2017

Pemilihan Rute Destinasi Objek Wisata di Kawasan Danau Toba Berbasis ANdroid Menggunakan Algoritma L-Deque

Pohan, Purnama Sari

PEMILIHAN RUTE DESTINASI OBJEK WISATA DI KAWASAN DANAU TOBA BERBASIS ANDROID MENGGUNAKAN ALGORITMA L-DEQUE

SKRIPSI

PURNAMA SARI POHAN
131402050

PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2017

PEMILIHAN RUTE DESTINASI OBJEK WISATA DI KAWASAN DANAU TOBA BERBASIS ANDROID MENGGUNAKAN ALGORITMA L-DEQUE

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Teknologi Informasi

PURNAMA SARI POHAN
131402050

PROGRAM STUDI S-1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2017

PERSETUJUAN

Judul	: PEMILIHAN RUTE DESTINASI OBJEK WISATA DI KAWASAN DANAU TOBA BERBASIS ANDROID MENGGUNAKAN ALGORITMA L-DEQUE
Kategori	: SKRIPSI
Nama	: PURNAMA SARI POHAN
Nomor Induk Mahasiswa	: 131402050
Program Studi	: SARJANA (S1) TEKNOLOGI INFORMASI
Fakultas	: ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA

Diluluskan di
Medan, 26 Oktober 2017

Komisi Pembimbing :

Dosen Pembimbing II	Dosen Pembimbing I
Dedy Arisandi,ST., M.T.	Baihaqi Siregar, S.Si., M.T.
NIP. 197908312009121002	NIP. 197901082012121002

Diketahui/disetujui oleh
Program Studi S1 Teknologi Informasi
Ketua,

Romi Fadillah Rahmat, B.Com.Sc.,M.Sc.
NIP. 19860303 201012 1 004

PERNYATAAN

PEMILIHAN RUTE DESTINASI OBJEK WISATA DI KAWASAN

DANAU TOBA BERBASIS ANDROID MENGGUNAKAN

ALGORITMA L-DEQUE

SKRIPSI

Saya menyatakan bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 26 Oktober 2017

Purnama Sari Pohan
131402050

PENGHARGAAN

Puji dan syukur kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya, sehingga Penulis dapat menyelesaikan penyusunan skripsi ini, sebagai syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi S1 Teknologi Informasi Universitas Sumatera Utara.

Penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar–besarnya kepada :

1. Bapak Prof. Dr. Runtung Sitepu, S.H., M.Hum selaku Rektor Universitas Sumatera Utara.
2. Bapak Prof. Opim Salim Sitompul, M.Sc., selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.
3. Bapak Romi Fadillah Rahmat, B.Com.Sc.,M.Sc. selaku Ketua Program Studi S1 Teknologi Informasi Universitas Sumatera Utara.
4. Bapak Baihaqi Siregar, S.Si., M.T. selaku Dosen Pembimbing I yang telah memberikan bimbingan, saran, masukan dan dukungan kepada penulis dalam pengerjaan skripsi ini.
5. Bapak Dedy Arisandi,ST., M.T. selaku Dosen Pembimbing II yang telah memberikan bimbingan, saran, masukan dan dukungan kepada penulis dalam pengerjaan skripsi ini.
6. Romi Fadillah Rahmat, B.Com.Sc.,M.Sc. selaku Dosen Pembanding I yang memberikan kritik dan saran untuk penyempurnaan skripsi ini.
7. Ibu Sarah Purnamawati, ST., M.Sc. selaku Dosen Pembanding II yang memberikan kritik dan saran untuk penyempurnaan skripsi ini.
8. Seluruh dosen dan pegawai di Fakultas Ilmu Komputer dan Teknologi Informasi USU, terkhususnya di Program Studi Teknologi Informasi.
9. Teristimewa Ayahanda Parluhutan Pohan dan Ibunda Hotna Sari Siregar yang tidak henti-hentinya memberikan doa, dukungan dan motivasi yang selalu menjadi semangat penulis untuk meyelesaikan skripsi ini.

10. Kakak, Abang dan Adik tersayang Efrida Hannum Pohan, Eri Anto Pohan, Martua Pohan, Holida Santi Pohan dan Hasanuddin Pohan beserta keluarga besar yang selalu memberikan doa, dukungan dan memotivasi penulis untuk menyelesaikan skripsi ini.
11. Kakak dan Abang mentor yaitu kak Bia dan bang Rio yang telah menjadi teman diskusi, tempat bertanya dan berbagi ilmu dalam penyelesaian skripsi ini.
12. Teman-teman yang luar biasa Nurul, Nikmah, Adel, Ayu, Nurajijah, Fatimah, Ani, Nina, Halimah, Ami, Rama, Melur, Bambang yang telah menjadi teman diskusi penulis dan senantiasa memberikan semangat.
13. Teman-teman Kom B 2013, serta teman-teman stambuk 2013 atas doa dan dukunganya sehingga penulis dapat menyelesaikan skripsi ini.
14. Dan semua pihak yang telah banyak membantu yang tidak bisa disebutkan satu-persatu.

Semoga semua kebaikan, bantuan, perhatian, serta dukungan yang telah diberikan kepada penulis mendapatkan berkat yang melimpah dari Allah SWT.

Medan, 26 Oktober 2017

Penulis,

Purnama Sari Pohan

ABSTRAK

Destinasi wisata adalah suatu kawasan spesifik atau tujuan wisata yang dipilih oleh orang-orang yang akan melakukan perjalanan wisata baik wisatawan lokal maupun wisatawan asing salah satunya yaitu Danau Toba. Danau Toba secara umum memiliki berbagi macam objek wisata yang cukup menarik diantaranya berupa:Pulau Samosir, Parapat, Ambarita dan lain sebagainya. Maka dari itu, diperlukan sebuah sistem informasi geografis dalam penentuan rute terdekat antar objek wisata untuk mempermudah wisatawan dalam memilih objek wisata yang diinginkan. Penelitian ini menggunakan algoritma L-Deque dimana algoritma L-Deque dapat digunakan suatu *linier list* yang menambah dan menghapus elemennya dapat dilakukan pada kedua sisi ujung *list*, tetapi tidak dapat dilakukan ditengah tengah *list*, sedangkan fungsinya yaitu untuk mengambil sebuah elemen dari *queue*. Berdasarkan pengujian, hasil penelitian yang diperoleh dari sistem yang dibangun menunjukkan bahwa algoritma L-Deque dapat memberikan solusi dalam penentuan rute terdekat dengan hasil tingkat efisiensi sebesar 8,33 %.

