

SKRIPSI

**MODULARISASI ALGORITMA SHORTEST PATH PADA
PERANGKAT LUNAK KIRI MENGGUNAKAN STRATEGY
PATTERN**



Muhammad Aldi Rivandi

NPM: 6182001029

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2025**

UNDERGRADUATE THESIS

**MODULARIZATION OF THE SHORTEST PATH ALGORITHM
ON KIRI SOFTWARE USING STRATEGY PATTERN**



Muhammad Aldi Rivandi

NPM: 6182001029

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2025**

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Kata-kata kunci: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Keywords: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

DAFTAR ISI

DAFTAR ISI	ix
DAFTAR GAMBAR	xi
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metodologi	2
1.6 Sistematika Pembahasan	2
2 LANDASAN TEORI	3
2.1 Strategy Pattern	3
2.2 MySQL	4
2.2.1 LineString	4
2.3 Algoritma Dijkstra	5
2.4 Algoritma Floyd-Warshall	5
2.5 NewMenjangan	5
A KODE PROGRAM	7
B HASIL EKSPERIMEN	9

DAFTAR GAMBAR

2.1	Struktur Strategy Pattern	3
B.1	Hasil 1	9
B.2	Hasil 2	9
B.3	Hasil 3	9
B.4	Hasil 4	9

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Perangkat lunak KIRI adalah perangkat lunak berbasis web yang dirancang untuk membantu pengguna menemukan rute perjalanan menggunakan angkutan kota (angkot). Pada perangkat lunak KIRI, pengguna dapat memasukkan titik awal perjalanan dan titik tujuan. KIRI kemudian akan mencari berbagai alternatif rute angkot yang bisa digunakan untuk mencapai tujuan tersebut.

Arsitektur aplikasi KIRI terbagi menjadi dua bagian utama. Bagian frontend, yang dinamakan Tirtayasa, dibangun menggunakan bahasa pemrograman PHP dan mengandalkan basis data MySQL untuk menyimpan serta mengelola data. Selain itu, Tirtayasa juga menggunakan framework CodeIgniter 3. Saat menerima permintaan pencarian, Tirtayasa meneruskannya ke bagian backend, yaitu NewMenjangan. Hasil dari NewMenjangan kemudian diformat agar dapat dibaca dengan baik oleh pengguna. Bagian ini diimplementasikan dalam bahasa pemrograman Java dan berperan penting dalam perhitungan rute optimal.

NewMenjangan merupakan program daemon yang berjalan secara otomatis saat server dinyalakan dan terus beroperasi hingga server dimatikan. Daemon sendiri adalah program komputer yang berjalan di latar belakang dan tidak berinteraksi langsung dengan pengguna¹. Pada saat eksekusi, NewMenjangan terhubung ke basis data MySQL untuk mengambil data rute angkot yang tersimpan dalam format *LineString*. *LineString* adalah tipe data geometri yang digunakan dalam basis data, untuk merepresentasikan garis atau rangkaian segmen garis². Setiap titik pada *LineString* merepresentasikan lokasi potensial untuk penumpang naik atau turun. Dari data tersebut, NewMenjangan membangun *weighted graph* dalam memori (RAM) dalam bentuk *adjacency list* dan melakukan prakomputasi. Setiap titik pada *LineString* menjadi *node*, dan antara titik ke- i dan titik ke- $(i+1)$ dihubungkan dengan *edge*. Jika ada dua titik dari rute angkot berbeda yang berdekatan (jarak di bawah konstanta tertentu), maka dibuatkan juga *edge*, yang menunjukkan kemungkinan seseorang dapat turun dari suatu angkot dan naik ke angkot lainnya untuk meneruskan perjalanan.

Saat NewMenjangan menerima permintaan pencarian dari titik A ke titik B, kedua titik tersebut dijadikan *node* sementara, dan dibuatkan *edge* sementara ke *node-node* yang sudah ada sebelumnya, jika jaraknya di bawah konstanta tertentu. Pencarian jarak terdekat pada graf tersebut dilakukan menggunakan algoritma Dijkstra versi teroptimasi (*priority queue* dengan struktur data *heap*). Proses ini dapat dilakukan secara paralel dengan aman (*thread-safe*) tanpa mengubah graf utama.

Pada saat ini algoritma yang digunakan KIRI masih terikat dengan algoritma Dijkstra. Oleh

¹<https://www.ibm.com/docs/en/aix/7.1?topic=processes->

²<https://dev.mysql.com/doc/refman/8.4/en/gis-linestring-property-functions.html>

karena itu, pada tugas akhir ini akan diimplementasikan algoritma lainnya, yaitu algoritma A-star dan Floyd Warshall sebagai *concrete strategy*. Selain itu, akan dilakukan juga penerapan arsitektur kelas *strategy pattern* sehingga aplikasi KIRI akan menjadi lebih fleksibel dalam pemilihan algoritma *shortest path* yang akan digunakan dan juga memudahkan apabila akan dilakukan perubahan atau perbaikan pada suatu algoritma yang digunakan.

