PENERAPAN ALGORITMA *DIJKSTRA* PADA APLIKASI JASA TRANSPORTASI *ONLINE* DI KOTA MEDAN

Abdul Alfattah Hidayah[1)], Samsudin,[2)], dan Triase[3)]

[1,2,3]Sistem Informasi, Fakultas Sains dan Teknologi Universitas Islam Negeri Sumatera Utara
Email: abdul.alfattah@uinsu.ac.id
Email: mr.samsudin@gmail.com
Email: triase@uinsu.ac.id

ABSTRACT

This study aims to determine the shortest route in the application of online transportation services which is the main attraction for entrepreneurs who want to start looking for their fortune in technology and transportation, given the enormous opportunity where the human population is very large. increased, and some human activities had to be carried out. fulfilled. Some online transportation service companies use paid features provided by Google to determine the shortest distance and route, thus costing a lot of money. In this study, to determine the shortest route on a journey, it is necessary to calculate using an algorithm method, namely the dijkstra algorithm which is an algorithm used to solve the problem of the shortest route or the shortest path from one point to another at a point. weighted graph, The distance between vertices is the weight value of each edge in the graph. A graph that has a weight must be positive (weight >= 0). Dijkstra's algorithm itself uses a greedy strategy in its operation, where in each step the chosen one with the smallest weight connects the selected node with other nodes that have not been selected.

Keywords: *dijkstra algorithm, online transportation service, shortest route,*

PENDAHULUAN

Di era *modern* saat ini, manusia sudah banyak melakukan revolusi ilmiah, baik dalam bidang fisika, astronomi, kimia, biologi, teknologi, dan beberapa ilmu pengetahuan lain. Salah satu bidang ilmiah yang mempunyai perkembangan sangat pesat adalah pada bidang teknologi. Teknologi sudah menjadi kebutuhan primer pada manusia saat ini, salah satunya ada pemanfaatan teknologi pada bidang angkutan dan transportasi yaitu ojol (ojek *online*) yang dapat mengantarkan penumpang kemana saja dan kapan saja. Aplikasi ojol inipun sudah menjadi daya tarik tersendiri untuk para pengusaha yang ingin memulai mencari peruntungan pada bidang teknologi dan transportasi, mengingat peluang yang cukup besar dimana jumlah populasi manusia yang semakin banyak, dan beberapa kesibukan manusia yang harus dipenuhi. Aplikasi ojol inipun tentunya akan dapat digunakan dalam beberapa macam aktifitas manusia sehari-hari seperti mengantar ke kantor, sekolah, pengantaran paket.

Beberapa perusahaan jasa transportasi *online* menggunakan fitur berbayar yang disediakan oleh *google* untuk menentukan jarak dan rute terpendek pada sebuah perjalanan, sehingga memakan biaya yang tidak sedikit dalam menentukan jarak dan rute terpendek tersebut. Untuk dapat menghemat biaya yang akan keluar, dibutuhkan sebuah sistem ataupun algoritma yang dapat menentukan rute terpendek pada sebuah perjalanan sehingga tidak perlu menggunakan fitur berbayar yang sudah disediakan oleh *google*. Dalam penelitian ini, untuk menentukan rute terpendek dalam sebuah perjalanan, maka dibutuhkan sebuah perhitungan dengan menggunakan metode algoritma yaitu algoritma *dijkstra* yang merupakan sebuah algoritma yang digunakan untuk menyelesaikan persoalan pencarian rute terpendek ataupun lintasan terpendek dari satu *vertex* ke *vertex* yang lainnya pada suatu *graf* yang berbobot, jarak antara *vertex* adalah nilai bobot dari setiap *edge* pada *graph*. Suatu *graf* yang mempunyai bobot, harus mempunyai nilai yang positif (bobot >= 0). Algoritma *Dijkstra* sendiri menggunakan strategi *greedy* dalam pengerjaannya, dimana pada setiap langkah dipilih sisi dengan bobot terkecil yang menghubungkan sebuah simpul yang sudah terpilih dengan simpul lain yang belum terpilih

