

Penerapan Teori Graf dalam Pemilihan Jadwal Kereta Transit

Wardatul Khoiroh - 13523001¹

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

wardatulkhoiroh11@gmail.com, 13523001@std.stei.itb.ac.id

Abstrak— penelitian ini berjudul penerapan teori graf dalam pemilihan jadwal kereta transit dengan tujuan mengkaji penerapan graf dalam menentukan jadwal *connecting train* dengan waktu perjalanan tercepat. Harapannya, makalah ini bisa memberikan pemahaman baru sekaligus menjadi inspirasi dalam perancangan jadwal kereta api yang lebih baik dan penentuan jadwal yang efisien. Penelitian ini bertujuan untuk mengidentifikasi konsep graf yang dapat diterapkan untuk menentukan jadwal kereta api transit tercepat. Metode yang digunakan kali ini menggunakan data kuantitatif berupa data waktu perjalanan kereta api dari stasiun asal menuju stasiun tujuan. Hasil penelitian ini menunjukkan bahwa dalam menentukan jadwal *connecting train*, penggunaan teori graf dapat membantu memvisualisasikan jalur perjalanan dan mengoptimalkan waktu transit antar kereta. Selain itu, penggunaan graf mampu meningkatkan efisiensi penjadwalan dengan mengurangi waktu tunggu penumpang serta meminimalkan kepadatan di stasiun-stasiun transit. Dengan demikian, penelitian ini tidak hanya memberikan solusi terhadap masalah penjadwalan kereta api tetapi juga berkontribusi pada peningkatan layanan transportasi publik di Indonesia.

Kata Kunci—Graf, *connecting train*, algoritma Dijkstra, penjadwalan.

I. PENDAHULUAN

Graf adalah salah satu cabang ilmu yang berkaitan dengan geometri dengan beberapa spesifikasinya. Graf seringkali digunakan dalam kebutuhan sehari-hari dalam memvisualisasikan persoalan seperti jalur kereta api, rangkaian Listrik, dan isomer senyawa kimia. Pada beberapa tahun terakhir, penggunaan transportasi umum mulai meningkat. Kebanyakan orang sekarang lebih memilih menggunakan transportasi umum karena harganya terjangkau dan memiliki rute jelas. Namun, hal ini menjadi masalah karena kepadatan penumpang yang tinggi setiap harinya. Hal ini menunjukkan perlunya optimasi dalam penjadwalan transportasi umum untuk mengurangi kepadatan dan meningkatkan kenyamanan pengguna. Masyarakat tak jarang juga lebih memilih menggunakan transportasi umum yaitu kereta dikarenakan tingkat keamanannya. Kereta sekarang memiliki keamanan yang ketat sehingga orang yang memasuki kereta adalah orang-orang yang memang memiliki kebutuhan tersendiri bukan mencuri dan sebagainya.

Kereta api jarak jauh menjadi salah satu pilihan utama

bagi Masyarakat dalam menentukan perjalanan jarak jauh, baik lintas kota maupun lintas provinsi. Kereta ini memiliki kecepatan operasional mencapai 120 km/jam dan kapasitas penumpang yang besar. Namun, terkadang dalam perjalanan jarak jauh tidak ada jadwal kereta api langsung sehingga perlu *connecting train*. Pemilihan *connecting train* ini menjadi suatu masalah apabila terjadi keterlambatan pada kereta pertama maupun jarak yang terlalu lama. Perlunya strategi dalam menentukan jadwal *connecting train* yang tepat agar efisiensi waktu transit dan juga tidak terburu-buru. Efisiensi waktu transit perlu agar waktu perjalanan tidak semakin lama karena tentunya setiap orang yang dalam perjalanan jauh ingin segera sampai di tujuan. Selain itu, waktu transit yang lama membuat kepadatan di area boarding.

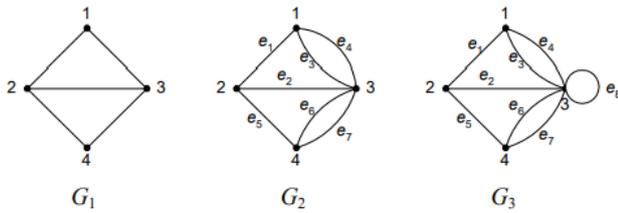
Meskipun ada beberapa penelitian yang membahas manajemen transportasi kereta api, masih terdapat kesenjangan dalam penerapan metode matematis dan algoritma optimasi untuk penjadwalan kereta transit. Banyak studi sebelumnya lebih fokus pada aspek kualitatif dan kurang mengintegrasikan pendekatan kuantitatif yang dapat memberikan solusi praktis terhadap masalah penjadwalan dan kepadatan penumpang. Penelitian ini bertujuan untuk menerapkan teori graf dalam pemilihan jadwal kereta transit, sebagai pendekatan baru yang lebih sistematis dalam mengatasi masalah yang ada. Dengan menggunakan model graf, diharapkan dapat dihasilkan jadwal yang lebih efisien dan mengurangi waktu tunggu serta kepadatan di stasiun-stasiun transit. Selain itu, dengan fokus pada pendekatan matematis ini, penelitian diharapkan dapat memberikan wawasan baru dan rekomendasi praktis bagi PT Kereta Api Indonesia dalam meningkatkan layanan mereka serta memenuhi harapan pengguna akan sistem transportasi yang lebih baik.

