semanTIK, Vol.3, No.1, Jan-Jun 2017, pp. 95-106
ISSN : 2502-8928 (Online)

TRAVELLING SALESMAN PROBLEM (TSP) UNTUK MENENTUKAN RUTE TERPENDEK BAGI KURIR KOTA KENDARI MENGGUNAKAN ALGORITMA GREEDY BERBASIS ANDROID

Augridita Prawidya C[*1], Bambang Pramono[2], L.M Bahtiar Aksara[3]
[*1,2,3]Jurusan Teknik Informatika, Fakultas Teknik Universitas Halu Oleo, Kendari
e-mail: [*1]**augriditaprawidya@gmail.com**, [2]bambangparamono09@gmail.com,
[3]anamogane@gmail.com

Abstrak

Kurir pelayanan *delivery order* Kota Kendari seringkali mengalami kesulitan untuk menentukan lintasan terpendek dalam pengantaran barang karena banyaknya alternatif jalan yang ada. Tidak jarang pula, kurir tidak mengenal dengan baik alamat yang dituju sehingga tidak dapat memperhitungkan jarak alamat tersebut dan bisa jadi melewati titik yang sama berulangkali sehingga lintasan yang dilewati tidak efisien. Untuk itu diperlukan suatu sistem yang dapat membantu kurir dalam menentukan lintasan terpendek dan dapat merepresentasikan data yang ada. Data tersebut dapat disimpan, diolah, dan disajikan dalam bentuk yang lebih sederhana serta terkomputerisasi sehingga memudahkan dalam penentuan lintasan terpendek.

Travelling Salesman Problem (TSP) adalah pencarian rute terpendek atau jarak minimum oleh seorang *salesman* dari suatu kota ke n-kota tepat satu kali dan kembali ke kota awal keberangkatan. TSP dapat diterapkan pada graph komplit berbobot yang memiliki total bobot sisi minimum, dimana bobot pada sisi adalah jarak. Rute TSP ini memuat semua titik pada graph tersebut tepat satu kali. Proses optimalisasi ini dilakukan dengan memperhitungkan fungsi heuristik yang akan mempersempit ruang pencarian. Hasil dari aplikasi ini berupa urutan alamat yang akan dikunjungi oleh kurir beserta lintasan terpendek antar alamat pada peta Kota Kendari.

Algoritma *Greedy* adalah algoritma yang memecahkan masalah langkah demi langkah dan merupakan salah satu metode dalam masalah optimasi. Pendekatan yang dilakukan dalam Algoritma *Greedy* adalah membuat pilihan yang terlihat memberikan perolehan terbaik yaitu dengan membuat pilihan *optimum local* pada setiap langkah dan diharapkan akan mendapatkan *solution optimum global.*

Kata kunci— *Android*, Ponsel, Rute Terpendek, *Travelling Salesman Problem*, SP, Kurir, Algoritma *Greedy*.

Abstract

Courir service order in Kendari city , sometimes to find trouble to determine the short route in the delivery, because of the many route which exist. For that we need system that can assists couriers in determining the short route and can represent data, and the data can be stored, processed, and presented in a simple form and computerized, to making it easier to determine the short route.

Travelling Salesman Problem (TSP) is the search for the shortest route or the minimum distance. For a salesman from one city to the n-city exactly once one return and bact to the first departure. TSP can be applied for complete graph weight having a total weight of the minimum side, where the weight of the side is the distance. TSP service contains all the points on the graph exactly once. This optimazation process is done by calculating the heuristic function that will narrow the search space. In the result of this application in the form of the order adres will be visited by courier along the short routest, between an adres on a map of the city of kendari.

The greedy algorithm is algorithm which solves the problem step by step and is one method of optimization problems the approach taken in the greedy algoritm is making choices that appear to

offer the best acquiston is to created a local optimum choice at each step and is expected to get a global optimum solution.

Keywords— *Android, Smartphone, Shortest Path, Travelling Salesman Problem, Courier,Greedy Algorithm*

1. PENDAHULUAN

Kemajuan teknologi tidak hanya menuntut kecepatan penyebaran informasi tetapi juga kecepatan pada aspek-aspek lainnya seperti penyampaian barang titipan, perbekalan, barang berharga atau bahkan dokumen usaha para pebisnis juga memiliki arus perpindahan yang cukup tinggi. Hal ini telah menginspirasi berdirinya berbagai usaha ekspedisi untuk memberikan jasa pengantaran.

Kurir perusahaan ekspedisi sebagai ujung tombak pelayanan seringkali mengalami kesulitan untuk menentukan rute yang akan dilalui dalam pengantaran barang karena banyaknya alamat yang menjadi tujuannya, untuk itu diperlukan suatu sistem yang dapat membantu kurir dalam menentukan rute yang akan dilaluinya dalam sekali pengantaran dan dapat menampilkan hasil pencarian jarak pada lebih dari satu titik tujuan yang telah diperoleh. Data tersebut dapat disimpan, diolah, dan disajikan dalam bentuk yang lebih sederhana serta terkomputerisasi sehingga memudahkan dalam penentuan lintasan terpendek.