Kata kunci : Destinasi wisata, Algoritma L-Deque, Sistem Informasi Geografis (SIG), *Shortest Path.*

ELECTION OF DESTINATION ROUTE OF TOURISM OBJECT IN THE AREA LAKE TOBA BASED ANDROID USING L-DEQUE ALGORITHM

ABSTRACT

A tourist destination is a specific area or destination chosen by those who will travel tours both local and foreign tourists an one of them is Lake Toba. Lake Toba in general has a wide range of attractions are quite interesting such as: Samosir Island, Parapat, Ambarita and others. Therefore, we need a geographical information system in determining the shortest route between the attractions to facilitate the tourists in choosing the desired attraction. This study uses L-Deque algorithm where in L-Deque algorithm can be used a *linear list* that add and remove elements can be done on both sides of the end of the *list,* but it can not be done amid *lists,* while the function is to take an element of the *queue.* Based on testing, results obtained research from system built show that L-deque algorithm can give solutions in determination route nearest with the result value effisicien by 8,33%.

Keywords : A tourist destination, Geographic Information Systems (GIS), L-Deque Algorithm, *Shortest Path.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

Halaman

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Destinasi wisata adalah suatu kawasan spesifik yang dipilih oleh pengunjung yang mana ia dapat tinggal dalam waktu tertentu (Hardinoto, 1996). Sesuai dengan penjelasan diatas maka dapat dikatakan bahwa destinasi merupakan tempat atau tujuan wisata oleh orang-orang yang akan melakukan perjalanan wisata baik wisatawan lokal maupun wisatawan asing.

Teknologi semakin berkembang sehingga setiap informasi dapat diakses dan dipublikasikan dalam waktu singkat, perkembangan teknologi dapat dimanfaatkan untuk memperkenalkan atau mempublikasikan semua keindahan alam yang berada di sekitar kita. Teknologi juga turut mengambil bagian menjadi media dan sarana informasi bagi para wisatawan dalam hal mencari dan menentukan destinasi objek wisata yang akan mereka tuju . Hal inilah yang menjadi motivasi penulis untuk memperkenalkan kepada publik setiap titik daerah asal Indonesia memiliki keindahan alam meliputi pemandangan, gunung, danau, pantai dan segala sesuatu dapat dijadikan sebagai destinasi wisata untuk semua kalangan salah satunya yaitu Danau Toba. Danau Toba merupakan objek wisata yang berada dikawasan 7 kabupaten pada provinsi Sumatera Utara. Lokasi wisata tersebut memiliki panjang hingga 100 kilometer, dan lebar 30 kilometer. Ditengah-tengah danau toba terdapat pulau vulkanik yang dikenal dengan nama Samosir. Pulau tersebut memiliki luas 640 kilometer persegi,atau setara dengan luas negara singapura.

Pada tahun 2019, Kementerian Pariwisata menargetkan adanya kunjungan satu juta wisatawan ke kawasan Danau Toba. Danau Toba secara umum memiliki berbagi macam objek wisata yang cukup menarik diantaranya berupa:Pulau Samosir, Parapat, Bakara, Pantai Lumban Silintong Balige, Pulau Sibandang, Tongging, Pusuk Buhit, Sipinsur, Simanindo, Ambarita, Pangguruan, Tomok, Tuktuk , Aek sipitu dai, Danau sidihoni, Danau aek natorang, Kampung siallagen, Rumah Pengasingan Mantan

Presiden RI Soekarno dan sebagainya. Dengan melihat berbagai macam objek wisata yang cukup menarik di kawasan danau toba tentunya dapat dijadikan modal untuk lebih mengembangkan wilayah ini sebagai daerah tujuan pariwisata sehingga penentuan rute objek wisata di kawasan danau toba sangat diperlukan untuk meminimalkan biaya dan menghemat waktu perjalanan. Semakin cepat wisatawan sampai ke tempat objek wisata yang satu maka semakin banyak juga waktu untuk mengunjungi tempat objek wisata lainnya.

Pada kasus ini penulis mengambil masalah untuk menentukan rute terdekat pada objek wisata di kawasan Danau Toba. Lintasan terpendek (*Shortest path*) dapat digunakan untuk mencari rute terdekat dari suatu tempat ke tempat tujuan yang diinginkan dengan menghasilkan jarak tempuh yang seminimum mungkin sehingga dapat menghemat waktu dan biaya perjalanan, Pencarian jalur terpendek ini telah banyak diterapkan di berbagai bidang sehingga dapat mengoptimasi kinerja suatu sistem. Pencarian suatu jalur tersebut dapat diselesaikan dengan menggunakan *graph*, g*raph* yang digunakan dalam mencari rute terdekat adalah *graph* terhubung dan *graph* berbobot.

Berdasarkan beberapa masalah di atas, maka untuk mempermudah menginformasikan kepada wisatawan dalam pemilihan destinasi objek wisata dibutuhkan sebuah sistem informasi geografis dalam penentuan rute yang harus dilalui berdasarkan jarak maupun waktu. Oleh karena itu maka penulis mengajukan penelitian dengan judul " Pemilihan rute destinasi objek wisata di kawasan Danau Toba berbasis android menggunakan algoritma L-Deque" dimana algoritma *L-Deque* dapat digunakan suatu *linier list* yang menambah dan menghapus elemennya dapat dilakukan pada kedua sisi ujung *list*, tetapi tidak dapat dilakukan ditengah tengah *list*, sedangkan fungsinya yaitu untuk mengambil sebuah elemen dari *queue* (Gallo & Pallotino, 1986). Dengan adanya aplikasi ini diharapkan dapat membantu para wisatawan dalam pemilihan rute destinasi objek wisata di kawasan danau toba dengan tepat sesuai dengan kebutuhan wisatawan.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, maka rumusan masalah yang akan diteliti adalah belum adanya penelitian untuk mengetahui rute terdekat dalam kasus antar objek wisata di kawasan Danau Toba sehingga belum diketahuinya jarak terdekat dan waktu yang optimal.

