Penerapan Algoritma Dijkstra dalam Pencarian Jalur Penerbangan Pesawat Komersial dengan Biaya Termurah

Ricky Yuliawan - 13517025
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517025@std.stei.itb.ac.id

***Abstract*—Pesawat komersial merupakan transportasi yang banyak digunakan oleh masyarakat ketika akan bepergian ke luar daerah, terutama ke daerah yang cukup jauh. Bepergian dengan pesawat komersial akan menghemat waktu perjalanan karena waktu tempuh menggunakan pesawat komersial dari suatu tempat ke tempat lainnya yang relatif cepat. Namun, mulai awal tahun 2019, biaya transportasi menggunakan pesawat komersial meningkat menjadi sangat mahal. Sehingga, masyarakat kebingungan untuk bepergian menggunakan pesawat komersial karena biayanya yang jauh lebih mahal dari biasanya. Untuk membantu menemukan rute penerbangan dari suatu tempat ke tempat lainnya yang paling murah diperlukan penerapan dari algoritma Dijkstra. Makalah ini akan menjelaskan bagaimana penerapan algoritma Dijkstra dalam pencarian jalur penerbangan pesawat dengan biaya termurah.**

Kata kunci : jalur, pesawat, algoritma Dijkstra, biaya

I. PENDAHULUAN

Pesawat udara adalah kendaraan yang mampu terbang di udara karena gaya angkat dari reaksi udara. Pesawat terbang merupakan kemajuan teknologi yang luar biasa bagi dunia ini. Dengan adanya pesawat udara, manusia bisa bepergian dari suatu tempat ke tempat yang lain dengan dalam waktu yang lebih cepat, karena pesawat udara memiliki kecepatan yang sangat cepat. Pesawat udara lebih berat dari udara, bersayap tetap, dan dapat terbang dengan tenaga sendiri. Pesawat udara dikendalikan oleh supir yang disebut dengan pilot.

Pesawat udara diklasifikasikan menjadi beberapa jenis. Salah satu jenis pesawat udara adalah pesawat komersial. Pesawat komersial adalah pesawat dengan penerbangan yang bagian dari penerbangan umum yang melibatkan pengoperasian pesawat untuk disewa. Salah satu contoh pesawat komersial adalah Pesawat Garuda Indonesia. Pesawat komersial sendiri sudah ada di seluruh tempat di dunia. Pesawat komersial beroperasi setiap hari membawa penumpang, bagasi, dan kargo yang banyak ke tempat yang lain. Pesawat komersial sudah menjadi transportasi umum yang banyak digunakan oleh masyarakat untuk berpergian ke suatu tempat yang cukup jauh. Pesawat komersial sendiri mengangkut penumpang dari terminal ke terminal lainnya, terminal untuk pesawat komersial disebut bandara atau bandar udara.

Gambar 1.1. Pesawat komersial
Sumber :
https://megapolitan.kompas.com/read/2016/09/10/18001051/berapa.rata-rata.umur.pesawat.terbang.yang.beroperasi.di.indonesia.ini.jawabannya.

Bandara atau bandar udara adalah tempat bagi pesawat udara untuk lepas landas ke udara dan mendarat dari udara. Bandara juga sering disebut dengan pelabuhan udara. Suatu bandara paling tidak memiliki sebuah landas pacu untuk lepas landas dan mendarat. Tetapi bandara-bandara besar biasanya dilengkapi berbagai fasilitas lain yang cukup lengkap yang berguna untuk menunjang aktivitas di bandara tersebut. Salah satu contoh bandara adalah Bandar Udara Husein Sastranegara yang terletak di Kota Bandung, Jawa Barat. Bandara memiliki kode unik sebagai identitas setiap bandara, yaitu kode IATA dan kode ICAO. Kode tersebut biasanya diambil dari nama bandara tersebut, daerah tempat bandara tersebut terletak, atau bisa juga dari nama kota yang dilayani. Misalnya, kode IATA Bandara Husein Sastranegara adalah BDO. Kode IATA Bandara Husein Sastranegara diambil dari nama kota yang dilayani bandara tersebut, yaitu Bandung.

Gambar 1.2. Bandara Husein Sastranegara

Sumber : https://mediaindonesia.com/read/detail/38757-wajah-baru-bandara-husein-sastranegara.html

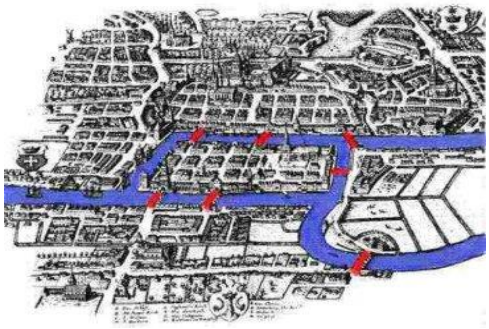
Dalam makalah ini, penulis akan menerapkan algoritma Dijkstra untuk menentukan jalur penerbangan pesawat komersil dengan biaya termurah. Dengan panduan rute penerbangan pesawat komersil dengan biaya termurah ini, diharapkan bisa terbantu untuk mencari rute penerbangan yang biayanya termurah menjadi lebih mudah.

