Implementasi Algoritma Bellman-Ford untuk Pencarian Jalur Terpendek Menuju Rumah Sakit di Kota Yogya Berbasis Android

Rendy Setiawan[1], R.Gunawan Santosa[2], Junius Karel Tampubolon[3]
Program Studi Informatika, Universitas Kristen Duta Wacana Yogyakarta
Jl. Dr. Wahidin Sudirohusodo No.5-25, Yogyakarta
[1]rendy.setiawan@ti.ukdw.ac.id
[2]gunawan@staff.ukdw.ac.id
[3]karel@staff.ukdw.ac.id

Abstract*— Yogyakarta City is a Student and Tourism City. Many tourist and student come to the city of Yogyakarta to study and vacation. But many of these newcomers who do not know the location of the hospital in the city of Yogyakarta. This research will try to create an application for finding the shortest path to the hospital using the Android-Based Bellman-Ford algorithm. The Bellman-Ford algorithm is one of the algorithms for searching the shortest path. In the process, the authors take data in the form of road coordinates and hospital coordinates, then make it into a single graph. After getting the graph, the author uses the Bellman-Ford algorithm to find the shortest path. The shortest path search results will be compared with results from Google Maps. The shortest path search test was carried out 20 times and successfully carried out and showed that the Bellman-Ford algorithm was able to provide the same shortest path as Google Maps by 80%, even the average distance issued was better by 30.2m than the average distance issued by Google Maps. This research also found that the more the number of vertices and edges in the data will affect the long process of algorithm for finding the shortest path because Bellman-Ford algorithm must check each vertex and edge.

Intisari—Kota Yogyakarta merupakan Kota Pelajar dan Kota Wisata. Banyak wisatawan maupun pelajar yang datang ke Kota Yogyakarta untuk belajar dan berlibur. Tetapi banyak dari pendatang tersebut yang tidak tahu mengenai lokasi dari rumah sakit di Kota Yogyakarta. Penelitian ini akan mencoba membuat aplikasi pencarian jalur terpendek menuju rumah sakit menggunakan algoritma Bellman-Ford berbasis Android. Algoritma Bellman-Ford merupakan salah satu algoritma untuk pencarian jalur terpendek. Dalam prosesnya, penulis mengambil data berupa koordinat jalan dan koordinat rumah sakit, lalu membuatnya menjadi satu kesatuan graf. Setelah mendapatkan graf, penulis menggunakan algoritma Bellman-Ford untuk mencari jalur terpendek. Hasil pencarian jalur terpendek akan dibandingkan dengan hasil pencarian dari Google Maps. Pengujian pencarian jalur terpendek dilakukan sebanyak 20 kali dan berhasil dilakukan dan menunjukan bahwa algoritma Bellman-Ford mampu memberikan jalur terpendek yang sama sebesar 80% dengan Google Maps, bahkan rata-rata jarak yang dikeluarkan lebih baik sebanyak 30.2m dibanding dengan rata-rata jarak yang dikeluarkan Google Maps. Pengujian ini juga menemukan bahwa semakin banyak jumlah verteks dan edge dalam data akan mempengaruhi lama proses algoritma untuk mencari jalur terpendek karena algoritma Bellman-Ford harus mengecek setiap verteks dan *edge*.

***Kata Kunci*— Bellman-Ford, Android, pencarian jalur terpendek**

I. PENDAHULUAN

Kota Yogyakarta merupakan Ibukota dari Provinsi Daerah Istimewa Yogyakarta yang merupakan Kota Pelajar dan Kota Wisata. Kota Yogya memiliki jumlah pendatang yang banyak baik pelajar dari luar kota maupun para wisatawan yang sedang berlibur. Dikarenakan Kota Yogya memiliki penduduk yang banyak baik itu penduduk tetap atau pendatang, maka secara otomatis pemerintah Kota Yogyakarta mengembangkan layanan infrastruktur kesehatan berupa Rumah Sakit.

Tidak semua lokasi atau keberadaan Rumah Sakit diketahui oleh masyarakat. Kurangnya informasi mengenai lokasi Rumah Sakit kadang membuat kesulitan masyarakat ataupun pendatang untuk menemukan Rumah Sakit terdekat atau jalan menuju Rumah Sakit yang berada di dekatnya.

Algoritma Bellman-Ford merupakan salah satu algoritma *path finding* yang digunakan untuk mencari *shortest path*. Algoritma Bellman-Ford terbukti dapat menemukan jalur terpendek dengan hasil yang selalu benar, tetapi waktu yang dibutuhkan untuk mencari jalur terpendek lama.

Christian [1] membuat penelitian tentang perbandingan algoritma Dijktsra dan Bellman-Ford dalam pencarian jarak terdekat, dan menyimpulkan bahwa algoritma Dijkstra lebih cepat dibandingkan dengan Bellman-Ford tetapi Bellman-Ford melakukan pencarian lebih rinci dan dapat menghitung beban negatif. Kemudian Kristyaningrum [2] juga melakukan penelitian perbandingan antara algoritma Dijkstra dan Bellman-Ford untuk pencarian jalur terpendek pada graf berarah yang hasil nya memilki kesimpulan yang sama dengan Christian.

Kurniawijaya membuat penelitian dengan merancang sebuah *web* pencarian pom bensin terdekat di Denpasar dengan menggunakan algoritma Dijkstra. Kurniawijaya menggunakan node *source* yang merupakan nama jalan yang kemudian akan mencari jarak ke semua node target yaitu pom bensin, tetapi dalam penelitian ini belum bisa menemukan posisi *user* berada dimana dan secara *realtime* [3].

Halim [4] membuat sebuah penelitian dengan judul Visualisasi Jalur Terpendek dengan algoritma Bellman-Ford pada Studi Kasus Jalan Darat Antar Kota (Pulau Jawa). Kesimpulan yang didapat dalam penelitian tersebut adalah bahwa algoritma pencarian jalur terpendek terbagi menjadi dua, yaitu algoritma pencarian jalur terpendek dari suatu node asal ke semua node yang lainnya dan algoritma pencarian jalur terpendek dari semua node asal ke semua node lainnya. Bellman-Ford bekerja dengan teliti karena membandingkan hasil pencarian sampai lintasan yang memuat N-1 jalur, sehingga hasil jalur terpendeknya tidak dirahukan lagi keabsahan hasilnya. Sebaliknya, karena algoritma Bellman-Ford meneliti seluruh kemungkinan yang ada, maka algoritma ini bekerja lebih lama dan kurang efisien.