1.3 Batasan Masalah

Adapun batasan masalah yang akan dibahas adalah:

1. Dalam kasus ini yang diteliti adalah sebuah grap berarah terhubung *(directed connected graph)* dengan menggunakan *vertex* yang telah ditentukan.
2. Bobot yang digunakan adalah jarak.
3. Pengambilan data bobot jarak antara *vertex* dan *edge* pada *graph* antar tempat wisata menggunakan bantuan dari *Google Maps* versi 2017.
4. Sistem ini berlaku hanya bebarapa objek wisata di kawasan Danau Toba dan titik tujuan yang akan dikunjungi sebanyak delapan belas titik yaitu tempat wisata yang paling sering dikunjungi di kawasan Danau Toba. Nama tempat wisata yang akan diteliti yaitu: Pulau Samosir, Parapat, Pantai Bulbul, Tongging, Pusuk Buhit, Sipinsur, Simanindo, Ambarita, pangguruan, tuktuk, aek sipitu dai,danau sidihoni, Sialanguan, rumah Pengasingan Mantan Presiden RI Soekarno, Balige Cultural Center, Haranggaol, dan Paropo.
5. Aplikasi ini menggunakan data titik awal yang diinginkan dari objek wisatadi kawasan Danau Toba yang telah ditentukan.
6. Menggunakan bahasa *java mobile programming*.
7. Lingkup kerja sistem adalah perangkat mobile yang menggunakan sistem operasi *Android*.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah menerapkan Algoritma *L-Deque* pada sistem informasi geografis dalam penentuan rute destinasi objek wisata di kawasan Danau Toba pada perangkat berbasis Android.

1.5 Manfaat Penelitian

Manfaat yang dihasilkan dari penelitian ini yaitu:

1. Membantu para wisatawan dalam menentukan rute destinasi objek wisata terdekat sesuai kebutuhan yang diinginkan.
2. Menambah wawasan dan pemahaman bagi penulis serta pembaca tentang penggunaan Algoritma *L-Deque*.
3. Membantu wisatawan meminimalkan biaya dan waktu dalam perjalanan.
4. Menjadi bahan pembelajaran bagi pembaca dan menjadi referensi untuk penelitian selanjutnya.

1.5 Metode Penelitian

Tahapan yang dilakukan dalam penelitian ini adalah:

1. Studi Literatur

   Pada tahap ini dilakukan pengumpulan referensi yang diperlukan dalam penelitian. Hal ini dilakukan untuk memperoleh informasi dan data yang diperlukan untuk penulisan skripsi ini. Referensi yang digunakan dapat berupa buku, jurnal, artikel, situs internet yang berkaitan dengan penelitian ini dengan beberapa topik seperti pengertian graf, jenis- jenis graf, pengertian algoritma, *shortest path*, algoritma *L-Deque* dan Sistem Informasi Geografis atau *Geographic Information Sistem* (SIG).

2. Pengumpulan dan Analisis Data

   Pada tahap ini dilakukan pengumpulan dan analisa data yang berhubungan dengan penelitian ini seperti fungsi algoritma *L-Deque* dalam penentuan rute terdekat antar tempat wisata di kawasan Danau Toba.

3. Perancangan Sistem

   Merancang sistem sesuai dengan rencana yang telah ditentukan, yaitu meliputi perancangan desain awal seperti perancangan tampilan *GUI* (*Graphic User Interface*). Proses perancangan ini berdasarkan pada batasan masalah dari penelitian ini.

4. Implementasi Sistem

   Pada tahap ini sistem telah selesai dikembangkan dengan algoritma *L-Deque* dalam penentuan rute terdekat antar objek wisata di kawasan Danau Toba ke dalam bentuk program.

5. Pengujian Sistem

   Pada tahap ini akan dilakukan pengujian terhadap sistem yang telah dikembangkan.

6. Dokumentasi Sistem

   Melakukan pembuatan dokumentasi sistem mulai dari tahap awal hingga pengujian sistem, untuk selanjutnya dibuat dalam bentuk laporan penelitian (skripsi).

1.7 Sistematika Penulisan

Agar pembahasan lebih sistematis, maka tulisan ini dibuat dalam lima bab, yaitu:

BAB 1 PENDAHULUAN

Bab ini akan menjelaskan mengenai latar belakang penelitian judul skripsi "Pemilihan Rute Destinasi Objek Wisata di kawasan Danau Toba berbasis Android Menggunakan Algoritma L-Deque". rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metode penelitian, tinjauan pustaka dan sistematika penulisan skripsi.

BAB 2 LANDASAN TEORI

Bab ini membahas tentang teori-teori yang berhubungan dengan pengertian graf, jenis- jenis graf, pengertian algoritma, sistem informasi geografis atau *Geographic Information Sistem* (SIG) , *shortest path,* algoritma *L-Deque,* dan sebagainya.

BAB 3 ANALISIS DAN PERANCANGAN SISTEM

Berisi tentang uraian analisis mengenai proses kerja dari algoritma *L-Deque* dalam penentuan rute terdekat antar objek wisata di kawasan Danau Toba tersebut yang terdiri dari *Unified Modeling Language(UML),* arsitektur umum serta perancangan *interface* pengguna.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Pada tahap ini dilakukan pembuatan sistem dan *coding* sesuai dengan analisis dan perancangan, kemudian melakukan pengujian sistem.

BAB 5 KESIMPULAN DAN SARAN

Bab terakhir akan memuat kesimpulan isi dari keseluruhan uraian dari bab-bab sebelumnya dan saran-saran dari hasil yang diperoleh yang diharapkan dapat bermanfaat dalam pengembangan selanjutnya.