II. LANDASAN TEORI

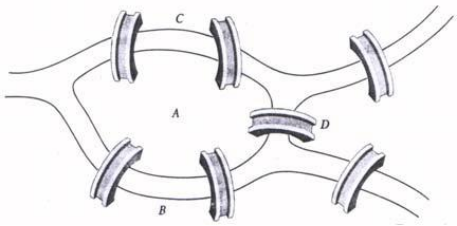
2.1. Teori Graf

Graf adalah himpunan benda-benda yang disebut "simpul" (vertex atau node) yang terhubung oleh "sisi" (*edge*) atau "busur" (*arc*). Biasanya graf digambarkan sebagai kumpulan titik-titik (melambangkan "simpul") yang dihubungkan oleh garis-garis (melambangkan "sisi") atau garis berpanah (melambangkan "busur"). Suatu sisi dapat menghubungkan suatu simpul dengan simpul yang sama. Sisi yang demikian dinamakan "gelang" (*loop*).[1]

Teori graf pertama kali ditulis oleh seorang matematikawan Swiss yang bernama Leonard Euler pada tahun 1736. Saat itu teori tersebut dibuat untuk menyelesaikan masalah jembatan Konigsberg. Dalam merepresentasikan jembatan Konigsberg dengan graf, simpul (*vertex*) pada graf tersebut menyatakan daratan, sedangkan sisi (*edge*) pada graf tersebut menyatakan jembatannya.



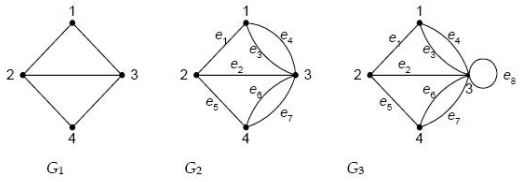
Gambar 2.1. Jembatan Konigsberg
Sumber : https://animath1994.wordpress.com/2018/01/05/kisah-7-jembatan-konigsberg/



Gambar 2.2. Graf yang merepresentasikan Jembatan Konigsberg
Sumber : https://ekamei08.wordpress.com/2016/10/03/teori-graf/

2.1.1. Definisi Graf

Graf G = (V, E), dimana
V adalah himpunan tidak-kosong dari simpul-simpul (*vertices*) = { v1 , v2 , ... , vn }
E adalah himpunan sisi (*edges*) yang menghubungkan sepasang simpul = {e1 , e2 , ... , en }



(G_1) graf sederhana, (G_2) multigraf, dan (G_3) multigraf

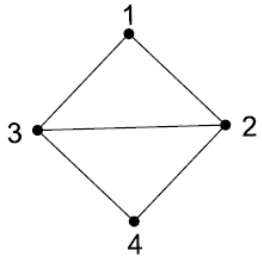
Gambar 2.3. Contoh Graf
Sumber : http://athayaniimtinan.blogspot.com/2017/12/pewarnaan-graf.html

Sisi dari graf juga ada yang namanya sisiganda (*mutiple edges*) dan gelang (*loop*). Contoh dari sisiganda adalah sisi $e_3 = (1, 3)$ dan sisi $e_4 = (1, 3)$ pada G_2, karena kedua sisi ini menghubungi dua buah simpul yang sama, yaitu simpul 1 dan simpul 3. Sedangkan contoh gelang adalah sisi $e_8 = (3, 3)$ pada G_3, karena sisi tersebut berawal dan berakhir pada simpul yang sama.

2.1.2. Jenis-jenis Graf

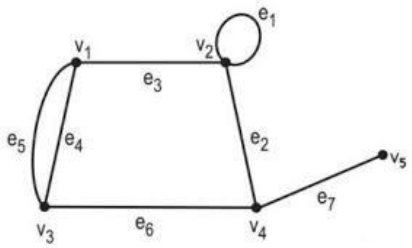
Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, maka graf digolongkan menjadi dua jenis:

1. Graf sederhana (*simple graph*).
   Graf sederhana adalah graf yang tidak mengandung gelang maupun sisi-ganda.[2]



Gambar 2.4. Graf Sederhana
Sumber : rabbitjeyek.blogspot.com/2011/12/teori-graf-3.html

