapus current dari path

Set visited[current] ke false

Publik:

Fungsi addEdge(src, dest)

Tambahkan dest ke adjList[src]

Tambahkan src ke adjList[dest] (graf tak berarah)

Fungsi DFS(start, goal)

Inisialisasi visited sebagai peta

Inisialisasi path sebagai vektor

Inisialisasi shortestPath sebagai vektor

Panggil DFSUtil(start, goal, visited, path, shortestPath)

Kembalikan shortestPath

Fungsi utama

Buat objek Graph g

Definisikan edges sebagai pasangan nama vertex

edges = [

("Gudang", "A"), ("Gudang", "S"), ("Gudang", "M"), ("Gudang", "B"),

("M", "D"), ("M", "C"), ("M", "B"), ("D", "C"), ("C", "B"), ("C", "E"),

("A", "B"), ("A", "N"), ("S", "A"), ("S", "R"), ("F", "N"), ("F", "A"),

("R", "H"), ("R", "I"), ("I", "H"), ("H", "G"), ("H", "S"), ("G", "F"),

("G", "S"), ("F", "S"), ("N", "B"), ("N", "Q"), ("N", "L"), ("F", "L"),

("B", "E"), ("B", "O"), ("B", "Q"), ("E", "O"), ("Q", "O"), ("Q", "K"),

("O", "K"), ("L", "Q"), ("L", "K"), ("L", "P"), ("P", "K"), ("P", "J"),

("K", "J")

]

Untuk setiap edge dalam edges

Panggil g.addEdge(edge.first, edge.second)

Definisikan start sebagai "Gudang"

Definisikan goal sebagai "J"

Panggil g.DFS(start, goal) dan simpan hasilnya dalam shortestPath

Jika shortestPath tidak kosong

Cetak "Rute terpendek dari" start "ke" goal ":"

Untuk setiap vertex dalam shortestPath

Cetak vertex dan panah ("->") jika itu bukan vertex terakhir

Jika tidak

Cetak "Tidak ada jalur dari" start "ke" goal

Kembalikan 0