LAPORAN TUGAS BESAR

STRUKTUR DATA

Laporan ini disusun untuk menyelesaikan tugas besar mata kuliah Struktur Data

Universitas
Telkom

Sistem Rekomendasi Rute
Antar Gedung Kampus

Disusun oleh:

Haafizd Alhabib Azwir	103022330089
Subhan Maulana Ahmad	103022300081

Deskripsi Masalah

Studi kasus ini berfokus pada pengembangan sistem rekomendasi rute antar gedung di lingkungan kampus. Masalah yang dihadapi adalah bagaimana membantu mahasiswa menemukan rute terbaik untuk berpindah antar gedung kampus yang terhubung melalui berbagai jalan setapak dengan jarak yang berbeda-beda. Sistem ini dibutuhkan untuk membantu mahasiswa mengoptimalkan waktu dan tenaga dalam perpindahan antar gedung.

Komponen Graf dalam Studi Kasus

- Vertex/Node: Gedung-gedung kampus
- Edge: Jalan setapak yang menghubungkan antar gedung
- Bobot Edge: Jarak dalam meter antara dua gedung
- Jenis Graf: Graf tidak berarah dan berbobot (undirected weighted graph)

Alasan Pemilihan Studi Kasus

Studi kasus ini dipilih karena memiliki karakteristik yang cocok atau ideal untuk struktur data graf. Selain mudah divisualisasikan dan dipahami, kasus ini merepresentasikan situasi nyata yang sering ditemui dalam kehidupan sehari-hari di dalam kampus. Implementasinya mencakup konsep-konsep dasar graf seperti vertex, edge, dan bobot, serta memungkinkan penerapan algoritma pencarian jalur.

Fitur yang Diimplementasikan

a. **Fitur Dasar:**
   - Penyimpanan struktur gedung dan rute menggunakan adjacency list
   - Penambahan rute baru antar gedung
   - Penampilan seluruh rute yang tersedia

b. Algoritma Pencarian:
   - Implementasi algoritma Dijkstra untuk mencari rute terpendek
   - Menampilkan jarak dan urutan gedung yang harus dilalui

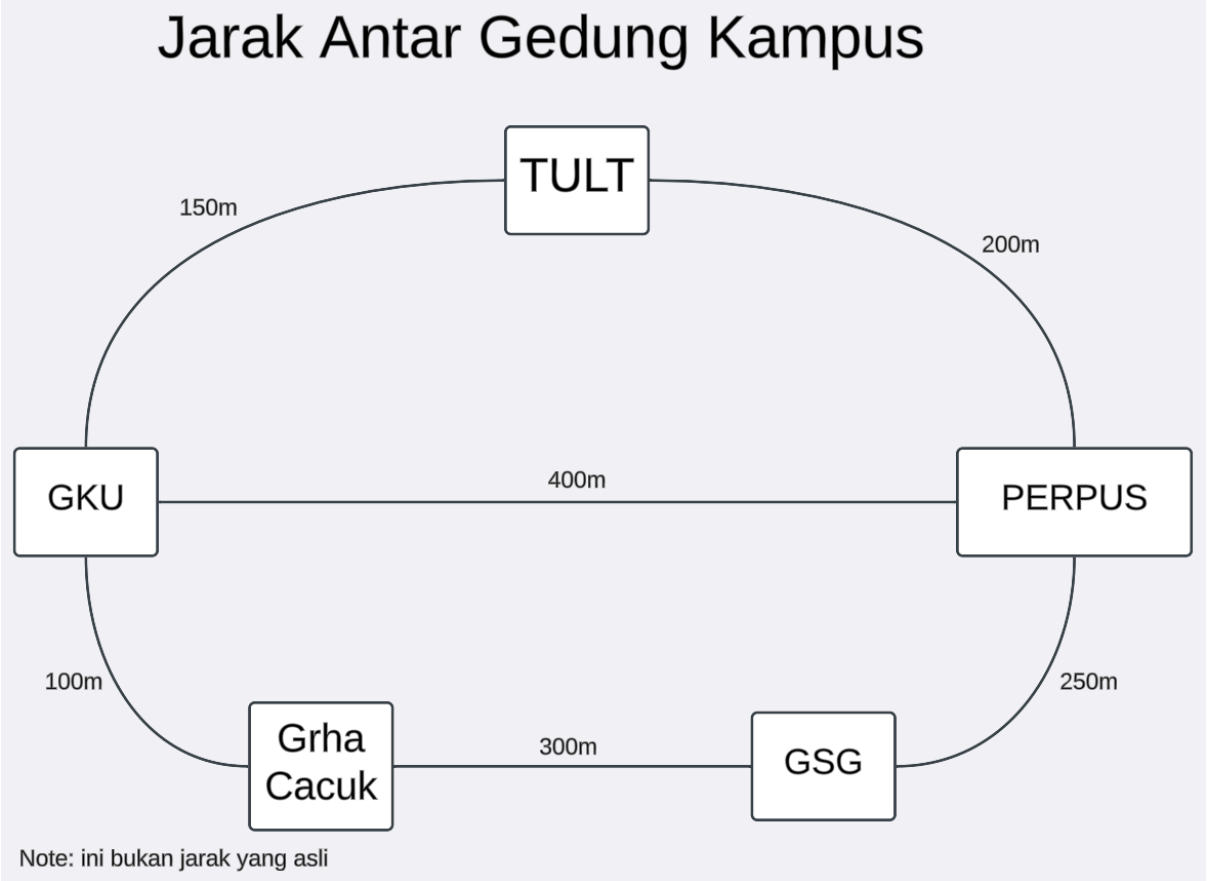
c. Fitur Tambahan:
   - Menghitung jumlah rute langsung dari suatu gedung
   - Mengidentifikasi gedung dengan konektivitas tertinggi