Di kota Kendari, transportasi adalah persoalan penting bagi masyarakat kota yang dinamis. Luasnya sebuah kota serta banyaknya jalan raya seringkali menyulitkan seseorang untuk mencari rute optimum, baik dari segi jarak maupun biaya yang dikeluarkan untuk berpergian dari satu tempat ke tempat lainnya.

Tujuan rute optimum adalah mendapatkan jarak yang optimal maupun biaya yang optimal untuk menempuh perjalanan dari titik asal ke titik tujuan. Penulis mengamati bahwa metode yang digunakan orang-orang pada umumnya terutama perusahaan bidang ekspedisi untuk mencari rute optimum tersebut belum cukup memuaskan.

Solusi yang dapat menyelesaikan permasalahan tersebut adalah dengan membangun suatu sistem yang dapat melakukan pencarian alamat yang memiliki rute perjalanan terpendek. Kurir dapat memilih alamat-alamat yang akan dikunjungi melalui sistem kemudian sistem akan memberikan daftar alamat beserta rute perjalanan terpendek yang dipilihnya. Solusi yang ditawarkan akan diwujudkan menggunakan *Algoritma Greedy*.

Algoritma Greedy merupakan algoritma yang membentuk solusi langkah per langkah. Pada setiap langkah tersebut akan dipilih keputusan yang paling optimal. Algoritma ini adalah salah satu penyelesaian *Travelling Salesman Problem* (TSP), TSP adalah permasalahan untuk mencari rute perjalanan terpendek setelah mengunjungi semua titik lokasi sehingga hal ini dapat diterapkan pada penelitian, kurir dapat mencari alamat dengan rute perjalanan terpendek setelah mengunjungi seluruh titik secara optimal.

Penelitian sebelumnya dilakukan oleh [1] dengan judul Perbandingan *Algoritma Greedy* dan *Brute Force* Dalam Simulasi Pencarian Koin. Penelitian ini membahas tentang perbandingan antara *AlgoritmaGreedy* dan *Brute Force* yang digunakan sebagai solusi masalah dalam simulasi pencarian koin. Simulasi pencarian koin ini direpresentasikan dengan sebuah papan berukuran *n x n* dengan koin tersebar secara acak di tiap kotak pada papan tersebut.

Terdapat sebuah robot yang akan bertugas untuk mengambil koin tersebut satu persatu. Algoritma *Greedy* yang digunakan adalah *Greedy by range* yaitu robot akan melakukan penyisiran area dengan titik awal berada pada kotak robot. Penyisiran dilakukan ke kanan dan ke bawah. Dapat dipastikan bahwa dengan adanya penyisiran, tidak ada koin yang tidak terambil. Berbeda dengan *Brute Force*, pencarian koin dilakukan dengan menelusuri setiap kotak, demikian seterusnya sehingga setiap kotak berhasil disinggahi. Dengan *Brute Force*, dapat dipastikan bahwa juga tidak mungkin ada koin yang tidak terambil

Tujuan dari penelitian ini adalah untuk merancang suatu program yang dapat menentukan rute terpendek lintasan yang akan dilalui dengan menggunakan *AlgoritmaGreedy* yang dapat membantu dan mempermudah kurir untuk mendapatkan urutan lintasan yang

akan dilaluinya dalam melakukan pengantaran barang.

2. METODE PENELITIAN

2.1 Android

Android adalah sebuah kumpulan perangkat lunak untuk perangkat *mobile* yang mencakup sistem operasi, *middleware* dan aplikasi utama *mobile.*Android memiliki empat karakteristik sebagai berikut : [2]

1. Terbuka

Android merupakan *open source*, dapat secara bebas diperluas untuk memasukkan teknologi baru yang lebih maju pada saat teknologi tersebut muncul. *Platform* ini akan terus berkembang untuk membangun aplikasi *mobile* yang inovatif.

2. Semua aplikasi dibuat sama

Android tidak memberikan perbedaan terhadap aplikasi utama dari telepon dan aplikasi pihak ketiga (*third-party application*). Semua aplikasi dapat dibangun untuk memiliki akses yang sama terhadap kemampuan sebuah telepon dalam menyediakan layanan dan aplikasi yang luas terhadap para pengguna.

3. Memecahkan hambatan pada aplikasi

Android memecah hambatan untuk membangun aplikasi yang baru dan inovatif. Misalnya, pengembang dapat menggabungkan informasi yang diperoleh dari *web* dengan data pada ponsel seseorang seperti kontak pengguna, kalender, atau lokasi geografis.

4. Pengembangan aplikasi yang cepat dan mudah

Android menyediakan akses yang sangat luas kepada pengguna untuk menggunakan *library* yang diperlukan dan *tools* yang dapat digunakan untuk membangun aplikasi yang semakin baik. Android memiliki sekumpulan *tools* yang dapat digunakan sehingga membantu para pengembang dalam meningkatkan produktivitas pada saat membangun aplikasi yang dibuat [3].

