Penerapan Algoritma Dijkstra dan Algoritma Kruskal dalam Merancang Trayek Angkutan Umum Perkotaan

Muhammad Nurdin Husen / 13517112
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517112@std.stei.itb.ac.id

***Abstrak*—Salah satu elemen yang memiliki peranan penting dalam kehidupan perkotaan adalah angkutan umum. Oleh karenanya, perlu adanya perhatian yang cukup serius untuk merancang trayek angkutan umum pada suatu perkotaan. Pada makalah ini, akan dijelaskan bagaimana merancang trayek angkutan umum suatu kota dengan menggunakan konsep algoritma *Greedy*, yaitu algoritma Dijkstra dan algoritma Kruskal. Dengan menggunakan algoritma Dijkstra dan Kruskal, diharapkan akan mendapatkan trayek dengan jarak paling pendek dari satu terminal ke terminal yang lainnya. Metode yang digunakan dalam makalah ini adalah dengan menggunakan studi pustaka terkait dengan algoritma *Greedy* khusunya algoritma Dijkstra dan algoritma Kruskal. Dengan adanaya makalah ini, diharapkan akan menambah pengetahuan pembaca mengenai algoritma Dijkstra dan Kruskal serta penerapannya dalam perancangan trayek suatu angkutan umum perkotaan serta perbandingan trayek yang dihasilkan dari kedua algortima tersebut. Selain itu, makalah ini diharapkan dapat menjadi rujukan untuk perancangan trayek angkutan perkotaan yang optimum.**

Kata kunci—Dijkstra, Greedy, Kruskal, Trayek

I. PENDAHULUAN

Pada kota-kota besar, angkutan umum sudah menjadi tulang punggung dalam permasalahan transportasi. Angkutan umum menjadi suatu elemen yang memiliki peranan penting karena perannya dalam pendistribusian barang, jasa, dan masyarakat. Selain itu, angkutan umum juga dapat menjadi solusi untuk mengatasi permasalahan kemacetan. Dengan adanya angkutan umum, penggunaan kendaraan pribadi akan berkurang dan berdampak berkurangnya volume kendaraan pada jalan raya. Angkutan umum juga mempermudah akses ke berbagai daerah di suatu kota. Dengan adanya angkutan umum, perekonomian suatu daerah juga akan mengalami pergerakan. Hal ini dikarenakan angkutan umum akan menjamah tempat-tempat strategis suatu kota yang akan membuat banyak pengunjung di kota tersebut. Sudah selayaknya angkutan umum menjadi perhatian serius pemerintah kota mengingat perannya yang cukup penting bagi kota tersebut. Perlu dipikirkan bagaimana merancang trayek angkutan umum agar trayek yang ada adalah trayek dengan jarak yang paling optimum. Dengan trayek yang jarak tempuhnya optimum, diharapkan akan terjadi penghematan waktu tempuh dan biaya operasional angkutan umum tersebut. Untuk mendapatkan trayek dengan jarak yang optimum akan diterapkan algoritma Dijkstra dan Kruskal. Kedua algoritma ini diterapkan dengan merepresentasikan terminal-terminal yang ada sebagai simpul-simpul pada graf dan jalan-jalan pada perkotaan sebagai lintasan-lintasan pada graf.

II. DASAR TEORI

A. *Teori Graf*

1. Definisi Graf

Graf G adalah pasangan himpunan dua buah himpunan V dan E, dalam hal ini V adalah himpunan simpul (*node*) yang tidak boleh kosong dan E adalah himpunan sisi (*edge*) yang menghubungkan sipul-simpul.

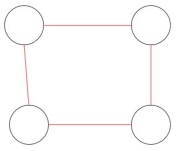
2. Jenis-jenis Graf

Graf dapat dikelompokkan berdasarkan beberapa sudut pandang. Pengelompokkan pertama didasarkan pada ada tidaknya sisi gelang atau sisi ganda pada suatu graf. Sisi gelang adalah sisi yang memiliki simpul awal dan simpul akhir yang sama sehingga membentuk sebuah *loop,* sedangkan sisi ganda adalah dua buah sisi menghubungkan dua buah simpul yang sama. Pengelompokkan yang kedua didasarkan orientasi arah pada sisi.

