KASUS OPTIMASI JALUR TERPENDEK DARI HUB DEPOK KE HUB CIBINONG MENGGUNAKAN ALGORITMA DIJKSTRA

Hendri Karisma, S.Kom., M.T

Program Studi Teknik Informatika

Semester Ganjil Tahun Ajaran 2024/2025

Nama Kelompok :

Fabian Alfarizi: 251552010039

4 February 2025

Sekolah Tinggi Manajemen Informatika dan Komputer TAZKIA
Jl. Raya Dramaga Blok Radar Baru No.8, RT.03/RW.03, Margajaya, Kec. Bogor Bar., Kota Bogor, Jawa Barat 16116, Indonesia

PENDAHULUAN

Pada era perkembangan teknologi dan bisnis saat ini, khususnya di bidang logistik, proses pengiriman barang dituntut agar dapat berjalan secara cepat dan efisien. Salah satu permasalahan yang sering muncul adalah penentuan jalur pengiriman yang paling pendek agar waktu tempuh dan biaya operasional dapat diminimalkan. Permasalahan ini dapat diselesaikan dengan pendekatan optimasi menggunakan konsep matematika dan algoritma.

Laporan ini disusun untuk **memenuhi tugas Ujian Akhir Semester (UAS) Semester 1** pada mata kuliah Optimasi / Matematika Diskrit / Teori Graf. Melalui tugas ini, mahasiswa diharapkan mampu memahami konsep graf serta mengimplementasikan algoritma pencarian jalur terpendek ke dalam sebuah program komputer.

Dalam tugas ini dibahas studi kasus optimasi jalur terpendek dari **Hub Depok** menuju **Hub Cibinong**. Permasalahan tersebut dimodelkan menggunakan graf berbobot, di mana setiap simpul (node) merepresentasikan lokasi hub, sedangkan sisi (edge) merepresentasikan jalur antar lokasi dengan bobot berupa jarak tempuh dalam satuan kilometer.

Algoritma yang digunakan untuk menyelesaikan permasalahan ini adalah **algoritma Dijkstra**, yaitu salah satu algoritma dalam teori graf yang digunakan untuk mencari jalur terpendek dari satu titik ke titik lainnya dengan bobot sisi bernilai non-negatif. Algoritma ini dipilih karena memiliki konsep yang jelas, mudah dipahami oleh mahasiswa semester awal, serta banyak digunakan dalam permasalahan nyata seperti navigasi dan logistik.

Diharapkan melalui laporan ini, mahasiswa tidak hanya memahami teori algoritma Dijkstra, tetapi juga mampu mengimplementasikannya dalam bentuk program serta menganalisis hasil yang diperoleh.

METHOD

2.1 Permodelan Graf

Permasalahan optimasi jalur pengiriman dari Hub Depok ke Hub Cibinong dimodelkan menggunakan graf berbobot. Graf digunakan karena mampu merepresentasikan hubungan antar lokasi beserta jarak tempuhnya secara sistematis.

Pada permodelan ini:

- Graf bersifat **tidak berarah (undirected graph)**, karena jalur antar lokasi dapat dilalui dua arah.
- Setiap **simpul (node)** merepresentasikan satu lokasi hub atau titik distribusi.
- Setiap **sisi (edge)** merepresentasikan jalur penghubung antar lokasi.
- **Bobot sisi (weight)** menyatakan jarak tempuh antar lokasi dalam satuan kilometer.

Node yang digunakan dalam graf ini adalah:

1. Hub Depok
2. Citayam
3. Bojong Gede
4. Hub Cibinong

Sedangkan hubungan antar node dapat dijelaskan sebagai berikut:

- Hub Depok terhubung dengan Citayam sejauh 7 km
- Hub Depok terhubung dengan Bojong Gede sejauh 10 km
- Citayam terhubung dengan Hub Cibinong sejauh 6 km
- Bojong Gede terhubung dengan Hub Cibinong sejauh 8 km

Dengan permodelan tersebut, graf dapat divisualisasikan sebagai jaringan rute pengiriman yang memiliki beberapa alternatif jalur. Tujuan dari algoritma optimasi adalah menentukan jalur dengan total bobot (jarak) paling kecil dari node awal ke node tujuan.