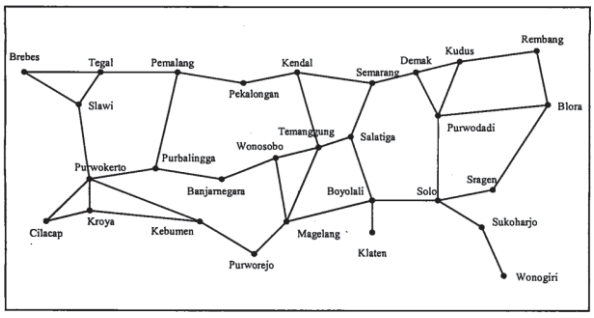
Dalam penelitian ini akan dicoba untuk merancang sebuah sistem berbasis Android yang dapat menampilkan lokasi dan mencari jalur terdekat menuju Rumah Sakit dengan menggunakan algoritma Bellman-Ford. Penelitian ini juga akan mencoba mengevaluasi perbedaan jarak dan keefektifan sistem yang menggunakan Bellman-Ford dengan fitur pencarian jalur Google Maps.

II. LANDASAN TEORI

Ilmu dan metode yang diterapkan dalam penelitian ini antara lain Graf, Android, *Application Programming Interface,* Google Maps dan Bellman-Ford.

A. Teori Graf

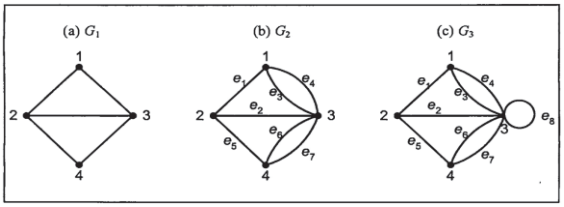
Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Representasi visual dari graf adalah dengan menyatakan objek dinyatakan sebagai noktah, bulatan atau titik, sedangkan hubungan antara objek dinyatakan dengan garis [5]. Sebagai contoh, Gambar 1 adalah sebuah peta jaringan jalan raya yang menghubungkan sejumlah kota di Provinsi Jawa Tengah. Sesungguhnya petatersebut adalah sebuah graf, yang dalam hal ini kota dinyatakan sebagai bulatan sedangkan jalan dinyatakan sebagai garis.



Gambar 1 Jaringan jalan raya di Jawa Tengah

Graf dapat dikelompokan menjadi beberapa kategori bergantung pada sudut pandang pengelompokanya. Jika berdasarkan ada tidaknya sisi ganda pada suatu graf, secara umum graf dapat diolongkan menjadi dua jenis:

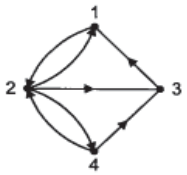
- Graf sederhana *(simple graph)*
  Graf yang tidak mengandung gelang maupun sisi-ganda. G_1 pada Gambar 2(a) adalah contoh graf sederhana.
- Graf tak sederhana (*unsimple graph)*
  Graf yang mengandung sisi ganda atau gelang. Ada dua macam graf tak sederhana, yaitu graf ganda (*multigraph*) dan graf semu (*pseudograph*). Graf ganda adalah graf yang mengandung *edge* ganda. Sedangkan graf semu adalah graf yang mengandung gelang (*loop*). G_2 pada Gambar 2(b), sisi e3 = (1,3) dan sisi e4 = (1,3) dinamakan *edge* ganda (*multiple graph* atau *parallel edges*) karena kedua verteks ini menghubungi dua buah verteks yang sama, yaitu verteks 1 dan verteks 3. Sedangkan G_3 pada Gambar 2(c), sisi e8 = (3,3) dinamakan gelang (*loop*) karena ia berawal dan berakhir pada simpul yang sama.



Gambar 2 Tiga buah graf (a) graf sederhana (b) graf ganda (c) graf semu

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

- Graf tak berarah *(undirected graph)*
  Graf yang sisinya tidak mempunyai orientasi arah. Pada graf tak berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan. Jadi (u,v) = (v,u) adalah sisi yang sama. Tiga buah graf pada Gambar 2 merupakan graf tak berarah.
- Graf berarah *(directed graph)*
  Graf yang setiap sisinya diberikan orientasi arah. Pada Gambar 3 dapat dilihat bahwa masing – masing verteks memiliki *edge* yang berarah.



Gambar 3 Graf berarah

B. Android

Menurut Safaat [6] Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Android juga dipuji sebagai "*platform mobile* pertama yang Lengkap, Terbuka, dan Bebas"

- Lengkap (*Complete Platform)*
  Para desainer dapat melakukan pendekatan yang komprehensif ketika mereka sedang mengembangkan *platform* android. Android merupakan sistem operasi yang aman dan banyak menyediakan *tools* dalam membangun *software* dan memungkinkan peluang pengembangan aplikasi.

JUTEI Edisi Volume.3 No.2 Oktober 2019
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2019.32.184

- Terbuka (*Open Platform)*
  Platform android disediakan melalui lisensi *open source*. Pengembang dapat dengan bebas untuk mengembangkan aplikasi. Android sendiri menggunakan Linux Kernel 2.6.
- Bebas (*Free Platform)*
  Android adalah *platform* / aplikasi yang bebas untuk dikembangkan. Tidak ada lisensi atau biaya *royalty* untuk dikembangkan pada *platform* android. Tidak ada biaya keanggotaan diperlukan. Tidak diperlukan biaya pengujian. Tidak ada kontrak yang diperlukan. Aplikasi untuk android dapat didistribusikan dan diperdagangkan dalam bentuk apapun.

Pada masa saat ini sebagian besar *vendor-vendor smartphone* sudah memproduksi *smartphone* berbasis Android, seperti Samsung, Xiaomi, LG, Huawei, Motorola, Nokia, Vivo, Oppo dan masih banyak lagi.

Aplikasi Android dapat dikembangkan pada beberapa sistem operasi, yaitu Windows, Mac OS, dan Linux.

C. GPS (Global Positioning System)

GPS atau biasa disebut dengan *Global Positioning System* merupakan sistem untuk menentukan posisi dan navigasi secara global dengan menggunakan satelit dan metode Triangulasi. Sistem tersebut merupakan sistem yang pertama kali dikembangkan oleh Departemen Pertahanan Amerika yang awalnya diperuntukan bagi kepentingan militer. NAVSTAR GPS (Navigation Satellite Timing and Ranging Global Positioning System) adalah nama asli dari Sistem GPS. [7]

Menurut Hudiono, Taufik, Koesmarijanto, dan Darmono [8] penerima GPS memperoleh sinyal dari beberapa satelit yang mengorbit bumi. Terdapat 24 susunan satelit yang mengitari bumi pada orbit pendek. Dengan susunan 21 satelit aktif dan 3 buah satelit sebagai cadangan. GPS dapat memberikan informasi posisi dan waktu dengan ketelitian sangat tinggi. Sistem GPS ini dapat digunakan oleh banyak orang sekaligus dalam segala cuaca, serta dapat memberikan posisi dan kecepatan tiga dimensi yang teliti dan juga informasi mengenai waktu secara kontinyu di seluruh dunia.