II. DASAR TEORI

A. Graf

Graf ditemukan pertama kali pada tahun 1736 untuk menyelesaikan persoalan jembatan Königsberg. Persoalan disini yaitu dapatkah seseorang melalui jembatan tersebut tepat sekali dan dapat kembali ke tempat semula. Graf G didefinisikan sebagai $G = (V, E)$ dimana V adalah himpunan tidak kosong dari simpul-simpul dan E adalah himpunan sisi yang menghubungkan sepasang simpul. Graf

dapat tidak memiliki sisi namun tidak boleh apabila tidak memiliki simpul. Berikut beberapa contoh graf:



Gambar 2.1 Contoh Graf
Sumber : Slide Graf Bagian 1

Graf terdiri dari dua jenis berdasarkan ada tidaknya sisi ganda dan gelang dalam graf yaitu graf sederhana dan graf tak-sederhana. Graf sederhana adalah graf yang tidak mengandung sisi ganda dan gelang sedangkan graf tak-sederhana kebalikannya. Graf tak-sederhana terdiri dari dua jenis yaitu graf ganda dan graf semu. Graf ganda adalah graf yang mengandung sisi ganda sedangkan graf semu adalah graf yang mengandung sisi gelang. Graf terdiri dari dua jenis berdasarkan orientasi arah pada sisinya yaitu graf tak-berarah dan graf berarah. Beberapa contoh pengaplikasian graf yaitu pada rangkaian listrik, isomer senyawa kimia karbon, jejaring makanan dalam biologi, pengujian program, pemodelan mesin jaja, jaringan jalan kereta api antar kota, *social network*, dan jaringan komputer.

Graf memiliki 12 terminologi diantaranya ketetanggaan, bersisian, simpul terpencil, graf kosong, derajat, lintasan, siklus atau sirkuit, keterhubungan, upagraf, upagraf merentang, *cut-set*, dan graf berbobot. Setiap terminologi memiliki pengertian dan aturannya masing-masing, sebagai berikut:

1. Ketetanggaan
Dikatakan bertetangga apabila kedua simpul terhubung secara langsung. Contoh pada gambar 2.1 graf G1 yaitu simpul 1 bertetangga dengan simpul 2 dan 3.
2. Bersisian
Dikatakan bersisian apabila sisi $e = (v_j, v_k)$ maka e bersisian dengan simpul v_j dan v_k . Contoh pada gambar 2.1 graf G1 yaitu sisi (2,3) bersisian dengan simpul 2 dan simpul 3.
3. Simpul Terpencil
Dikatakan simpul terpencil apabila simpul tidak memiliki sisi yang bersisian dengannya.
4. Graf Kosong
Dikatakan graf kosong apabila graf dengan himpunan sisinya adalah himpunan kosong.
5. Derajat
Derajat adalah jumlah sisi yang bersisian dengan simpul tersebut dengan notasinya $d(v)$. Teorema Lemma Jabat Tangan berbunyi bahwa jumlah derajat semua simpul pada graf adalah genap yaitu dua kali jumlah sisinya.
6. Lintasan
Panjang lintasan adalah jumlah sisi dalam lintasan tersebut.

7. Siklus atau Sirkuit
Ialah lintasan yang memiliki awal dan akhir pada simpul yang sama.
8. Keterhubungan
Graf dikatakan terhubung apabila terdapat lintasan dari simpul v_1 ke v_2 .
9. Upagraf atau *Subgraph*
10. Upagraf Merentang
11. *Cut-Set*
12. Graf Berbobot

Ada beberapa graf khusus yaitu graf lengkap, graf lingkaran, graf teratur, dan graf *bipartite*. Ada tiga cara dalam merepresentasikan graf yaitu dengan matriks ketetanggaan, matriks bersisian, dan senarai ketetanggaan. Graf isomorfik adalah dua graf yang sama namun beda secara geometri. Syarat dari graf isomorfik adalah mempunyai jumlah simpul yang sama, mempunyai jumlah sisi yang sama, dan mempunyai jumlah simpul yang sama berderajat tertentu. Graf planar adalah graf yang dapat digambarkan pada bidang datar tanpa sisi bersilangan. Graf tak-planar adalah kebalikannya. Pengaplikasian dari graf planar ini saat perancangan IC tidak boleh ada kawat yang saling bersilangan.

Lintasan dan sirkuit ada dua jenis yaitu euler dan hamilton. Lintasan euler adalah lintasan yang melalui masing-masing sisi dalam graf sekali. Sedangkan sirkuit euler adalah sirkuit yang melalui masing-masing sisi dalam graf sekali. Lintasan hamilton adalah lintasan yang melalui masing-masing simpul dalam graf sekali. Sedangkan sirkuit hamilton adalah sirkuit yang melalui masing-masing simpul dalam graf sekali.

Beberapa pengaplikasian graf dalam kehidupan sehari-hari yaitu dalam menentukan lintasan terpendek, persoalan pedagang keliling, persoalan tukang pos Cina, dan pewarnaan graf.

B. Algoritma Dijkstra

Menurut Wikipedia, algoritma Dijkstra adalah algoritma dalam pemecahan permasalahan jarak terpendek dalam suatu graf berarah dengan bobot-bobot garisnya nonnegative. Contohnya apabila simpul dilambangkan dengan kota-kota dan sisi dilambangkan dengan jarak antar kota tersebut maka algoritma ini dapat digunakan untuk menentukan jarak terpendek antara dua kota. Algoritma ini dikembangkan oleh Edsger Wybe Dijkstra pada tahun 1959. Prinsip utama dalam algoritma ini adalah menggunakan iterasi dengan menambahkan sisi yang terhubung dengan titik awal v_0 berdasarkan jarak terdekat. Misalkan X adalah himpunan sisi yang telah ditambahkan ke pohon. Fungsi Dijkstra-nextEdge(G, X) memilih sisi yang titik ujungnya berderajat lebih dari satu (disebut simpul non-pohon) dan memiliki jarak terpendek dari titik awal v_0 . Jika terdapat lebih dari satu pilihan sisi dengan jarak yang sama, salah satu sisi dapat dipilih secara acak.