1.2 Rumusan Masalah

1. Bagaimana cara melakukan perubahan kode pada NewMenjangan untuk menerapkan *strategy pattern*?
2. Bagaimana mengimplementasikan algoritma A-star dan Floyd Warshall sebagai *concrete strategy*?

1.3 Tujuan

1. Melakukan perubahan arsitektur kelas dengan menerapkan *strategy pattern*.
2. Melakukan implementasi algoritma A-star dan Floyd Warshall.

1.4 Batasan Masalah

...

1.5 Metodologi

Metodologi yang akan digunakan dalam pembuatan tugas akhir ini adalah sebagai berikut:

1. Melakukan eksplorasi fungsi-fungsi dan cara kerja perangkat lunak KIRI.
2. Mempelajari modul-modul yang terdapat pada Tirtayasa dan NewMenjangan.
3. Mempelajari bahasa pemrograman PHP dan framework CodeIgniter 3.
4. Melakukan studi literatur mengenai penerapan arsitektur kelas *strategy pattern*.
5. Mempelajari cara kerja algoritma Dijkstra, A-star, dan Floyd Warshall.
6. Mengubah implementasi algoritma Dijkstra yang sudah ada ke dalam *strategy pattern*.
7. Mengimplementasikan algoritma A-star dan Floyd Warshall.
8. Melakukan pengujian dan eksperimen.
9. Menulis dokumen tugas akhir.

1.6 Sistematika Pembahasan

Tugas akhir ini akan disusun menjadi beberapa bab sebagai berikut:

- **Bab 1:** Pendahuluan
Bab ini berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan.
- **Bab 2:** Landasan Teori ...
- **Bab 3:** Analisis ...

BAB 2

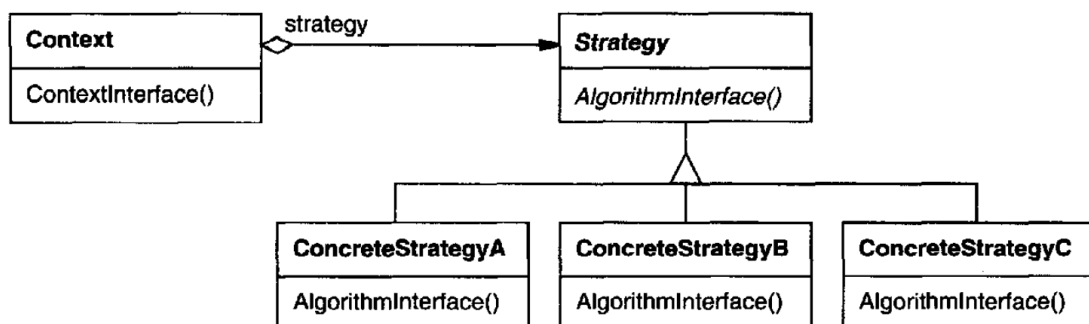
LANDASAN TEORI

2.1 Strategy Pattern

Strategy Pattern adalah sebuah design pattern yang memungkinkan objek untuk memilih saat runtime. Ini mendefinisikan sekumpulan algoritma, merangkum masing-masing algoritma, dan membuatnya dapat dipertukarkan. Hal ini memungkinkan klien untuk memilih algoritma berdasarkan kebutuhan mereka tanpa mengubah kode yang menggunakan algoritma tersebut. Ide utama penerapan Strategy Pattern adalah merancang antarmuka fleksibel yang dapat bekerja dengan algoritma berbeda tanpa perlu mengubah kode ketika algoritma baru diperkenalkan.

Strategy Pattern cocok digunakan dalam skenario-skenario berikut:

1. Ketika ada beberapa kelas terkait yang hanya berbeda dalam perilakunya. Strategy Pattern memungkinkan kelas dikonfigurasi dengan salah satu dari banyak kemungkinan perilaku.
2. Ketika ada kebutuhan untuk varian algoritma yang berbeda.
3. Ketika suatu algoritma melibatkan data yang harus tetap tersembunyi dari klien. Strategy Pattern memastikan bahwa struktur data yang kompleks dan spesifik algoritma tidak terekspos.
4. Ketika sebuah kelas mendefinisikan banyak perilaku, dan perilaku ini diimplementasikan melalui beberapa pernyataan kondisional. Dalam kasus seperti ini, Strategy Pattern menyederhanakan kode dengan memindahkan cabang kondisional terkait ke dalam kelas Strategi masing-masing



Gambar 2.1: Struktur Strategy Pattern

2.2 MySQL

MySQL adalah *open source relational database management system* (RDBMS) yang digunakan untuk menyimpan dan mengelola data yang dikembangkan oleh Oracle. MySQL adalah salah satu sistem manajemen basis data paling populer di dunia. SQL adalah singkatan dari *Structured Query Language* yang merupakan bahasa pemrograman yang digunakan untuk mengambil, memperbarui, menghapus, dan memanipulasi data dalam database relasional. Sebagai database relasional, MySQL menyimpan data dalam tabel baris dan kolom yang disusun dalam skema. Skema mendefinisikan bagaimana data diatur dan disimpan serta menjelaskan hubungan antara berbagai tabel.