Sinyal GPS diterima oleh alat penerima di permukaan, dan digunakan untuk menentukan posisi, kecepatan, arah, dan waktu. Sistem yang serupa dengan GPS antara lain: GLONASS Rusia, Galileo Uni Eropa, IRNSS India.

D. Application Programming Interface

Menurut Tulach [9] API atau *Application Programming Interface* bukan hanya satu *set* dan *method* atau fungsi dan *signature* yang sederhana. Akan tetapi API, yang bertujuan utama untuk mengatasi "*clueless*" dalam membangun *software* yang berukuran besar, berawal dari sesuatu yang sederhana sampai ke yang kompleks dan merupakan perilaku komponen yang sulit dipahami. Secara sederhana dapat dipahami dengan membayangkan kekacauan yang akan timbul bila mengubah database atau skema XML.

Jadi dapat disimpulkan bahwa API merupakan suatu kumpulan perintah, *class*, fungsi dan protokol yang memungkinkan suatu *software* berhubungan dengan *software* lainnya.

E. Google Maps

Google Maps merupakan layanan pemetaan web yang dikembangkan oleh Google. Google Maps memberikan citra satelit, peta jalan, panorama 360º dan kondisi lalu lintas.

Pada tahun 2010 Google Maps API merupakan API yang paling popular di internet. Pencatatan yang dilakukan pada bulan Mei 2010 ini menyatakan bahwa 43% *mashup* (situs web dan aplikasi yang menggabungkan dua atau lebih sumber data) menggunakan Google Maps API. Pencatatan tersebut dilakukan oleh Svennerberg [10]

Beberapa tujuan dari penggunaan Google Maps API adalah untuk mencari peta, melihat lokasi, dan sebagainya. Hampir semua hal yang berhubungan dengan peta dapat memanfaatkan Google Maps.

F. Algoritma Bellman Ford

Algoritma Bellman-Ford merupakan salah satu algoritma yang digunakan untuk pencarin jalur terpendek dalam suatu graf. Sama seperti algoritma Dijkstra, algoritma Bellman-Ford digunakan untuk menghitung jarak terpendek dari satu sumber menuju ke tujuan pada sebuah graf yang berbobot. Maksud dari satu sumber ialah bahwa algoritma Bellman-Ford menghitung semua jarak terpendek pada graf yang bermula dari satu titik node. Keunggulan algoritma Bellman-Ford terhadap algoritma pencarian yang lain adalah algoritma ini dapat meghitung bobot negatif.

Berikut merupakan *pseudocode* Algoritma Bellman-Ford yang dikutip dari buku *Introduction to Algorithm* [11]:

```
BELLMAN-FORD(G, w, s)
1. INITIALIZE-SINGLE-SOURCE(G,s)
2. for i = 1 to |G.V| - 1
3.         for each edge (u,v) ϵ G.E
4.                 RELAX(u,v,w)
5. for each edge (u,v) ϵ E
6.          if v.d > u.d + w(u, v)
7.                         return FALSE
8. return TRUE
```

Pada pseudocode di atas terdapat fungsi Initialize-Single-Source dan Relax yang dimana penjelasan nya dapat dilihat di bawah ini:

```
INITIALIZE–SINGLE–SOURCE(G,s)
1. For each vertex v ϵ G.V
2.          v.d = ∞
3.         v. π = NIL
4. s.d = 0
```

```
RELAX(u, v, w)
1. if v.d > u.d + w(u, v)
2.         v.d  = u.d + w(u, v)
3.         v. π = u
```

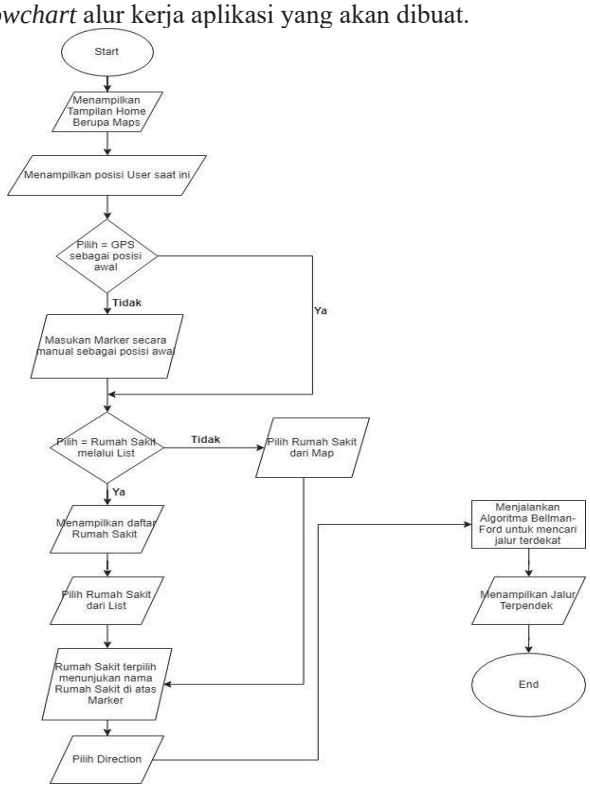
Penjelasan dari *pseudocode* diatas adalah sebagai berikut:

1. Inisialisasi graf G, bobot di setiap edge yang disimbolkan w dengan verteks asal s.
2. Inisialisasi semua nilai verteks menjadi infinity.
3. Inisialisasi predecessor menjadi nil.
4. Inisialisasi nilai verteks awal menjadi 0.