2.2 Graf

a) Definisi Graf

Graf *G* didefenisikan sebagai pasangan himpunan (*V,E*), ditulis dengan notasi *G* = (*V,E*), yang dalam hal ini *V* adalah himpunan tidak kosong dari simpul-simpul (*vertices* atau *node*) dan *E* adalah himpunan sisi (*edges* atau *arcs*) yang menghubungkan sepasang simpul [4].

Simpul pada graf dapat dinomori dengan huruf, seperti *a,b,c,d, ...*, atau dengan bilangan asli 1,2,3, … atau juga gabungan dengan keduanya. Sedangkan untuk sisi yang menghubungkan antara simpul *u* dan *v* dinyatakan dengan (*u,v*) atau dapat dinyatakan dengan lambang e_1,e_2,e_3, … dengan 1,2,3 adalah indeks. Dapat dikatakan bahwa jika *e* merupakan sisi yang menghubungkan simpul *u* dengan *v*, maka *e* dapat ditulis sebagai *e* = (*u,v*).

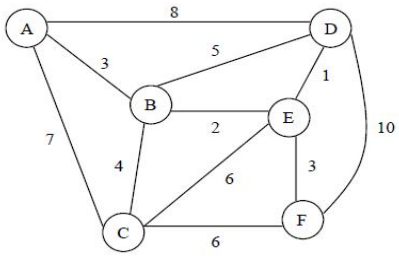
Dalam aplikasinya, setiap simpul pada graf dapat dijadikan sebagai objek kehidupan, yaitu sebagai objek titik jaringan pesan atau komunikasi, lokasi penempatan kerja, titik kota, jalur transportasi dan lain sebagainya. Sedangkan untuk sisi graf dijadikan sebagai bobot jarak, waktu, biaya dan kendala lainnya. Dan juga busur (*arcs*) adalah yang menunjukan hubungan atau relasi dari sepasang simpul.

b) Jenis Graf

Berdasarkan orientasi arah pada sisi dan bobotnya, maka secara umum graf dibedakan atas empat jenis : [5]

1. Graf tidak berarah dan berbobot

Graf yang setiap sisinya tidak mempunyai arah anak panah tetapi memiliki bobot pada setiap sisinya. Urutan pasangan simpul yang terhubung oleh sisi tidak diperhatikan. Sehingga *(u,v) = (v,u)* adalah sisi yang sama. Gambar 1 menunjukkan graf tidak berarah dan berbobot.

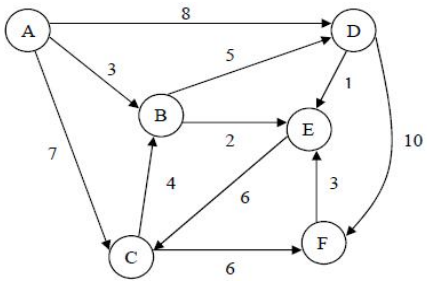


Gambar 1 Graf Tidak Berarah dan Berbobot

2. Graf berarah dan berbobot

Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah. Secara umum sisi berarah disebut dengan

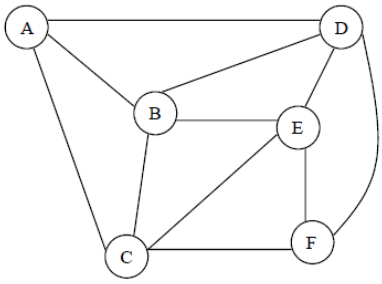
busur (*arc*). Pada graf berarah *(u,v)* dan *(v,u)* menyatakan dua buah busur yang berbeda, dalam arti kata bahwa *(u,v)* □*(v,u)*. Jadi untuk busur *(u,v)* simpul *u* dinamakan simpul asal dan simpul *v* dinamakan simpul terminal atau simpul tujuan. Gambar 2 menunjukkan graf berarah dan berbobot.



Gambar 2 Graf Berarah dan Berbobot

3. Graf tidak berarah dan tidak berbobot

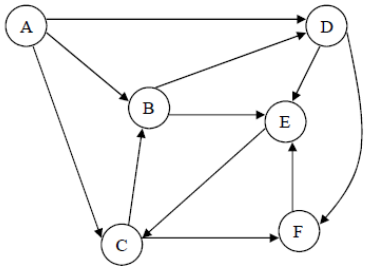
Graf yang setiap sisinya tidak mempunyai arah dan tidak mempunyai bobot apapun. Gambar 3 menunjukkan graf tidak berarah dan tidak berbobot.



Gambar 3 Graf Tidak Berarah dan Tidak Berbobot

4. Graf berarah dan tidak berbobot

Graf yang setiap sisinya mempunyai arah tetapi tidak mempunyai bobot apapun. Gambar 4 menunjukkan graf berarah dan tidak berbobot.



