Penerapan Program Dinamis dalam Menentukan Tiket Kereta Termurah Dari Bandung ke Lumajang

Ivan Hendrawan Tan – 13522111[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
[1]*ivanhendrawantan@gmail.com*

Abstrak*—Di era digital ini terdapat begitu banyak jenis strategi algoritma yang beredar yang dapat membantu menyelesaikan berbagai masalah dari yang sederhana hingga yang kompleks, salah satu contohnya adalah menentukan tiket kereta api termurah dari Kota Bandung ke Kabupaten Lumajang. Permasalahan yang terdapat dalam makalah ini yaitu belum adanya perjalanan kereta secara langsung dari Bandung ke Lumajang. Penerapan pemrograman dinamis ini memungkinkan untuk mengevaluasi semua kemungkinan jalur perjalanan dan memilih opsi yang paling murah. Makalah ini dibuat dengan tujuan menambah wawasan baru tentang pemanfaatan teknologi informasi dalam sektor transportasi, khususnya dalam pencarian solusi optimal untuk perjalanan kereta api yang ekonomis.

***Kata Kunci*—Kereta api, pemrograman dinamis, murah**

I. PENDAHULUAN

[2]Gambar 1. Ilustrasi kereta api

Di era transportasi sekarang, terdapat begitu banyak jenis pilihan transportasi untuk kemana-mana, baik seperti bus, kereta api, kapal laut, dan pesawat. Akan tetapi, masih ada beberapa lokasi yang tidak mampu dijangkau secara langsung dari tujuan asal karena keterbatasan tertentu seperti lokasi yang cukup ekstrem, kurangnya peminat dengan tujuan tersebut, dan sebagainya. Hal ini dapat berdampak kepada kesulitan untuk memilih solusi yang dapat dijadikan sebagai pilihan termurah.

Dalam konteks kereta api, diperlukan beberapa kali transit untuk dapat tiba di stasiun tujuan. Akan tetapi, pemilihan stasiun untuk transit dapat membingungkan karena terdapat cukup banyak pilihan stasiun untuk transit. Ditambah lagi perhitungan ongkos biaya tiket yang berbeda-beda juga dapat menyulitkan untuk memilih stasiun mana yang harus dipilih agar biaya totalnya paling murah.

Makalah ini akan membahas masalah pilihan stasiun transit dari Kota Bandung menuju Kabupaten Lumajang yang dapat diselesaikan dengan Program Dinamis. Bagian pertama dari makalah ini akan membahas awal mula masalah ini. Bagian kedua akan membahas mengenai dasar teori yang digunakan untuk memecahkan permasalahan ini. Bagian ketiga adalah bagian implementasi dengan dasar teori yang digunakan. Bagian keempat membahas hasil yang ditemukan. Yang terakhir, bagian kelima akan memberi kesimpulan untuk makalah ini.

II. DASAR TEORI

A. *Graf*

Graf adalah sebuah representasi objek-objek diskrit dan hubungan yang dimiliki oleh objek-objek tersebut. Graf memiliki dua komponen penting yaitu simpul (vertex) dan sisi (edge). Graf dapat direpresentasikan sebagai sebuah tuple dengan 2 elemen yang ditulis sebagai berikut

$$G = (V, E)$$

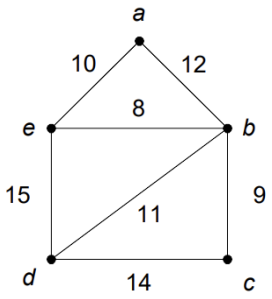
dengan $V=\{ v_1, v_2, v_3, \ldots, v_n\}$ adalah himpunan tidak kosong dari simpul-simpul (vertices) dan $E=\{e_1, e_2, e_3, \ldots, e_n\}$adalah himpunan sisi (edges) yang menghubungkan sepasang simpul.