Batasan Implementasi Saat Ini

Implementasi saat ini memiliki beberapa batasan berikut ini:

- Hanya memperhitungkan jarak sebagai bobot
- Mengasumsikan semua rute dapat dilalui dua arah
- Terbatas pada jumlah gedung yang telah ditentukan
- Tidak memperhitungkan faktor dinamis seperti kepadatan rute

Ilustrasi Graf



Desain Algoritma

1. Implementasi menggunakan adjacency list untuk menyimpan graf
2. Menggunakan Algoritma Dijkstra untuk mencari rute terpendek
3. Fungsi tambahan untuk:
    - Menghitung jumlah rute langsung dari suatu gedung
    - Mencari gedung dengan konektivitas tertinggi

Program yang diimplementasikan mencakup:

1. Struktur dasar graf menggunakan adjacency list
2. Fungsi untuk menambah edge dan menampilkan rute
3. Algoritma Dijkstra untuk pencarian rute terpendek
4. Dua fungsi tambahan sesuai kebutuhan
5. Program utama untuk demonstrasi

Pseudocode

Function createGraph(Input/output G : Graph)

```
FUNCTION createGraph(input/output G : Graph)
{ I.S.: G sembarang
  F.S.: Terbentuk graph G kosong dengan first(G) = NULL }
ALGORITMA
    first(G) ← NULL
```

Procedure addVertex(input/output G : Graph, input name : string)

```
PROCEDURE addVertex(input/output G : Graph, input name : string)
{ I.S.: G terdefinisi, name terdefinisi dan unik (belum ada di graph)
  F.S.: Vertex baru dengan info = name ditambahkan ke dalam graph G sebagai vertex terakhir }
KAMUS LOKAL
    newVertex : pointer to Vertex
    current : pointer to Vertex
ALGORITMA
    newVertex ← new Vertex
    name(newVertex) ← name
    firstEdge(newVertex) ← NULL
    next(newVertex) ← NULL
    visited(newVertex) ← false
    distance(newVertex) ← MAX_DIST
    parent(newVertex) ← ""

    if first(G) = NULL then
        first(G) ← newVertex
    else
        current ← first(G)
        while next(current) ≠ NULL do
            current ← next(current)
        next(current) ← newVertex
```

Function findVertex(input G : Graph, input name : string) ⟶ pointer to Vertex

```
FUNCTION findVertex(input G : Graph, input name : string) → pointer to Vertex
{ I.S. : G terdefinisi, name terdefinisi
 F.S. : Mengembalikan address vertex dengan info = name jika ditemukan,NULL jika tidak ditemukan }
KAMUS LOKAL
    current : pointer to Vertex
ALGORITMA
    current ← first(G)
    while current ≠ NULL do
        if name(current) = name then
            return current
        current ← next(current)
    return NULL
```

Procedure addEdge(input/output G : Graph, input src, dest : string, input weight : integer)

```
PROCEDURE addEdge(input/output G : Graph, input src,dest : string, input weight : integer)
{ I.S. : G terdefinisi, src dan dest adalah nama vertex yang sudah ada di graph, weight bernilai positif
  F.S. : Edge baru terbentuk yang menghubungkan vertex src dan dest dengan bobot weight. Karena graph tidak berarah, edge dibuat dua arah }
KAMUS LOKAL
    srcVertex, destVertex : pointer to Vertex
    newEdge : pointer to Edge
ALGORITMA
    srcVertex ← findVertex(G, src)
    destVertex ← findVertex(G, dest)

    if (srcVertex ≠ NULL) and (destVertex ≠ NULL) then
        { Membuat edge dari src ke dest }
        newEdge ← new Edge
        dest(newEdge) ← dest
        weight(newEdge) ← weight
        next(newEdge) ← firstEdge(srcVertex)
        firstEdge(srcVertex) ← newEdge

        { Membuat edge dari dest ke src }
        newEdge ← new Edge
        dest(newEdge) ← src
        weight(newEdge) ← weight
        next(newEdge) ← firstEdge(destVertex)
        firstEdge(destVertex) ← newEdge
```

Procedure findShortestPath(input/output G : Graph, input src, dest : string)

```
PROCEDURE findShortestPath(input/output G : Graph, input src,dest : string)
{ I.S.: G terdefinisi, src dan dest adalah nama vertex yang ada di graph
  F.S.: Menampilkan rute terpendek dari vertex src ke dest beserta total jaraknya. Jika tidak ada rute yang tersedia, menampilkan pesan error }
KAMUS LOKAL
  srcVertex, current, minVertex : pointer to Vertex
  edge : pointer to Edge
  minDist : integer
  path : array[1..100] of string
  pathSize : integer
ALGORITMA
  srcVertex ← findVertex(G, src)
  if srcVertex = NULL then
    return

  { Inisialisasi semua vertex }
  current ← first(G)
  while current ≠ NULL do
    visited(current) ← false
    distance(current) ← MAX_DIST
    parent(current) ← ""
    current ← next(current)

  distance(srcVertex) ← 0
```

Referensi

GeeksforGeeks, "Graph Data Structures: Adjacency List and Matrix", https://www.geeksforgeeks.org/graph-and-its-representations/

Adeel Javaid, “Understanding Dijkstra Algorithm”, Januari 2013, https://www.researchgate.net/publication/273264449_Understanding_Dijkstra_Algorithm#:~:text=Dijkstra)%20solves%20the%20problem%20of,single-source%20shortest%20paths%20problem.

Raj Shah, Graphs and Real-Life Applications, 20 Desember 2020, https://rajshah001.medium.com/graphs-and-real-life-application-28759b77b833

GeeksforGeeks, "Applications of Graph Theory in Real Life", https://www.geeksforgeeks.org/applications-of-graph-theory/