Pada setiap iterasi, fungsi ini menambahkan sisi terkait dan simpul ujungnya ke dalam pohon lintasan terpendek.

Misalnya, jika ada dua sisi e_1 dan e_2 , dengan titik ujung dari kedua sisi tersebut berada di luar pohon (simpul non-pohon), maka sisi dengan jarak paling pendek ke v_0 akan dipilih.

Dalam algoritma Dijkstra, misalkan $w(e)$ adalah bobot dari sisi e pada graf berbobot, dan $w(q)$ adalah bobot dari sisi q yang ditambahkan ke pohon Dijkstra yang sedang dibangun. Titik awal yang digunakan untuk membangun pohon Dijkstra adalah v_0 . Jika v merupakan titik pangkal dengan nilai bobot terendah dan y adalah titik ujung dari sisi q yang terpilih, maka jarak $d(v_0, y)$ dapat dihitung sebagai $d(v_0, v) + w(q)$. Artinya, ketika sisi q dipilih pada iterasi ke- i , titik y harus memiliki nilai bobot minimum dibandingkan dengan titik lainnya.

C. Connecting Train

Connecting train adalah salah satu fitur dari KAI. Fitur ini dapat membantu pengguna dalam perjalanan jauh yang membutuhkan transit. Fitur ini biasanya muncul saat pencarian dua stasiun tidak ada yang avail entah itu memang kereta yang langsung dari dua stasiun tersebut tidak ada maupun tiketnya habis. Fitur ini mengobinasikan dua kereta sehingga pengguna dapat sampai di tujuan yang diinginkan. Namun, pemilihan *connecting train* tidak sembarangan. Perlu jeda yang cukup agar tidak tertinggal kereta berikutnya. Fungsi dari fitur ini yaitu:

1. Mencari tiket kereta api saat rute yang diinginkan tidak memiliki akses kereta secara langsung
2. Mencari tiket kereta api saat jadwal yang diinginkan dari rute perjalanan tidak sesuai
3. Mencari tiket kereta api saat kuota tiket kereta api dari rute yang diinginkan telah habis

Cara kerja dari *connecting train* sebagai berikut:

1. Maksimal hanya 2 kereta
2. Kedua kereta dapat berasal dari kelas yang sama atau gabungan
3. Pencarian *connecting train* dilakukan dengan kombinasi kereta yang memiliki jarak transit minimal 1 jam dan maksimal 24 jam
4. Terdapat kebijakan dalam keterlambatan kereta api pertama

III. PEMBAHASAN

Sebuah graf merupakan struktur yang terdiri dari simpul-simpul yang saling terhubung, di mana setiap graf pasti memiliki simpul, meskipun tidak semua graf harus memiliki sisi. Graf dapat dibedakan menjadi beberapa jenis berdasarkan karakteristik tertentu. Dalam konteks penerapan graf pada penjadwalan kereta api, kompleksitas dan variasi aplikasinya sangat beragam, sehingga dalam penelitian ini akan diangkat sebuah studi kasus mengenai *connecting train* dari Bandung ke Klakah. Pemilihan rute ini didasarkan pada pengalaman pribadi penulis yang telah melakukan perjalanan pulang kampung ke Klakah atau Lumajang, sehingga penulis memiliki pemahaman

yang lebih baik mengenai beberapa rute yang dilalui. Berikut penjelasan penerapan graf dan algoritma dijkstra secara lengkap:

A. Penggunaan Graf

Pada penelitian kali ini, graf yang digunakan adalah graf berarah dengan simpul awal menandakan stasiun asal dan simpul tujuan menandakan stasiun tujuan. Pada persoalan kali ini, simpul akan menandakan stasiun-stasiun kereta api jarak jauh sedangkan sisi akan menandakan arah kereta api berjalan dengan bobotnya berupa waktu perjalanan kereta api tersebut. Berikut daftar jadwal kereta api dari Bandung ke Klakah:

Tabel 3.1 Jadwal Kereta Api Bandung Klakah

NO	Nama Kereta	Stasiun Asal	Stasiun Tujuan	Durasi	Harga/Pax
1.	Turangga	BD	SGU	10j 11m	629.000
	Probowangi	SGU	KK	04j 15m	
2.	Turangga Panoramic	BD	SGU	10j 11m	1.029.000
	Probowangi	SGU	KK	04j 15m	
3.	Turangga	BD	SGU	10j 11m	750.000
	Pandalungan	SGU	KK	05j 18m	
4.	Turangga Panoramic	BD	SGU	10j 11m	1.150.000
	Pandalungan	SGU	KK	05j 18m	
5.	Ciremai	BD	SMT	08j 09m	680.000
	Pandalungan	SMT	KK	08j 30m	
6.	Ciremai	BD	CN	04j 17m	740.000
	Pandalungan	CN	KK	12j 22m	
7.	Malabar	BD	YK	06j 47m	595.000
	Wijayakusuma	YK	KK	09j 54m	
8.	Malabar	BD	MN	08j 49m	615.000
	Wijayakusuma	MN	KK	07j 52m	
9.	Malabar	BD	SLO	07j 37m	595.000
	Wijayakusuma	SLO	KK	09j 04m	
10.	Malabar	BD	KYA	05j 05m	665.000
	Wijayakusuma	KYA	KK	11j 36m	
11.	Malabar	BD	KTS	09j 48m	615.000
	Wijayakusuma	KTS	KK	06j 53m	
12.	Argo Wilis Panoramic	BD	SGU	10j 55m	750.000
	Blambangan Ekspres	SGU	KK	07j 40m	
13.	Argo Wilis	BD	SGU	10j 55m	785.000
	Blambangan Ekspres	SGU	KK	07j 40m	
14.	Argo Wilis	BD	KTS	08j 34m	920.000
	Wijayakusuma	KTS	KK	10j 17m	
15.	Lodaya	BD	YK	07j 00m	735.000
	Wijayakusuma	YK	KK	12j 36m	
16.	Argo Parahyangan	BD	JNG	02j 43m	755.000
	Blambangan Ekspres	JNG	KK	17j 07m	
17.	Mutiara Selatan	BD	SGU	11j 08m	625.000
	Logawa	SGU	KK	10j 14m	
18.	Lodaya	BD	KYA	04j 56m	630.000
	Logawa	KYA	KK	17j 26m	
19.	Harina	BD	CN	04j 01m	860.000
	Ranggajati	CN	KK	18j 30m	