Graf berarah (directed graph atau digraph) adalah graf yang setiap sisinya diberikan orientasi atau tanda arah untuk menunjukkan arah gerak dari sisi tersebut. Akibat sisi dalam graf berarah memiliki arah, hubungan antara dua simpul tidak simetris yang berarti bahwa keberadaan sisi dari A ke B tidak mengimplikasikan keberadaan sisi dari B ke A.

Graf berbobot adalah jenis graf di mana setiap sisi memiliki nilai tertentu yang disebut bobot. Bobot ini biasanya mewakili biaya, jarak, waktu, atau parameter kuantitatif lain yang berkaitan dengan sisi.

Graf berbobot dapat direpresentasikan menggunakan:

- Matriks Bobot: Sebuah matriks dua dimensi di mana entri w(i, j) menunjukkan bobot sisi antara simpul i dan j. Jika tidak ada sisi antara i dan j, maka entri tersebut bisa diisi dengan nilai khusus seperti ∞ (tak hingga) atau nilai khusus lain yang menandakan "tidak terhubung".
- Daftar Adjacensi: Setiap simpul memiliki daftar yang menyimpan pasangan simpul lain dan bobot sisi yang menghubungkannya.



[3]Gambar 2. Ilustrasi graf berbobot

B. Pemrograman Dinamis

Pemrograman dinamis adalah salah satu jenis algoritma untuk menyelesaikan masalah yang melibatkan pengambilan keputusan sekuensial dan optimasi, terutama ketika masalah tersebut dapat dibagi menjadi sub-masalah yang lebih kecil. Algoritma ini pertama kali dikenalkan oleh Richard Bellman pada tahun 1950-an.

Perbedaan yang mencolok dari pemrograman dinamis dari algoritma lainnya yaitu pendekatannya memecah masalah besar menjadi sub-masalah yang lebih kecil dan lebih mudah untuk dikerjakan. Sub-masalah ini sering kali tumpang tindih, dan pemrograman dinamis menggunakan pendekatan 'bottom-up' dimana solusi dari sub-masalah disimpan dan digunakan kembali sehingga menghindari pengulangan perhitungan yang tidak perlu.

Keunggulan utama pemrograman dinamis terletak pada kemampuannya untuk mengoptimalkan seluruh proses pengambilan keputusan dengan menemukan solusi terbaik di setiap tahapnya, berbeda dengan metode greedy yang hanya membuat pilihan lokal terbaik pada setiap tahapan tanpa mempertimbangkan keseluruhan.

Dalam konteks lainnya, pemrograman dinamis telah diterapkan dalam berbagai aplikasi seperti pengoptimalan rute, pengelolaan sumber daya, jadwal dan alokasi, hingga dalam bidang keuangan dan pengelolaan inventaris. Dengan perkembangan teknologi komputer dan algoritma, pemrograman dinamis terus berkembang dan menemukan aplikasi baru dalam menyelesaikan masalah yang kompleks dan dinamis dalam dunia nyata.

Karakteristik utama dari pemrograman dinamis adalah sebagai berikut:

1. Overlapping Subproblems

Pemrograman dinamis diterapkan pada masalah yang dapat dipecah menjadi sub-masalah yang lebih kecil, di mana sub-masalah ini sering kali muncul berulang kali. Dalam banyak kasus, solusi optimal untuk masalah utama bergantung pada solusi dari sub-masalah yang sama.

2. Optimal Substructure

Hal ini menunjukkan bahwa solusi optimal dari masalah dapat dibangun secara efisien dari solusi optimal sub-masalahnya. Jika masalah dapat menunjukkan struktur optimal sub ini, maka pendekatan pemrograman dinamis akan sangat efektif.

3. Memoization atau Tabulation

Salah satu prinsip utama dalam pemrograman dinamis adalah menyimpan hasil dari sub-masalah yang telah dipecahkan untuk menghindari perhitungan berulang-ulang yang sama. Teknik ini disebut memoization jika penyimpanan dilakukan secara "on-demand" atau tabulation jika dilakukan dengan sistematis dalam bentuk tabel.