Gambar 4 Graf Berarah dan Tidak Berbobot

c) Representasi Graf

Terdapat beberapa cara mempresentasikan graf, tiga di antaranya yang sering digunakan adalah matriks ketetanggaan, matriks bersisian dan senarai ketetanggaan [6].

1. Matriks Ketetanggan

Misalkan $G = (V,E)$ adalah graf dengan n simpul, $n \geq 1$. Matriks ketetanggaan G adalah matriks yang berukuran $n \times n$. Bila matriks tersebut dinamakan $A=[a_{ij}]$, maka a_{ij}= 1jika simpul i dan j bertetangga atau terhubung, sebaliknya a_{ij}= 0 jika simpul i dan j tidak bertetangga atau tidak terhubung.

Matriks bertetanggaan hanya berisi 0 dan 1, selain dengan angka 0 dan 1, elemen matriks dapat juga dinyatakan dengan nilai *false* (menyatakan 0) dan *true* (menyatakan 1). Matriks ketetanggaan nol-satu tidak dapat digunakan untuk mempresentasikan graf yang mempunyai sisi ganda (graf ganda).

2. Matriks Berisisan

Matriks bersisian menyatakan kebersisian simpul dengan sisi. Misalkan $G = (V,E)$ adalah graf dengan n simpul dan m buah sisi. Matriks bersisian G adalah matriks yang berukuran $n \times m$. Baris menunjukkan label simpul, sedangkan kolom menunjukkan label sisi. Bila matriks tersebut dinamakan $A=[a_{ij}]$, maka a_{ij}= 1jika simpul i bersisian dengan j, sebaliknya a_{ij}= 0jika simpul i tidak bersisian dengan simpul j.

Matriks bersisian dapat digunakan untuk mempresentasikan graf yang mengandung sisi ganda atau sisi gelang. Derajat setiap simpul i dapat dihitung dengan menghitung jumlah seluruh elemen pada baris i (kecuali pada graf yang mengandung gelang atau *looping)*. Jumlah elemen matriks bersisian adalah $n \times m$.

3. Senarai Ketetanggaan

Jika dilihat dari segi implementasinya terhadap komputer matriks tetanggaan memiliki kebutuhan ruang memori yang boros. Hal ini dikarenakan matriksnya memiliki banyak elemen nol sedangkan di dalam komputer elemen nol tidak perlu disimpan.

Untuk mengatasi hal tersebut maka senarai ketetanggaan yang lebih baik. Senarai ketetanggaan mengurutkan simpul-simpul yang bertetangga dengan setiap simpul di dalam graf [7].

2.3 Jasa Pengiriman Barang

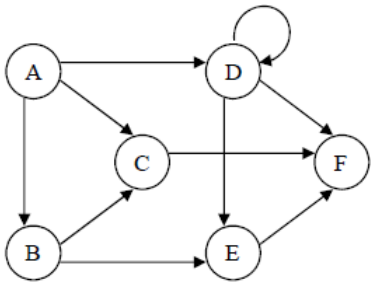
Jasa pengiriman barang adalah suatu organisasi laba/perusahaan yang bergerak dibidang jasa pengiriman barang. Akhir-akhir ini jasa pengiriman barang ini sangat diminati penggunanya, karena dapat dipercaya, dan sangat memuaskan. Pelanggan tidak perlu lagi repot untuk mengirimkan barang, karena pelanggan hanya perlu pergi ke tempat-tempat cabang dari jasa pengiriman barang tersebut. Hanya dengan memberikan alamat tujuan yang lengkap, hitung berat barang, dan hitung jarak kota awal ke kota tujuan, dari situ dapat dihitung total biaya yang diperlukan untuk pengiriman barang.

2.4 Permasalahan Optimasi

Secara umum pencarian jalur terpendek dapat dibagi menjadi dua metode, yaitu metode konvensional dan metode heuristik. Metode konvensional diterapkan dengan menggunakan perhitungan matematika murni, sedangkan metode heuristik diterapkan dengan menggunakan perhitungan kecerdasan buatan. [8].

Persoalan lintasan terpendek di dalam graf merupakan salah satu persoalan optimasi. Graf yang digunakan dalam pencarian lintasan terpendek adalah graf berbobot (*weighted graph*), yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot. Bobot pada sisi graf dapat dinyatakan sebagai jarak antar kota, waktu pengiriman pesan dan lain-lain.

Gambar 5 menunjukkan graf berarah ABCDEF dan tidak berbobot.



Gambar 5 Graf Berarah ABCDE

Dapat dilihat pada Gambar 5 untuk melakukan suatu perjalanan dari simpul awal A ke simpul tujuan F, maka terdapat beberapa pilihan jalur yang mungkin untuk ditempuh, yaitu :

Jalur ke-1 : A B C F
Jalur ke-2 : A B E F
Jalur ke-3 : A C F
Jalur ke-4 : A D E F
Jalur ke-5 : A D E F

Dari uraian jalur dapat ditentukan jalur atau lintasan terpendek dengan mencari jarak suatu jalur antara simpul-simpulnya kemudian membandingkan dengan jarak pada jalur yang lain dan menentukan total jarak yang terpendek atau yang paling kecil.