Pada tabel 3.1 tertera beberapa alternatif *connecting train* dari Bandung ke Klakah. Beberapa alternatif tersebut memiliki perbedaan stasiun singgah, waktu durasi perjalanan, maupun harga tiket per paxnya. Pada

tabel tersebut, diurutkan berdasarkan durasi perjalanan mulai dari yang tercepat hingga terlama. Dalam rangka memudahkan perhitungan, waktu menunggu di stasiun transit dimasukkan dalam durasi perjalanan kereta kedua. Berdasarkan data tersebut, simpul mewakili stasiun-stasiun kereta api pada kolom stasiun asal, seperti Bandung (BD), Surabaya Gubeng (SGU), Klakah (KK), dan stasiun-stasiun lain yang dilalui. Sedangkan sisi mewakili waktu perjalanan antara dua stasiun pada kolom durasi dalam perhitungan akan dijadikan bentuk menit, contohnya pada baris 1 10j 11m akan diubah menjadi 611 menit. Variabel nama dan harga disini sebagai tambahan informasi saja, namun harga bisa dijadikan sebagai parameter setelah ditemukan beberapa jadwal yang memiliki waktu tercepat sama. Semua data pada table tersebut bersumber pada aplikasi Access by KAI.

B. Perhitungan Algoritma Dijkstra

Perjalanan kereta api Bandung Klakah tidak hanya menggunakan satu kereta namun dua. Hal inilah yang disebut *connecting train*. Oleh karena itu, durasi perjalanan dihitung dari awal keberangkatan kereta pertama, dilanjutkan waktu transit antara kereta pertama dan kedua, terakhir waktu perjalanan kereta kedua.

Contoh perhitungannya yaitu pada baris pertama:

- ✓ Rute perjalanannya: Bandung (BD) → Surabaya Gubeng (SGU) → Klakah (KK).
- ✓ Durasi kereta pertama (Turangga): 10 jam 11 menit dikonversi menjadi menit = 611 menit
- ✓ Durasi waktu tunggu dan durasi kereta kedua (Probowangi): 4 jam 15 menit dikonversi menjadi menit = 255 menit
- ✓ Total durasi perjalanan: $611 + 255 = 866$ menit = 14 jam 26 menit

Perhitungan durasi total tersebut dilakukan pada seluruh baris di tabel. Pada baris kedua didapatkan total durasi perjalanannya sama yaitu 14 jam 26 menit. Setelah didapat durasi total semua alternatif jadwal kereta apinya, kita dapat menentukan rute perjalanan mana yang tercepat. Algoritma Dijkstra digunakan untuk menentukan rute perjalanan tercepat dari satu stasiun ke stasiun lainnya. Berikut langkah-langkah perhitungannya:

1. Menentukan node awal dan memberikan bobot jaraknya nol pada node tersebut. Pada studi kasus kali ini, simpul awalnya adalah stasiun Bandung atau BD dan ditetapkan jaraknya dengan diri sendiri adalah 0. Namun, untuk simpul lainnya atau stasiun lainnya diinisialisasi jaraknya dengan nilai tak terhingga. Hal ini agar memudahkan dalam mencari nilai minimum nantinya. Contoh: $BD(0)$, $SGU(\infty)$, $KK(\infty)$.
2. Simpul awal yaitu BD ditandai sebagai simpul yang sedang diproses untuk menghitung jaraknya dengan simpul tetangga.
3. Memeriksa semua jarak antara simpul tetangga dengan simpul BD berdasarkan graf berarahnya. Jarak disini sebagai waktu durasi antara dua

simpul atau dua stasiun. Apabila jarak terbaru lebih kecil daripada jarak yang sebelumnya, maka nilai jarak akan berubah menjadi yang sekarang. Tidak lupa juga disimpan detail dari simpul tersebut. Contoh: $BD(0)$, $SGU(611)$, $SMT(489)$, $KK(\infty)$.

4. Simpul yang sudah dihitung jaraknya ditandai agar tidak diproses Kembali.
5. Ulangi proses pada Langkah 3 hingga semua simpul bertetangga telah dicek dan mencapai tujuan yaitu Klakah (KK)

Pemrosesan jarak dilakukan dengan memperhatikan nama simpulnya juga. Apabila ada dua sisi dengan simpul awal dan akhir sama maka akan dipilih sisi dengan jarak terkecil. Apabila berbeda, maka dijadikan sebagai pembandingan di akhir saat telah mencapai total durasi (BD → KK). Jadi, data jarak dari BD ke stasiun transit tidak langsung dibandingkan dengan jarak BD ke stasiun transit lainnya kecuali keduanya bersimpul sama. Data jarak baru dibandingkan dan dicari nilai minimalnya setelah mencapai KK. Contohnya:

- ✓ Baris pertama dan keempat memiliki rute sama yaitu $BD \rightarrow SGU \rightarrow KK$.
- ✓ Durasi perjalanan $BD \rightarrow SGU$ dari kedua baris sama sehingga tidak memengaruhi perhitungan.
- ✓ Durasi perjalanan $SGU \rightarrow KK$ dari kedua baris berbeda sehingga dilakukan perhitungan mencari minimal yaitu pada baris pertama dan data itu yang nantinya akan dibandingkan dengan rute perjalanan lain.