(Harahap & Khairina, 2017). Node digunakan didalam perhitungan pada algoritma *dijkstra,* dikarenakan algorima *Dijkstra* menggunakan *graph* berarah pada pengerjaanya untuk penentuan rute lintasan terpendek. Algoritma ini berfungsi untuk menemukan lintasan terpendek berdasarkan titik yang mempunyai bobot paling kecil ataupun terkecil dari satu titik ke titik lainnya (Ardana & Saputra, 2016). Sedangkan menurut (Triase & Aprilia, 2020) algoritma *Dijkstra* merupakan salah satu algoritma yang banyak digunakan dan sangat popular dalam menyelesaikan sebuah permasalahan dalam pencarian jalur terpendek. Dalam penyelesaian-nya algoritma ini memiliki teknik sumber tunggal untuk menentukan jalur terpendek. Algoritma ini mempunya sebuah teori dasar yaitu, dalam setiap *vertex* tertentu mempunyai peran sebagai titik jalur terpendek untuk ke semua titik lain yang ingin dituju. Algoritma *Dijkstra* ini tidak hanya terfokus dalam mencari jalur terpendek dari setiap *vertex*, namun dapat mencari jalur ke semua *vertex* yang ada.

METODE PENELITIAN

Pendekatan pada penelitian ini didasarkan dengan menggunakan model pengembangan sistem pada perangkat lunak yaitu metode pengembangan sistem *Prototype*. Menurut (Widiyanto, 2018), metode *Prototype* merupakan sebuah proses dalam pembuatan model secara sederhana terhadap sebuah *software* yang memungkinkan pengguna memiliki gambaran mengenai sistem yang akan dibuat dan memungkinkan melakukan pengujian terhadap sistem. Metode *Prototype* sendiri memungkinkan untuk seorang pengembang dan pemakai dapat melakukan interaksi dalam proses pembuatan sistem, sehingga memudahkan pengembang dalam melakukan pemodelan terhadap sistem yang akan dibuat.

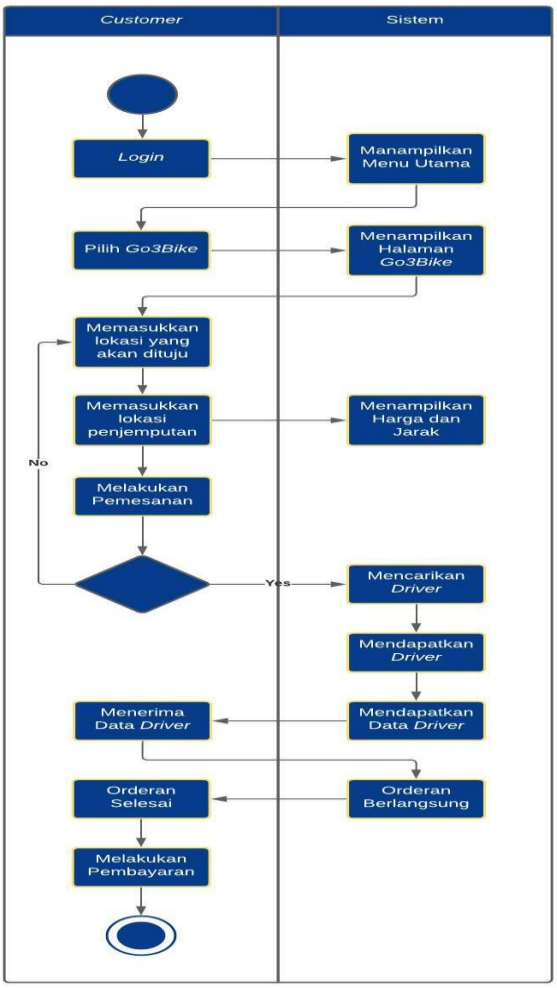
Menurut (Mubarok et al., 2015), adapun tahapan-tahapan dalam penggunaan metode *Prototype* yaitu:

1. *Listen to Customer*, dalam tahap ini penulis melakukan komunikasi untuk mengidentifikasi tentang bagaimana sistem yang diinginkan dan diharapkan pada perusahaan yang berkaitan, sehingga penulis dapat merancang sebuah sistem usulan yang didapatkan melalui identifikasi pada keinginan perusahaan terkait. Penulis akan melakukan identifikasi sistem, yakni:
2. *Build Prototype*, pada tahap ini bertujuan untuk memberikan gambaran tampilan yang akan dikerja-kan beserta gambaran tentang tahap-tahap yang akan dikerjakan. Dalam melakukan tahap ini dapat sangat membantu dalam mendefinisikan arsitektur sistem yang akan dibuat secara keseluruhan.
3. *Revise* (*Prototype Evaluation*), pada tahap ini pelanggan akan melakukan evaluasi kepada *prototype* yang sudah di desain, apabila sudah sesuai maka akan masuk ke proses selanjutnya yaitu *coding*. Apabila *prototype* belum sesuai keinginan maka penulis akan melakukan revisi terhadap *prototype* dengan mengulang langkah-langkah sebelumnya.
4. *Coding*, merupakan penerjemahan desain dalam bahasa pemrograman yang bisa dikenali oleh komputer. Dalam penelitian ini, penulis akan melakukan *coding* menggunakan bahasa pemrograman Java melalui *Android Studio*.
5. *Testing*, pada tahap ini penulis akan menguji coba sistem yang sudah dibuat agar benar-benar sesuai dengan kebutuhan. Tahapan yang dilakukan untuk testing adalah pengetesan program untuk menemukan kesalahan-kesalahan ataupun *bug* yang mungkin terjadi.
6. *Implementation and Maintenance*, pada tahap ini sistem yang sudah siap untuk digunakan akan di implementasikan pada pengguna dan nantinya melakukan pengembangan lebih lanjut terhadap sistem tersebut.

HASIL DAN PEMBAHASAN

Tahap Perancangan

Dalam melakukan orderan, seorang aktor *customer* pastinya mempunyai aktifitas-aktifitas yang akan dilalui agar dapat melakukan pesanan terhadap jasa transportasi *online* yang akan dirancang. Pada *activity* ini, *user* dapat melakukan pemesanan jasa *transportasi online*, dimana *customer* setelah melakukan *login* dan berada pada halaman utama, *customer* dapat memilih menu *Bike* kemudian memasukkan alamat yang akan dituju, dan juga memasukkan alamat penjemputan. Setelah mengisi data alamat yang dituju dan juga alamat penjemputan, maka *customer* dapat langsung melakukan proses transaksi ataupun melakukan pemesanan. Setelah melakukan pemesanan, sistem secara otomatis akan mencarikan *driver* yang sedang *online*. Orderan akan segera berlangsung ketika sistem sudah mendapatkan *driver*. Ketika orderan telah selasai, maka *customer* akan melakukan aktifitas pembayaran. Adapun aktifitas-aktifitas pada *customer* telah dirancang oleh penulis, yakni:

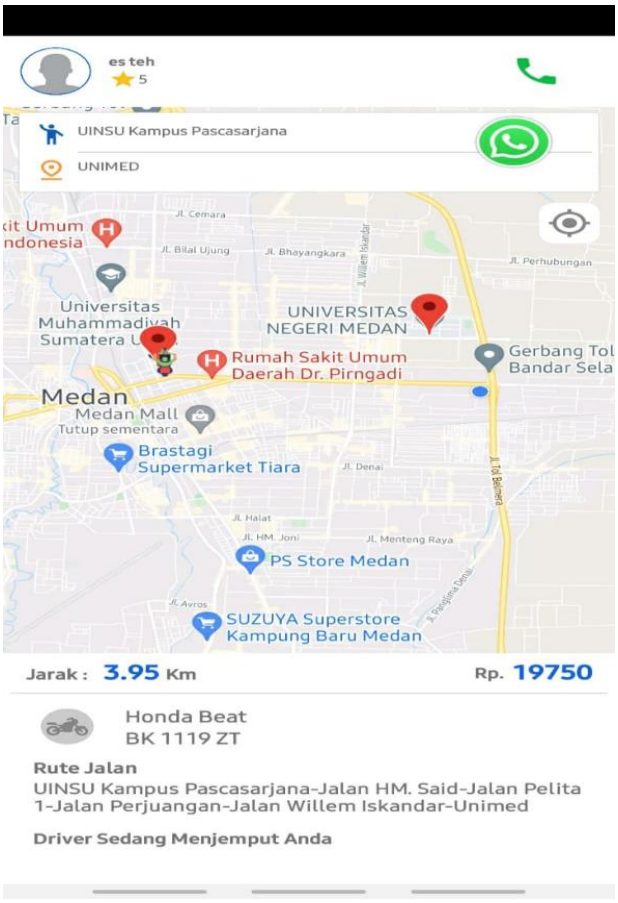


Gambar 1 *Diagram Activity* Aplikasi yang akan dibangun

Tahap Implementasi

Gambar 2 Menu Utama Aplikasi Jasa Transportasi *Online*

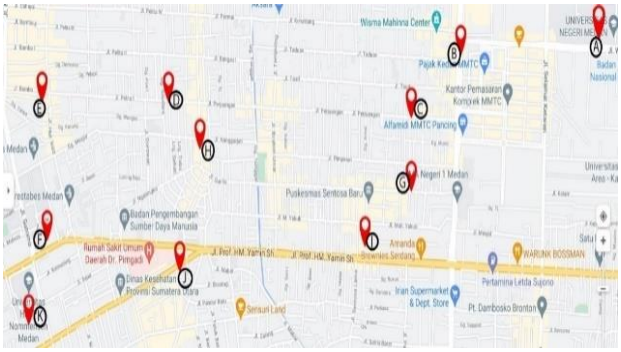
Gambar 3 Menu *Bike* Aplikasi Jasa Transportasi *Online*



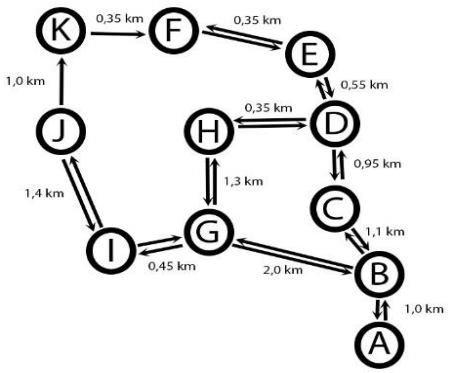
Gambar 4 Tampilan Mendapatkan *Driver* dan Keterangan Rute Terpendek Algoritma *Dijkstra*

Tahap Pengujian Algoritma *Dijkstra*

Pada penelitian ini, aplikasi jasa transportasi *online* yang akan dibangun nantinya menggunakan algoritma *dijkstra* sebagai penentuan rute terpendek dan juga jarak yang akan ditempuh. Adapun contoh kasus yang diangkat penulis mengenai pencarian rute terpendek seperti tertera pada gambar berikut ini:



Gambar 5 Contoh Kasus Algoritma *Dijkstra*



Gambar 6 Graf Contoh Kasus Algoritma Dijkstra

Seperti yang tertera pada gambar 6 diatas yang merupakan sebuah *graf* berarah dan berbobot yang merupakan hasil penerjemahan dari gambar 5, dapat dilihat pada kasus yang diangkat penulis menetapkan *node* ataupun *vertex* berjumlah sebanyak 11 titik *node*. Adapun *node-node* tersebut yakni:

Tabel 1 Daftar Lokasi Rute Terpendek

Node	**Daerah/Jalan**
A	UNIMED
B	Jalan Willem Iskandar
C	Jalan Perjuangan
D	Jalan Pelita I
E	Jalan H. M Said
F	Kampus Pasca Sarjana UINSU
G	Jalan Gurilla
H	Jalan Rakyat
I	Jalan Sentosa Baru
J	Jalan Prof. M. Yamin
K	Universitas HKBP Nommensen

Dari Tabel 1, terdapat jarak antar *vertex* ataupun *node* yang sudah ditentukan penulis berdasarkan jarak yang tertera pada *Google Maps* menggunakan satuan jarak kilometer (km) yakni:

Tabel 2 Tabel Jarak Antara Node

Node Awal	**Node Tujuan**	**Jarak (Kilometer)**
A	B	1,0 km
B	A	1,0 km
B	C	1,1 km
B	G	2,0 km
C	B	1,1 km
C	D	0,95 km
D	C	0,95 km
D	E	0,55 km
D	H	0,35 km
E	D	0,55 km
E	F	0,35 km
F	E	0,35 km
G	B	2,0 km
G	H	1,3 km
G	I	0,45 km
H	D	0,35 km
H	G	1,3 km
I	G	0,45 km
I	J	1,4 km
J	I	1,4 km
J	K	1,0 km
K	F	0,35 km

Adapun perhitungan manual algoritma *dijkstra* yang telah dihitung penulis menurut contoh kasus yang diangkat penulis seperti yang tertera pada gambar 5 sebagai titik-titik lokasi yang tertera pada *google maps*, gambar 6 yang merupakan hasil *graph* dari terjemahan pada gambar 5, dan juga Tabel 1 yang didalamnya terdapat daftar lokasi dan juga jarak antara satu *vertex* dengan *vertex* yang lainnya adalah sebagai berikut:

Tabel 3 Hasil Perhitungan Manual *Dijkstra Node* F

C	A	B	C	D	E	F	G	H	I	J	K
	∞	∞	∞	∞	∞	0_F	∞	∞	∞	∞	∞
F	∞	∞	∞	∞	0,35	0_F	∞	∞	∞	∞	∞
E	∞	∞	∞	$0,9_E$	0,35	0_F	∞	∞	∞	∞	∞
D	∞	∞	1,85	$0,9_E$	0,35	0_F	∞	1,25	∞	∞	∞
H	∞	∞	1,85	$0,9_E$	0,35	0_F	2,55	1,25	∞	∞	∞
C	∞	2,95	1,85	$0,9_E$	0,35	0_F	2,55	1,25	∞	∞	∞
G	∞	2,95	1,85	$0,9_E$	0,35	0_F	2,55	1,25	$3,0_G$	∞	∞
B	3,95	2,95	1,85	$0,9_E$	0,35	0_F	2,55	1,25	$3,0_G$	∞	∞
I	3,95	2,95	1,85	$0,9_E$	0,35	0_F	2,55	1,25	$3,0_G$	$4,4_I$	∞
A	3,95	2,95	1,85	$0,9_E$	0,35	0_F	2,55	1,25	$3,0_G$	$4,4_I$	∞
J	3,95	2,95	1,85	$0,9_E$	0,35	0_F	2,55	1,25	$3,0_G$	$4,4_I$	$5,4_J$