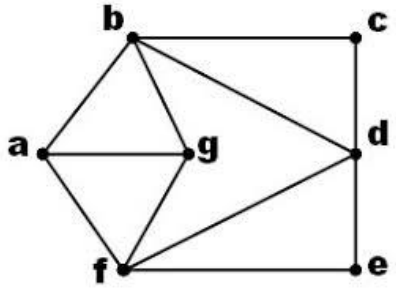
2. Graf tak-sederhana (*unsimple-graph*).
   Graf tak-sederhana adalah graf yang mengandung sisi ganda atau gelang.[2]



Gambar 2.5. Graf Tak-Sedehana
Sumber :
https://eprints.uny.ac.id/28787/2/c.BAB%20II.pdf

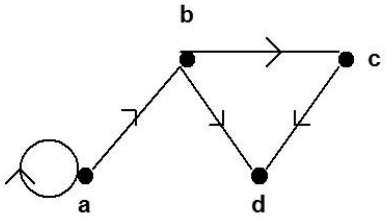
Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas 2 jenis:

1. Graf tak-berarah (*undirected graph*)
   Graf tak-berarah adalah graf yang sisinya tidak mempunyai orientasi arah.[2]



Gambar 2.6. Graf Tak-Berarah
Sumber : http://rangkum-ilmu.blogspot.com/2014/01/lintasan-dan-sirkuit-hamilton.html

2. Graf berarah (*directed graph* atau *digraph*)
   Graf berarah adalah graf yang setiap sisinya diberikan orientasi arah.[2]



Gambar 2.7. Graf Berarah
Sumber :
strukturdata01.wordpress.com/2015/01/26/graph/

2.1.3. Terminologi Graf

1. Ketetanggaan (*Adjacent*)
   Dua buah simpul dikatakan bertetangga bila keduanya terhubung langsung.[2]
2. Bersisian (*Incidency*)
   Untuk sembarang sisi e = (v_j, v_k) dikatakan e bersisian dengan simpul v_j, atau e bersisian dengan simpul v_k. [2]
3. Simpul Terpencil (*Isolated Vertex*)
   Simpul terpencil adalah simpul yang sama sekali tidak memiliki sisi yang bersisian dengannya. [2]
4. Graf Kosong (*null graph* atau *empty graph*)
   Graf yang sama sekali tidak mempunyai himpunan sisi atau kosong (N_n).
5. Derajat (*Degree*)
   Derajat suatu simpul adalah jumlah sisi yang bersisian dengan simpul tersebut. [2]
   Notasi: d(v).
6. Lintasan (*Path*)
   Lintasan yang panjangnya n dari simpul awal v_0 ke simpul tujuan vn di dalam graf G ialah barisan berselang-seling simpul-simpul dan sisi-sisi yang berbentuk v_0, e_1, v_1, e_2, v_2,... , v_{n-1}, e_n, v_n sedemikian sehingga $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2)$, ... , $e_n = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf G. [2]
7. Siklus (*Cycle*) atau Sirkuit (*Circuit*)
   Lintasan yang berawal dan berakhir pada simpul yang sama disebut sirkuit atau siklus. Panjang sirkuit adalah jumlah sisi dalam sirkuit tersebut. [2]
8. Terhubung (*Connected*)
   Dua buah simpul v1 dan simpul v2 disebut terhubung jika terdapat lintasan dari v_1 ke v_2. Suatu graf dikatakan graf terhubung (*connected graph*) jika untuk setiap pasang simpul v_i dan v_j dalam himpunan V terdapat lintasan dari v_i ke v_j. Jika tidak, maka G disebut graf tak-terhubung (*disconnected graph*).
   Graf berarah G dikatakan terhubung jika graf tidak berarahnya terhubung (graf tidak berarah dari G diperoleh dengan menghilangkan arahnya).
   Dua simpul, u dan v, pada graf berarah G disebut terhubung kuat (*strongly connected*) jika terdapat lintasan berarah dari u ke v dan juga lintasan berarah dari v ke u.
   Jika u dan v tidak terhubung kuat tetapi terhubung pada graf tidak berarahnya, maka u dan v dikatakan terhubung lemah (*weakly coonected*).
   Graf berarah G disebut graf terhubung kuat (*strongly connected graph*) apabila untuk setiap pasang simpul sembarang u dan v di G, terhubung kuat. Kalau tidak, G disebut graf terhubung lemah. [2]
9. Upagraf (*Subgraph*) dan Komplemen Upagraf
   Misalkan G = (V, E) adalah sebuah graf. $G_1 = (V_1, E_1)$ adalah upagraf (*subgraph*) dari G jika $V_1 \subseteq V$ dan $E_1 \subseteq E$.

Komplemen dari upagraf G_1 terhadap graf G adalah graf $G_2 = (V_2, E_2)$ sedemikian sehingga $E_2 = E - E_1$ dan V_2 adalah himpunan simpul yang anggota-anggota E_2 bersisian dengannya.