Berdasarkan ada tidaknya sisi gelang atau sisi ganda, graf dapat dikelompokkan menjadi dua jenis :

a. Graf Sederhana

Graf sederhana (*simple graph*) adalah graf yang tidak mengandung sisi ganda maupun sisi gelang. Yang artinya adalah setiap pasangan-pasangan sisi (u,v) akan sapa dengan pasangan sisi (v,u)

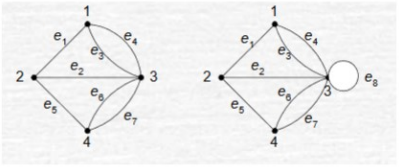


Gambar 1 : graf sederhana

b. Graf tak Sederhana

Graf tak sederhana (*unsimple graph*) adalah graf yang mengandung sisi ganda atau sisi gelang. Graf tak sederhana terdiri dari dua jenis, yaitu graf ganda dan graf semu. Graf ganda adalah graf tak sederhana yang

tidak terdapat sisi gelang, sedangkan graf semu adalah graf tak sederhana yang mengandung sisi gelang.



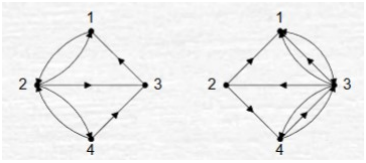
Gambar 2 : (a) graf ganda, (b) graf semu

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

Berdasarkan orientasi arah pada sisi graf, graf dapat diklasifikasikan menjadi dua jenis :

a. Graf berarah

Graf berarah (*directed graph*) adalah graf yang setiap sisinya terdapat orientasi arah. Artinya untuk setiap pasangan sisi (u,v) tidak sama denga pasangan sisi (v,u). Pada sisi (u,v) memiliki simpul asal u dan simpul akhir v, sedangkan pada sisi (v,u) memiliki simpul asal dan simpul akhir berturut-turut v dan u.



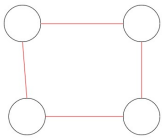
Gambar 3 : graf berarah

Sumber :

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

b. Graf tak berarah

Graf tak berarah (*undirected graph*) adalah graf yang setiap sisinya tidak memiliki orientasi arah. Artinya untuk setiap pasangan sisi (u,v) akan sama dengan pasangan sisi (v,u).



Gambar 4 : graf tak berarah

3. Lintasan dan Sirkuit

Lintasan adalah barisan simpul-simpul dan sisi-sisi yang menunjukkan arah dari simpul awal u ke simpul akhir v. Barisan simpul-simpul dan sisi-sisi tersebut menunjukkan simpul-simpul dan sisi-sisi yang dilewati untuk menuju simpul v dari simpul u. Banyaknya simpul dan sisi menggambarkan panjang lintasan.

Sirkuit pada graf adalah lintasan yang memiliki simpul awal dan simpul akhir yang sama.

4. Lintasan dan Sirkuit Euler

Lintasan Euler adalah lintasan yang melalui setiap sisi pada graf tepat satu kali. Jika pada lintasan Euler menghubungkan sebuah simpul yang sama (lintasan kembali ke simpul asal), lintasan itu disebut sirkuit Euler. Graf yang memiliki sirkuit Euler sering disebut sebagai graf Euler dan graf yang terdapat lintasan Euler disebut sebagai graf semi-Euler. Sebuah graf tidak berarah dapat diidentifikasi sebagai graf Euler (terdapat sirkuit Euler) jika dan hanya jika setiap simpul berderajat genap. Sebuah graf tidak berarah dapat dikatakan graf semi-Euler (terdapat lintasan Euler) jika dan hanya jika graf tersebut terhubung dan memiliki dua buah simpul berderajat ganjil atau tidak ada simpul berderajat ganjil sama sekali.

5. Lintasan dan Sirkuit Hamilton

Lintasan Hamilton adalah lintasan yang melalui setiap simpul pada graf tepat satu kali. Jika pada lintasan Hamilton menghubungkan sebuah simpul yang sama (lintasan kembali ke simpul asal), lintasan itu disebut sirkuit Hamilton. Graf yang memiliki sirkuit Hamilton disebut graf Hamilton dan graf yang hanya memiliki lintasan Hamilton saja disebut graf semi-Hamilton. Syarat cukup agar graf sederhana G dengan jumlah simpul n lebih dari tiga disebut graf Hamilton adalah derajat setiap simpul paling sedikit $n/2$.