2.5 Algoritma *Greedy*

Algoritma *Greedy* adalah salah satu metode pemecahan persoalan optimasi yang paling populer. Secara harafiah, *greedy* memiliki arti tamak atau rakus. Orang yang tamak akan mengambil sebanyak mungkin apa yang tersedia tanpa memikirkan konsekuensi ke depan.

Untuk menentukan solusi, *Algoritma Greedy* memiliki kendala (*constraint*) dan fungsi optimasi. Solusi yang memenuhi semua kendala disebut solusi layak, dan solusi layak yang mengoptimalkan fungsi optimasi disebut solusi optimum. [9]

a) Prinsip Utama Algoritma *Greedy*

Algoritma *Greedy* adalah algoritma yang memecahkan masalah langkah demi langkah dan merupakan salah satu metode dalam masalah optimasi. Algoritma *Greedy* membentuk solusi langkah per langkah sebagai berikut:

1. Terdapat banyak pilihan yang perlu diekspolarasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya.
2. Pendekatan yang digunakan di dalam Algoritma *Greedy* adalah membuat pilihan yang terlihat memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal pada setiap langkah dan diharapkan akan mendapatkan solusi optimum global.

Algoritma *Greedy* didasarkan pada pemindahan *edge* per *edge* dan pada setiap langkah yang diambil tidak memikirkan konsekuensi ke depan, *Greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada serta sebagian

masalah *Greedy* tidak selalu berhasil memberikan solusi yang benar-benar oprimum tapi pasti memberikan solusi yang mendekati nilai optimum.

b) Elemen Algoritma *Greedy*

Elemen-elemen yang digunakan dalam penerapan Algoritma*Greedy* antara lain:

1. Himpunan kandidat (C)
   Himpunan ini berisi elemen-elemen pembentuk solusi.
2. Himpunan solusi (S)
   Himpunan ini berisi kandidat-kandidat yang terpilih sebagai solusi dari permasalahan optimasi yang akan diselesaikan.
3. Fungsi seleksi
   Fungsi ini akan memilih kandidat yang paling memungkinkan untuk mencapai solusi optimal. Sesuai dengan prinsip Algoritma *Greedy*, kandidat yang sudah dipilih pada suatu langkah tidak bisa diubah di langkah selanjutnya.
4. Fungsi kelayakan
   Fungsi ini akan memeriksa kelayakan suatu kandidat yang telah dipilih. Dalam arti, kandidat tersebut dan himpunan solusi yang terbentuk tidak melanggar *constraints* yang ada. Bila kandidat layak, maka kandidat tersebut akan dimasukkan ke dalam himpunan solusi, dan jika kandidat tersebut tidak layak, maka kandidat akan dibuang dan tidak akan dipertimbangkan lagi dalam pencarian solusi optimum.
5. Fungsi objektif
   Fungsi ini akan membuat nilai solusi maksimum atau minimum, sesuai dengan jenis optimasi apa yang dibutuhkan.

Skema umum dari Algoritma *Greedy* dapat dituliskan sebagai berikut :

1. Inisiasi *S* dengan kosong.
2. Pilih sebuah kandidat dari *C* (dengan *select* ()).
3. Kurangi *C* dengan kandidat yang terpilih.
4. Periksa apakah kandidat yang dipilih tersebut bersama-sama dengan *S* membentuk solusi yang layak (dengan *feasible* ()). Jika ya, masukkan kandidat ke *S*; jika tidak, buang kandidat tersebut dan tidak perlu ditelaah lagi.
5. Periksa apakah *S* yang sudah terbentuk telah memberikan solusi yang lengkap (dengan *solution* ()). Jika ya, berhenti; jika tidak, ulangi dari langkah kedua.

2.6 *Unified Modelling Language* (UML)

Unified Modeling Language (UML) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan kebutuhan, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman beorientasi objek. [10]

UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak.

Dengan UML, desainer dapat melihat konsep global suatu desain. Desain kemudian dapat dijadikan panduan dalam proses pengembangan dan rekayasa perangkat lunak. Selain itu, UML dapat menjadi media komunikasi gagasan antara pengembang perangkat lunak dengan pengguna.

Bagian-bagian UML yang pada pembuatan aplikasi ini adalah:

a. *Use Case Diagram*

Diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

b. *Activity Diagram*

Diagram aktivitas menggambarkan aliran kerja (*workflow*) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor.

c. *Class Diagram*

Diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas yang akan dibuat untuk membangun sistem. Kelas memiliki atribut dan metode atau operasi.

3. HASIL DAN PEMBAHASAN

Dalam bab ini dibahas mengenai pembuatan program dan hasil uji coba program yang telah dirancang dan diimplementasikan dengan menggunakan aplikasi *Android Studio*.