2.2 Algoritma Dijkstra

Algoritma Dijkstra bekerja dengan cara mencari jarak minimum secara bertahap dari node awal ke node lainnya. Langkah-langkah algoritma Dijkstra adalah sebagai berikut:

1. Menginisialisasi jarak semua node dengan nilai sangat besar (tak hingga)
2. Menentukan jarak node awal bernilai 0
3. Memilih node dengan jarak terkecil yang belum dikunjungi
4. Memperbarui jarak node tetangga jika ditemukan jalur yang lebih pendek
5. Menandai node yang sudah diproses
6. Mengulangi langkah hingga seluruh node diproses

EXPERIMENT AND RESULT

3.1 Skenario Pengujian

Pengujian dilakukan dengan skenario sebagai berikut:

Titik awal: Hub Depok

Titik tujuan: Hub Cibinong

Metode: Algoritma Dijkstra

Bahasa pemrograman: Go (Golang)

3.2 Hasil Pengujian

Berdasarkan hasil eksekusi program, diperoleh jalur terpendek sebagai berikut:

Hub Depok → Citayam → Hub Cibinong

Total jarak tempuh: 7 km + 6 km = 13 km

Hasil tersebut menunjukkan bahwa jalur melalui Citayam lebih optimal dibandingkan jalur melalui Bojong Gede yang memiliki total jarak 18 km.

DISCUSSION

Dari hasil pengujian dapat dilihat bahwa algoritma Dijkstra mampu menentukan jalur terpendek secara tepat sesuai dengan bobot jarak yang diberikan. Algoritma ini mengevaluasi setiap kemungkinan jalur dan memilih jalur dengan total jarak paling kecil.

Implementasi graf menggunakan adjacency list memudahkan dalam penambahan node dan edge baru. Namun, pada program ini algoritma hanya menampilkan jarak terpendek tanpa menampilkan urutan jalur secara detail.

KESIMPULAN

Berdasarkan hasil pembahasan dan pengujian yang telah dilakukan, dapat disimpulkan bahwa:

1. Algoritma Dijkstra dapat digunakan untuk menyelesaikan permasalahan optimasi jalur terpendek pada sistem logistik.
2. Jalur terpendek dari Hub Depok ke Hub Cibinong adalah melalui Citayam dengan total jarak 13 km.
3. Pemodelan graf berbobot efektif untuk merepresentasikan permasalahan rute pengiriman.

PENJELASAN PROGRAM

6.1 Struktur Program

Program ditulis menggunakan bahasa pemrograman Go dengan memanfaatkan struktur data graf. Program terdiri dari beberapa bagian utama, yaitu deklarasi struktur data, fungsi algoritma Dijkstra, dan fungsi utama (main).

6.2 Penjelasan Kode

- Edge digunakan untuk menyimpan informasi tujuan node dan jarak
- Graph merupakan representasi graf dalam bentuk adjacency list
- Fungsi Dijkstra digunakan untuk menghitung jarak terpendek dari node awal ke seluruh node lainnya
- Fungsi main berisi data graf, pemanggilan algoritma, dan penampilan hasil

Saran dan Future Work

Beberapa saran untuk pengembangan selanjutnya antara lain:

1. Menambahkan fitur penelusuran jalur (path tracking) agar rute dapat ditampilkan secara lengkap
2. Menggunakan priority queue (heap) agar algoritma lebih efisien untuk graf berukuran besar
3. Menambahkan parameter lain seperti waktu tempuh atau biaya pengiriman
4. Mengembangkan sistem ke dalam aplikasi berbasis web atau mobile

Daftar Pustaka

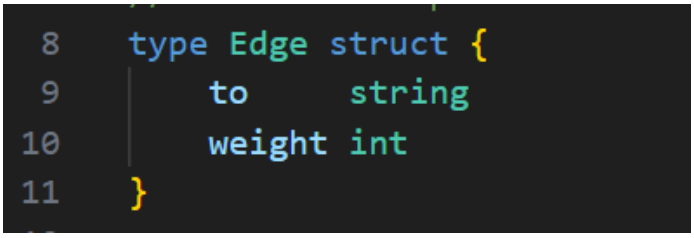
1. Munir, Rinaldi. Matematika Diskrit. Bandung: Informatika, 2012.
2. Suyanto. Algoritma dan Pemrograman. Yogyakarta: Andi Offset, 2015.
3. Rosa A.S & Shalahuddin, M. Rekayasa Perangkat Lunak. Bandung: Informatika, 2018.
4. Modul Mata Kuliah Optimasi dan Teori Graf, Program Studi Informatika.