5. Melakukan perulangan sebanyak jumlah verteks (|G.V|) – 1.
6. Melakukan perulangan terhadap setiap edge E (dari verteks u ke verteks v).
7. Masuk ke fungsi Relax. Lakukan pengecekan apakah bobot dari verteks asal s ke verteks v (v.d) lebih besar dari verteks awal ke verteks u (u.d) + bobot dari verteks u ke verteks v (w(u,v)).
8. Jika langkah ke 1 di fungsi relax benar, maka bobot dari verteks awal ke verteks u (u.d) + bobot dari verteks u ke verteks v (w(u,v)) akan disimpan dan menggantikan bobot dari verteks asal s ke verteks v (v.d).
9. Lalu verteks u akan menyimpan ke verteks yang menuju ke verteks v (v. π) yang berguna untuk menyimpan jalur yang akan dilalui sebagai solusi langkah terpendek.
10. Untuk langkah ke 5 sampai ke 7 digunakan untuk pengecekan ada atau tidak nya negative-weight cycle pada graf. Dengan cara melakukan perulangan untuk setiap edge E dari verteks u ke verteks v.
11. Lalu melakukan pengecekan apakah bobot dari verteks asal ke verteks v (v.d) lebih besar daripada bobot verteks asal ke verteks u (u.d) + bobot dari verteks u ke verteks v (w (u, v))
12. Jika benar, berarti terdapat cycle yang memiliki total bobot negatif pada graf, maka akan mengembalikan nilai false.

III. METODOLOGI PENELITIAN

Penelitian dilakukan dengan cara pembuatan aplikasi berbasis android untuk mencari jalur terpendek menuju rumah sakit. Penulis merancang dan membuat sebuah *flowchart* alur kerja aplikasi yang akan dibuat.

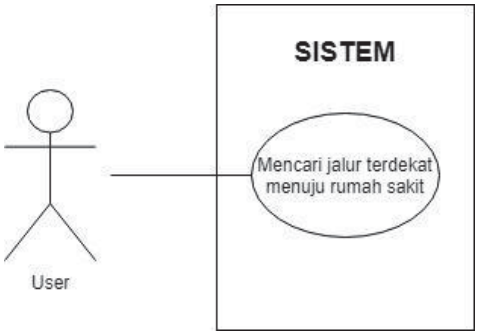


Gambar 4 Flowchart aplikasi yang akan dibuat

Flowchart diatas menunjukan alur sistem dalam melakukan pencarian jalur terpendek menuju rumah sakit terdekat. Pada tahap pertama sistem akan menampilkan tampilan utama yang berupa *map* Kota Yogyakarta dan sistem akan mendeteksi lokasi user secara *realtime* menggunakan GPS. Lalu user harus memilih apakah ingin menggunakan GPS / lokasi user saat ini sebagai lokasi awal dalam pencarian atau ingin menentukan lokasi sendiri dengan cara memilih lokasi yang diinginkan. Jika user ingin menentukan lokasi sendiri, maka user harus menentukan marker ke posisi tertentu dengan cara menekan posisi yang diinginkan di map, lalu akan muncul sebuah marker yang akan digunakan sebagai titik awal.

Setelah itu user memilih rumah sakit yang ingin dituju. User dapat memilih rumah sakit dengan cara mencari dari map atau memilih dari list rumah sakit yang tersedia. Jika user ingin memilih dari rumah sakit, maka user harus memencet tombol list, lalu sistem akan menampilkan list rumah sakit yang tersedia. User memilih rumah sakit yang ingin dituju lalu sistem akan menampilkan lokasi rumah sakit yang dipilih beserta nama dari rumah sakit tersebut. Jika user tidak memilih dari list, maka user harus mencari rumah sakit yang ingin dituju melalui peta yang disediakan dan memilih marker rumah sakit yang ingin dituju. Setelah itu user memilih tombol Direction dan sistem akan mencari jalur terdekat dari posisi awal ke rumah sakit yang ingin dituju dengan menggunakan Algoritma Bellman-Ford. Setelah mendapatkan jalur terpendek, sistem akan mewarnai jalan tersebut dan menampilkan nya di dalam *map*.

Secara garis besar, penulis membuat sistem ini hanya memiliki satu tujuan utama, yaitu mencari jalur terpendek menuju rumah sakit.



Gambar 5 Use case diagram sistem

Untuk menyimpan data di aplikasi, penulis merancang sebuah basis data menggunakan SQLiteStudio v3.1.1 dengan membuat dua tabel sebagai berikut:

graph	
id	INTEGER
simpul_awal	INTEGER
simpul_tujuan	INTEGER
jalur	TEXT
bobot	DOUBLE

rumahsakit	
id	INTEGER
simpul	INTEGER
rumahsakit	VARCHAR
latitude	DOUBLE
longitude	DOUBLE

Gambar 6 Tabel data graph dan rumah sakit

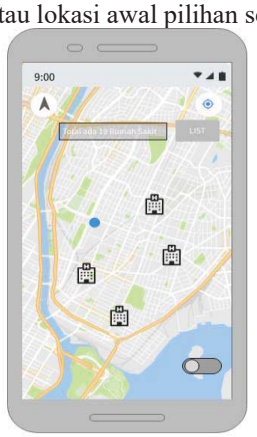
Tabel data "rumahsakit" digunakan untuk menyimpan data-data tentang rumah sakit. Lokasi rumah disimpan dalam bentuk koordinat yang disimpan dalam data latitude dan

JUTEI Edisi Volume.3 No.2 Oktober 2019
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2019.32.184

longitude, lalu lokasi rumah sakit di graf disimpan di dalam data simpul. Sedangkan tabel "graph" digunakan untuk menyimpan data jalur atau *edge* yang saling menghubungkan verteks satu sama lainnya sehingga membentuk satu kesatuan *graph*.

Program diimplementasikan menggunakan Bahasa Pemograman Java dan menggunakan aplikasi Android Studio v3.5 dengan Gradle v3.5. Rancangan antarmuka memiliki beberapa halaman, yaitu:

1. Tampilan Menu Awal
   Pada tampilan awal sistem ini, sistem akan menunjukan lokasi *user* saat ini serta menampilkan seluruh posisi rumah sakit yang ada di kota Yogyakarta. Pada Gambar 7 dapat dilihat bahwa titik biru menandakan posisi *user* saat ini, dan simbol rumah sakit menunjukan rumah sakit yang ada / tersedia. Terdapat juga tombol list yang berguna untuk menampilkan daftar-daftar rumah sakit yang tersedia. Sedangkan tombol *toggle* digunakan untuk menentukan posisi awal apakah ingin menggunakan lokasi gps atau lokasi awal pilihan sendiri.



Gambar 7 Tampilan awal

2. Tampilan Menu List
   Pada tampilan menu *list*, sistem akan menampilkan daftar rumah sakit yang tersedia. Daftar rumah sakit hanya akan menampilkan nama dari rumah sakit. Dan jika *user* menekan salah satu rumah sakit dari dalam daftar tersebut, maka sistem akan menampilkan lokasi rumah sakit tersebut.

