Penerapan Teori Graf dalam Pemilihan Jadwal Kereta Transit

Wardatul Khoiroh - 13523001[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*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 mengurasi 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.

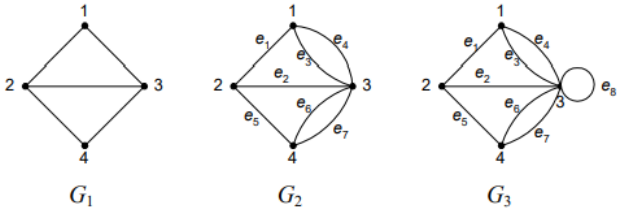
Meskipun ada beberapa penelitian yang membahas manajemen transportasi kereta api, masih terdapat kesenjangan dalam penerapan metode matematis dan algoritma optimasi untuk penjadwalan kereta transit. Banyak studi sebelumnya lebih fokus pada aspek kualitatif dan kurang mengintegrasikan pendekatan kuantitatif 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 Konigsberg. 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 menghubungan 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 1bertetangga dengan simpul 2 dan 3.
2. Bersisian
   Dikatakan bersisian apabila sisi e = (vj,vk) maka e bersisian dengan simpul vj dan vk. 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 kai 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 terhubungn apabila terdapat lintasan dari simpul v1 ke v2.
9. Upagraf atau *Subgraph*
10. Upagraf Merentang
11. *Cut-Set*
12. Graf Berbobot

Ada beberapa graf khusus yaitu graf lengkap, graf lingkarab, 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 permasalhan 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 terhubungan dengan titik awal v0 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 v0. 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 e1 dan e2, dengan titik ujung dari kedua sisi tersebut berada di luar pohon (simpul non-pohon), maka sisi dengan jarak paling pendek ke v0 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 v0. Jika v merupakan titik pangkal dengan nilai bobot terendah dan y adalah titik ujung dari sisi q yang terpilih, maka jarak d(v0,y) dapat dihitung sebagai d(v0,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 dpaat 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 perjalannnya 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 adala 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(∞), KK(∞).
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(∞).
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 pembanding 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 → SGU → KK.
- ✓ Durasi perjalanan BD → SGU dari kedua baris sama sehingga tidak memengaruhi perhitungan.
- ✓ Durasi perjalanan SGU → 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 → SGU → KK dan BD → SMT → 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 → SGU → KK dengan nama kereta Turangga dan Probowangi.

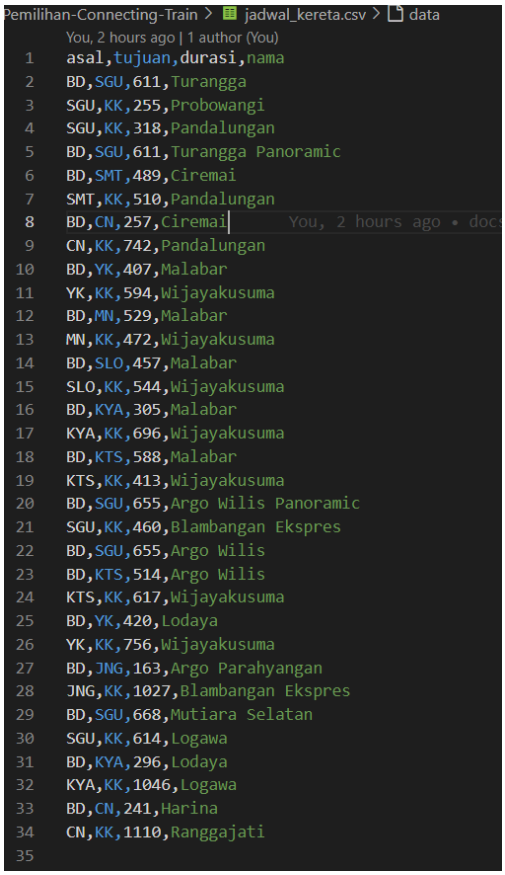
IV. IMPLEMENTASI PROGRAM

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 bernamakan 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

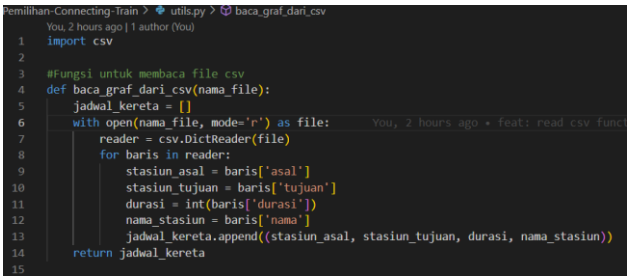
File ini berfungsi untuk menyimpan database dari *connecting train* Bandung Klakah. Pemilihan jenis file ".csv" ini untuk memudahkan proses pengambilan database serta file ini sudah familiar bagi saya. Database yang disimpan terdiri dari stasiun asal (asal), stasiun tujuan (tujuan), durasi perjalanan (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:



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:



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 (inf), kecuali pada stasiun awal berjarak nol. Semua node yang belum dikumjungi 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 ssebelumnya, 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 me-*return* jalur perjalanan, waktu toal perjalanan, dan nama keretanya. Berikut tampilan dari file algoritma_dijkstra.py:

```
Pemilihan-Connecting-Train > algoritma_dijkstra.py > dijkstra
#Fungsi Algoritma Dijkstra dengan nama_stasiunrmasi jadwal kereta
def dijkstra(jadwal_kereta, awal, akhir):
    # Membuat graf dari data jadwal_kereta
    # Node/simpul adalah stasiun, sisi berbobot durasi
    graf = {}
    for stasiun_asal, stasiun_tujuan, durasi, nama_stasiun in jadwal_kereta:
        if stasiun_asal not in graf:
            graf[stasiun_asal] = []
        graf[stasiun_asal].append((stasiun_tujuan, durasi, nama_stasiun))  # Tambah
        if stasiun_tujuan not in graf:
            graf[stasiun_tujuan] = []  # Pastikan stasiun_tujuan juga tercatat seba

    # Inisialisasi jarak dan node sebelumnya
    jarak = {node: float('inf') for node in graf}
    jarak[awal] = 0
    node_sebelumnya = {node: None for node in graf}
    node_belum_dikunjungi = list(graf.keys())  # Semua node yang belum dikunjungi
    nama_stasiun_tercepat = {}

    while node_belum_dikunjungi:
        # Pilih node dengan jarak terpendek
        node_saat_ini = min(node_belum_dikunjungi, key=lambda node: jarak[node])
        node_belum_dikunjungi.remove(node_saat_ini)

        # Jika node tersebut sudah memiliki jarak tak terjangkau, lanjutkan
        if jarak[node_saat_ini] == float('inf'):
            break  # Jika jarak ke node tersebut tidak bisa dijangkau

        # Perbarui jarak untuk setiap tetangga node
        for tetangga, durasi, nama_stasiun in graf.get(node_saat_ini, []):  # Gunak
            jarak_saat_ini = jarak[node_saat_ini] + durasi
            if jarak_saat_ini < jarak[tetangga]:
                jarak[tetangga] = jarak_saat_ini
                node_sebelumnya[tetangga] = node_saat_ini
                nama_stasiun_tercepat[tetangga] = nama_stasiun
```

Gambar 4.3 File algoritma_dijkstra.py

Sumber: Dokumen penulis

```
    # Mencari jalur dan nama_stasiun kereta yang tercepat
    jalur = []
    node_saat_ini = akhir
    while node_saat_ini is not None:
        jalur.append(node_saat_ini)
        node_saat_ini = node_sebelumnya[node_saat_ini]

    jalur = jalur[::-1]  # Balikkan jalur untuk mendapatkan uruta

    # Menghitung waktu total perjalanan
    waktu_total = jarak[akhir]

    # Menampilkan hasil
    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 yyang 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:

```
Pemilihan-Connecting-Train > main.py > main
import os
from algoritma_dijkstra import dijkstra
from utils import baca_graf_dari_csv
import sys

# Menetapkan encoding ke UTF-8 di awal untuk print judul
sys.stdout.reconfigure(encoding='utf-8')

def main():
    template_ascii = """
KERETA API BAKA
"""
    print(template_ascii)

    # Untuk menerima input string dengan format seperti scanf
    node_awal = input("Masukkan node awal (contoh: BD): ")
    node_akhir = input("Masukkan node akhir (contoh: KK): ")

    # Membaca graf dari file CSV
    nama_file_csv = os.path.join(os.path.dirname(__file__), "jadwal_kereta.csv")
    jadwal_kereta = baca_graf_dari_csv(nama_file_csv)

    # Jalankan algoritma Dijkstra manual
    jalur, waktu_total, nama_stasiun_tercepat = dijkstra(jadwal_kereta, node_awal, node_akhir)
```

Gambar 4.5 File main.py

Sumber: Dokumen penulis

```
    print("==========================================================================================")
    # Menampilkan hasil
    print(f"Jalur Tercepat dari {node_awal} ke {node_akhir}: {jalur}")
    print(f"Durasi Perjalanan Tercepat: {waktu_total // 60} jam {waktu_total % 60} menit")
    print("Detail Kereta yang Digunakan:")
    for i in range(len(jalur) - 1):
        asal = jalur[i]
        tujuan = jalur[i + 1]
        print(f"- Dari {asal} ke {tujuan}: {nama_stasiun_tercepat[tujuan]}")

    template_ascii_footer = """
SELAMAT JALAN!
"""
    print(template_ascii_footer)