Perbandingan rute perjalanan sebagai berikut, contoh:

- ✓ Pada baris satu dan lima memiliki rute yang berbeda yaitu $BD \rightarrow SGU \rightarrow KK$ dan $BD \rightarrow SMT \rightarrow KK$.
- ✓ Lakukan perhitungan sebelumnya yaitu mencari total durasi perjalanan minimum dari rute yang sama didapat baris satu 14 jam 26 menit atau 866 menit dan baris lima 16 jam 39 menit atau 999 menit.
- ✓ Bandingkan total durasi tersebut sehingga didapat minimal total durasinya 866 menit. Data minimal inilah yang akan digunakan untuk perbandingan dengan rute berbeda lainnya sesuai konsep algoritma Dijkstra.

Dari perhitungan algoritma Dijkstra tersebut, didapatkan bahwa rute perjalanan Bandung Klakah yang memiliki total durasi tercepat adalah pada baris satu dan dua yaitu $BD \rightarrow SGU \rightarrow KK$ dengan nama kereta Turangga dan Probawangi.

IV. IMPLEMENTASI PROGRAM

Dengan menggunakan prinsip teori graf dan penerapan algoritma dijkstra, kita dapat memilih jadwal kereta mana yang seharusnya dinaiki agar menghemat waktu menunggu saat transit. Dalam rangka memudahkan

perhitungan, saya membuat *source code* yang memberikan hasil *connecting train* mana yang seharusnya dipilih dengan durasi perjalanan terpendek. Program dibuat dalam beberapa file dengan file intinya bernama *main.py*. Bahasa pemrograman yang digunakan yaitu python. Tentu dalam python memiliki banyak *library*. Pada program kali ini, saya menggunakan library *network* dan *matplotlib* untuk menampilkan graf atau denah dari database *connecting train* Bandung Klakah. Terdapat lima file penting bernama *algoritma_dijkstra.py*, *graf.py*, *jadwal_kereta.csv*, *main.py*, dan *utils.py*. Pada masing-masing file terdapat fungsi dengan tujuan masing-masing. Perhatikan penjelasan berikut untuk lebih rincinya.

1. File *jadwal_kereta.csv*

File ini berfungsi untuk menyimpan database dari *connecting train* Bandung Klakah. Pemilihan jenis file “.csv” ini untuk memudahkan proses pengambilan database serta file ini sudah familiar bagi saya. Database yang disimpan terdiri dari stasiun asal (*asal*), stasiun tujuan (*tujuan*), durasi perjalanan (*durasi*), dan nama keretanya (*nama*). Tiap data tersebut berbeda-beda tipenya. Pada *asal* dan *tujuan* memiliki tipe data string yaitu kumpulan char yang berupa singkatan dari nama stasiun tersebut. Contohnya untuk Stasiun Bandung disimpan dengan *BD* dan Stasiun Klakah disimpan dengan *KK*. Selanjutnya, tipe data dari *durasi* yaitu integer yang menandakan lama waktu kereta jalan dalam menit. Terakhir, tipe data dari *nama* adalah string yang menandakan nama keretanya. Berikut tampilan dari file *jadwal_kereta.csv*:

```

Pemilihan-Connecting-Train > jadwal_kereta.csv > data
You, 2 hours ago | 1 author (You)
1 asal,tujuan,durasi,nama
2 BD,SGU,611,Turangga
3 SGU,KK,255,Probowangi
4 SGU,KK,318,Pandalungan
5 BD,SGU,611,Turangga Panoramic
6 BD,SMT,489,Ciremai
7 SMT,KK,510,Pandalungan
8 BD,CN,257,Ciremai | You, 2 hours ago • doc
9 CN,KK,742,Pandalungan
10 BD,YK,407,Malabar
11 YK,KK,594,Wijayakusuma
12 BD,MN,529,Malabar
13 MN,KK,472,Wijayakusuma
14 BD,SLO,457,Malabar
15 SLO,KK,544,Wijayakusuma
16 BD,KYA,305,Malabar
17 KYA,KK,696,Wijayakusuma
18 BD,KTS,588,Malabar
19 KTS,KK,413,Wijayakusuma
20 BD,SGU,655,Argo Wilis Panoramic
21 SGU,KK,460,Blambangan Ekspres
22 BD,SGU,655,Argo Wilis
23 BD,KTS,514,Argo Wilis
24 KTS,KK,617,Wijayakusuma
25 BD,YK,420,Lodaya
26 YK,KK,756,Wijayakusuma
27 BD,JNG,163,Argo Parahyangan
28 JNG,KK,1027,Blambangan Ekspres
29 BD,SGU,668,Mutiara Selatan
30 SGU,KK,614,Logawa
31 BD,KYA,296,Lodaya
32 KYA,KK,1046,Logawa
33 BD,CN,241,Harina
34 CN,KK,1110,Ranggajati
35