Gambar 8 Tampilan list

3. Tampilan Menu Direction
   Pada tampilan menu *direction*, sistem akan menampilkan jalur terpendek menuju rumah sakit. Jalur akan ditampilkan di *map* dengan garis biru yang merupakan jalur menuju rumah sakit tersebut. Terdapat juga tombol *back* untuk kembali ke menu awal. Jarak menuju rumah sakit juga akan dicatat dan ditampilkan seperti yang ditunjukan di Gambar 9.

Gambar 9 Tampilan Direction

Penulis mengambil data dari API Google Maps untuk mengimplementasikan *Map* ke dalam *device* Android. Sistem yang dibuat penulis harus terhubung dengan *https://code.google.com/apis/console* untuk mengirimkan *permission* kepada Google untuk proses pengambilan *API key* dan Google akan memberikan *API Key* untuk dipakai dalam sistem. Berikut merupakan keterangan *property* dari API Google Maps:

1. *LocationManager* = berfungsi untuk mengatur lokasi.
2. *CameraPosition* = berfungsi untuk mengatur layar tampilan Maps.
3. *animateCamera* = berfungsi untuk memindahkan layar tampilan dengan animasi ke tujuan yang telah ditetapkan.
4. *moveCamera* = berfungsi untuk memindahkan layar tampilan ke tujuan yang telah ditetapkan tanpa menggunakan animasi.
5. *setMyLocationEnabled* = berfungsi untuk menambahkan fungsi tombol yang akan mengarahkan langsung kepada posisi user saat ini.
6. *Clear* = berfungsi untuk membersihkan Google Maps menjadi seperti semula.
7. *OnMapClickListener* = berfungsi untuk memasang listener jika map ditekan oleh user.
8. *OnMapLongClickListener* = berfungsi untuk memasang listener jika map ditekan lama oleh user.
9. *addMarker* = berfungsi untuk membuat dan menambah Marker kedalam map.
10. *addPolyline* = berfungsi untuk membuat dan menambah Polyline.
11. *onLocationChanged* = berfungsi untuk memindahkan koordinat ketika user berpindah tempat.

Implementasi Algoritma Bellman-Ford dalam sistem dapat dilihat di class BellmanFord yang didapat dari postingan williamfiset di *website* Github. Sebelum memanggil *class* BellmanFord dan menjalankan algoritma Bellman-Ford, penulis harus menghubungkan dengan *database* untuk mengambil titik verteks, jalan, dan data

rumah sakit yang sudah ditandai oleh penulis. Koordinat verteks dan jalan tersebut akan saling terhubung dan menjadi suatu *graph*.

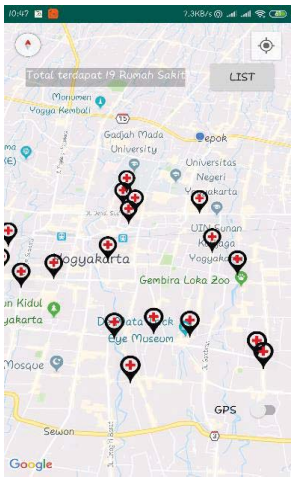
Penulis juga harus mencari posisi koordinat dari *user* saat ini ataupun posisi koordinat dari pilihan *user* sebagai titik awal dari pencarian jalur terpendek, lalu dari koordinat awal tersebut penulis akan mencari jarak ke verteks terdekat yang ada di *database* dan verteks yang terpilih menjadi verteks terdekat dari posisi awal tersebut akan menjadi verteks awal dalam pencarian jalur terpendek.

Agar dapat menjalankan algoritma Bellman-Ford dan memanggil *class* BellmanFord sistem harus memiliki *graph*, verteks awal, dan verteks tujuan. Verteks tujuan didapat dengan cara *user* memilih rumah sakit yang ingin dituju dan rumah sakit tersebut akan menjadi verteks tujuan

1. Mendapatkan lokasi koordinat *user* melalui GPS atau koordinat titik yang dibuat oleh *user*.
2. Menghitung jarak antara lokasi awal dengan setiap vertex yang ada di *database*.
3. Verteks dengan jarak terdekat akan menjadi verteks awal.
4. Verteks tujuan merupakan rumah sakit yang telah dipilih oleh *user*.
5. Mengambil seluruh data verteks dan jalur dari *database*.
6. Membentuk *graph* menjadi seperti *adjacency matrix* dalam bentuk *array* 2 dimensi dengan ukuran sebesar banyak *edge.*
7. Isi nilai *array* yang verteksnya saling berhubung atau *adjacency* dengan bobot jalur yang diperoleh dari *database.*
8. Buat *array* bernama *dist* dengan tipe data *double* sebesar ukuran *graph*.
9. Isi semua nilai dist dengan nilai tak terhingga dan isi nilai dist indeks menjadi 0.
10. Buat *array* bernama *prev* bertipe *Integer* yang digunakan untuk menyimpan verteks sebelum guna membuat jalur terpendek.
11. Untuk setiap verteks, lakukan relaksasi untuk setiap *edge* nya.
12. Relaksasi adalah pengecekan apakah bobot verteks sebelumnya ditambah bobot jalur menuju verteks selanjutnya lebih besar dari pada nilai bobot dari verteks selanjutnya itu sendiri. Jika iya, maka nilai bobot verteks selanjutnya menjadi bobot verteks sebelumnya ditambah dengan bobot jalur menuju verteks selanjutnya.
13. Lakukan perulangan dan pengecekan untuk kedua kalinya untuk mengecek apakah ada *negative cycle* didalam *graph* tersebut.
14. Panggil fungsi *reconstructShortestPath*, dan setelah mendapat jalur terpendek, akan disimpan kedalam sebuah *list* yang berbentuk *Integer* bernama *path*.
15. Ambil koordinat setiap verteks yang ada di *path* dari *database* jalur tabel *graph*.
16. Lalu gambar jalur tersebut menggunakan *polyline*.
17. Jalur terpendek menggunakan algoritma Bellman-Ford telah berhasil dibuat.

