LAPORAN JURNAL KOMPUTA

Penentuan Jalur Terpendek Antar Desa Menggunakan Algoritma Dijkstra Berbasis Graph

Muhammad Ikhsan Azis[1], Muhammad Dimas Putra Sudrajat[2]

Teknik Informatika, STMIK TAZKIA Bogor

---

1. Pendahuluan

Perkembangan infrastruktur transportasi di wilayah pedesaan memiliki peran penting dalam mendukung mobilitas masyarakat dan kelancaran distribusi barang serta jasa. Ketersediaan jalur yang efisien antar desa menjadi faktor utama dalam menghemat waktu tempuh, biaya perjalanan, dan meningkatkan aksesibilitas antar wilayah. Namun, dalam praktiknya sering terdapat berbagai alternatif jalur yang dapat dilalui, sehingga diperlukan suatu metode untuk menentukan jalur yang paling optimal.

Permasalahan penentuan jalur terpendek dapat dimodelkan sebagai permasalahan optimasi yang melibatkan hubungan antar lokasi dan jarak antar jalur. Dalam ilmu komputer dan matematika diskrit, permasalahan ini dapat direpresentasikan menggunakan struktur data graph, di mana setiap desa direpresentasikan sebagai simpul (node) dan jalan penghubung antar desa direpresentasikan sebagai sisi (edge) yang memiliki bobot berupa jarak atau waktu tempuh.

Salah satu algoritma yang umum digunakan untuk menyelesaikan permasalahan jalur terpendek adalah Algoritma Dijkstra. Algoritma ini mampu menentukan jarak minimum dari satu titik awal ke seluruh titik lain dalam graph berbobot dengan nilai bobot non-negatif. Dengan keunggulan tersebut, Algoritma Dijkstra banyak diterapkan dalam berbagai sistem navigasi dan perencanaan rute.

Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem penentuan jalur terpendek antar desa menggunakan Algoritma Dijkstra berbasis graph. Sistem yang dibangun diharapkan dapat memberikan solusi jalur yang optimal serta menjadi contoh penerapan konsep graph dan algoritma pencarian jalur terpendek dalam pemrograman menggunakan bahasa Go (Golang).

2. Metodologi

Metodologi penelitian yang digunakan dalam penentuan jalur terpendek antar desa ini terdiri dari beberapa tahapan, mulai dari pemodelan permasalahan hingga implementasi algoritma. Pendekatan yang digunakan bertujuan untuk memperoleh solusi jalur paling optimal berdasarkan jarak minimum.

2.1 Pemodelan Graph

Permasalahan jalur antar desa dimodelkan menggunakan struktur data graph berbobot. Setiap desa direpresentasikan sebagai simpul (node), sedangkan jalan yang menghubungkan antar desa direpresentasikan sebagai sisi (edge). Bobot pada setiap sisi menunjukkan jarak tempuh antar desa yang dinyatakan dalam satuan kilometer.

Graph yang digunakan dalam penelitian ini merupakan graph tak berarah (undirected graph), karena jalur antar desa diasumsikan dapat dilalui dua arah. Secara matematis, graph dinyatakan sebagai $G = (V, E)$, dengan V sebagai himpunan simpul dan E sebagai himpunan sisi berbobot.

2.2 Algoritma Dijkstra

Algoritma Dijkstra digunakan untuk menentukan jalur terpendek dari satu desa asal ke desa tujuan. Algoritma ini bekerja dengan cara menghitung jarak minimum secara bertahap dari titik awal ke seluruh simpul lain dengan memilih simpul yang memiliki jarak sementara paling kecil.

Langkah-langkah Algoritma Dijkstra yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Menentukan desa asal sebagai titik awal.
2. Menginisialisasi jarak seluruh simpul dengan nilai tak hingga, kecuali simpul awal yang bernilai nol.
3. Memilih simpul dengan jarak terkecil yang belum dikunjungi.
4. Memperbarui jarak simpul tetangga jika ditemukan jarak yang lebih pendek.
5. Mengulangi proses hingga simpul tujuan ditemukan atau seluruh simpul telah dikunjungi.