Komponen graf (*connected component*) adalah jumlah maksimum upagraf terhubung dalam graf G. Pada graf berarah, komponen terhubung kuat (*strongly connected component*) adalah jumlah maksimum upagraf yang terhubung kuat. [2]

10. Upagraf Rentang (*Spanning Subgraph*)
    Upagraf $G_1 = (V_1, E_1)$ dari $G = (V, E)$ dikatakan upagraf rentang jika $V_1 = V$ (yaitu G_1 mengandung semua simpul dari G). [2]
11. *Cut-Set*
    Cut-set dari graf terhubung G adalah himpunan sisi yang bila dibuang dari G menyebabkan G tidak terhubung. Jadi, *cut-set* selalu menghasilkan dua buah komponen. [2]
12. Graf Berbobot (*Weighted Graph*)
    Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot). [2]

2.1.4. Beberapa Graf Khusus

1. Graf Lengkap (*Complete Graph*)
   Graf lengkap adalah graf sederhana yang setiap simpulnya mempunyai sisi ke semua simpul lainnya. Jumlah sisi pada graf lengkap yang terdiri dari n buah simpul adalah $n(n-1)/2$.[2]
2. Graf Lingkaran
   Graf lingkaran adalah graf sederhana yang setiap simpulnya berderajat dua.[2]
3. Graf Teratur (*Regular Graphs*)
   Graf teratur adalah graf yang setiap simpulnya mempunyai derajat yang sama. Jumlah sisi pada graf teratur adalah $nr/2$. [2]
4. Graf Bipartite (*Bipartite Graph*)
   Graf bipartit adalah graf G yang himpunan simpulnya dapat dipisah menjadi dua himpunan bagian V_1 dan V_2, sedemikian sehingga setiap sisi pada G menghubungkan sebuah simpul di V_1 ke sebuah simpul di V_2. [2]

2.2. Algoritma Greedy

Algoritma greedy merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Kata "*greedy*" berasal dari bahasa inggris yang artinya rakus, tamak atau loba. Greedy memiliki prinsip, yaitu "*take what you get now!*", yang artinya "*ambil apa yang kamu dapatkan sekarang!*". Algoritma greedy membentuk solusi langkah per langkah (*step by step*). Pada setiap langkah, terdapat banyak pilihan yang perlu dievaluasi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan.

Algoritma greedy menggunakan pendekatan penyelesaian masalah dengan mencari nilai maksimum sementara (*local maximum*) pada setiap langkahnya. Pada kebanyakan kasus, algoritma greedy tidak akan menghasilkan solusi paling optimal, begitupun algoritma greedy biasanya memberikan solusi yang mendekati nilai optimum dalam waktu yang cukup cepat. Algoritma greedy merupakan algoritma yang bersifat heuristik, mencari nilai maksimal sementara dengan harapan akan mendapatkan solusi yang cukup baik. Meskipun tidak selalu mendapatkan solusi terbaik (optimum), algoritma greedy umumnya memiliki kompleksitas waktu yang cukup baik, sehingga algoritma ini sering digunakan untuk kasus yang memerlukan solusi cepat meskipun tidak optimal seperti sistem real-time atau game.

Algoritma greedy memiliki beberapa fungsionalitas dasar, yaitu:

1. Himpunan kandidat, C.
   Himpunan kandidat merupakan himpunan yang berisi elemen-elemen pembentuk solusi.
2. Himpunan solusi, S.
   Himpunan solusi merupakan himpunan yang berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.
3. Fungsi seleksi – dinyatakan dengan predikat SELEKSI.
   Fungsi seleksi merupakan fungsi yang pada setiap langkah memilih kandidat yang paling memungkinkanmencapai solusi optimal.
4. Fungsi kelayakan – dinyatakan dengan predikat LAYAK.
   Fungsi kelayakan merupakan fungsi yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala (*constrains*) yang ada.
5. Fungsi obyektif
   Fungsi obyektif merupakan fungsi yang memaksimumkan atau meminimumkan nilai solusi.[3]

2.3. Algoritma Dijkstra

Algoritma Dijkstra adalah algoritma greddy yang digunakan untuk memecahkan permasalahan jarak terpendek (shortest path problem) untuk sebuah graf berarah (directed graph) dengan bobot-bobot sisi (edge weights) yang bernilai tak-negatif. Algoritma Dijkstra ditemukan oleh Edsger W. Dikstra pada tahun 1959. Nama algoritma ini diambil dari nama penemunya.