```

Gambar 4.1 File *jadwal_kereta.csv*

Sumber: Dokumen penulis

2. File *utils.py*

File ini berfungsi untuk menyimpan fungsi dasar dalam pengelolaan csv. Pada file berikut, terdapat satu fungsi yaitu *baca_graf_dari_csv*. Fungsi ini digunakan untuk membaca file csv. Proses pembacaan dimulai dengan membuat array kosong bernama *jadwal_kereta*. Selanjutnya, file csv dibuka dan dibaca per baris dengan memasukkan tiap jenis data ke suatu variabel. Setelahnya, data tersebut ditambahkan ke array *jadwal_kereta* dan siap untuk digunakan dalam perhitungan. Berikut tampilan dari file *utils.py*:

```

Pemilihan-Connecting-Train > utils.py > baca_graf_dari_csv
You, 2 hours ago | 1 author (You)
1 import csv
2
3 #Fungsi untuk membaca file csv
4 def baca_graf_dari_csv(nama_file):
5     jadwal_kereta = []
6     with open(nama_file, mode='r') as file:
7         reader = csv.DictReader(file)
8         for baris in reader:
9             stasiun_asal = baris['asal']
10            stasiun_tujuan = baris['tujuan']
11            durasi = int(baris['durasi'])
12            nama_stasiun = baris['nama']
13            jadwal_kereta.append((stasiun_asal, stasiun_tujuan, durasi, nama_stasiun))
14        return jadwal_kereta
15

```

Gambar 4.2 File *utils.py*

Sumber: Dokumen penulis

3. File *algoritma_dijkstra.py*

File ini berfungsi untuk menyimpan fungsi perhitungan algoritma dijkstra. Pada file berikut, terdapat satu fungsi yaitu *dijkstra*. Fungsi ini digunakan untuk perhitungan algoritma Dijkstra dengan data *jadwal_kereta*. Fungsi ini

memiliki tiga parameter yaitu data jadwal_kereta, awal sebagai stasiun asal, dan akhir sebagai stasiun akhir. Proses perhitungan dimulai dengan membuat graf kosong. Selanjutnya, dilakukan iterasi dari database dengan menambahkan setiap barisnya ke dalam graf dari stasiun asal. Selanjutnya, fungsi menginisialisasi jarak semua node dengan nilai tak terhingga (∞), kecuali pada stasiun awal berjarak nol. Semua node yang belum dikunjungi disimpan dalam daftar.

Setelah itu, dilakukan proses *while-do* yaitu saat node tersebut belum dikunjungi maka akan memprosesnya. Proses *while-do* ini dimulai dengan memilih node dengan jarak terpendek atau minimal. Apabila jarak terbaru lebih pendek dari jarak sebelumnya, nilai jarak dan detail keretanya akan diperbarui. Node yang telah dihitung akan dihapus dari daftar dan proses akan berlanjut ke node berikutnya yang belum dikunjungi. Iterasi akan selesai saat node telah dikunjungi semua lalu mencari jalur dan nama kereta yang tercepat. Urutan jalur dibalik agar sesuai dan perlu juga menghitung waktu total perjalanan. Terakhir, fungsi akan *return* jalur perjalanan, waktu total perjalanan, dan nama keretanya. Berikut tampilan dari file algoritma_dijkstra.py:

```

1 #Fungsi Algoritma Dijkstra dengan nama stasiunmasi jadwal kereta
2 def dijkstra(jadwal_kereta, awal, akhir):
3     # Membuat graf dari data jadwal kereta
4     # Node/simpul adalah stasiun, sisi berbobot durasi
5     graf = {}
6     for stasiun_asal, stasiun_tujuan, durasi, nama_stasiun in jadwal_kereta:
7         if stasiun_asal not in graf:
8             graf[stasiun_asal] = []
9         graf[stasiun_asal].append((stasiun_tujuan, durasi, nama_stasiun)) # Tambah
10        if stasiun_tujuan not in graf:
11            graf[stasiun_tujuan] = [] # Pastikan stasiun_tujuan juga tercatat seb
12
13    # Inisialisasi jarak dan node sebelumnya
14    jarak = {node: float('inf') for node in graf}
15    jarak[awal] = 0
16    node_sebelumnya = {node: None for node in graf}
17    node_belum_dikunjungi = list(graf.keys()) # Semua node yang belum dikunjungi
18    nama_stasiun_tercepat = {}
19
20    while node_belum_dikunjungi:
21        # Pilih node dengan jarak terpendek
22        node_saat_ini = min(node_belum_dikunjungi, key=lambda node: jarak[node])
23        node_belum_dikunjungi.remove(node_saat_ini)
24
25        # Jika node tersebut sudah memiliki jarak tak terjangkau, lanjutkan
26        if jarak[node_saat_ini] == float('inf'):
27            break # Jika jarak ke node tersebut tidak bisa dijangkau
28
29        # Perbarui jarak untuk setiap tetangga node
30        for tetangga, durasi, nama_stasiun in graf.get(node_saat_ini, []): # Gunak
31            jarak_saat_ini = jarak[node_saat_ini] + durasi
32            if jarak_saat_ini < jarak[tetangga]:
33                jarak[tetangga] = jarak_saat_ini
34                node_sebelumnya[tetangga] = node_saat_ini
35            nama_stasiun_tercepat[tetangga] = nama_stasiun
36

```

Gambar 4.3 File algoritma_dijkstra.py
Sumber: Dokumen penulis

```

37 # Mencari jalur dan nama stasiun kereta yang tercepat
38 jalur = []
39 node_saat_ini = akhir
40 while node_saat_ini is not None:
41     jalur.append(node_saat_ini)
42     node_saat_ini = node_sebelumnya[node_saat_ini]
43
44 jalur = jalur[::-1] # Balikkan jalur untuk mendapatkan urutan
45
46 # Menghitung waktu total perjalanan
47 waktu_total = jarak[akhir]
48
49 # Menampilkan hasil
50 return jalur, waktu_total, nama_stasiun_tercepat