BAB 2

LANDASAN TEORI

2.1 Destinasi wisata

Destinasi wisata adalah suatu kawasan spesifik yang dipilih oleh pengunjung yang mana ia dapat tinggal dalam waktu tertentu (Hardinoto, 1996). Sesuai dengan penjelasan diatas maka dapat dikatakan bahwa destinasi merupakan tempat atau tujuan wisata oleh orang-orang yang akan melakukan perjalanan wisata baik wisatawan lokal maupun wisatawan asing.

2.2 Sistem Informasi Geografis (SIG)

2.2.1 Pengertian Sistem Informasi Geografis (SIG)

Sistem Informasi Geografis atau *Geographic Information Sistem* (SIG) merupakan suatu sistem informasi yang berbasis komputer, dirancang untuk bekerja dengan menggunakan data yang memiliki informasi spasial (bereferensi keruangan). Sistem ini mengcapture, mengecek, mengintegrasikan, memanipulasi, menganalisa, dan menampilkan data yang secara spasial merefrensikan kepada kondisi bumi. Teknologi SIG mengintegrasikan operasi-operasi umum database, seperti *query* dan analisa statistik, dengan kemampuan visualisasi dan analisa yang unik yang dimiliki oleh pemetaan. Kemampuan inilah yang membedakan SIG dengan Sistem Informasi lainnya yang membuatnya menjadi berguna berbagai kalangan untuk menjelaskan kejadian, merencanakan strategi, dan memprediksi apa yang terjadi (Aini, 2011).

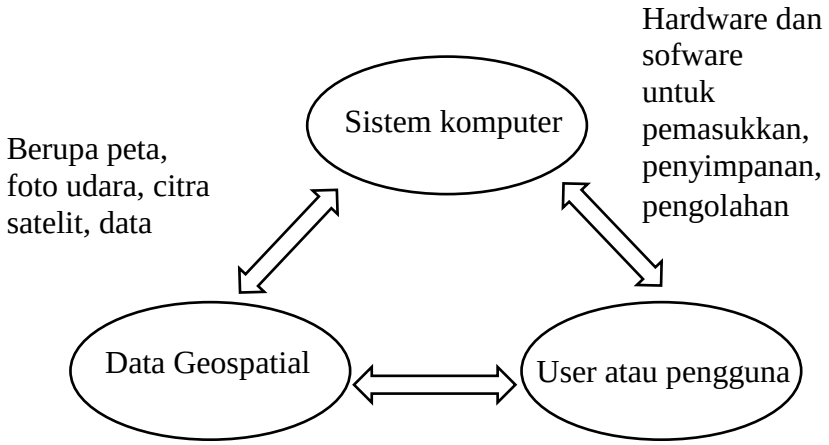
Beberapa manfaat dari SIG adalah mengetahui jarak antara satu daerah dengan daerah lain, memberikan alternatif jalan dari satu daerah ke daerah lain, memberi informasi seputar daerah yang diinginkan, menemukan lokasi kecelakaan dengan cepat, dan masih banyak lagi informasi yang dapat diperoleh dengan menggunakan bantuan SIG tersebut (Chang,2004).

Dalam suatu sistem informasi geografis, terdapat beberapa komponen utama yang saling berintegrasi dan saling terkait, yaitu :

- Sistem komputer (*Hardware* dan *Software)*

- Data Geospatial.
- User atau pengguna.

Pada gambar 2.1 adalah komponen utama sistem informasi geografis yang terdiri atas perangkat keras, perangkat lunak, data geografis, dan sumber daya manusia yang bekerja bersama secara efektif untuk memasukan, menyimpan, memperbaiki, memperbaharui mengelola, memanipulasi, mengintegrasikan, menganalisa, dan menampilkan data dalam suatu informasi berbasis geografis (Sumaja. 2013).



Gambar 2.1 Komponen GIS

2.2.2 Subsistem SIG

SIG dapat diuraikan menjadi beberapa subsistem sebagai berikut:

a. Data *Input*

Subsistem ini bertugas untuk mengumpulkan, mempersiapkan, dan menyimpan data spasial dan atributnya dari berbagai sumber. Subsistem ini pula yang bertanggung jawab dalam mengonversikan atau mentransformasikan format-format data aslinya ke dalam format yang dapat digunakan oleh perangkat SIG yang bersangkutan.

b. Data *Output*

Subsistem ini bertugas untuk menampilkan atau menghasilkan keluaran (termasuk mengekspornya ke format yang dikehendaki) seluruh atau sebagian basis data (spasial) baik dalam bentuk *softcopy* maupun *hardcopy* seperti halnya tabel, grafik, report, peta, dan lain sebagainya.

c. Data *Management*

Subsistem ini mengorganisasikan baik data spasial maupun tabel-tabel atribut terkait ke dalam sebuah sistem basis data sedemikian rupa hingga mudah dipanggil kembali atau di*retrieve*, diupdate, dan diedit.

d. Data *Manipulation* & *Analysis*

Subsistem ini menentukan informasi-informasi yang dapat dihasilkan oleh SIG. Selain itu sub-sistem ini juga melakukan manipulasi (evaluasi dan penggunaan fungsi-fungsi dan operator matematis & logika) dan pemodelan data untuk menghasilkan informasi yang diharapkan.

2.3 Global positioning system (GPS)

Global positioning system (GPS) atau sistem pemosisi global menggunakan sistem yang digunakan menentukan posisi dipermukaan bumi dengan sinkronisasi sinyal satelit. Dengan bantuan GPS seseorang dapat mengetahui posisi objek yang diinginkannya dengan bantuan perangkat yang memiliki sensor GPS didalamnya. GPS bekerja ketika sejumlah satelit yang berada di orbit Bumi memancarkan sinyalnya ke Bumi kemudian sinyal tersebut ditangkap oleh sebuah alat penerima yang nantinya diubah menjadi informasi berupa titik lokasi dari alat penerima tersebut. Karena alat ini bergantung penuh pada satelit maka sinyal satelit merupakan hal yang paling penting untuk mendapatkan informasi posisi objek yang berupa titik koordinat. Untuk itu perlu diperhatikan hal-hal yang dapat mengganggu sinyal satelit antara lain adalah kondisi geografis, alat yang menggunakan gelombang elektromagnetik, gedung dan sinyal yang memantul. Keakuratan maupun ketepatan menjadi perhatian utama dalam sistem ini untuk mendapatkan sebuah lokasi atau koordinat. Tingkat akurasi pada GPS lebih sering dipengaruhi oleh faktor lingkungan disekitarnya. Ketika alat GPS berada di sebuah lembah, maka tingkat akurasinya akan jauh lebih rendah daripada di puncak gunung.