if __name__ == "__main__":
    main()
```

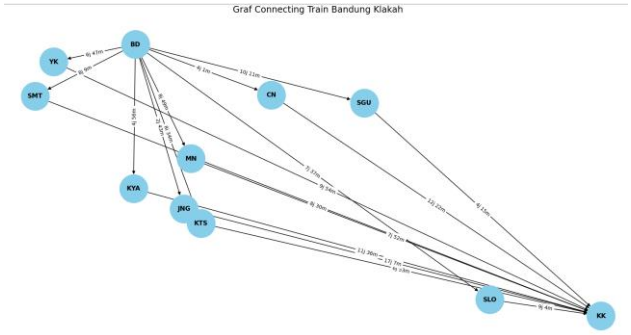
Gambar 4.6 Lanjutan File main.py

Sumber: Dokumen penulis

5. File graf.py

Dalam file ini, graf akan dibuat divisualisasikan menggunakan Pustaka *network* dan *matplotlib* dalam python. Pertama, data jadwal kereta di list dengan stasiun sebagai node atau simpul dan sisinya durasi perjalanan. Graf dibentuk menggunakan struktur *digraph* atau graf terarah. Proses pembentukan graf dimulai dengan iterasi data jadwal kereta dengan bila dua stasiun sudah memiliki hubungan maka dipilih hubungan dengan waktu tercepat. Jika belum ada, maka dibentuk sisi baru pada graf. Setelah graf terbentuk, graf divisualisasi menggunakan Pustaka *matplotlib* yang diatur oleh algoritma tata letak *spring layout*. Bobot sisi ditampilkan dengan format jam dan menit. Penggunaan warna, ukuran, dan label disesuaikan dengan kehendak pribadi. Graf diberi judul

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

Berikut tampilan dari file graf.py:

```
Pemilihan-Connecting-Train > graf.py > ...
import networkx as nx
import matplotlib.pyplot as plt

# Data kereta termasuk nama kereta, durasi perjalanan dalam menit
edges = [
    ("BD", "SGU", 611),  # Turangga
    ("SGU", "KK", 255),   # Probowangi
    ("BD", "SGU", 611),  # Turangga Panoramic
    ("SGU", "KK", 318),   # Pandalungan
    ("BD", "SMT", 489),    # Ciremai
    ("SMT", "KK", 510),   # Pandalungan
    ("BD", "CN", 257),    # Ciremai
    ("CN", "KK", 742),   # Pandalungan
    ("BD", "YK", 407),    # Malabar
    ("YK", "KK", 594),    # Wijayakusuma
    ("BD", "MN", 529),    # Malabar
    ("MN", "KK", 472),    # Wijayakusuma
    ("BD", "SLO", 457),   # Malabar
    ("SLO", "KK", 544),    # Wijayakusuma
    ("BD", "KYA", 305),    # Malabar
    ("KYA", "KK", 696),   # Wijayakusuma
    ("BD", "KTS", 588),   # Malabar
    ("KTS", "KK", 413),   # Wijayakusuma
    ("BD", "SGU", 655),  # Argo Wilis Panoramic
    ("SGU", "KK", 460),   # Blambangan Ekspres
    ("BD", "SGU", 655),  # Argo Wilis
    ("BD", "KTS", 514),   # Argo Wilis
    ("KTS", "KK", 617),  # Wijayakusuma
    ("BD", "YK", 420),         # Lodaya
    ("YK", "KK", 756),   # Wijayakusuma
    ("BD", "JNG", 163),   # Argo Parahyangan
    ("JNG", "KK", 1027),   # Blambangan Ekspres
    ("BD", "SGU", 668),   # Mutiara Selatan
    ("SGU", "KK", 614),  # Logawa
    ("BD", "KYA", 296),   # Lodaya
    ("KYA", "KK", 1046),  # Logawa
    ("BD", "CN", 241),     # Harina
```

Gambar 4.8 File graf.py
Sumber: Dokumen penulis

```
Pemilihan-Connecting-Train > graf.py > ...
# Create a graph using MultiDiGraph to allow multiple edges
G = nx.DiGraph()  # This allows multiple edges between the same nodes

# Add edges with weights (durations in minutes)
for station1, station2, duration in edges:
    # Check if an edge already exists between the two stations
    if G.has_edge(station1, station2):
        # If the edge already exists, compare and keep the minimum duration
        current_duration = G[station1][station2]['weight']
        G[station1][station2]['weight'] = min(current_duration, duration)
    else:
        # If the edge doesn't exist, just add it
        G.add_edge(station1, station2, weight=duration)

fig, ax = plt.subplots(figsize=(20, 12))
# Create visualization of the graph
pos = nx.spring_layout(G, seed=42)  # Position of nodes for visualization
# Draw nodes, edges, and labels
nx.draw(G, pos, with_labels=True, node_color='skyblue', node_size=2000, font_size=10, font_weight='bold')

# Get all the edge weights and format them in hours and minutes
labels = nx.get_edge_attributes(G, 'weight')

# Convert minutes to hours and minutes (e.g., 255 minutes to 4h 15m)
labels = {edge: f"{weight // 60}j {weight % 60}m" for edge, weight in labels.items()}

# Draw edge labels with the correct duration format
nx.draw_networkx_edge_labels(G, pos, edge_labels=labels, font_size=8,ax=ax)

# Title
ax.set_title("Graf Connecting Train Bandung Klakah", fontsize=14)

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()
```

Gambar 4.9 Lanjutan 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 penentuaan 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 rute-rutenya sehingga lebih sistematis. Proses perhitungan dalam mencari rute tercepat menggunakan algoritma Dijkstra dengan langka-langkah yang sistemastis dimulai dari inisialisasi 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 Probowangi, 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/wrdtlkhoir/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