```

Gambar 4.4 Lanjutan File algoritma_dijkstra.py
Sumber: Dokumen penulis

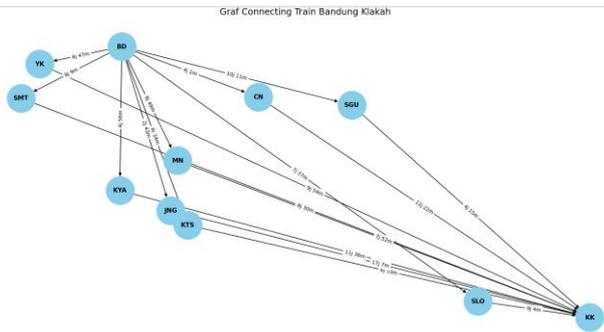
4. File main.py

File ini berfungsi untuk menyimpan fungsi utama program yang dirancang untuk mencari jalur kereta tercepat antara dua stasiun dengan *connecting train*. Pada file berikut, terdapat satu fungsi yaitu main. Fungsi ini digunakan untuk memanggil fungsi lain yang sudah dibuat pada file terpisah. Proses utama dimulai dengan meminta pengguna melakukan input manual di terminal yaitu node awal (stasiun asal) dan node akhir (stasiun tujuan). Selanjutnya, program akan membaca data graf dari file csv yang akan dikonversi menjadi graf berbobot. Terakhir, program akan mulai menghitung rute tercepat dengan menjalankan fungsi algoritma Dijkstra. Hasil dicetak di terminal berdasarkan formatnya. Berikut tampilan dari file main.py:

```

1 import os
2 from algoritma_dijkstra import dijkstra
3 from utils import baca_graf_dari_csv
4 import sys
5
6 # Menetapkan encoding ke UTF-8 di awal untuk print judul
7 sys.stdout.reconfigure(encoding='utf-8')
8
9 def main():
10     template_ascii = """
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
```

pada bagian atasnya yaitu “Graf Connecting Train Bandung Klakah”. Selain itu, agar graf tidak saling bertumpukan, perlu digunakan metode `plt.tight_layout()`. Hasil akhirnya sebagai berikut:



Gambar 4.7 Hasil graf.py
Sumber: Dokumen penulis

```

Pemilihan-Connecting-Train > graf.py > ...
41 # create a graph using MultiDiGraph to allow multiple edges
42 G = nx.MultiDiGraph() # This allows multiple edges between the same nodes
43
44 # Add edges with weights (durations in minutes)
45 for station1, station2, duration in edges:
46     # check if an edge already exists between the two stations
47     if G.has_edge(station1, station2):
48         # If the edge already exists, compare and keep the minimum duration
49         current_duration = G[station1][station2]['weight']
50         G[station1][station2]['weight'] = min(current_duration, duration)
51     else:
52         # If the edge doesn't exist, just add it
53         G.add_edge(station1, station2, weight=duration)
54
55 fig, ax = plt.subplots(figsize=(20, 12))
56 # Create visualization of the graph
57 pos = nx.spring_layout(G, seed=42) # Position of nodes for visualization
58 # Draw nodes, edges, and labels
59 nx.draw(G, pos, with_labels=True, node_color='skyblue', node_size=2000, font_size=10, font_weight='bold')
60
61 # Get all the edge weights and format them in hours and minutes
62 labels = nx.get_edge_attributes(G, 'weight')
63
64 # Convert minutes to hours and minutes (e.g., 255 minutes to 4h 15m)
65 labels = {edge: f'{weight // 60} {weight % 60}m' for edge, weight in labels.items()}
66
67 # Draw edge labels with the correct duration format
68 nx.draw_networkx_edge_labels(G, pos, edge_labels=labels, font_size=8, ax=ax)
69
70 # Title
71 ax.set_title("Graf Connecting Train Bandung Klakah", fontsize=14)
72
73 # Adjust layout to prevent overlap
74 plt.tight_layout()
75
76 # Show the plot
77 plt.show()

```

Gambar 4.9 Lanjutan File graf.py
Sumber: Dokumen penulis

Berikut tampilan dari file graf.py:

```

Pemilihan-Connecting-Train > graf.py > ...
1 import networkx as nx
2 import matplotlib.pyplot as plt
3
4 # Data kereta termasuk nama kereta, durasi perjalanan dalam menit
5 edges = [
6     ("BD", "SGU", 611), # Turangga
7     ("SGU", "KK", 255), # Probawangli
8     ("BD", "SGU", 611), # Turangga Panoramic
9     ("SGU", "KK", 318), # Pandalungan
10    ("BD", "SMT", 489), # Ciremai
11    ("SMT", "KK", 510), # Pandalungan
12    ("BD", "CN", 257), # Ciremai
13    ("CN", "KK", 742), # Pandalungan
14    ("BD", "YK", 407), # Malabar
15    ("YK", "KK", 594), # Wijayakusuma
16    ("BD", "MN", 529), # Malabar
17    ("MN", "KK", 472), # Wijayakusuma
18    ("BD", "SLO", 457), # Malabar
19    ("SLO", "KK", 544), # Wijayakusuma
20    ("BD", "KYA", 305), # Malabar
21    ("KYA", "KK", 696), # Wijayakusuma
22    ("BD", "KTS", 588), # Malabar
23    ("KTS", "KK", 413), # Wijayakusuma
24    ("BD", "SGU", 655), # Argo Wilis Panoramic
25    ("SGU", "KK", 460), # Blambangan Ekspres
26    ("BD", "SGU", 655), # Argo Wilis
27    ("BD", "KTS", 514), # Argo Wilis
28    ("KTS", "KK", 617), # Wijayakusuma
29    ("BD", "YK", 420), # Lodaya
30    ("YK", "KK", 756), # Wijayakusuma
31    ("BD", "JNG", 163), # Argo Parahyangan
32    ("JNG", "KK", 1027), # Blambangan Ekspres
33    ("BD", "SGU", 668), # Mutiara Selatan
34    ("SGU", "KK", 614), # Logawa
35    ("BD", "KYA", 296), # Lodaya
36    ("KYA", "KK", 1046), # Logawa
37    ("BD", "CN", 241), # Harina

```

Gambar 4.8 File graf.py
Sumber: Dokumen penulis

6. Pengujian

Pengujian dilakukan dalam dua tipe yaitu saat input yang diterima valid dan saat input yang diterima tidak valid. Input dianggap valid saat node awal dan node akhir berbeda, contoh dan hasil dapat dilihat pada gambar 4.10. Sedangkan, input tidak valid saat node awal dan node akhir sama, contoh dan hasil dapat dilihat pada gambar 4.11.