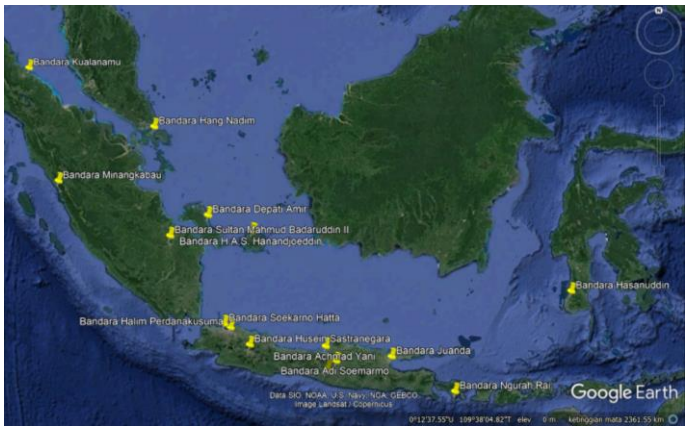
Algoritma Dijkstra bekerja dengan membuat jalur ke satu simpul optimal pada setiap langkah. Langkah-langkah algoritma Dijkstra adalah sebagai berikut.

1. Tentukan titik mana yang akan menjadi node awal, lalu beri bobot jarak pada node pertama ke node terdekat satu per satu, Dijkstra akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap.
2. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada node awal dan nilai tak hingga terhadap node lain (belum terisi) 2.
3. Set semua node yang belum dilalui dan set node awal sebagai "Node keberangkatan"
4. Dari node keberangkatan, pertimbangkan node tetangga yang belum dilalui dan hitung jaraknya dari titik keberangkatan. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru
5. Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah dilalui sebagai "Node dilewati". Node yang dilewati tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
6. Set "Node belum dilewati" dengan jarak terkecil (dari node keberangkatan) sebagai "Node Keberangkatan" selanjutnya dan ulangi langkah 5.[4]

III. Penerapan Algoritma Dijkstra dalam Pencarian Jalur Penerbangan Pesawat Komersil dengan Biaya Termurah

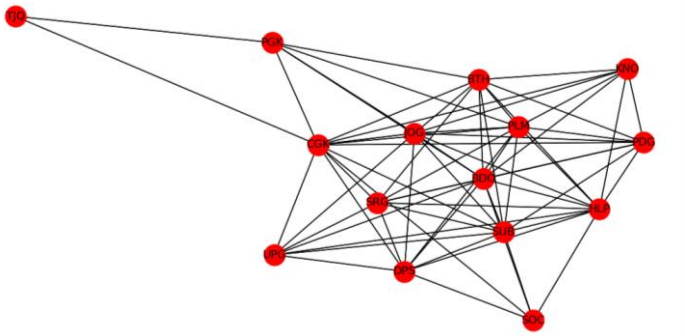
Pertama-tama, untuk penerapan algoritma Dijkstra dalam pencarian jalur penerbangan pesawat komersil dengan biaya termurah, perlu mengubah contoh dari peta rute penerbangan bandara-bandara di Indonesia menjadi sebuah graf. Sebagai contoh, saya mengambil 15 bandara di Indonesia beserta jalur dari 15 bandara yang terhubung. 15 bandara yang saya jadikan contoh adalah :

1. Bandara Soekarno Hatta, Jakarta (CGK)
2. Bandara Husein Sastranegara, Kota Bandung (BDO)
3. Bandara Halim Perdanakusuma, Jakarta (HLP)
4. Bandara Depati Amir, Pangkalpinang (PGK)
5. Bandara H.A.S Hanandjoeddin, Tanjung Pandan (TJQ)
6. Bandara Sultan Mahmud Badaruddin II, Palembang (PLM)
7. Bandara Kualanamu, Medan (KNO)
8. Bandara Achmad Yani, Semarang (SRG)
9. Bandara Djuanda, Surabaya (SUB)
10. Bandara Adi Sutjicpto, Yogyakarta (JOG)
11. Bandara Adi Soemarmo, Surakarta (SOC)
12. Bandara Ngurah Rai, Denpasar (DPS)
13. Bandara Hasanuddin, Makassar (UPG)
14. Bandara Minangkabau, Padang (PDG)
15. Bandara Hang Nadim, Batam (BTH)



Gambar 3.1. Contoh bandara yang diambil
Sumber : Google Earth

Untuk mengubah menjadi graf, saya akan merepresentasikan bandara-bandara nya sebagai simpul graf, dan jalur penerbangannya sebagai sisi graf. Untuk cost dari graf tersebut adalah biaya tiket penerbangan dari rute tersebut. Untuk memudahkan, nama bandara diganti dengan kode IATA.