Uji coba dilakukan untuk mengetahui apakah program dapat berjalan sebagaimana mestinya dengan lingkungan uji coba yang telah ditentukan serta dilakukan sesuai dengan skenario uji coba.

3.1 Implementasi

Tahap ini bertujuan untuk mengetahui apakah kebutuhan sistem yang telah diintegrasikan telah terpenuhi. Implementasi antarmuka terbagi menjadi beberapa bagian, diantaranya adalah:

a) *Flash Screen*

Pada sesi ini *user* akan melihat logo dan menu *loading* yang sedang berjalan. Gambar 6 menunjukkan tampilan *Flash Screen*.

Gambar 6 *Flash Screen*

b) Menu Utama

Gambar 7 merupakan tampilan menu saat membuka aplikasi.

Gambar 7 Halaman Menu

c) Halaman *Costumer*

Gambar 8 merupakan tampilan saat *user* memilih menu *view*, pada menu ini terdapat daftar alamat-alamat pelanggan yang telah diinput ke dalam *database*.

Gambar 8 Menu *Costumer*

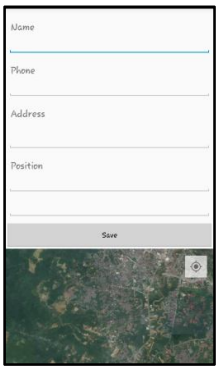
d) Halaman *Edit*

Gambar 9 merupakan tampilan saat *user* memilih salah satu alamat yang terdapat pada menu *costumer*, halaman ini berfungsi untuk mengedit identitas pelanggan yang telah diinput sebelumnya.

Gambar 9 Halaman *Edit*

e) Halaman *Add Costumer*

Gambar 10 merupakan tampilan saat *user* memilih menu *Add Costumer*, pada menu ini berisi *form* yang berisi identitas pelanggan yang akan dimasukan ke dalam *database*.



Gambar 10 Menu *Add Costumer*

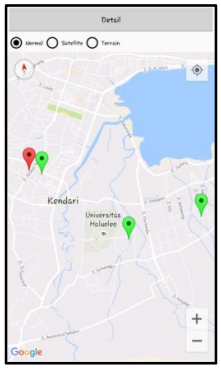
f) Halaman *Routes*

Gambar 11 merupakan tampilan saat *user*memilih menu *Routes*. Menu ini menampilkan daftar alamat-alamat yang telah diinput untuk kemudian di pilih sebagai titik tujuan. Terdapat tombol pilih setelah titik-titik tujuan telah selesai ditentukan.

Gambar 11 Menu *Routes*

g) Halaman *Map*

Gambar 12 merupakan tampilan saat *user* menekan tombol pilih pada menu *routes*, halaman ini akan menampilkan gambaran rute dari titik awal ke semua titik tujuan yang telah dipilih sebelumnya. Terdapat tombol *detail* untuk melihat perincian rutenya.



Gambar 12 Tampilan *Map*

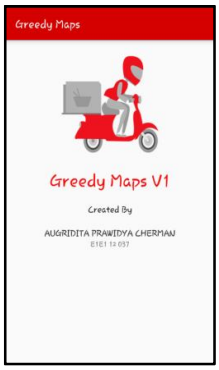
h) Halaman Detail

Gambar 13 merupakan tampilan saat *user* menekan tombol *detail* pada halaman *Map*, pada halaman ini terdapat hasil dari jarak yang telah di hitung antara titik awal ke titik tujuan pertama, kedua, ketiga dst.

i) Halaman *About*

Gambar 14 merupakan tampilan saat *user* membuka menu *About*, pada menu ini terdapat sekilas info mengenai aplikasi.

Gambar 13 Halaman *Detail*



Gambar 14 Menu *About*

3.2 Uji Coba Sistem

Dalam penggunaan aplikasi ini, masih terdapat perbedaan dalam penentuan estimasi jarak. Kali ini penulis akan melakukan tiga kali percobaan contoh kasus dalam penentuan rute dan jarak ke titik tujuan yang ada di kota Kendari.

a) Menentukan Data Uji

Tahap ini merupakan tahap untuk memilih data uji yang digunakan dalam proses penentuan jarak. Tabel 1 menunjukkandatauji yang digunakan merupakan data pelanggan tetap Burjo Kendari berdasarkan dari hasil observasi lapangan.