4. Bottom-Up atau Top-Down Approach
   - Bottom-Up: Dikenal juga sebagai tabulation, di mana solusi untuk masalah dibangun mulai dari kasus dasar terkecil hingga mencapai kasus yang diinginkan. Ini biasanya dilakukan melalui iterasi dan penyimpanan solusi dari sub-masalah dalam tabel.
   - Top-Down: Pendekatan ini menggunakan memoization untuk memecahkan masalah dari atas ke bawah, memecahnya menjadi sub-masalah yang lebih kecil hanya jika diperlukan dan menyimpan solusi sub-masalah tersebut untuk digunakan kembali.

5. Pola Pemecahan Masalah yang Terstruktur

Pemrograman dinamis memerlukan perumusan yang jelas dari "fungsi keputusan", yang biasanya menggambarkan bagaimana solusi optimal untuk masalah yang lebih besar dapat didefinisikan dalam istilah solusi dari masalah yang lebih kecil.

6. Deterministik

Metode ini sering kali digunakan dalam konteks yang deterministik, di mana setiap pilihan yang dibuat berdasarkan pengambilan keputusan yang dihitung akan mengarah pada hasil yang pasti, meskipun masalah awal mungkin kompleks.

C. Algoritma Floyd-Warshall

Algoritma Floyd-Warshall merupakana salah satu pendekatan pemrograman dinamis yang unik, efektif dalam menyelesaikan masalah jarak terpendek antara setiap pasangan simpul dalam graf berbobot. Algoritma ini berguna dalam graf yang memiliki bobot baik positif maupun negatif, selama tidak terdapat siklus berbobot negatif yang dapat mengakibatkan perhitungan jarak terpendek menjadi tidak terdefinisi dan berpotensi menyebabkan perulangan tanpa batas dalam perhitungan tersebut.

Dalam menerapkan algoritma ini, setiap simpul graf dianggap sebagai simpul perantara dalam proses pembaruan matriks cost. Melalui iterasi yang dilakukan sebanyak n^3 kali

dengan n adalah jumlah simpul dalam graf. Algoritma ini mengupdate nilai-nilai dalam matriks cost berdasarkan pemikiran bahwa harga termurah antara dua simpul dapat diperpendek dengan memasukkan simpul lain sebagai titik perantara. Hasilnya adalah matriks yang menyajikan harga termurah antara semua pasangan simpul, yang sangat berguna dalam optimasi biaya.

Kendati efektivitasnya dalam menghadapi graf padat dan kemampuannya dalam menangani bobot negatif menjadi kelebihan utama, algoritma Floyd-Warshall juga memiliki kekurangan. Salah satu kekurangan signifikan adalah kompleksitas waktu operasionalnya, yaitu $O(n^3)$, yang bisa sangat membebani dalam kasus graf dengan jumlah simpul yang besar. Ini berarti bahwa dalam graf yang sangat luas, waktu yang diperlukan untuk menyelesaikan perhitungan bisa menjadi sangat lama. Selain itu, algoritma ini juga memerlukan alokasi memori yang besar karena perlu menyimpan sebuah matriks, yang ukurannya meningkat secara kuadratik dengan jumlah simpul dalam graf.

Dalam praktiknya, algoritma Floyd-Warshall sering dijadikan sebagai solusi teoretis dan pendidikan karena menunjukkan prinsip-prinsip pemrograman dinamis secara jelas. Namun, untuk aplikasi dunia nyata dengan graf berukuran sangat besar, biasanya dipertimbangkan pendekatan lain yang lebih efisien dari segi waktu dan memori, seperti algoritma Dijkstra atau Bellman-Ford, tergantung pada karakteristik spesifik dari graf yang dihadapi.