2.4 Algoritma

2.4.1 Pengertian Algoritma

Algoritma ialah sebuah urutan atau langkah-langkah penyelesaian yang logis dari suatu masalah, dibuat dengan notasi yang mudah dipahami sehingga dapat dilaksanakan oleh pemroses (Utami, 2004).

Kata algoritma mempunyai asal kata dari nama penulis buku Arab yang terkenal yaitu Abu Ja'far Muhammad ibnu Musa al-Khuwarizmi (al-Khuwarizmi dibaca oleh orang Barat menjadi *algorism*). Al-Khuwarizmi menulis buku yang berjudul *Kitab al jabar wal-muqabala*, yang artinya "Buku pemugaran dan pengurangan". Dari judul buku itu juga diperoleh akar kata "aljabar" (*algebra*). Perubahan kata *algorism* menjadi *algorithm* muncul karena kata *algorism* sering dikelirukan dengan *arithmetic*, sehingga akhiran -*sm* berubah menjadi *thm*. Karena perhitungan dengan angka Arab sudah menjadi hal yang biasa, maka lambat laun kata *algorithm* berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna aslinya. Dalam bahasa Indonesia, kata algorithm diserap menjadi algoritma (Munir, 2014).

2.4.2 Karakteristik Algoritma

Menurut Bapak "Analisa Algoritma" (Knuth, 1973) algoritma yang baik dan benar harus memiliki karakteristik sebagai berikut :

1. *Input* : Sebuah algoritma harus memiliki nol input atau lebih dari pengguna sebagai data masukan untuk diolah.
2. *Output* : Sebuah algoritma harus memiliki sebuah ouput yang merupakan hasil dari olahan data.
3. *Finiteness* : Sebuah algoritma harus terbatas dan berakhir *(terminate)* setelah melakukan sejumlah langkah proses.
4. *Defineteness* : Sebuah Algoritma tidak menimbulkan makna ganda *(ambigious).*
5. *Effectiveness* : Langkah-langkah algoritma dikerjakan dalam waktu yang wajar

2.5 Algoritma L-Deque

Deque adalah sebuah daftar yang menggabungkan sifat-sifat dari kedua antrian dan tumpukan atau antrian. Sebuah *deque* adalah daftar di mana penambahan dan penghapusan yang mungkin di kedua ujung. Sebuah *deque* sudah tekenal digunakan dalam algoritma *D'Esopo-Pape*, algoritma ini dipanggil *L-Deque*. Dalam Q *deque* digunakan di penambahan *L-Deque* dibuat di kedua ujungnya, sementara penghapusan dibuat di kepala (Gallo & Pallotino, 1986).

Dequeue terdiri dari dua jenis, yaitu :

1. Input -Restricted-Deque

   Deque yang operasi pemasukan elemen datanya hanya dapat dilakukan di satu ujung kanannya *(right),* tetapi dapat menghapus dari kedua ujungnya (*left* dan *right).*

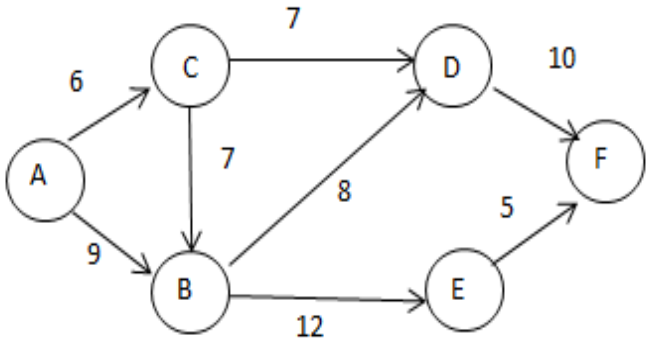
2. Output-Restricted-Deque

   Deque yang operasi pemasukan elemen datanya dapat dilakukan melalui kedua ujungnya (*left* dan *right*), tetapi hanya dapat menghapus dari ujung kanannya (*right).*

Langkah-langkah Algoritma *L-Deque* untuk mencari rute terpendek dapat dirumuskan sebagai berikut:

1. Masukkan seluruh *vertex* yang ada pada *graph* ke daftar data deque.
2. Tentukan *vertex* yang akan menjadi *vertex* asal dan berinilai untuk *distance* dari *vertex source* = 0, dan yang lain *infinite*(∞).
3. Periksa semua *vertex* yang dapat dilalui dari *vertex* asal, dan periksa setiap *edges* (u,v) yang ada pada *graph* dan tentukan d[v] untuk rute terpendek pertama dengan cara :
   i. Hitung d[v] > d[u] + *edges* [u][v].
   ii. Jika d[[v] > d[u] + *edges* [u][v] maka d[v] = d[u] + *edges* [u][v] dan tukar nilai *infinite* (∞) pada *vertex* yang telah dikunjungi tersebut dengan nilai d[v]. Dan jika ada nilai d[v] pada *vertex* tersebut dan nilai d[v] yang baru dihasilkan lebih kecil maka tukar nilai d[v] dengan nilai d[v] yang baru.
4. Dengan cara yang sama, ulangi langkah no. 3 untuk menentukan rute terpendek berikutnya sampai *vertex* yang ada dalam data deque sama dengan null.

Contoh sederhana penyelesaian sebuah *graph* untuk menentukan rute terpendek menggunakan algoritma *L-Deque* dengan *vertex* awal A dan *vertex* akhir D dapat dilihat pada gambar 2.2 dibawah ini.