B. Pohon

1. Definisi Pohon

Pohon (*tree*) adalah graf berarah terhubung yang tidak memiliki sirkuit di dalamnya. Setiap pasang simpul pada pohon terhubung dengan lintasan tunggal.

2. Pohon Merentang Minimum

Pohon merentang (*spanning tree*) dari sebuah graf terhubung adalah upagraf merentang yang merupakan sebuah pohon. Pohon merentang ini didapatkan dengan menghilangkan sirkuit di dalam graf.

Pohon merentang minimum (*minimum spanning tree*) adalah pohon merentang pada sebuah graf yang memiliki total bobot minimum. Pada sebuah graf dimungkinkan adanya lebih dari satu buah pohon merentang minimum. Pohon merentang minimum juga dimungkinkan menghasilkan bobot yang sama dengan pohon rentang minimum lain untuk graf yang sama.

C. Algoritma Greedy

Algoritma *Greedy* adalah salah satu algoritma yang popular untuk memecahkan permasalahan optimasi (persoalan mencari solusi optimum). Suatu solusi dikatakan optimum jika hasil yang dihasilkan adalah hasil yang memiliki nilai minimum atau maksimum. Prinsip yang digunakan pada algoritma *Greedy* adalah prinsip rakus. Pada setiap langkah algoritma *Greedy* akan dihasilkan solusi hasil dari evaluasi pilihan-pilihan yang tersedia. Solusi yang dihasilkan pada tiap langkah tersebut adalah pilihan yang optimum. Harapannya adalah jika pada setiap langkah solusi yang dihasilkan adalah solusi optimum, langkah berikutnya akan mengarah pada solusi yang optimum secara global.

Elemen-elemen pada algoritma *Greedy* adalah himpunan kandidat C, himpunan solusi S, fungsi seleksi, fungsi

kelayakan, dan fungsi obyektif. Algoritma Greedy akan melakukan pencarian sebuah himpunan bagian *S* dari himpunan kandidat *C* dan *S* harus memenuhi beberapa kriteria yang ditetntukan, yaitu menyatakan solusi dan *S* dioptimalisasi oleh fungsi obyektif. Dengan kata lain, algoritma *Greedy* akan mengambil solusi yang optimum denagn fungsi obyektif pada setiap langkah dan solusi yang diambil haruslah lolos pada fungsi kelayakan.

D. Algoritma Dijkstra

Algoritma Dijkstra salah satu algoritma yang menerapkan prinsip algoritma *Greedy* yang ditemukan oleh Edsger W. Dijkstra. Algoritma ini adalah algoritma yang optimal pada persoalan pencarian lintasan terpendek. Secara umum algoritma Dijkstra adalah mengambil sisi yang memiliki bobot minimum untuk setiap langkahnya. Sisi yang diambil menghubungkan sebuah simpul yang sudah terpilih dengan simpul lain yang belum terpilih. Lintasan yang dipilih haruslah lintasan terpendek diantara semua lintasan ke simpul-simpul yang belum terpilih.

E. Algoritma Kruskal

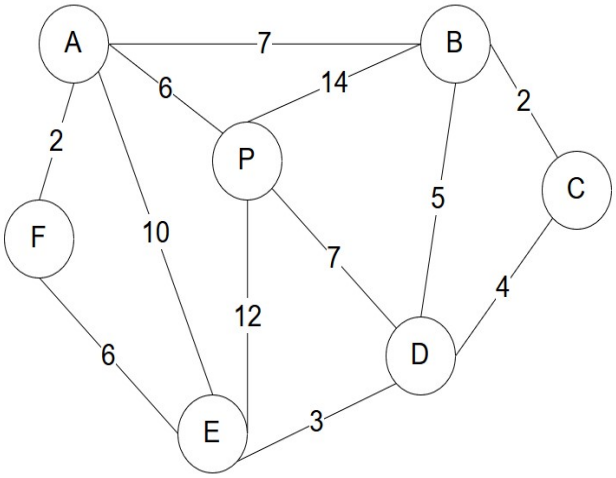
Algoritma Kruskal adalah algoritma lain yang juga menerapkan prinsip algoritma *Greedy*. Algoritma ini adalah algoritma yang akan mencari sebuah pohon merentang minimum (*minimum spanning tree*) pada seabuah graf berbobot yang terhubung. Secara umum, algoritma Kruskal adalah memilih sisi yang memiliki bobot minimum yang tidak membentuk sirkuit pada pohon, kemudian menambahkan sisi tersebut sebagai solusi. Langkah tersebut diulangi sampai sebanyak jumlah simpul dikurangi satu.