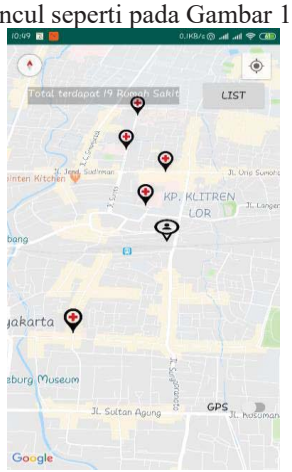
IV. HASIL DAN ANALISIS

Pada saat aplikasi dibuka di *smartphone* android, sistem akan menampilkan tampilan menu awal yang berupa map yang terdapat juga posisi *user* saat ini dan posisi rumah sakit yang tersedia berupa *marker* yang dapat dilihat di Gambar 10. Posisi *user* akan disimbolkan dengan titik biru sedangkan rumah sakit akan berbentuk *marker* hitam. Posisi *user* akan terdeteksi secara *realtime* menggunakan GPS. Terdapat juga tombol *compass* yang berfungsi sebagai petunjuk arah dan mengembalikan tampilan map kembali ke utara, tombol GPS yang berfungsi memindahkan tampilan map ke posisi user saat ini, tombol *List* yang berfungsi menampilkan halaman *list*, dan tombol *toggle* untuk menentukan apakah ingin menggunakan GPS sebagai titik awalan atau tidak.



Gambar 10 Tampilan awal aplikasi

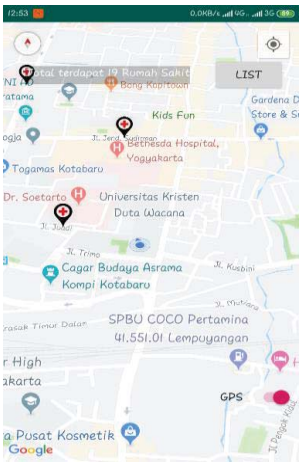
User dapat menentukan posisi / titik awal dengan cara menahan lama posisi yang diinginkan, lalu akan muncul marker posisi awal. *Marker* posisi awal dapat juga dipindah ke posisi yang diinginkan user secara dinamis melalui *map*. *Marker* akan muncul seperti pada Gambar 11.



Gambar 11 Menentukan posisi marker awal

Tetapi *marker* pilihan *user* akan menghilang seperti pada Gambar 12 jika *user* menekan tombol *toggle* yang mengaktifkan GPS sebagai titik awalan, dan akan muncul lagi jika *user* menonaktifkan GPS.

JUTEI Edisi Volume.3 No.2 Oktober 2019
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2019.32.184



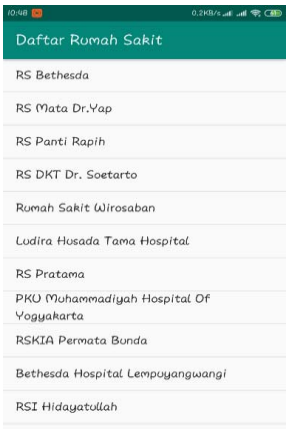
Gambar 12 Mengaktifkan GPS sebagai titik awalan

Setelah itu *user* dapat memilih rumah sakit yang ingin dituju dengan cara klik *marker* rumah sakit tersebut seperti pada contoh Gambar 13. Saat *marker* rumah sakit di tekan, akan muncul nama rumah sakit yang akan dituju dan akan muncul tombol *direction* yang digunakan untuk mencari jalur terpendek menuju rumah sakit tersebut.

Gambar 13 Tampilan saat marker rumah sakit dipilih

Tampilan menu *list* dapat diakses melalui tombol *list* yang berada di tampilan menu awal. Pada halaman menu *list* ini, akan ditampilkan daftar – daftar rumah sakit yang tersedia untuk dituju dan tersedia di kota Yogyakarta. Pada saat rumah sakit dipilih dari daftar rumah sakit, sistem akan menampilkan tampilan awal dengan posisi *map* akan langsung tertuju ke rumah sakit yang dipilih *user* sebelumnya.

Untuk menggunakan fungsi *direction*, *user* harus terlebih dahulu memilih tujuan rumah sakit yang akan dituju seperti contoh pada Gambar 13. Pada saat *user* menekan tombol *direction*, sistem akan mencari jalur terpendek dari verteks awal menuju ke rumah sakit tujuan user. Setelah mendapatkan jalur terpendek, sistem akan mewarnai biru jalur terpendek tersebut. Sistem juga akan memberikan keterangan verteks apa saja yang dilewati dan berapa jarak menuju rumah sakit tersebut dalam satuan meter dan memberikan waktu lama mencari jalur tersebut dalam satuan *milisecond*.



Gambar 14 Tampilan menu list

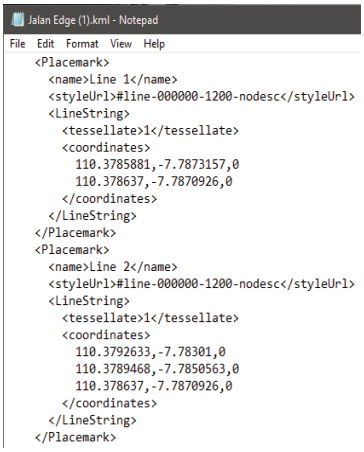
Gambar 15 Hasil pencarian jalur

Sebelum masuk ke dalam hasil dan analisis sistem, penulis akan menjelaskan bagaimana penulis mendapatkan data penelitian beserta analisis.

Pengumpulan data dilakukan dengan cara penulis menandai rumah sakit dan persimpangan jalan raya di Kota Yogyakarta melalui Google My Maps. Google My Maps dipergunakan untuk membuat peta baru atau mengedit peta yang telah dibuat yang disediakan oleh Google. Pengumpulan data lainya juga diambil penulis melalui *google.co.id/maps*.

Setelah menandai rumah sakit dan persimpangan jalan raya di kota Yogyakarta, penulis mengambil data *map* tersebut dengan cara mengekspor *map* tersebut ke *file* bertipe KML. Isi dari KML tersebut berupa koordinat latitude dan longitude dari titik yang telah penulis tandai. Setelah mendapatkan koordinat tersebut, penulis memasukan koordinat tersebut ke dalam database.

Gambar 16 Jalan yang sudah ditandai di Google My Maps



Gambar 17 Isi dari File KML

Penulis mencoba beberapa titik jalan yang sudah ditandai di *database* untuk mencoba dan mengetahui optimalitas Algoritma Bellman-Ford pada sistem pencarian jalur terpendek menuju rumah sakit dan membandingkan nya dengan aplikasi pencarian jalur dari Google yaitu Google Maps. Total data percobaan yang dilakukan penulis adalah 20 kali percobaan.