PENJELASAN KODE PROGRAM

6.3 Package dan Import

- package main menandakan bahwa program ini merupakan program utama yang dapat dieksekusi.
- fmt digunakan untuk menampilkan output ke layar.
- math digunakan untuk mengambil konstanta MaxInt32 sebagai nilai awal jarak tak hingga.

6.2 Struktur Data Edge



Struktur Edge digunakan untuk merepresentasikan sisi pada graf, yang terdiri dari:

- to: node tujuan
- weight: bobot sisi berupa jarak antar node

6.3 Struktur Data Graph

Graph direpresentasikan dalam bentuk adjacency list, di mana:

- string sebagai key menunjukkan nama node
- []Edge berisi daftar edge yang terhubung dengan node tersebut

Struktur ini dipilih karena efisien dalam penggunaan memori dan mudah dikembangkan.

6.4 Fungsi Dijkstra

```
16  func Dijkstra(graph Graph, start string) map[string]int {
```

Fungsi ini digunakan untuk menghitung jarak terpendek dari node awal ke seluruh node lainnya.

6.4.1 Inisialisasi

```
17      dist := make(map[string]int)
18      visited := make(map[string]bool)
19
20      for node := range graph {
21          dist[node] = math.MaxInt32
22      }
23      dist[start] = 0
```

- dist menyimpan jarak terpendek dari node awal ke setiap node
- visited menandai node yang sudah diproses
- Semua jarak diinisialisasi dengan nilai maksimum, kecuali node awal yang bernilai 0

6.4.2 Pemilihan Node dengan Jarak Minimum

```
25      for {
26          current := ""
27          minDist := math.MaxInt32
28
29          for node, d := range dist {
30              if !visited[node] && d < minDist {
31                  minDist = d
32                  current = node
33              }
34          }
```

Bagian ini digunakan untuk memilih node yang memiliki jarak paling kecil dan belum dikunjungi.

6.4.3 Relaksasi Edge

```
39
40          visited[current] = true
41
42          for _, edge := range graph[current] {
43              if dist[current]+edge.weight < dist[edge.to] {
44                  dist[edge.to] = dist[current] + edge.weight
45              }
46          }
```

Proses relaksasi dilakukan untuk memperbarui jarak node tetangga jika ditemukan jalur yang lebih pendek melalui node saat ini.

6.4.4 Hasil Fungsi

```
49      return dist
```

Fungsi mengembalikan jarak terpendek dari node awal ke seluruh node dalam graf.

6.5 Fungsi Main

```
7   func main() {
```

Fungsi utama yang berisi data graf dan pemanggilan algoritma Dijkstra.

6.4.5 Inisialisasi Graf

```
8       graph := Graph{
9           "Hub Depok": {
10              {"Citayam", 7},
11              {"Bojong Gede", 10},
12          },
13          "Citayam": {
14              {"Hub Depok", 7},
15              {"Hub Cibinong", 6},
16          },
17          "Bojong Gede": {
18              {"Hub Depok", 10},
19              {"Hub Cibinong", 8},
20          },
21          "Hub Cibinong": {
22              {"Citayam", 6},
23              {"Bojong Gede", 8},
24          },
25      }
```

Graf didefinisikan sesuai dengan permodelan jalur antar hub beserta jaraknya.

6.4.6 Penentuan Node Awal dan Tujuan

```
27      start := "Hub Depok"
28      end := "Hub Cibinong"
29
```

Node awal dan node tujuan ditentukan sesuai dengan studi kasus.

6.4.7 Pemanggilan Algoritma dan Output

```
29
30      dist := Dijkstra(graph, start)
31
32      fmt.Println("=== Optimasi Jalur Terpendek (Graph) ===")
33      fmt.Println("Dari :", start)
34      fmt.Println("Ke   :", end)
35      fmt.Println("Jarak Terpendek:", dist[end], "km")
```

Program memanggil fungsi Dijkstra dan menampilkan hasil jarak terpendek ke layar.