**Pseudocode untuk Program C++ Djikstra**

**Inisialisasi Graf**

**1. Inisialisasi CityGraph dengan jumlah simpul tertentu:**

- Buat daftar adjacency `adjList` dengan ukuran `vertices`.

- Buat daftar nama vertex `vertexNames` dengan ukuran `vertices`.

- Buat pemetaan `vertexIndexMap` untuk memetakan nama vertex ke indeks masing-masing.

**Menetapkan Nama Vertex**

**2. Tetapkan nama-nama vertex:**

- Jika ukuran daftar `names` tidak sama dengan jumlah vertex, cetak kesalahan dan keluar.

- Tetapkan `names` ke `vertexNames`.

- Isi `vertexIndexMap` dengan nama-nama vertex dan indeksnya masing-masing.

**Menambahkan Jalan Antar Kota**

**3. Tambahkan jalan (edges) antara vertex dengan waktu tertentu (bobot):**

- Konversi nama vertex `u` dan `v` ke indeksnya `uIndex` dan `vIndex`.

- Tambahkan edge dari `uIndex` ke `vIndex` dengan bobot yang ditentukan `time`.

- Tambahkan edge dari `vIndex` ke `uIndex` dengan bobot yang sama (graf tak berarah).

**Menemukan Jalur Terpendek**

**4. Temukan jalur terpendek antara dua vertex menggunakan algoritma Dijkstra:**

- Inisialisasi array jarak `dist` untuk menyimpan jarak terpendek dari simpul awal ke semua simpul lainnya.

- Inisialisasi priority queue `pq` untuk menyimpan dan mengambil vertex dengan jarak minimum.

- Set jarak dari simpul awal ke 0 dan dorong ke dalam priority queue.

- Selama priority queue tidak kosong:

- Ekstrak vertex `u` dengan jarak minimum dari priority queue.

- Untuk setiap tetangga `v` dari vertex `u`, jika ditemukan jalur yang lebih pendek ke `v` melalui `u`, perbarui jaraknya dan dorong `v` ke dalam priority queue.

- Setelah semua vertex diproses, rekonstruksi jalur terpendek dari simpul awal ke simpul akhir menggunakan stack.

**Fungsi Utama**

5. **Fungsi utama:**

- Buat graf dengan 20 vertex.

- Tetapkan nama-nama vertex.

- Tambahkan edges (jalan) antara vertex dengan bobot yang ditentukan.

- Tentukan simpul awal dan akhir.

- Panggil fungsi untuk menemukan jalur terpendek dari simpul awal ke simpul akhir.

**Pseudocode Detil**

Kelas CityGraph

Pribadi:

adjList: Daftar daftar adjacency untuk menyimpan graf

vertexNames: Daftar untuk menyimpan nama vertex

vertexIndexMap: Peta untuk menyimpan pemetaan nama vertex ke indeks

numVertices: Jumlah vertex dalam graf

Publik:

Fungsi CityGraph(vertices)

Inisialisasi adjList dengan ukuran vertices

Inisialisasi vertexNames dengan ukuran vertices

Set numVertices ke vertices

Fungsi setVertexNames(names)

Jika ukuran names tidak sama dengan numVertices

Cetak "Jumlah nama vertex tidak sesuai dengan jumlah vertex dalam graf!"

Keluar

Set vertexNames ke names

Untuk setiap nama dalam names

Peta nama ke indeksnya dalam vertexIndexMap

Fungsi addRoad(u, v, time)

uIndex = Dapatkan indeks dari vertex u menggunakan vertexIndexMap

vIndex = Dapatkan indeks dari vertex v menggunakan vertexIndexMap

Tambahkan (vIndex, time) ke adjList[uIndex]

Tambahkan (uIndex, time) ke adjList[vIndex]

Fungsi vertexNameToIndex(name)

Jika name ada dalam vertexIndexMap

Kembalikan indeks dari name

Jika tidak

Cetak "Nama vertex tidak ada!"