Penulis membandingkan data keluaran sistem dengan data pada Google Maps. Data tersebut akan dibandingkan dengan cara menghitung jarak terdekat menuju rumah sakit dengan posisi *user* saat ini atau posisi awal pilihan *user*, dalam satuan meter. Perhitungan jarak sistem dihitung dengan cara titik koordinat *user* saat ini atau posisi awal yang telah ditentukan oleh user dikurangi dengan koordinat rumah sakit yang menjadi tujuan *use.*

TABEL I
HASIL PERCOBAAN SISTEM

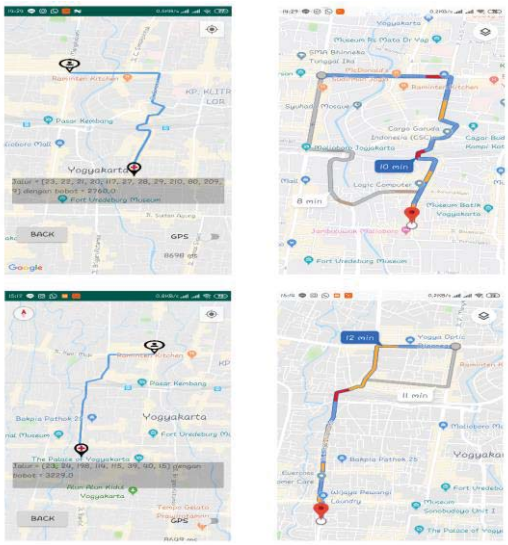
Percobaan	Waktu Proses	Hasil Sistem	Keterangan
1	8623 ms	1718 m	Titik ukdw - Panti Rapih
2	8371 ms	4896 m	Titik ukdw - Wirosaban
3	10406 ms	3132 m	Titik ukdw - PKU
4	8683 ms	2334 m	Titik ukdw - Siloam
5	8599 ms	4145 m	Tugu - Siloam
6	8698 ms	2760 m	Tugu - Lempuyawangi
7	8829 ms	2478 m	Tugu - Ludira Husada
8	8649 ms	3229 m	Tugu - RSKIA Rachmi
9	9571 ms	2151 m	Lampu merah Jl.Solo - Happy Land
10	8653 ms	1127 m	Lampu merah Jl.Solo - Bethesda
11	8739 ms	1695 m	Lampu merah Jl.Solo - Dr.Yap
12	10421 ms	1579 m	Malioboro - PKU
13	8700 ms	1833 m	Malioboro - Ludira
14	8353 ms	3972 m	Malioboro - RS PRATAMA
15	9419 ms	1398 m	Malioboro - DKT
16	8594 ms	2861 m	Taman Siswa - Wirosaban
17	8731 ms	3486 m	Prawirotaman - RS Ismangoen
18	8570 ms	4039 m	Galeria - RS Bhakti Ibu
19	8635 ms	2650 m	Galeria - Happy Land
20	8586 ms	1522 m	Batikan - Wirosaban
AVR	**8891,5 ms**	**2650,25 m**	

Pada Tabel 1 dapat dilihat bahwa penulis telah melakukan 20 kali percobaan dan sistem dapat mencari jalur terpendek menuju rumah sakit yang dituju dengan rata-rata waktu proses sebesar 8891,5 ms. Sistem juga selalu berhasil mencari jalur terpendek menuju rumah sakit selama 20 kali percobaan. Hasil dari perbandingan terbagi menjadi 2 hasil, yaitu pencarian jalur yang dicari sama dengan Google Maps dan berbeda dengan Google Maps.

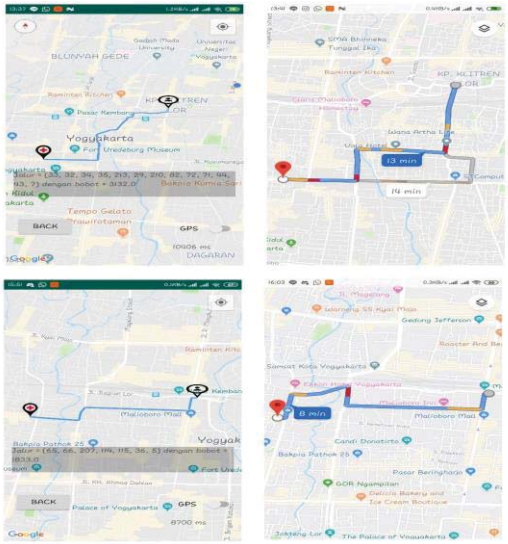
Pada Gambar 18 dapat dilihat bahwa hasil pencarian jalur terpendek yang dihasilkan oleh sistem sama dengan Google Maps.

Pada Gambar 19 dapat dilihat bahwa hasil pencarian jalur terpendek yang dihasilkan oleh sistem berbeda dengan *Google Maps*. Hal ini terjadi karena keterbatasannya data jalanan yang disimpan di *database* sistem.

JUTEI Edisi Volume.3 No.2 Oktober 2019
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2019.32.184



Gambar 18 Hasil keluaran sistem sama dengan Google Maps



Gambar 19 Hasil keluaran sistem berbeda dengan Google Maps

TABEL II
TABEL PERBANDINGAN SISTEM DENGAN GOOGLE MAPS

Percobaan	Hasil Sistem	Hasil Google	Selisih Jarak	Rute yang dilalui	Keterangan
1	1718 m	1700 m	18 m	SAMA	Titik ukdw - Panti Rapih
2	4896 m	4850 m	46 m	SAMA	Titik ukdw - Wirosaban
3	3132 m	3398 m	-266 m	BEDA	Titik ukdw - PKU
4	2334 m	2300 m	34 m	SAMA	Titik ukdw - Siloam
5	4145 m	4155 m	-10 m	SAMA	Tugu - Siloam
6	2760 m	2730 m	30 m	SAMA	Tugu - Lempuyawangi
7	2478 m	2383 m	95 m	BEDA	Tugu - Ludira Husada
8	3229 m	3240 m	-11 m	SAMA	Tugu - RSKIA Rachmi
9	2151 m	2200 m	-49 m	SAMA	Lampu merah Jl.Solo - Happy Land
10	1127 m	1100 m	27 m	SAMA	Lampu merah Jl.Solo - Bethesda
11	1695 m	1640 m	55 m	SAMA	Lampu merah Jl.Solo - Dr.Yap
12	1579 m	1524 m	55 m	SAMA	Malioboro - PKU
13	1833 m	1994 m	-161 m	BEDA	Malioboro - Ludira
14	3972 m	4350 m	-378 m	BEDA	Malioboro - RS PRATAMA
15	1398 m	1389 m	9 m	SAMA	Malioboro - DKT
16	2861 m	2915 m	-54 m	SAMA	Taman Siswa - Wirosaban
17	3486 m	3550 m	-64 m	SAMA	Prawirotaman - RS Ismangoen
18	4039 m	4000 m	39 m	SAMA	Galeria - RS Bhakti Ibu
19	2650 m	2680 m	-30 m	SAMA	Galeria - Happy Land
20	1522 m	1511 m	11 m	SAMA	Batikan - Wirosaban
AVR	**2650,25 m**	**2680,45 m**	**-30,2 m**		