Gambar 3.2. Graf bandara yang jadi contoh
Sumber : Networkx Python

Lalu, dengan memasukkan bobot antara graf dengan biaya penerbangan rute tersebut, saya merepresentasikan matriks ketetanggaan yang berisi bobot setiap sisi seperti berikut:

	CGK	BDO	HLP	PGK	TJQ	PLM	KNO	SRG	SUB	JOG	SOC	DPS	UPG	PDG	BTH
CGK	0	∞	∞	876	730	666	1077	636	840	723	704	996	1477	1171	10800
BDO	∞	0	342	974	∞	1211	1682	676	879	656	1103	974	1397	1419	13446

HLP	∞	332	0	∞	∞	828	1951	817	1211	980	1023	1306	1701	1485	1551
PGK	836	947	∞	0	431	424	∞	∞	∞	1102	∞	∞	∞	∞	779
TJQ	685	∞	∞	426	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
PLM	631	1201	828	429	∞	0	1152	1164	1001	1050	∞	1596	∞	984	707
KNO	1681	1722	2001	∞	∞	1202	0	∞	∞	2069	∞	∞	∞	779	1285
SRG	621	666	766	∞	∞	992	∞	0	634	∞	∞	928	1129	1755	1420
SUB	865	909	1297	∞	∞	1041	∞	771	0	799	771	613	942	1252	1533
JOG	661	646	817	1107	∞	1050	2092	∞	759	0	∞	751	949	1402	1432
SOC	543	1093	968	∞	∞	∞	∞	∞	731	∞	0	591	∞	∞	∞
DPS	954	1014	1628	∞	∞	1612	∞	899	623	927	641	0	879	∞	∞
UPG	1457	1402	1716	∞	∞	∞	∞	1421	988	1039	∞	908	0	∞	∞
PDG	1126	1399	1475	∞	∞	974	775	1431	988	1039	∞	∞	∞	0	2335
BTH	1075	1266	1473	794	∞	817	1245	1344	1523	1356	∞	∞	∞	2330	0

Tabel 3.1. Representasi Matriks Ketetanggaan dari bandara yang dipilih

Untuk mencobanya, misalnya saya dari Pangkalpinang hendak pergi liburan ke Padang. Kita akan mencoba menemukan rute penerbangan dengan biaya temurah dari Pangkalpinang ke Makassar menggunakan algoritma Dijkstra. Langkah-langkahnya berdasarkan langkah-langkah yang telah dituliskan pada Landasan Teori.

Dengan menggunakan langkah–langkah penerapan algoritma Dijkstra yang telah dijelaskan pada Landasan Teori, maka langkah pertama yang harus kita lakukan adalah memasukkan node awal, yaitu PGK. Lalu memasukkan nilai biaya penerbangan ke node yang bertetanggaan. Setelah itu masukkan nilai 0 untuk biaya node awal tersebut terhadap node awal itu sendiri, lalu untuk yang belum terisi, masukkan nilai tak hingga.

	CGK	BDO	HLP	PGK	TJQ	PLM	KNO	SRG	SUB	JOG	SOC	DPS	UPG	PDG	BTH
PGK	836	947	∞	0	431	424	∞	∞	∞	1102	∞	∞	∞	∞	779

Dari node keberangkatan, pertimbangkan node tetangga yang belum dilalui dan hitung jaraknya dari titik keberangkatan. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.

Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah dilalui sebagai “Node dilewati”. Node yang dilewati tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.

Set “Node belum dilewati” dengan jarak terkecil (dari node keberangkatan) sebagai “Node Keberangkatan” selanjutnya dan ulangi langkah 5. Sehingga akan membentuk proses seperti ini di bawah ini.

Proses pada simpul CGK.

	CGK	BDO	HLP	PGK	TJQ	PLM	KNO	SRG	SUB	JOG	SOC	DPS	UPG	PDG	BTH
PGK	836	947	∞	0	431	424	1953	1512	1674	1102	1538	1830	2311	2005	779
Previous	PGK	PGK	-	-	PGK	PGK	CGK	CGK	CGK	PGK	CGK	CGK	CGK	CGK	PGK

Proses pada simpul BDO.

	CGK	**BDO**	HLP	PGK	TJQ	PLM	KNO	SRG	SUB	JOG	SOC	DPS	UPG	PDG	BTH
PGK	836	947	1286	0	431	424	1953	1512	1674	1102	1538	1830	2311	2005	779
Previous	PGK	PGK	BDO	-	PGK	PGK	CGK	CGK	CGK	PGK	CGK	CGK	CGK	CGK	PGK

Proses pada simpul HLP.

	CGK	**BDO**	**HLP**	PGK	TJQ	PLM	KNO	SRG	SUB	JOG	SOC	DPS	UPG	PDG	BTH
PGK	836	947	1286	0	431	424	1953	1512	1674	1102	1538	18830	23111	2005	779
Previous	PGK	PGK	BDO	-	PGK	PGK	CGK	CGK	CGK	PGK	CGK	CGK	CGK	CGK	PGK

Proses pada simpul PLM.