III. PENERAPAN ALGORITMA DIJKSTRADALAM MERANCANG TRAYEK ANGKUTAN UMUM YANG OPTIMUM

Pada suatu kota, salah satu hal terpenting untuk menunjang keseharian kota tersebut adalah angkutan umum. Angkutan umum pada suatu kota akan melayani trayek dari terminal pusat *P*, kebeberapa halte dan akan kembali ke terminal pusat setelah mencapai halte akhir trayek tersebut. Misalkan pada suatu kota terdapat 6 buah halte, yakni halte *A*, *B*, *C*, *D*, *E*, dan *F* yang memiliki posisi tertentu pada suatu daerah. Akan dibuat suatu rancangan trayek angkutan umum yang melayani rute dari terminal pusat ke setiap halte dan akan kembali ke terminal pusat setelah mencapai halte terakhir yang memiliki jarak lintasan optimum untuk setiap trayeknya.

Untuk mempermudah dalam memahami persoalan diatas, akan direpresentasikan peta suatu kota menjadi sebuah graf. Setiap halte yang telah disebutkan dan terminal pusat direpresentasikan sebagai simpul graf. Sisi-sisi pada graf merepresentasikan jalan-jalan yang menghubungkan tiap halte atau terminal pada kota tersebut. Bobot-bobot pada sisi graf merepresentasikan jarak atau panjang jalan yang menghubungkan tiap halte atau terminal. Setelah itu akan dicari lintasan-lintasan optimum dari terminal ke setiap halte yang ada dengan menggunakan algoritma Dijkstra dan Kruskal.

Berikut adalah representasi peta dan letak halte serta terminal pada kota tersebut :



Gambar 5 : instansiasi permasalahan menjadi graf

Untuk memperjelas gambar 7, berikut adalah legenda nama-nama simpul pada graf :

P : terminal pusat

A : halte A

B : halte B

C : halte C

D : halte D

E : halte E

F : halte F

Pada graf tersebut, bobot menyatakan jarak-jarak pada tiap halte atau terminal yang bersisian dalam satuan kilometer. Angkutan umum akan berangkat dari terminal pusat menuju tiap halte untuk setiap trayeknya. Akan dioptimasi jarak tempuh angkutan umum untuk setiap trayek yang ada.

Langkah-langkah penyelesaian untuk mencari trayek dengan jarak optimum menggunakan algoritma Dijkstra adalah sebagai berikut :

1. Menginisialisasi bobot tiap halte dari terminal pusat dengan jarak tak hingga
2. Pada setiap langkah, ambil sisi yang memiliki bobot minimum yang menghubungkan sebuah simpul yang sudah terpilih denga simpul lain yang belum terpilih. Kemudian bandingkan lintasantersebut dengan sisi yang tersimpan sebelumnya. Lintasan yang dipilih haruslah merupakan lintasan optimum (terpendek). Kemudian *update* jaraknya.

Berikut adalah penyelesaian untuk permasalahan tersebut menggunakan algoritma Dijkstra :

- Iterasi pertama

  Menginisialisasi bobot tiap halte dengan tak hingga

Tabel 1 : iterasi pertama

Simpul Ekspan	Jarak ke-					
	A	B	C	D	E	F
-	~	~	~	~	~	~

- Iterasi kedua

Tabel 2 : iterasi kedua

Simpul Ekspan	Jarak ke-					
	A	B	C	D	E	F
P	6	14	~	7	12	~
Simpul Ekspan	Lintasan ke-					
	A	B	C	D	E	F
P	P	P	~	P	P	~

- Iterasi ketiga

Tabel 3 : iterasi ketiga

Simpul Ekspan	Jarak ke-					
	A	B	C	D	E	F
A	6	13	~	7	12	8
Simpul Ekspan	Lintasan ke-					
	A	B	C	D	E	F
P	P	PA	~	P	P	PA