Keterangan:
c = *current*
∞ = tak hingga (tidak terhubung)
d = *distance*

Tabel 4 Hasil Perhitungan Manual *Dijkstra Node* F

Node	Rumus (km)	*Node*	Rumus (km)
A	= *c*+*d* = B + 1,0 = 2,95 + 1,0 = 3,95 km	G	= *c*+*d* = H + 1,3 = 1,25 + 1,3 = 2,55 km
B	= *c* + *d* = C + 1,1 = 1,85+ 1,1 = 2,95 km	H	= *c*+*d* = D + 0,35 = 0,9 + 0,35 = 1,25 km
C	= *c*+*d* = D + 0,95 = 0,9 + 0,95 = 1,85 km	I	= *c*+*d* = G + 0,45 = 2,55 + 0,45 = 3 km
D	= *c*+*d* = E + 0,55 = 0,35 + 0,55 = 0,9 km	J	= *c*+*d* = I + 1,4 = 3 + 1,4 = 4,4 km
E	= *c*+*d* = F + 0,35 = 0 + 0,35 = 0,35 km	K	= *c*+*d* = J + 1,0 = 4,4 + 1,0 = 5,4 km
F	= *c*+*d* = 0 + 0 = 0 km		

Berdasarkan perhitungan *node* F, dapat disimpulakan *node* F mempunyai panjang lintasan masing-masing, yakni:

a. F → A = F-E-D-C-B-A dengan panjang lintasan 3,95km
b. F → B = F-E-D-C-B, dengan panjang lintasan 2,95km
c. F → C = F-E-D-C, dengan panjang lintasan 1,85km
d. F → D = F-E-D, dengan panjang lintasan 0,9km
e. F → E = dengan panjang lintasan 0,35km
f. F → G = F-E-D-H-G, dengan panjang lintasan 2,55km
g. F → H = F-E-D-H, dengan panjang lintasan 1,25km
h. F → I = F-E-D-H-G-I, dengan panjang lintasan 3,0km
i. F → J = F-E-D-H-G-I-J, dengan panjang lintasan 4,4km
j. F → K = F-E-D-H-G-I-K, dengan panjang lintasan 5,4km

KESIMPULAN

Berdasarkan penelitian yang telah penulis lakukan yaitu adalah mengimplementasikan algoritma *Dijkstra* dalam aplikasi transportasi *online* dapat ditarik kesimpulan yaitu Algoritma *dijkstra* sangat membantu mendapatkan rute terpendek pada sebuah orderan dengan sangat optimal dan dapat dimanfaatkan dalam pemetaan pada aplikasi transportasi *online* yang dibuat untuk mencari rute-rute terpendek pada sebuah perjalanan dengan memanfaatkan *node-node* yang sudah ditentukan pada suatu *graf* yang berbobot, dimana masing-masing pada *node* ataupun *vertex* tersebut sudah memiliki nilai jarak yang sudah ditetapkan.

DAFTAR PUSTAKA

Ardana, D., & Saputra, R. (2016). Penerapan Algoritma Dijkstra pada Aplikasi Pencarian Rute Bus Trans Semarang. *Seminar Nasional Ilmu Komputer*.

Harahap, M. K., & Khairina, N. (2017). Pencarian Jalur Terpendek dengan Algoritma Dijkstra. *SinkrOn*, *2*(2), 18. https://doi.org/10.33395/sinkron.v2i2.61

Mubarok, F., Harliana, H., & Hadijah, I. (2015). Perbandingan Antara Metode RUP dan Prototype Dalam Aplikasi Penerimaan Siswa Baru Berbasis Web. *Creative Information Technology Journal*, *2*(2), 114. https://doi.org/10.24076/citec.2015v2i2.42

Triase, T., & Aprilia, R. (2020). Implementasi Penyaluran Paket Online Shop Menggunakan Algoritma FIFO dan Dijkstra. *Query: Journal of Information Systems*, *4*(1), 60–67.

Widiyanto, W. W. (2018). Analisa Metodologi Pengembangan Sistem Dengan Perbandingan Model Perangkat Lunak Sistem Informasi Kepegawaian Menggunakan Waterfall Development Model, Model Prototype, Dan Model Rapid Application Development (Rad). *Jurnal Informa Politeknik Indonusa Surakarta ISSN*, *4*(1), 34–40. http://www.informa.poltekindonusa.ac.id/index.php/informa/article/view/34