III. Implementasi dan Pembahasan

Cara kerja program dinamis ini akan dijelaskan dalam beberapa tahap sebagai berikut.

A. *Pembacaan File Harga*

Mula-mula, buat file harga untuk mengetahui biaya tiket kereta api dari suatu kota ke kota lainnya. Jika terdapat total n kota, maka jumlah barisnya ada sebanyak n dengan n angka untuk setiap barisnya. Angka 0 berarti biaya untuk ke kota itu sendiri, angka -1 menandakan tidak ada jalur dari kota itu ke kota tujuan, sedangkan sisanya merupakan harga tiket kereta api yang tersedia. Setelah itu, baca file harga tadi dan ubah menjadi sebuah matriks dua dimensi yang bertipe integer. Untuk setiap -1 yang ada, ubah nilai -1 menjadi sebuah nilai integer terbesar untuk merepresentasikan nilai tak hingga. Akan tetapi, nilai tak hingga yang ditampilkan pada luaran program akan diganti dengan -1 untuk memudahkan pembacaan.

B. *Rekonstruksi Awal Jalur Terpendek Antara Pasangan Simpul*

Untuk menghasilkan jalur transit yang terpendek dan termurah, maka diperlukan sebuah matriks yang bernama next dengan ukuran sebesar n x n dengan n adalah banyaknya kota. Matriks next berfungsi sebagai data untuk merekonstruksi jalur terpendek antara pasangan simpul dalam graf. Setelah menjalankan algoritma Floyd-Warshall untuk menghitung biaya termurah, kita juga perlu untuk mengetahui jalur spesifik yang menghasilkan biaya tersebut, bukan hanya panjang jaraknya.

```
int[][] next = new int[NUM_CITIES][NUM_CITIES];
        for (int i = 0; i < NUM_CITIES; i++) {
            for (int j = 0; j < NUM_CITIES; j++) {
                if (cost[i][j] != INF) {
                    next[i][j] = j;
                } else {
                    next[i][j] = -1;
                }
            }
        }
```

Gambar 3. Source code program dalam mengisi matriks next

C. *Penghitungan Biaya dan Pemrosesan Jarak*

Proses penghitungan biaya termurah dilakukan dengan iterasi sebanyak tiga kali dengan masing-masing sebanyak jumlah kota. Pengecekan dilakukan terlebih dahulu terhadap cost[i][k] dan cost [k][j] untuk memastikan bahwa nilainya bukan tak hingga dan cost[i][k] ditambah cost[k][j] kurang dari cost[i][j]. Jika ketiga syarat tersebut memenuhi, maka nilai cost[i][j] akan diisi dengan cost[i][k] ditambah dengan cost[k][j]. Lalu next[i][j] juga diisi dengan next[i][k].

```
for (int k = 0; k < NUM_CITIES; k++) {
    for (int i = 0; i < NUM_CITIES; i++) {
        for (int j = 0; j < NUM_CITIES; j++) {
            if (cost[i][k] != INF && cost[k][j] != INF && cost[i][k] + cost[k][j] < cost[i][j]) {
                cost[i][j] = cost[i][k] + cost[k][j];
                next[i][j] = next[i][k];
            }
        }
    }
}
```

Gambar 4. Source code program untuk menghitung biaya termurah

D. *Proses Jalur yang Dihasilkan*

Setelah semua proses tadi, maka kita telah menghasilkan sebuah matriks harga termurah dan sebuah matriks rute yang berdasarkan cara mendapatkan harga termurah tersebut. Untuk mengetahui rute dari suatu kota ke kota lainnya, maka diperlukan 3 buah data yaitu indeks kota asal (start), indeks kota tujuan (end), dan matriks next yang sudah diprose