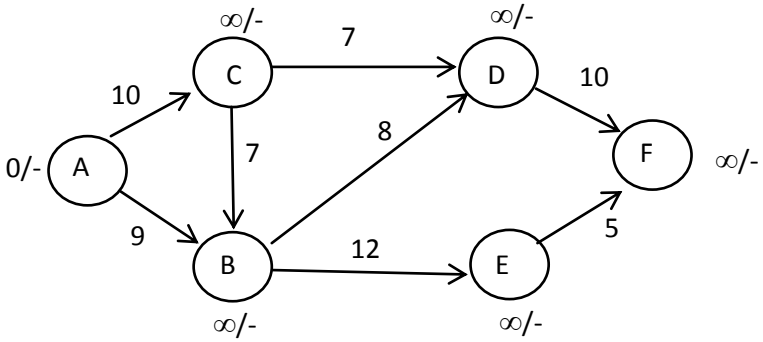
Gambar 2.2 Graph Berarah Terhubung

Langkah 1:

Masukkan seluruh *vertex* yang ada pada *graph* ke daftar data deque. Daftar deque [A, B, C, D, E, F].

Langkah 2:

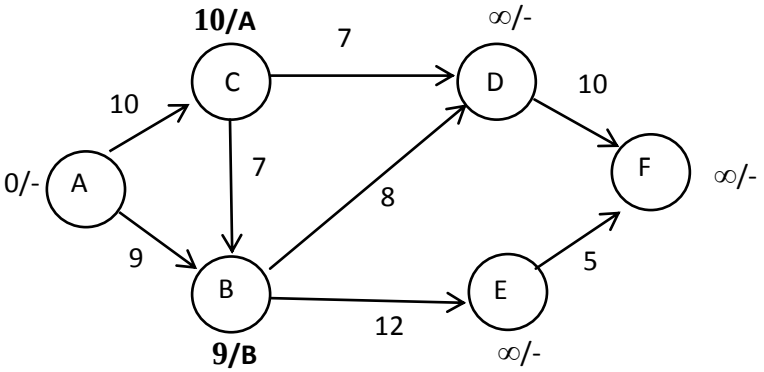
Vertex asal diatas pada *graph* ialah *vertex* A, tandai *vertex* tersebut dengan nilai 0 dan beri nilai *infinite*(∞) pada *vertex* lainnya. Dapat dilihat pada gambar 2.3.



Gambar 2.3 *Graph* Penentuan *Vertex* Asal

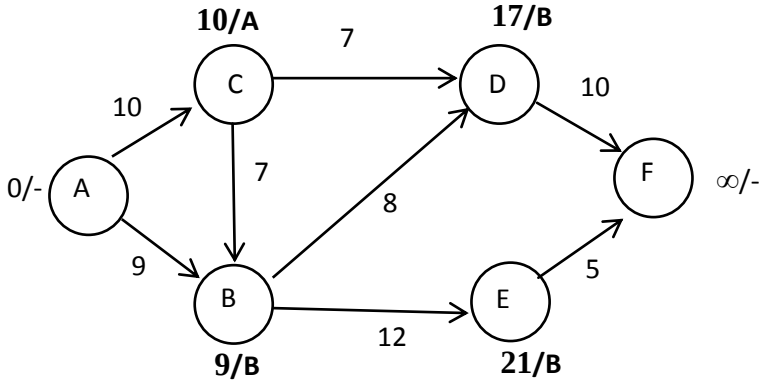
Langkah 3:

Vertex yang dapat dilalui dari *vertex A* ialah *vertex* B dengan bobot jarak 9 dan juga *vertex* C dengan bobot jarak 10. Hitung d[v] untuk *vertex* B, d[v] > d[u] + *edges* [u][v], d[B] > d[A] + *edges* [A][B], d[B] > 0 + 9, inf > 6 sehingga diperoleh d[B] = 9 dan tukar nilai inf (∞) pada *vertex* B dan hitung untuk *vertex* C, d[v] > d[u] + *edges* [u][v], d[C] > d[A] + *edges* [A][C], d[B] > 0 + 10, inf > 10, sehingga didapat d]C] = 10 dan tukar nilai inf (∞) pada *vertex* C. Dapat dilihat pada gambar 2.4.



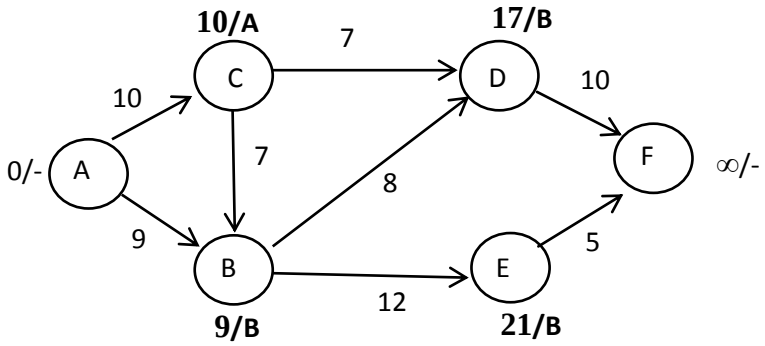
Gambar 2.4 Perhitungan *Graph* dari *Vertex* A

Kemudian pilih *edges* yang dapat dilalui dari *vertex* B, yaitu BE dengan bobot jarak 12. Hitung d[v] untuk *vertex* E, d[v] > d[u] + *edges* [u][v], d[E] > d[B] + *edges* [B][E], d[B] > 9 + 12, inf > 21 sehingga diperoleh d[E] = 21 dan tukar nilai inf (∞) pada *vertex* E, *edges* BD juga dapat dilalui dari *vertex* B dengan bobot jarak 8, hitung untuk *vertex* D, d[v] > d[u] + *edges* [u][v], d[D] > d[B] + *edges* [B][D], d[B] > 9 + 8, inf > 17, sehingga didapat d[D] = 17 dan tukar nilai inf (∞) pada *vertex* D. Dapat dilihat pada gambar 2.5.