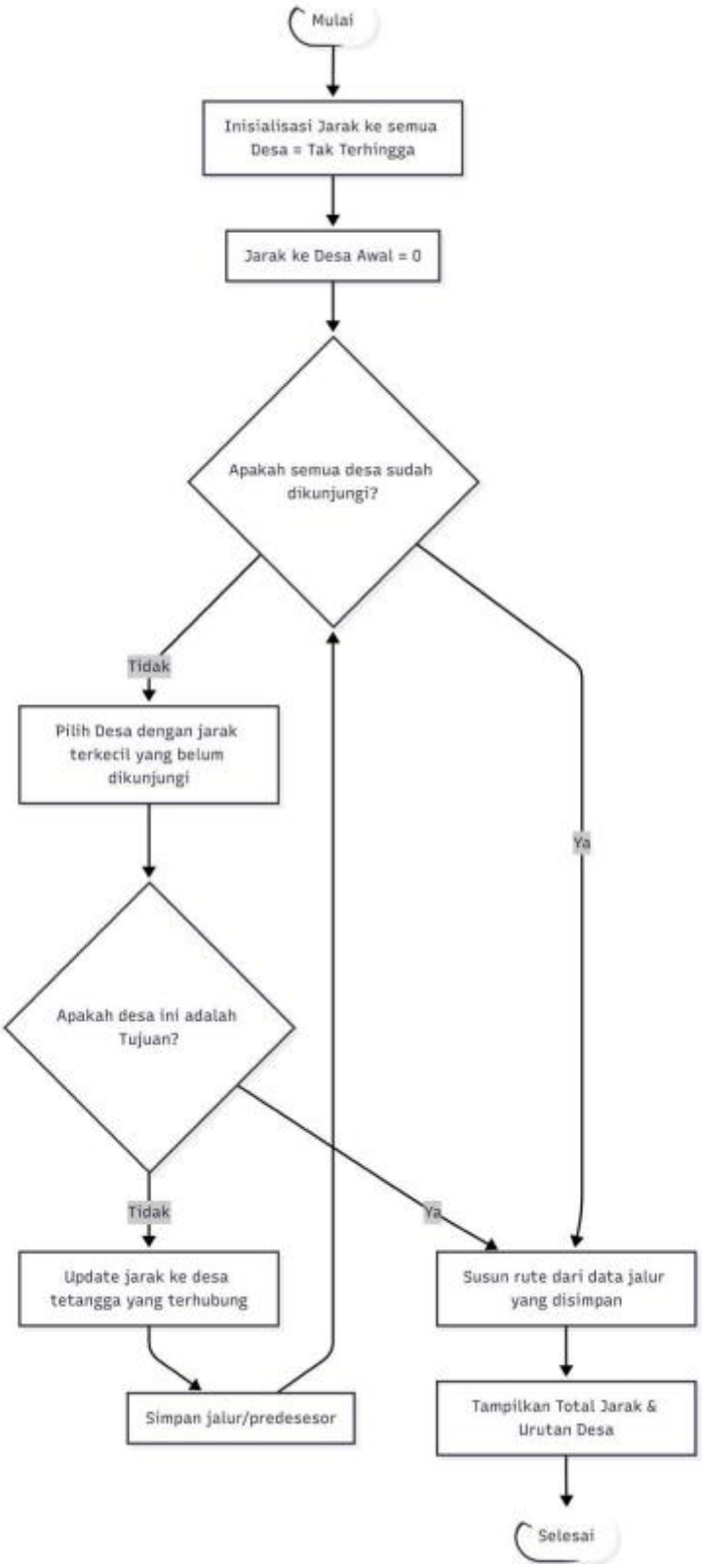
Algoritma ini dipilih karena efisien dalam menangani graph berbobot dengan bobot non-negatif dan memiliki kompleksitas waktu yang relatif baik.

2.3 Implementasi Sistem

Implementasi algoritma dilakukan menggunakan bahasa pemrograman Go (Golang). Struktur data graph direpresentasikan dalam bentuk adjacency list untuk menghemat penggunaan memori dan meningkatkan efisiensi pemrosesan.

Program menerima input berupa data desa, jarak antar desa, desa asal, dan desa tujuan. Selanjutnya, sistem akan memproses data tersebut menggunakan Algoritma Dijkstra dan menghasilkan output berupa jalur terpendek serta total jarak tempuh.