Mula-mula, cek apakah elemen next[start][end] bernilai -1, jika ya maka keluarkan hasil tidak ada. Jika tidak bernilai -1, maka lanjutkan pengecekan dengan loop selama nilai start tidak sama dengan nilai end. Di dalam loop, ubah nilai start menjadi nilai dari elemen next[start][end], lalu keluarkan hasil start yang baru ke layar dan ulangi lagi loop tersebut hingga nilai start sama dengan nilai end.

```java
public static void printPath(int start, int end, int[][] next) {
    if (next[start][end] == -1) {
        System.out.println("No path");
        return;
    }
    System.out.print("Path: " + (start + 1));
    while (start != end) {
        start = next[start][end];
        System.out.print(" -> " + (start + 1));
    }
    System.out.println();
}
```

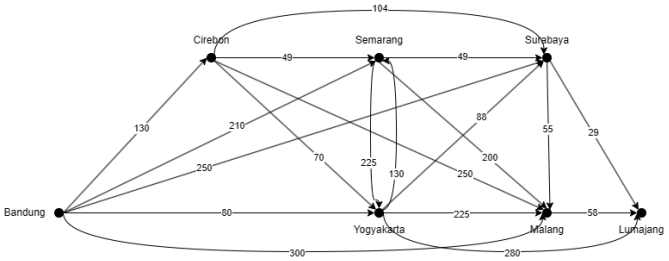
Gambar 5. Source code program untuk mencetak jalur yang dihasilkan

E. Pengujian dan Analisis

Dalam pengujian kali ini, data harga yang didapatkan berasal dari aplikasi tiket.com dengan perjalanan pada tanggal 17 Juli 2024. Berikut adalah tabel yang berisi daftar harga yang saya gunakan untuk pengujian ini.

Kota Keberangkatan	Kota Tujuan	Harga (dalam ratus ribu rupiah)
Bandung	Cirebon	130
Bandung	Yogyakarta	80
Bandung	Semarang	210
Bandung	Surabaya	250
Bandung	Malang	300
Cirebon	Yogyakarta	70
Cirebon	Semarang	49
Cirebon	Surabaya	104
Cirebon	Malang	250
Yogyakarta	Semarang	130
Yogyakarta	Surabaya	88
Yogyakarta	Malang	225
Yogyakarta	Lumajang	280
Semarang	Yogyakarta	225
Semarang	Surabaya	49
Semarang	Malang	200
Surabaya	Malang	55
Surabaya	Lumajang	29
Malang	Lumajang	58

Tabel 1. Daftar harga tiket kereta api



Gambar 6. Ilustrasi Graf berdasarkan tabel 1

Dari tabel 1, dapat dibuat sebuah file harga.txt yang isinya seperti pada gambar berikut.

```
0 130 80 210 250 300 -1
-1 0 70 49 104 250 -1
-1 -1 0 130 88 225 280
-1 -1 225 0 49 200 -1
-1 -1 -1 -1 0 55 29
-1 -1 -1 -1 -1 0 58
-1 -1 -1 -1 -1 -1 0
```

Gambar 7. Ilustrasi penulisan harga dalam file

Nama Kota	Indeks
Bandung	0
Cirebon	1
Yogyakarta	2
Semarang	3
Surabaya	4
Malang	5
Lumajang	6

Tabel 2. Daftar indeks kota

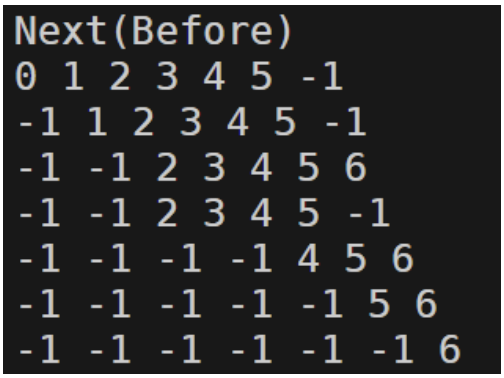
Cara membaca matriks pada gambar 6 adalah nilai elemen matriks pada baris i dan kolom j berarti harga yang dibutuhkan dari kota i menuju kota j. Jika harga yang didapat -1, hal ini berarti tidak ada rute untuk dari kota i menuju kota j secara langsung. Contohnya matriks baris 1 (indeks 0) kolom 6 (indeks 5) yang berarti harga tiket dari Kota Bandung ke Kota Malang adalah 300 ribu rupiah.