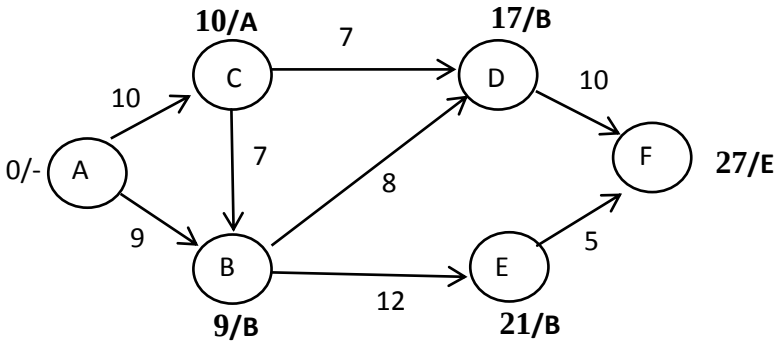
Gambar 2.5 Perhitungan *Graph* dari *Vertex* B

Pilih *edges* yang dapat dilalui dari *vertex* C yaitu CD dengan bobot jarak 7. Hitung d[v] ntuk *vertex* D, d[v] > d[u] + *edges* [u][v], d[D] > d[C] + *edges* [C][D], 17 > 10 + 17 "tidak" sehingga nilai d[D] sebelumnya yang diambil untuk *vertex* D, *edges* CB juga dapat dilalui dari *vertex* C dengan bobot jarak 7, hitung untuk *vertex* B, d[v] > d[u] + *edges* [u][v], d[B] > d[C] + *edges* [C][B], 9 > 10 + 7, 9 > 17 "tidak" sehingga d[B] = 17 sebelumnya yang diambil untuk *vertex* B. Dapat dilihat pada gambar 2.6.



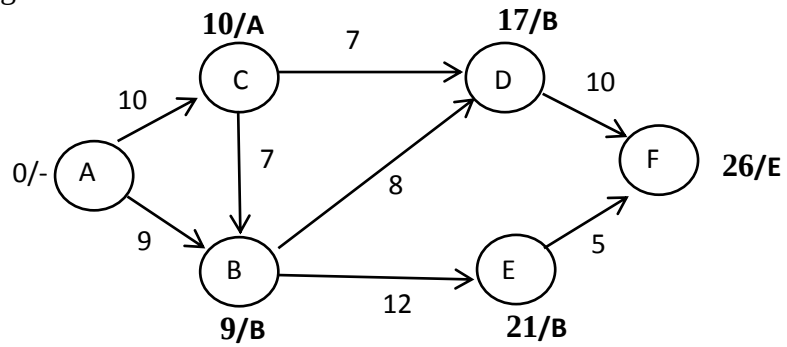
Gambar 2.6 Perhitungan *Graph* dari *Vertex* C

Pilih *edges* yang dapat dilalui dari *vertex* D yaitu DF dengan bobot jarak 10, . Hitung d[v] ntuk *vertex* F, d[v] > d[u] + *edges* [u][v], d[F] > d[D] + *edges* [D][F], d[F] > 17 + 10, inf > 27 sehingga didapat d[F] = 27 dan tukar nilai inf pada *vertex* F. Dapat dilihat pada gambar 2.7.



Gambar 2.7 Perhitungan *Graph* dari *Vertex* D

Pilih *edges* yang dapat dilalui dari *vertex* E yait EF dengan bobot jarak 5, . Hitung d[v] ntuk *vertex* F, d[v] > d[u] + *edges* [u][v], d[F] > d[E] + *edges* [E][F], 27 > 21 + 5, 27 > 26 “ya” sehingga nilai d[F] sebelumnya yang diambil untuk *vertex* F. Dapat dilihat pada gambar 2.8.



Gambar 2.8 Perhitungan *Graph* dari *Vertex* E

Hasil yang didapat dari contoh sederhana diatas untuk menentuka rute terpendek dengan algoritma L-Deque ialah :

AB = 9 (*direct)*

AC = 10 *(direct)*

AD = AB + BD = 17

AE = AB + BE = 21

AF = AB + BE + EF = 26

2.6 Teori Dasar Graph

Teori *graph* merupakan salah satu cabang ilmu matematika yang banyak digunakan dalam bidang ilmu seperti di bidang teknologi komputer, fisika, kimia, arsitektur, listrik dan di bidang lainnya. Teori *graph* ini juga banyak digunakan dalam kehidupan sehari-hari seperti pengaturan lampu lalu lintas, *travelling salesman problem*, penjadwalan matakuliah di kampus dan lain-lain.

2.6.1. Defenisi Graph

Graph didefinisikan sebagai pasangan himpunan yang terdiri dari *vertex* (V) dan *edges* (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. Simpul pada graf dapat dinomori dengan huruf, seperti a, b, c...dst, dengan bilangan asli 1, 2, 3...dst, atau gabungan keduanya. Sedangkan sisi yang menghubungkan simpul dengan simpul v dinyatakan dengan pasangan (u, v) atau dinyatakan dengan lambang e1, e2....en dengan kata lain, jika e adalah sisi yang menghubungkan simpul u dengan simpul v, maka e dapat ditulis sebagai e = (u, v). Secara geometri graf digambarkan sebagai sekumpulan noktah (simpul) di dalam bidang dwimatra yang dihubungkan dengan sekumpulan garis (sisi). (Ziad, 2013).

2.6.2. Jenis - Jenis Graph

Dilihat dari ada tidaknya gelang atau sisi ganda pada suatu graf, maka secara umum graf dapat dibagi menjadi dua jenis, yaitu:

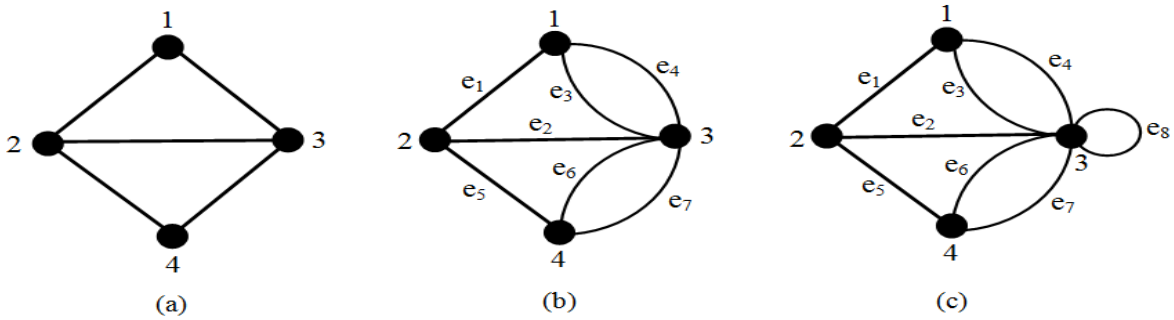
1. Graf Sederhana (*simple graph*) adalah graf yang tidak memiliki garis paralel ataupun *loop*. Titik-titik pada graf sederhana dihubungkan tepat dengan satu

garis ke setiap titik yang lain dan tidak ada garis yang titik awal dan akhirnya sama (Pinem, 2014).