2.4 Flowchart



3. Experiment and Result

3.1 Skenario Pengujian

Pengujian sistem dilakukan untuk mengetahui kemampuan Algoritma Dijkstra dalam menentukan jalur terpendek antar desa berdasarkan jarak minimum. Pengujian menggunakan beberapa data desa yang direpresentasikan dalam bentuk graph berbobot. Setiap pengujian dilakukan dengan menentukan satu desa sebagai titik awal dan satu desa sebagai titik tujuan.

Data jarak antar desa dimasukkan ke dalam sistem secara manual sesuai dengan skenario yang telah ditentukan. Sistem kemudian dijalankan untuk menghitung jalur terpendek berdasarkan bobot jarak yang tersedia.

3.2 Data Uji

Titik Awal	Titik Tujuan	Jarak(KM)
Desa A	Desa B	4
Desa A	Desa C	2
Desa B	Desa C	5
Desa B	Desa D	10
Desa C	Desa E	3
Desa E	Desa D	4

3.3 Proses Pengujian

3.3.1 Inisialisasi Graf

Data desa dan jarak dimasukkan ke dalam struktur *Graph* yang berisi node dan bobot jarak antar node.

3.3.2 Inisialisasi Jarak Awal

Semua jarak node diatur ke nilai maksimum (*math.MaxInt32*) kecuali node awal yang diatur bernilai 0.

3.3.3 Proses Algoritma Dijkstra

- Sistem memilih node dengan jarak terkecil yang belum dikunjungi.
- Jarak ke node tetangga diperbarui jika ditemukan jarak yang lebih kecil.
- Node yang telah diproses ditandai sebagai visited.
- Proses berhenti ketika node tujuan ditemukan atau tidak ada node yang dapat dijangkau.

3.3.4 Penyusunan Jalur Terpendek

Jalur terpendek disusun menggunakan struktur *prev* dengan menelusuri kembali node dari tujuan ke titik awal.

3.3.5 Menampilkan Hasil

Sistem menampilkan:

- Titik awal dan tujuan
- Total jarak terpendek
- Urutan desa yang dilalui

3.4 Souce Code

```go
package main

import (
    "fmt"
    "math"
)

// Edge merepresentasikan hubungan antar desa dan jaraknya
type Edge struct {
    Node   string
    Weight int
}

// Graph adalah kumpulan data desa dan jalannya
type Graph map[string][]Edge

// Dijkstra adalah fungsi utama untuk mencari jalur terpendek
func Dijkstra(graph Graph, start string, end string) (int, []string) {
    // 1. Inisialisasi jarak awal dengan nilai tak terhingga (infinity)
    distances := make(map[string]int)
    for node := range graph {
        distances[node] = math.MaxInt32
    }
    distances[start] = 0

    // 2. Simpan jejak jalur yang dilewati
    prev := make(map[string]string)
    visited := make(map[string]bool)

    for len(visited) < len(graph) {
        // Cari node dengan jarak terkecil yang belum dikunjungi
        currNode := ""
        minDist := math.MaxInt32

        for node, dist := range distances {
            if !visited[node] && dist < minDist {
                minDist = dist
                currNode = node
            }
```

```
        for node, dist := range distances {
            if !visited[node] && dist < minDist {
                minDist = dist
                currNode = node
            }
        }

        // Jika tidak ada node lagi yang bisa dijangkau
        if currNode == "" {
            break
        }

        visited[currNode] = true

        // Jika sudah sampai tujuan, kita bisa berhenti (opsional, tapi lebih efisien)
        if currNode == end {
            break
        }

        // Update jarak ke tetangga-tetangganya
        for _, edge := range graph[currNode] {
            newDist := distances[currNode] + edge.Weight
            if newDist < distances[edge.Node] {
                distances[edge.Node] = newDist
                prev[edge.Node] = currNode
            }
        }
    }

    // 3. Menyusun jalur dari hasil 'prev'
    path := []string{}
    curr := end
    for curr != "" {
        path = append([]string{curr}, path...)
```

```
69          curr = prev[curr]
70      }
71
72      return distances[end], path
73  }
74
75  func main() {
76      // Contoh Data Desa (Graph)
77      // Kamu bisa mengubah nama desa dan jaraknya di sini
78      desaGraph := Graph{
79          "Desa A": {{"Desa B", 4}, {"Desa C", 2}},
80          "Desa B": {{"Desa A", 4}, {"Desa C", 5}, {"Desa D", 10}},
81          "Desa C": {{"Desa A", 2}, {"Desa B", 5}, {"Desa E", 3}},
82          "Desa D": {{"Desa B", 10}, {"Desa E", 4}},
83          "Desa E": {{"Desa C", 3}, {"Desa D", 4}},
84      }
85      start := "Desa A"
86      end := "Desa D"
87      jarak, rute := Dijkstra(desaGraph, start, end)
88      // Menampilkan Hasil
89      fmt.Println("=== Program Optimasi Jalur Terpendek ===")
90      fmt.Printf("Titik Awal : %s\n", start)
91      fmt.Printf("Tujuan     : %s\n", end)
92      fmt.Println("---------------------------------------")
93
94      if jarak == math.MaxInt32 {
95          fmt.Println("Jalur tidak ditemukan.")
96      } else {
97          fmt.Printf("Total Jarak Terpendek: %d KM\n", jarak)
98          fmt.Printf("Rute yang Dilewati   : ")
99          for i, desa := range rute {
100             fmt.Print(desa)
101             if i < len(rute)-1 {
102                 fmt.Print(" -> ")
103             }
104         }
105         fmt.Println()
106     }
```