- Iterasi keempat

Tabel 4 : iterasi keempat

Simpul Ekspan	Jarak ke-					
	A	B	C	D	E	F
D	6	12	11	7	10	8
Simpul Ekspan	Lintasan ke-					
	A	B	C	D	E	F
P	P	PD	PD	P	PD	PA

- Iterasi kelima

Tabel 5 : iterasi kelima

Simpul Ekspan	Jarak ke-					
	A	B	C	D	E	F
F	6	12	11	7	10	8
Simpul Ekspan	Lintasan ke-					
	A	B	C	D	E	F
P	P	PD	PD	P	PD	PA

- Iterasi keenam

Tabel 6 : iterasi keenam

Simpul Ekspan	Jarak ke-					
	A	B	C	D	E	F
E	6	12	11	7	10	8
Simpul Ekspan	Lintasan ke-					
	A	B	C	D	E	F
P	P	PD	PD	P	PD	PA

- Iterasi ketujuh

Tabel 7 : iterasi ketujuh

Simpul Ekspan	Jarak ke-					
	A	B	C	D	E	F
C	6	12	11	7	10	8
Simpul Ekspan	Lintasan ke-					
	A	B	C	D	E	F
P	P	PD	PD	P	PD	PA

- Iterasi kedelapan

Tabel 8 : iterasi kedelapan

Simpul Ekspan	Jarak ke-					
	A	B	C	D	E	F
B	6	12	11	7	10	8
Simpul Ekspan	Lintasan ke-					
	A	B	C	D	E	F
P	P	PD	PD	P	PD	PA

Karena semua simpul pada graf sudah diperiksa semua, iterasi berhenti dan menghasilkan trayek dari terminal pusat ke tiap halte dengan lintasan dan jarak sebagai berikut:

Tabel 9 : lintasan dan jarak tiap trayek

Trayek	Lintasan	Jarak
A	P → A → P	6
B	P → D → B → D → P	12
C	P → D → C → D → P	11
D	P → D → P	7
E	P → D → E → D → P	10
F	P → A → F → A → P	8

Langkah-langkah penyelesaian untuk mencari trayek dengan jarak optimum menggunakan algoritma Kruskal :

I. Mengurutkan sisi-sisi yang ada berdasarkan bobot yang menaik.
II. Menginisialisasi solusi sebagai pohon kosong
III. Memilih sisi (u,v) dengan bobot minimum yang tidak membentuk sirkuit di pohon solusi. Kemudian tambahkan (u,v) ke dalam pohon solusi. Lakukan langkah ini sebanyak jumlah simpul dikurangi satu.

Berikut adalah penyelesaian permasalahan dengan menggunakan algoritma Kruskal :

Tabel 10 : urutan bobot menaik dari graf

Sisi	Bobot
(A,F)	2
(B,C)	2
(D,E)	3
(C,D)	4
(B,D)	5
(E,F)	6
(A,P)	6
(A,B)	7
(D,P)	7
(A,E)	10
(E,P)	12
(B,P)	14

Setelah mengurutkan semua sisi menjadi terurut naik, kemudian kita akan menjalankan iterasinya :

- Langkah awal
  Pohon solusi masih kosong, semua simpul tidak ada yang terhubung

- Langkah pertama

Tabel 11 : langkah pertama

Sisi	Bobot	Sisi pada Pohon Solusi
(A,F)	2	(A,F)

- Lankah kedua

Tabel 12 : langkah kedua Kruskal

Sisi	Bobot	Sisi pada Pohon Solusi
(B,C)	2	(A,F), (B,C)

- Langkah ketiga

Tabel 13 : langkah ketiga Kruskal

Sisi	Bobot	Sisi pada Pohon Solusi
(D,E)	3	(A,F), (B,C), (D,E)

- Langkah keempat

Tabel 14 : langkah keempat Kruskal

Sisi	Bobot	Sisi pada Pohon Solusi
(C,D)	4	(A,F), (B,C), (D,E), (C,D)