Jika file harga tadi telah dimasukkan, maka program akan menghasilkan luaran seperti berikut.

```
Cost(Before)
0 130 80 210 250 300 -1
-1 0 70 49 104 250 -1
-1 -1 0 130 88 225 280
-1 -1 225 0 49 200 -1
-1 -1 -1 -1 0 55 29
-1 -1 -1 -1 -1 0 58
-1 -1 -1 -1 -1 -1 0
```

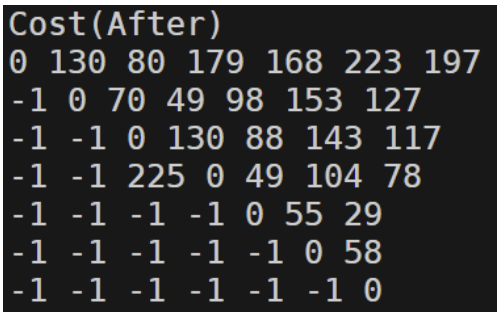
Gambar 8. Potongan pertama hasil luaran program berdasarkan kasus permasalahan

Gambar di atas merupakan potongan pertama dari keseluruhan hasil luaran program. Disini matriks cost telah diisi sesuai dengan apa yang ada di dalam file harga.txt.



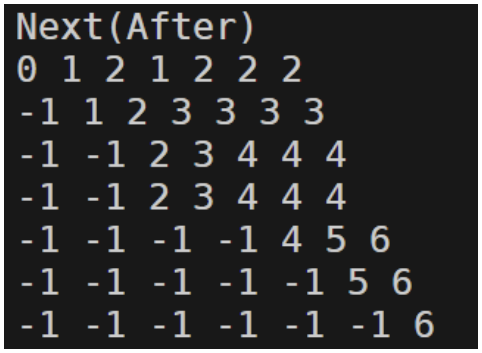
Gambar 9. Potongan kedua hasil luaran program berdasarkan kasus permasalahan

Gambar 9 ini adalah isi dari matriks next yang nantinya akan merepresentasikan rute untuk mendapatkan harga tiket termurah.



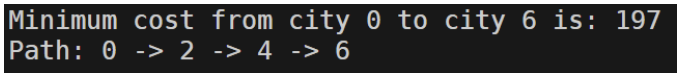
Gambar 10. Potongan ketiga hasil luaran program berdasarkan kasus permasalahan

Gambar 10 di atas merupakan matriks biaya yang telah diproses dan merupakan hasil final untuk menentukan biaya termurah dari simpul i ke simpul j. Pada kasus ini, untuk berangkat dari Kota Bandung (indeks 0) ke Kabupaten Lumajang (indeks 6), harga tiket termurah yang dapat dibeli memiliki harga total 197 ribu rupiah.



Gambar 11. Potongan keempat hasil luaran program berdasarkan kasus permasalahan

Gambar 11 menampilkan hasil perubahan yang terjadi pada matriks next yang digunakan untuk menentukan simpul tujuan. Untuk berangkat dari Bandung (indeks 0) ke Lumajang (indeks 6), pertama-tama baca elemen dari next[0][6] yang bernilai 2, selanjutnya elemen dari next[2][6] adalah 4, lalu elemen dari next[4][6] adalah 6. Maka rute stasiun yang didapat adalah 0 – 2 – 4 – 6.



Gambar 12. Hasil luaran program

Biaya termurah dari Kota Bandung ke Kabupaten Lumajang adalah 197 ribu rupiah dengan rute Bandung – Yogyakarta – Surabaya – Lumajang.