Manfaat utama MySQL meliputi hal berikut:

- **Ease Of use.** Pengembang dapat menginstal MySQL dengan mudah, dan database mudah dikelola.
- **Reliability.** MySQL adalah salah satu database yang paling matang dan banyak digunakan. Teknologi ini telah diuji dalam berbagai skenario selama hampir 30 tahun, termasuk oleh banyak perusahaan terbesar di dunia.
- **Scalability.** MySQL berskala untuk memenuhi permintaan aplikasi yang paling banyak diakses. Arsitektur replikasi asli MySQL memungkinkan organisasi, termasuk Facebook, Netflix, dan Uber, meningkatkan aplikasi untuk mendukung puluhan juta pengguna atau lebih.
- **Performance.** MySQL adalah sistem database tanpa administrasi yang terbukti berkinerja tinggi dan hadir dalam berbagai edisi untuk memenuhi hampir semua permintaan.
- **High availability.** MySQL menghadirkan serangkaian teknologi replikasi asli dan terintegrasi penuh untuk ketersediaan tinggi.
- **Security.** Keamanan data mencakup perlindungan data dan kepatuhan terhadap peraturan industri dan pemerintah.
- **Flexibility.** Penyimpanan Dokumen MySQL memberi pengguna fleksibilitas maksimum dalam mengembangkan aplikasi database tradisional bebas skema SQL dan NoSQL.

2.2.1 LineString

Di MySQL, LineString adalah tipe data spasial yang digunakan untuk merepresentasikan geometri linier, seperti jalur atau garis, dalam ruang dua dimensi. MySQL menyediakan berbagai fungsi spasial untuk bekerja dengan geometri LineString, termasuk mengukur panjangnya, menentukan perpotongannya, dan banyak lagi. Fitur-fitur ini sangat berguna dalam aplikasi sistem informasi geografis dimana data spasial seperti jalan atau sungai perlu disimpan dan dianalisis.

Format untuk mendefinisikan LINESTRING adalah sebagai berikut:

LINESTRING(x1 y1, x2 y2, x3 y3, ...)

2.3 Algoritma Dijkstra

Algoritma Dijkstra merupakan algoritma greedy yang digunakan untuk mencari jalur terpendek antara suatu node sumber dan seluruh node lainnya pada suatu graf yang tidak memiliki bobot negatif $G = (V, E)$, dengan V untuk sekumpulan simpul dan E untuk sekumpulan sisi. Algoritma Dijkstra selalu memilih simpul yang memiliki bobot terkecil atau terdekat, oleh karena itu Dijkstra termasuk kedalam algoritma greedy. Dijkstra bekerja dengan menjelajahi node terdekat secara progresif dengan jarak terpendek yang diketahui, kemudian memperbarui jarak tetangganya, jika ditemukan jalur yang lebih pendek. Proses ini berlanjut hingga semua node telah dieksplorasi. *Running time* algoritma Dijkstra bergantung pada bagaimana kita mengimplementasikan *min-priority queue*.

2.4 Algoritma Floyd-Warshall

Algoritma Floyd-Warshall merupakan algoritma dynamic-programming yang digunakan untuk mencari jalur terpendek antara semua pasangan simpul dalam graf berbobot. Algoritma Floyd-Warshall bisa digunakan untuk graf berarah dan juga tidak berarah. *Running time* untuk algoritma Floyd-Warshall, yaitu $O(V^3)$, dengan V adalah jumlah simpul pada graf. Perbedaan algoritma Floyd-Warshall dan algoritma Dijkstra, yaitu algoritma Floyd-Warshall dapat menangani graf yang memiliki bobot negatif, dengan catatan tidak adanya siklus negatif.

2.5 NewMenjangan

NewMenjangan merupakan bagian *BackEnd* dari KIRI. Pada NewMenjangan terdapat berbagai modul yang dirancang menggunakan bahasa pemrograman java. NewMenjangan merupakan program daemon1 yang berjalan secara otomatis saat server dinyalakan dan terus beroperasi hingga server dimatikan yang berfungsi untuk melakukan perhitungan rute.

LAMPIRAN A

KODE PROGRAM

Kode A.1: MyCode.c

```
1 // This does not make algorithmic sense,
2 // but it shows off significant programming characters.
3
4 #include<stdio.h>
5
6 void myFunction( int input, float* output ) {
7     switch ( array[i] ) {
8         case 1: // This is silly code
9             if ( a >= 0 || b <= 3 && c != x )
10                 *output += 0.005 + 20050;
11             char = 'g';
12             b = 2^n + ~right_size - leftSize * MAX_SIZE;
13             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
14             strcpy(a,"hello_$@?");
15         }
16         count = ~mask | 0x00FF00AA;
17     }
18 }
19
20 // Fonts for Displaying Program Code in LATEX
21 // Adrian P. Robson, nepsweb.co.uk
22 // 8 October 2012
23 // http://nepsweb.co.uk/docs/progfonts.pdf
```

Kode A.2: MyCode.java

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID; //store the ID of all vertices
12    protected ArrayList<Double> closeDist; //store the distance of all vertices
13    protected int totaltrj; //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35
36 }
```


LAMPIRAN B

HASIL EKSPERIMEN

Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4