Keluar

Fungsi findShortestPath(startVertex, endVertex)

startIdx = Dapatkan indeks dari startVertex menggunakan vertexIndexMap

endIdx = Dapatkan indeks dari endVertex menggunakan vertexIndexMap

Inisialisasi array dist dengan ukuran numVertices, set semua nilai ke infinity

Set dist[startIdx] ke 0

Inisialisasi priority queue pq

Dorong (0, startIdx) ke dalam pq

Sementara pq tidak kosong

u = Ekstrak vertex dengan jarak minimum dari pq

Jika jarak u dalam dist kurang dari jarak dalam pq, lanjutkan

Untuk setiap (v, weight) dalam adjList[u]

Jika dist[u] + weight < dist[v]

Perbarui dist[v] ke dist[u] + weight

Dorong (dist[v], v) ke dalam pq

Jika dist[endIdx] masih infinity

Cetak "Tidak ada jalur dari startVertex ke endVertex!"

Keluar

Inisialisasi stack path

Set current ke endIdx

Sementara current tidak sama dengan startIdx

Dorong current ke dalam path

Set current ke vertex sebelumnya dalam jalur terpendek

Dorong startIdx ke dalam path

Cetak "Jalur terpendek:"

Sementara path tidak kosong

Pop vertex dari path dan cetak namanya

Fungsi utama

Buat CityGraph g dengan 20 vertex

Definisikan names sebagai [ 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'Z' ]

Panggil g.setVertexNames(names)

Panggil g.addRoad('Z', 'A', 6)

Panggil g.addRoad('Z', 'S', 11)

Panggil g.addRoad('Z', 'M', 7)

Panggil g.addRoad('Z', 'B', 8)

Panggil g.addRoad('M', 'D', 8)

Panggil g.addRoad('M', 'C', 5)

Panggil g.addRoad('M', 'B', 8)

Panggil g.addRoad('D', 'C', 8)

Panggil g.addRoad('C', 'B', 8)

Panggil g.addRoad('C', 'E', 8)

Panggil g.addRoad('A', 'B', 10)

Panggil g.addRoad('A', 'N', 7)

Panggil g.addRoad('S', 'A', 6)

Panggil g.addRoad('S', 'R', 12)

Panggil g.addRoad('F', 'N', 7)

Panggil g.addRoad('F', 'A', 8)

Panggil g.addRoad('R', 'H', 16)

Panggil g.addRoad('R', 'I', 12)

Panggil g.addRoad('I', 'H', 18)

Panggil g.addRoad('H', 'G', 11)

Panggil g.addRoad('H', 'S', 8)

Panggil g.addRoad('G', 'F', 5)

Panggil g.addRoad('G', 'S', 8)

Panggil g.addRoad('F', 'S', 8)

Panggil g.addRoad('N', 'B', 8)

Panggil g.addRoad('N', 'Q', 6)

Panggil g.addRoad('N', 'L', 5)

Panggil g.addRoad('F', 'L', 7)

Panggil g.addRoad('B', 'E', 10)

Panggil g.addRoad('B', 'O', 10)

Panggil g.addRoad('B', 'Q', 9)

Panggil g.addRoad('E', 'O', 8)

Panggil g.addRoad('Q', 'O', 5)

Panggil g.addRoad('Q', 'K', 8)

Panggil g.addRoad('O', 'K', 8)

Panggil g.addRoad('L', 'Q', 4)

Panggil g.addRoad('L', 'K', 11)

Panggil g.addRoad('L', 'P', 9)

Panggil g.addRoad('P', 'K', 7)

Panggil g.addRoad('P', 'J', 10)

Panggil g.addRoad('K', 'J', 9)

Definisikan startVertex sebagai 'Z'

Definisikan endVertex sebagai 'J'

Panggil g.findShortestPath(startVertex, endVertex)

Kembalikan 0