Selain untuk permasalahan dari Bandung ke Lumajang, matriks cost dan next setelah diproses juga dapat digunakan untuk mencari biaya termurah dari kota ke kota tanpa harus sesuai dengan permasalahan. Hal ini merupakan salah satu keunggulan yang didapat dari penggunaan algoritma Floyd-Warshall yang menghitung seluruh jarak terpendek antara semua pasangan simpul.

IV. KESIMPULAN

Makalah ini berhasil memaparkan penerapan program dinamis yang efektif dalam mencari tiket termurah dari Kota Bandung ke Kabupaten Lumajang. Melalui penerapan konsep tabulation, program ini membangun solusi secara bertahap dari kasus yang paling sederhana hingga yang paling kompleks. Dengan metode iteratif yang dilakukan sebanyak tiga kali, berlandaskan pada algoritma Floyd-Warshall, makalah ini tidak hanya mencapai tujuan utamanya tetapi juga menawarkan solusi untuk masalah harga tiket di berbagai kota lainnya yang terdapat dalam file harga.

Algoritma Floyd-Warshall, yang terkenal dengan kemampuannya menghitung jalur terpendek antara semua pasangan simpul dalam sebuah graf, diadaptasi untuk menentukan biaya termurah. Proses iterasi ini memungkinkan pengumpulan data secara komprehensif, dimana setiap iterasi memperbarui matriks cost dengan informasi terkini mengenai tarif terendah yang tersedia. Hasilnya, matriks cost yang dihasilkan menjadi sumber daya yang berharga dan memberikan wawasan mendalam mengenai dinamika harga tiket di seluruh jaringan transportasi yang diteliti. Dengan demikian, pendekatan ini tidak hanya menunjukkan kekuatan algoritma Floyd-Warshall dalam konteks teoretis tetapi juga aplikasinya yang praktis dalam membantu konsumen menemukan pilihan tarif terbaik yang tersedia.

V. PENUTUP

Akhir kata, penulis ingin mengucapkan puji syukur kepada Tuhan Yang Maha Esa karena atas berkat dan rahmat-Nya, yang telah membantu penulis dalam proses pembuatan makalah yang berjudul "Penerapan Program Dinamis dalam Menentukan Tiket Kereta Termurah Dari Bandung ke Lumajang". Penulis juga ingin mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen pengampu kelas 01 mata kuliah IF2211 Strategi Algoritma yang telah memaparkan ilmunya sehingga dapat dipahami oleh penulis. Tak luput juga, penulis

ingin berterima kasih kepada teman-teman beserta keluarga yang telah memberikan dukungan kepada penulis selama proses pembuatan makalah ini. Seluruh bantuan dan juga motivasi yang didapat sangatlah bermakna bagi penulis. Penulis juga ingin berterima kasih kepada semua sumber referensi yang telah sangat membantu dalam proses pembuatan makalah ini. Harapannya, makalah ini dapat membantu bagi yang kesulitan dalam mencari tiket transit termurah. Akhir kata, penulis ingin memohon maaf bila terdapat kesalahan kata dalam makalah ini karena penulis jugalah seorang manusia biasa.

LAMPIRAN

Tautan repositori:
https://github.com/Bodleh/ImplementasiProgramDinamis.git

DAFTAR PUSTAKA

[1] R. Munir, "IF2211 Strategi Algoritma - Semester II Tahun 2023/2024", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. Diakses pada 11 Juni 2024.

[2] https://www.grid.id/read/043947205/merinding-cerita-mistis-soal-kursi-13-d-kereta-api-harina-konon-diisi-penumpang-misterius-pt-kai-langsung-buka-suara. Diakses pada 11 Juni 2024

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf. Diakses pada 11 Juni 2024

[4] https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16. Diakses pada 11 Juni 2024

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

Ivan Hendrawan Tan 13522111