Pada Tabel 2 dapat dilihat hasil pencarian jalur terpendek yang dihasilkan oleh sistem dan hasil pencarian jalur terpendek yang dihasilkan oleh Google Maps. Dari 20 kali percobaan yang dilakukan, penulis mendapatkan hasil sebesar 100% memiliki selisih jarak yang berbeda antara Google Maps dengan hasil keluaran dari sistem dengan rentang antara 9m sampai dengan 378m. Pada tabel dapat dilihat bahwa sebesar 45% dari 20 kali percobaan keluaran hasil sistem lebih cepat dari keluaran hasil dari Google Maps dan sebesar 55% keluaran hasil Google Maps lebih cepat dibandingkan dengan keluaran sistem, dikarenakan jarak antar jalan di data sistem dengan data Google Maps tidak sama persis.

Dari 20 kali percobaan yang dilakukan oleh penulis, sebanyak 16 kali atau sekitar 80% menunjukan rute yang sama antara keluaran sistem dengan keluaran Google Maps, sedangkan 4 kali atau sekitar 20% menunjukan rute yang berbeda antara keluaran sistem dengan Google Maps dikarenakan sistem hanya melalui jalan yang sudah ditandai oleh penulis. Dan dari data diatas dapat disimpulkan bahwa data keluaran sistem lebih cepat jika dibandingkan dengan Google Maps, karena data keluaran sistem memiliki rata-rata jarak yang lebih kecil sebesar 2650,25m dibandingkan dengan rata-rata jarak yang dikeluarkan oleh Google Maps yaitu sebesar 2680,45m.

V. KESIMPULAN

Berdasarkan hasil pengujian dan analisis sistem yang telah dilakukan, maka dapat disimpulkan bahwa:

1. Algoritma Bellman-Ford dapat diimplementasikan untuk mencari jalur terpendek menuju rumah sakit berbasis Android.
2. Algoritma Bellman-Ford yang diimplementasikan oleh sistem hampir mendekati data dari Google Maps karena dari 20 kali percobaan didapatkan selisih jarak yang kecil, yaitu sebesar 30,2 meter dari rata-rata jarak antara hasil keluaran sistem dengan Google Maps.
3. Algoritma Bellman-Ford dalam pencarian jalur terpendek menghasilkan sebesar 80% atau sebanyak 16 kali sistem mengeluarkan jalur yang sama dengan jalur yang dikeluarkan Google Maps. Hal ini dikarenakan perbedaan kelengkapan data verteks, *edge*, dan bobot pada graf yang dimiliki sistem dengan yang dimiliki Google Maps.
4. Banyaknya jumlah verteks dan *edge* dalam data mempengaruhi lama proses algoritma untuk mencari jalur terpendek karena algoritma Bellman-Ford harus mengecek setiap verteks dan *edge*. Rata-rata sistem mencari jalur selama 8891,5ms dengan banyak verteks 215 dan *edge* sebanyak 528.

Adapun saran yang diberikan penulis kepada pengembang selanjutnya adalah sebagai berikut:

1. Pengembang selanjutnya dapat memperhatikan tingkat kemacetan jalan.
2. Pengembang selanjutnya dapat memberikan navigasi dan rute yang akan dituju ke rumah sakit tersebut.
3. Pengembang selanjutnya dapat menambahkan titik jalan secara dinamis agar lebih mudah untuk menambah data dan koordinat jalan.
4. Sistem dapat dikembangkan dengan algoritma yang berbeda atau mengkombinasikan algoritma Bellman-Ford dengan algoritma lainnya untuk mencari jalur terpendek.

UCAPAN TERIMA KASIH

Dalam menyelesaikan penelitian ini, penulis mengucapkan terima kasih kepada Bapak Gunawan Santosa dan Bapak Junius Karel yang telah mengarahkan penelitian dengan baik. Juga diucapkan terima kasih kepada keluarga dan teman-teman seperjuangan program studi Informatika UKDW angkatan 2015 yang selalu setia mendukung dan mendoakan setiap saat.

DAFTAR PUSTAKA

[1] Christian, A. (2013). Studi Literatur Perbandingan Algoritma Dijkstra dan Bellman-Ford dalam Pencarian Jarak Terdekat. Undergraduate Thesis, Duta Wacana Christian University.

[2] Kristyaningrum, I. (2009). Perbandingan Algoritma Dijkstra dan Bellmanford Untuk Pencarian Jalur Terpendek Pada Graf Berarah. *Undergraduate thesis, Duta Wacana Christian University*.

[3] Kurniawijaya, P. A. (2010). Pencarian Pom Bensin Terdekat di Denpasar Menggunakan Algoritma Dijktsra. *Undergraduate thesis, Duta Wacana Christian University*.

[4] Halim, J. (1998). Visualisasi Jalur Terpendek dengan Algoritma Bellman-Ford Pada Studi Kasus Jalan Darat Antar Kota (Pulau Jawa). *Undergraduate Thesis, Duta Wacana Christian University*.

[5] Munir, R. (2010). Matematika Diskrit. Bandung: Penerbit Informatika Bandung.

[6] Safaat, N. (2012). Pemrograman Aplikasi Mobile Smartphone Dan Tablet Pc Berbasis Android. Bandung: Informatika Bandung.

[7] Susilo, Y. S., Pranjoto, H., & Gunadhi, A. (2014). Sistem Pelacakan dan Pengamanan Kendaraan Berbasis GPS dengan Menggunakan Komunikasi GPRS. *Jurnal Ilmiah Widya Teknik*, 22.

[8] Hudiono, Taufik, M., Koesmarijanto, & Darmono, H. (2018). *Sistem Komunikasi Radio dan Laboratorium.* Polinema.

[9] Tulach, J. (2008). *Practical API Design: Confessions of a Java Framework Architect*. Apress.

[10] Svennerberg, G. (2010). *Beginning Google Maps API 3*. Apress.

[11] Cormen, T. H. (2009). *Introduction to Algorithms* (Vol. III). Cambridge: The MIT Press.

JUTEI Edisi Volume.3 No.2 Oktober 2019
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2019.32.184