- Langkah kelima

Tabel 15 : langkah kelima Kruskal

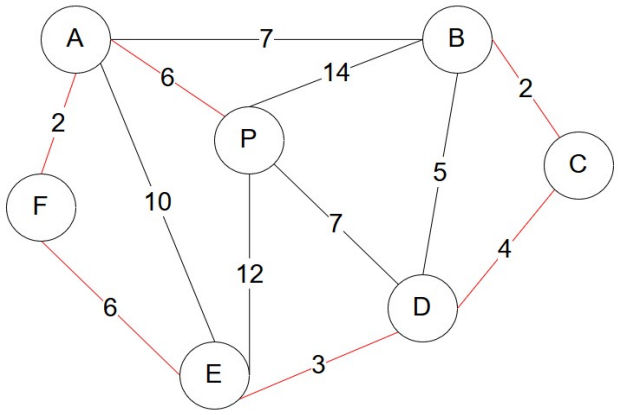
Sisi	Bobot	Sisi pada Pohon Solusi
(E,F)	6	(A,F), (B,C), (D,E), (C,D), (E,F)

- Langkah keenam

Tabel 16 : langkah keenam Kruskal

Sisi	Bobot	Sisi pada Pohon Solusi
(A,P)	6	(A,F), (B,C), (D,E), (C,D), (E,F), (A,P)

Karena iterasi sudah dilakukan sebanyak jumlah simpul dikurangi satu, algoritma berhenti dan didapatkan pohon merentang minimum sebagai berikut :



Gambar 6 : pohon merentang minimum peta kota

Lintasan graf yang berwarna merah adalah pohon merentang minimum yang dihasilkan. Jarak (dalam kilometer) tiap halte dari terminal pusat adalah sebagai berikut :

Tabel 17 : Jarak tiap halte terhadap terminal pusat

Halte	Jarak
A	6
B	23
C	21
D	17
E	14
F	8

Pada algoritma Kruskal untuk permasalahan ini, hanya perlu dibuat satu trayek saja, sebab jika kita membuat trayek dari terminal pusat ke halte *B*, semua halte lain juga akan dikunjungi sebelum akhirnya angkutan umum sampai di halte *B*.

IV. ANALISIS TRAYEK YANG DIHASILKAN PANA ALGORITMA DIJKSTRA DAN ALGORITMA KRUSKAL

Pada permasalahan mencari lintasan terpendek dari terminal menuju halte-halte untuk angkutan umum suatu kota, algoritma Dijkstra dapat memberikan jarak lintasan yang lebih optimal dibandingkan dengan algoritma Kruskal. Karena dalam algoritma Dijkstra, untuk setiap langkah yang dipilih adalah lintasan yang optimal dari terminal ke setiap halte yang ada. Hal ini akan menghasilkan lintasan yang optimal juga secara global untuk setiap halte yang ada. Berbeda halnya dengan algoritma Kruskal yang mencari total lintasan dengan jarak yang optimum untuk setiap langkahnya. Pada algoritma Kruskal, yang menjadi fokus adalah mencari nilai optimum untuk lintasan secara keseluruhan bukan tiap simpul. Keoptimalan algoritma Dijkstra dibandingkan Kruskal dapat dilihat dari jarak yang dihasilkan untuk tiap halte pada *tabel 9* dan *tabel 17*.

V. KESIMPULAN

Algoritma Dijkstra ini sangat baik dalam menghasilkan trayek dengan jarak yang optimum disbandingkan dengan algortima Kruskal. Algoritma ini juga bisa diterapkan dalam permasalahan mencari lintasan terpendek. Namun, algoritma Kruskal dapat memberikan solusi optimum untuk jumlah trayek yang dibutuhkan karena algoritma ini menjamin semua simpul bisa diakses. Akan tetapi, algoritma Kruskal ini sangat bergantung pada graf yang dihasilkan dari permasalahan yang ada. Dengan didapatnya trayek ini, diharapkan mampu menjadi rujukan dalam merencanakan trayek angkutan umum yang optimal.

VI. UCAPAN TERIMA KASIH

Sebelumnya penulis ingin mengucapkan syukur kepada Allah SWT karena berkat rahmat-Nya penulis dapat menyelesaikan makalah ini. Selanjutnya penulis ingin menyampaikan terima kasih kepada orang tua penulis dan juga kepada dosen-dosen pengajar mata kuliah Strategi Algoritma 2018/2019.

REFERENSI

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/Algoritma-Greedy-(2019).pdf diakses pada 25 April 2019 pukul 21.55

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf diakses pada 25 April 2019 pukul 21.57

[3] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf diakses pada 25 April 2019 pukul 22.05

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

Muhammad Nurdin Husen - 13517112