	CGK	**BDO**	**HLP**	**PGK**	**TJQ**	**PLM**	KNO	SRG	SUB	JOG	SOC	DPS	UPG	PDG	BTH
PGK	836	947	1286	0	431	424	1573	1512	1422	1102	1538	18830	23111	1405	779
Previous	PGK	PGK	BDO	-	PGK	PGK	PLM	CGK	PLM	PGK	CGK	CGK	CGK	PLM	PGK

Proses pada simpul KNO.

	CGK	**BDO**	**HLP**	**PGK**	**TJQ**	**PLM**	**KNO**	SRG	SUB	JOG	SOC	DPS	UPG	PDG	BTH
PGK	836	947	1286	0	431	424	1573	1512	1422	1102	1538	18830	23111	1405	779
Previous	PGK	PGK	BDO	-	PGK	PGK	PLM	CGK	PLM	PGK	CGK	CGK	CGK	PLM	PGK

Proses pada simpul SRG.

	CGK	**BDO**	**HLP**	**PGK**	**TJQ**	**PLM**	**KNO**	**SRG**	SUB	JOG	SOC	DPS	UPG	PDG	BTH
PGK	836	947	1286	0	431	424	1573	1512	1422	1102	1538	18830	23111	1405	779
Previous	PGK	PGK	BDO	-	PGK	PGK	PLM	CGK	PLM	PGK	CGK	CGK	CGK	PLM	PGK

Proses pada simpul SUB.

	CGK	**BDO**	**HLP**	**PGK**	**TJQ**	**PLM**	**KNO**	**SRG**	**SUB**	JOG	SOC	DPS	UPG	PDG	BTH
PGK	836	947	128	0	431	424	157	151	142	110	153	183	231	140	779
Previous	PGK	PGK	BDO	-	PGK	PGK	PLM	CGK	PLM	PGK	CGK	CGK	CGK	PLM	PGK

			6				3	2	2	2	8	0	1	5	
Previous	PGK	PGK	BDO	-	PGK	PGK	PLM	CGK	PLM	PGK	CGK	CGK	CGK	PLM	PGK

Proses pada simpul JOG.

	CGK	**BDO**	**HLP**	**PGK**	**TJQ**	**PLM**	**KNO**	**SRG**	**SUB**	**JOG**	SOC	DPS	UPG	PDG	BTH
PGK	836	947	1286	0	431	424	1573	1512	1422	1102	1538	1715	2044	1405	779
Previous	PGK	PGK	BDO	-	PGK	PGK	PLM	CGK	PLM	PGK	CGK	JOG	JOG	PLM	PGK

Proses pada simpul SOC.

	CGK	**BDO**	**HLP**	**PGK**	**TJQ**	**PLM**	**KNO**	**SRG**	**SUB**	**JOG**	**SOC**	DPS	UPG	PDG	BTH
PGK	836	947	1286	0	431	424	1573	1512	1422	1102	1538	1715	2044	1405	779
Previous	PGK	PGK	BDO	-	PGK	PGK	PLM	CGK	PLM	PGK	CGK	JOG	JOG	PLM	PGK

Proses pada simpul DPS.

	CGK	**BDO**	**HLP**	**PGK**	**TJQ**	**PLM**	**KNO**	**SRG**	**SUB**	**JOG**	**SOC**	**DPS**	UPG	PDG	BTH
PGK	836	947	1286	0	431	424	1573	1512	1422	1102	1538	1715	2044	1405	779
Previous	PGK	PGK	BDO	-	PGK	PGK	PLM	CGK	PLM	PGK	CGK	JOG	JOG	PLM	PGK

Proses pada simpul UPG.

	CGK	**BDO**	**HLP**	**PGK**	**TJQ**	**PLM**	**KNO**	**SRG**	**SUB**	**JOG**	**SOC**	**DPS**	**UPG**	PDG	BTH
PGK	836	947	1286	0	431	424	1573	1512	1422	1102	1538	1715	2044	1405	779
Previous	PGK	PGK	BDO	-	PGK	PGK	PLM	CGK	PLM	PGK	CGK	JOG	JOG	PLM	PGK

Proses pada simpul PDG.

	CGK	**BDO**	**HLP**	**PGK**	**TJQ**	**PLM**	**KNO**	**SRG**	**SUB**	**JOG**	**SOC**	**DPS**	**UPG**	**PDG**	BTH
PGK	836	947	1286	0	431	424	1573	1512	1422	1102	1538	1715	2044	1405	779
Previous	PGK	PGK	BDO	-	PGK	PGK	PLM	CGK	PLM	PGK	CGK	JOG	JOG	PLM	PGK

Proses pada simpul BTH.