3.5 Output / Hasil code

```
=== Program Optimasi Jalur Terpendek ===
Titik Awal : Desa A
Tujuan     : Desa D
---------------------------------------
Total Jarak Terpendek: 9 KM
Rute yang Dilewati   : Desa A -> Desa C -> Desa E -> Desa D
```

3. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem yang telah dilakukan, dapat disimpulkan bahwa algoritma Dijkstra berhasil diterapkan untuk menentukan jalur terpendek antar desa menggunakan representasi graf berbobot. Implementasi sistem menggunakan bahasa pemrograman Go (Golang) mampu menghitung jarak minimum secara akurat dari titik awal menuju titik tujuan.

Hasil pengujian menunjukkan bahwa sistem dapat menghasilkan jalur terpendek dengan total jarak minimum sebesar 9 km melalui rute Desa A – Desa C – Desa E – Desa D. Jalur yang diperoleh sesuai dengan prinsip kerja algoritma Dijkstra, di mana pemilihan simpul dilakukan berdasarkan jarak terkecil yang belum dikunjungi.

Dengan demikian, penelitian ini membuktikan bahwa algoritma Dijkstra efektif dan efisien dalam menyelesaikan permasalahan optimasi jalur terpendek pada jaringan antar desa. Sistem yang dikembangkan dapat dijadikan sebagai dasar untuk pengembangan lebih lanjut, seperti penerapan pada sistem navigasi berbasis peta digital atau optimasi distribusi logistik wilayah.

4. Daftar Pustaka

[1] E. W. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[2] A. Cantona, F. Fauziah & W. Winarsih, “Implementasi Algoritma Dijkstra pada Pencarian Rute Terpendek ke Museum di Jakarta,” *Jurnal Teknologi dan Manajemen Informatika*, 2025.

[3] D. Rahmadi, T. N. Putri, D. A. Ilmi, S. M. Rohmah & D. N. Safinka, “Optimasi Jalur Perjalanan Antara Jakarta dan Surabaya Menggunakan Algoritma Dijkstra,” *Al-Aqlu: Jurnal Matematika, Teknik dan Sains*, 2025.

[4] A. Pramadjaya & I. Rohmawati, “Analisis Rute Terpendek Menuju Universitas Pamulang dengan Implementasi Algoritma Dijkstra,” *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, 2025.

[5] E. Y. Christin & Y. F. Riti, “Perbandingan Penerapan Algoritma Dijkstra dan Algoritma Kruskal untuk Menentukan Rute Terpendek dari Taman Puspa Garden Menuju SMAN 4 Sidoarjo,” *Jurnal Teknologi Informatika dan Komputer*, 2024.