Tabel 1 Data Uji

Nama	Alamat	Koordinat *(Lat, Long)*
Putri	Perdos UHO, Blok 5 No.4	-4.011228, 122.529269
Widi	Perumnas Multigraha, Poasia	-4.007314, 122.549351

Akbar	Briton Sorumba	-3.991224, 122.508093
Nola	Jl. Bahagia No. 46A	-4.003977, 122.502208
Heny	Lrg. Anaway Kampus UHO, Asrama Masita	-4.009176, 122.516720
Fitri	Jl. Ahmad Yani, Lr. Flamboyan No 35b	-3.983715, 122.506209
Fitria	BTN. Bonggoeya Lestari Blok B/7	-4.007972, 122.507142
Astika	Jl. Wayong, Toko Setia Jaya	-3.977268, 122.501041
Kiga	Jl. Balaikota III No.87	-3.977625, 122.505921

b) Pengujian

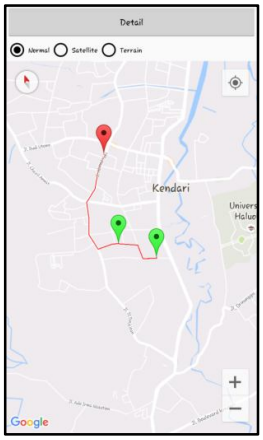
Dalam penggunaan aplikasi ini, masih terdapat perbedaan dalam penentuan estimasi jarak. Kali ini akan dilakukan dua kali pengujian dalam penentuan rute dan jarak ke alamat-alamat pelanggan berdasarkan tabel data uji.

1. Pengujian Pertama

Gambar 15 menampilkan halaman menu *Routes* di mana *user* memilih dua alamat yang akan dihitung jaraknya masing-masing dari titik awal. Setelah memberi centang pada kolom yang tersedia, *user* menekan tombol pilih untuk melakukan proses perhitungan.

Gambar 15 Pemilihan Alamat yang akan Dituju (Pengujian Pertama)

Gambar 16 merupakan hasil dari penentuan rute dari titik awal ke tiga titik yang telah dipilih *user* sebelumnya. Posisi awal *user* berlokasi di Burjo Kendari (ikon berwarna merah).



Gambar 16 Penentuan Jalur Terdekat dari Titik Awal (Pengujian Pertama)

Gambar 17 menampilkan halaman *detail* yang menjabarkan secara rinci hasil perhitungan jarak terpendek yang telah dilakukan oleh sistem.

Gambar 17 Detail (Pengujian Pertama)

Berdasarkan hasil perhitungan, rute terpendek dari Burjo Kendari yaitu Nola dengan jarak 2189 meter dari titik awal. Nola akan menjadi titik awal baru untuk menentukan titik tujuan selanjutnya.

Setelah melakukan proses perhitungan ulang dengan menggunakan Nola sebagai titik awal, maka didapatkan hasil yaitu rute terpendek dari Nola adalah Fitria dengan jarak 857 meter dan merupakan titik tujuan terakhir.

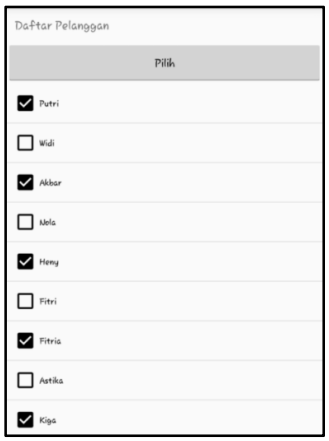
Dari pengujian tersebut maka dapat disimpulkan bahwa rute yang harusnya dilewati oleh kurir adalah Nola dan yang terakhir adalah Fitria. Tabel 2 menunjukkan rekap hasil pengujian.

Tabel 2 Rekap Hasil Pengujian Pertama

Posisi Awal	Posisi Terdekat	Jarak	Keterangan
Burjo Kendari	Nola	2189 meter	Nola merupakan titik tujuan pertama yang akan dilalui
Nola	Fitria	857 meter	Fitria merupakan titik tujuan terakhir yang akan dilalui

2. Pengujian Kedua

Gambar 18 menampilkan halaman menu *Routes* di mana *user* memilih lima alamat yang akan dihitung jaraknya masing-masing dari titik awal. Setelah memberi centang pada kolom yang tersedia, *user* menekan tombol pilih untuk melakukan proses perhitungan.

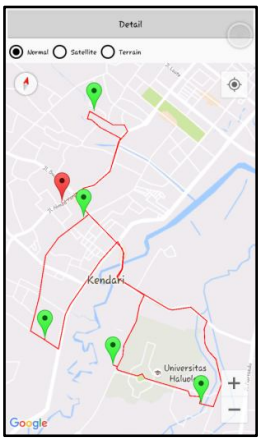


Gambar 18 Pemilihan Alamat yang akan Dituju (Pengujian Kedua)

Gambar 18 menampilkan halaman menu *Routes* di mana *user* memilih lima alamat yang akan dihitung jaraknya masing-masing dari titik awal. Setelah memberi centang pada kolom yang tersedia, *user* menekan tombol pilih untuk melakukan proses perhitungan.

Gambar 19 merupakan hasil dari penentuan rute dari titik awal ke tiga titik yang telah dipilih *user* sebelumnya. Posisi awal *user* berlokasi di Burjo Kendari (ikon berwarna merah).