	CGK	**BDO**	**HLP**	**PGK**	**TJQ**	**PLM**	**KNO**	**SRG**	**SUB**	**JOG**	**SOC**	**DPS**	**UPG**	**PDG**	**BTH**
PGK	836	947	1286	0	431	424	1573	1512	1422	1102	1538	1715	2044	1405	779
Previous	PGK	PGK	BDO	-	PGK	PGK	PLM	CGK	PLM	PGK	CGK	JOG	JOG	PLM	PGK

Dengan menggunakan algoritma Dijkstra pada langkah di atas, maka kita akan mendapatkan bahwa rute penerbangan Pangkalpinang ke Padang yang biayanya termurah adalah 1.405.000, dengan rutenya adalah PGK-PLM-PDG (Bandara Depati Amir – Bandara Sultan Mahmud Badaruddin II – Bandara Minangkabau.

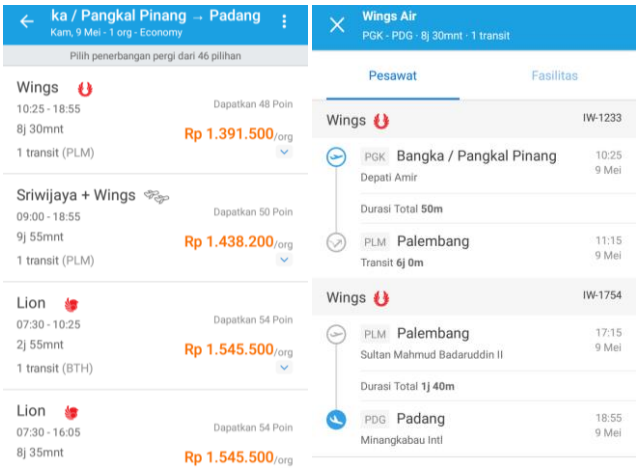
Saya juga membuat program algoritma Dijkstra untuk menentukan jalur penerbangan dengan biaya termurah. Berikut adalah hasil pencarian tersebut menggunakan program yang saya buat.

```
C:\Users\acer\Documents>python makalahstima.py
Masukkan kode IATA bandara keberangkatan : PGK
Masukkan kode IATA bandara tujuan : PDG
Rute dengan biaya termurah :
deque(['PGK', 'PLM', 'PDG'])
```

Gambar 3.4. Hasil dengan program Algoritma Dijkstra
Sumber : Dokumen Penulis

Dengan menggunakan program pencarian menggunakan algoritma Dijkstra yang saya buat juga, menghasilkan rute penerbangan yang sama, yaitu PGK-PLM-PDG.

Untuk memastikan apakah hasil tersebut benar. Maka saya mencari harga tiket Pangkalpinang ke Padang melalui aplikasi Traveloka. Berikut hasil dari aplikasi Traveloka :



Gambar 3.4. Pencarian Tiket PGK - PDG
Sumber : Traveloka

Dari hasil pencarian dari aplikasi Traveloka tersebut dapat disimpulkan bahwa algoritma Dijkstra dapat memberikan rute penerbangan dengan biaya termurah. Dengan penjelasan tersebut juga, tulisan ini sudah dapat mengaplikasikan pencarian rute penerbangan dengan biaya termurah menggunakan algoritma Dijkstra.

IV. Kesimpulan

Dalam menentukan jalur penerbangan dengan biaya termurah, algoritma Dijkstra dapat digunakan. Melalui graf, kita bisa menetapkan bandara-bandara sebagai simpul, lalu jalur penerbangannya sebagai sisi, serta biaya penerbangannya sebagai nilai bobotnya. Dimulai dari menghubungkan jalur antar bandara dengan data jalur penerbangan dari masing-masing bandara, kita bisa menemukan jalur dengan biaya termurah menggunakan algoritma Dijkstra. Sehingga terbentuklah jalur penerbangan dengan biaya termurah dari suatu bandara ke bandara yang lainnya. Dengan adanya jalur ini, orang-orang yang hendak bepergian dengan pesawat, dapat menemukan jalur penerbangan termurah.

V. Ucapan Terima Kasih

Pertama-tama, penulis mengucapkan puji syukur kepada Allah swt. atas segala nikmat yang telah diberikan-Nya, sehingga penulis bisa menyelesaikan makalah ini tepat pada waktunya. Penulis juga ingin mengucapkan terima kasih kepada dosen mata kuliah IF2211 Strategi Algoritma yang pernah mengajari penulis, khususnya kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc. yang telah mengajar kelas K-01 selama satu semester ini dan memberikan ilmu kepada penulis. Penulis juga ingin mengucapkan terima kasih kepada keluarga dan teman-teman penulis yang telah memberikan motivasi dan mengirimkan doa kepada penulis.

VI. REFERENSI

[1] Biggs, N.; Lloyd, E.; Wilson, R. (1986), Graph Theory, 1736–1936, Oxford University Press.

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf

[3] Munir, R. *Strategi Algoritma*. Bandung: Informatika Bandung, 2009.

[4] https://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/

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, 26 April 2019

Ricky Yuliawan 13517025