Gambar 4.10 Pengujian 1
Sumber: Dokumen penulis



Gambar 4.11 Pengujian 2
Sumber: Dokumen penulis

V. KESIMPULAN

Penerapan graf dalam penentuan jadwal kereta api transit digunakan dengan graf berarah sebagai jaringan kereta api dimana simpul-simpulnya mewakili stasiun-stasiun kereta api perjalanan Bandung Klakah sedangkan sisinya mewakili arah perjalanan dengan bobot durasi perjalanan. Disajikan tabel yang menunjukkan berbagai

alternatif *connecting train* dari Bandung ke Klakah dengan kolom nomor, nama kereta, stasiun awal, stasiun tujuan, durasi perjalanan, dan harga per paxnya. Melalui tabel jadwal kereta, memudahkan dalam analisis rutenya sehingga lebih sistematis. Proses perhitungan dalam mencari rute tercepat menggunakan algoritma Dijkstra dengan langkah-langkah yang sistematis dimulai dari inialisasi jarak, pemilihan simpul terpendek, hingga pembaruan jarak untuk tetangga. Proses ini dilakukan dengan iterasi sesuai banyaknya simpul tetangga.

Dari hasil perhitungan yang dilakukan, dapat disimpulkan bahwa rute tercepat untuk perjalanan dari Bandung ke Klakah adalah melalui Surabaya Gubeng (SGU), dengan total durasi perjalanan sebesar 14 jam 26 menit (866 menit). Rute ini melibatkan dua kereta, yaitu Turangga dan Probwangi, yang menunjukkan efisiensi waktu dalam perjalanan. Penerapan algoritma Dijkstra dalam konteks penjadwalan kereta api tidak hanya meningkatkan efisiensi operasional, tetapi juga memberikan informasi penting bagi penumpang mengenai waktu perjalanan dan pilihan rute terbaik. Penelitian ini memberikan kontribusi signifikan terhadap pengembangan sistem transportasi publik yang lebih baik di Indonesia. Untuk penelitian selanjutnya, disarankan untuk mempertimbangkan faktor-faktor lain seperti harga tiket dan kenyamanan perjalanan sebagai parameter tambahan dalam menentukan rute terbaik. Selain itu, penggunaan teknologi informasi untuk mengintegrasikan data real-time dapat meningkatkan akurasi dan relevansi informasi yang diberikan kepada pengguna. Dengan demikian, penerapan graf dan algoritma Dijkstra dalam studi kasus ini menunjukkan potensi besar dalam meningkatkan manajemen transportasi kereta api serta memberikan pengalaman perjalanan yang lebih baik bagi pengguna.

VI. UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya, sehingga saya dapat menyelesaikan makalah ini dengan baik. Saya juga ingin mengucapkan terima kasih kepada pihak-pihak yang telah memberikan dukungan, bantuan, dan motivasi selama proses penulisan makalah ini. Ucapan terima kasih saya sampaikan kepada:

1. Dr. Ir. Rinaldi, M.T. selaku dosen pengajar yang telah memberikan arahan dalam mengerjakan makalah ini serta masukan yang sangat berharga.
2. Muhammad Su'udi dan Siti Aminah selaku orang tua yang selalu memberikan doa, semangat, dan dukungan moral.
3. Teman-teman yang turut memberikan ide, diskusi, dan motivasi dalam menyelesaikan makalah ini.
4. Semua pihak yang tidak bisa saya sebutkan satu per satu, yang telah membantu baik secara langsung maupun tidak langsung.

Semoga makalah ini dapat memberikan manfaat dan kontribusi yang positif bagi pembaca. Saya menyadari bahwa makalah ini masih jauh dari sempurna, oleh karena

itu saran dan kritik yang membangun sangat saya harapkan.

VI. LAMPIRAN

Kode program secara lengkap dapat dilihat pada link github berikut: <https://github.com/wrdtlk choir/Pemilihan-Connecting-Train.git>. Serta video penjelasan dari makalah ini dapat dilihat pada link youtube berikut: <https://youtu.be/GRgUZYUOrc>.

REFERENSI

- [1] Munir, Rinaldi. (2024). Graf (Bagian 1). 5 Januari 2025, dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>.
- [2] Munir, Rinaldi. (2024). Graf (Bagian 2). 5 Januari 2025, dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf>.
- [3] Munir, Rinaldi. (2024). Graf (Bagian 3). 6 Januari 2025, dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf>.
- [4] Wibawana, Widhia Arum. (2023). Apa Itu *Connecting Train*? Simak Fungsi dan Cara Menggunakannya. 6 Januari 2025, dari <https://news.detik.com/berita/d-7102271/apa-itu-connecting-train-simak-fungsi-dan-cara-menggunakannya>.
- [5] Wikipedia. (2022). Algoritma Dijkstra. 6 Januari 2025, dari https://id.wikipedia.org/wiki/Algoritma_Dijkstra.
- [6] Zaki, Abdul. Algoritma Dijkstra : Teori dan Aplikasinya. 6 Januari 2025, dari <https://jmua.fmipa.unand.ac.id/index.php/jmua/article/viewFile/332/323>.

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, 8 Januari 2025



Wardatul Khoiroh
13523001