Gambar 20 menampilkan halaman *detail* yang menjabarkan secara rinci hasil perhitungan jarak terpendek yang telah dilakukan oleh sistem. Berdasarkan hasil perhitungan, rute terpendek dari Burjo Kendari yaitu Akbar dengan jarak 697 meter dari titik awal. Akbar akan menjadi titik awal baru untuk menentukan titik tujuan selanjutnya.



Gambar 19 Penentuan Jalur Terdekat dari Titik Awal (Pengujian Kedua)

Gambar 20 Detail (Pengujian Kedua)

Setelah melakukan proses perhitungan ulang dengan menggunakan Akbar sebagai titik awal, maka didapatkan hasil yaitu rute terpendek dari Akbar adalah Fitria dengan jarak 2200 meter, kemudian proses selanjutnya dilakukan dengan menggunakan Fitria sebagai titik awal dan Heny merupakan titik tujuan

selanjutnya dengan jarak 4393 meter dari Astika.

Selanjutnya Heny digunakan sebagai titik awal baru dan didapatkan rute selanjutnya yaitu Putri dengan jarak 1639 dari Heny, kemudian dengan menggunakan Putri sebagai titik awal, Kiga menjadi titik tujuan terakhir dengan jarak 6541 dari Putri.

Dari pengujian tersebut maka dapat disimpulkan bahwa rute yang harusnya dilewati oleh kurir adalah Akbar, Fitria, Heny, Putri kemudian yang terakhir adalah Kiga. Tabel 5 menunjukkan rekap hasil pengujian.

Tabel 3 Rekap Hasil Pengujian Kedua

Posisi Awal	Posisi Terdekat	Jarak	Keterangan
Burjo Kendari	Akbar	697 meter	Akbar merupakan titik tujuan pertama yang akan dilalui
Akbar	Fitria	2200 meter	Fitria merupakan titik tujuan kedua yang akan dilalui
Fitria	Heny	4393 meter	Heny merupakan titik tujuan ketiga yang akan dilalui
Heny	Putri	1639 meter	Putri merupakan titik tujuan keempat yang akan dilalui
Putri	Kiga	6541 meter	Kiga merupakan titik tujuan terakhir yang akan dilalui

4. KESIMPULAN

Setelah dilakukan serangkaian uji coba dan analisis terhadap aplikasi penentuan rute terpendek yang dibuat, maka dapat diambil kesimpulan bahwa Algoritma *Greedy* dapat diimplementasikan untuk pencarian rute terpendek terhadap sistem yang *real time*, dan sistem navigasi yang bisa diakses dengan telepon seluler yang mampu menunjukan rute yang paling pendek menuju ke lokasi tujuan dengan waktu komputasi yang lebih cepat.

5. SARAN

Saran dalam penelitian ini yaitu:

1. Untuk penelitian selanjutnya, diharapkan untuk mengembangkan sistem *input* koordinat lokasi secara otomatis.
2. Aplikasi ini dapat dikembangkan dan menyelesaikan persoalan lintasan terpendek dalam cakupan yang lebih besar dan mengimplementasikannya dengan bahasa pemograman yang berbeda dan menggunakan algoritma yang berbeda pula, agar dapat menghasilkan nilai *output* yang lebih akurat dan optimal.
3. Kelemahan aplikasi ini adalah tidak adanya batas maksimal wilayah, jadi untuk penelitian selanjutnya diharapkan batas wilayah sudah bisa ditampilkan.

DAFTAR PUSTAKA

[1] Indra, M., 2008, *Perbandingan Algoritma Greedy dan Brute Force Dalam Simulasi Pencarian Koin*, Bandung : Program Studi Teknik Informatika, ITB.

[2] Pakpaha, F.S., 2015, *Aplikasi Wisata Sumut Memanfaatkan Fasilitas Google Map Pada Smartphone Berbasis Android*, Universitas Sumatra Utara.

[3] Priawan, M., 2013, *Teknologi, Smartphone, dan Android.*

[4] Munir, R., 2004, *Algoritma Greedy*. Departemen Teknik Informatika. Institut Teknologi Bandung. Bandung.

[5] Poetra dan Fajrul H., 2010, *Aplikasi Penggunaan Algoritma Tabu Search Pada Pencarian Jalur Terpendek*, Universitas Sumatra Utara.

[6] Romelta, E., 2009, *Metode Pencarian Lintasan Terpendek dalam Graf*, Institut Teknologi Bandung, Bandung.

[7] Budayasa, I.K., 2007, *Teori Graf dan Aplikasinya*, Surabaya, Unesa University Press.

[8] Sundari dan Sri, 2010, *Eksentrisitas Di graf Pada Graf Cycel*, Universitas Sumatra Utara.

[9] Wiradeva, A.K., 2007, *Perbandingan Kompleksitas Penerapan Algoritma Greedy untuk Beberapa Masalah*, Bandung: Institut Teknologi Bandung.

[10] Rosa, A., 2011, *Rekayasa Perangkat Lunak*, Bandung: Modula.