2. Graf Tidak Sederhana (*unsimple graph*) adalah graf yang memiliki *loop* atau garis paralel. Graf tidak sederhana kemudian terbagi lagi menjadi graf semu (pseudo*graph*) dan graf ganda (*multiple graph*) (Pinem, 2014). Graf semu adalah graf yang mengandung gelang (*loop*). Graf semu lebih umum daripada graf ganda, karena sisi pada graf semu dapat terhubung ke dirinya sendiri sedangkan Graf ganda adalah graf yang mengandung sisi ganda.

   Sisi ganda yang menghubungkan sepasang simpul bisa lebih dari dua buah (Munir, 2014).

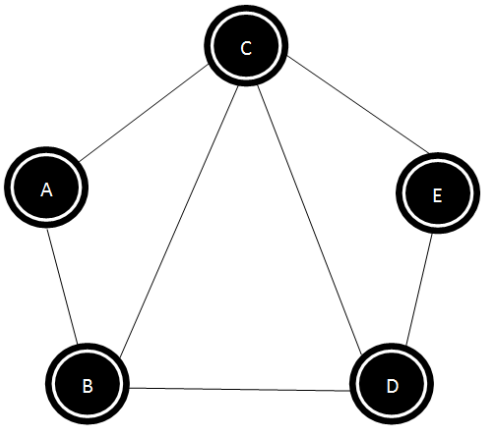
Loop merupakan suatu sisi dapat menghubungkan suatu simpul dengan simpul yang sama. Pada gambar 2.9 adalah contoh graf sederhana, graf ganda dan graf semu.



Gambar 2. 9 (a) Graf Sederhana, (b) Graf Ganda, dan (c) Graf Semu (Munir, 2014)

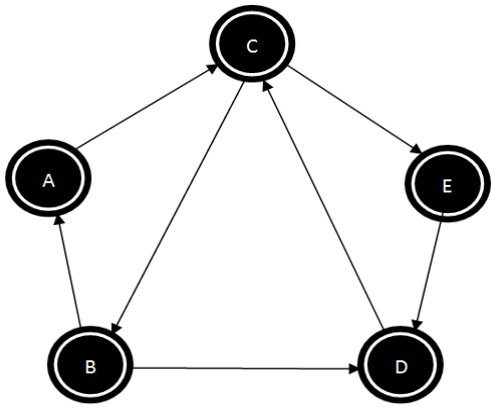
Sisi pada graf dapat mempunyai orientasi arah. Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas dua jenis:

1. Graf tidak berarah (*undirected graph*) adalah graf yang sisinya tidak mempunyai orientasi arah. Pada graf ini, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan (Munir, 2014). Pada gambar 2.10 adalah contoh graf tidak berarah.



Gambar 2.10 Graf Tidak Berarah

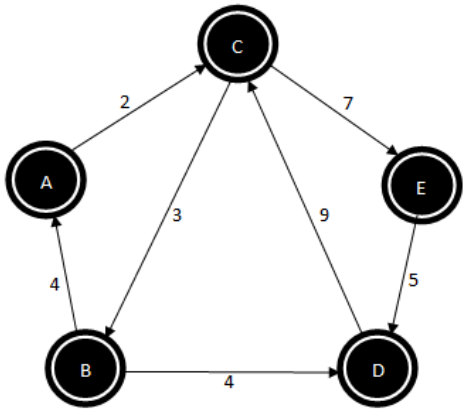
2. Graf berarah (*directed graph* atau *digraph*) adalah graf yang setiap sisinya diberikan orientasi arah. Pada graf berarah, (*u*,*v*) dan (*v*,*u*) menyatakan dua buah busur yang berbeda, dengan kata lain (*u*,*v*) ≠ (*v*,*u*). Untuk busur (*u*,*v*), simpul *u* dinamakan simpul asal (*initial vertex*) dan simpul *v* dinamakan simpul terminal (*terminal vertex*) (Munir, 2014). Pada gambar 2.11 adalah contoh graf berarah.



Gambar 2.11 Graf Berarah

2.6.3 Graf Berbobot (Weighted Graph)

Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot). Bobot pada tiap sisi dapat berbeda-beda bergantung pada masalah yang dimodelkan dengan graf. Bobot dapat menyatakan jarak antara dua buah kota, waktu tempuh pesan (*message*) dari sebuah simpul komunikasi ke simpul komunikasi lain (dalam jaringan komputer), ongkos produksi, dan sebagainya (Munir, 2014). Pada gambar 2.12 adalah contoh graf berbobot.



Gambar 2.12 Graf Berbobot

2.6 Shortest Path (Jalur Terpendek)

Shortest Path (Jalur Terpendek) adalah pencarian suatu masalah untuk menemukan lintasan terpendek antara dua atau lebih simpul yang saling berhubungan. Lintasan terpendek adalah jalur yang dilalui dari suatu *vertex* ke *vertex* lain dengan besar atau nilai pada *edges* yang jumlah akhirnya dari *vertex* awal ke *vertex* akhir paling kecil. Lintasan terpendek adalah lintasan minimum yang diperlukan untuk mencapai suatu tempat dari tempat lain. Lintasan minimum yang dimaksud dapat dicari dengan menggunakan graf. Graf yang digunakan adalah graf yang berbobot yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot (Hayati & Yohanes, 2014). Contoh *Shortest Path* dapat dilihat pada Gambar 2.12, dimana *vertex* A sebagai *vertex* awal dan *vertex* tujuan. Maka didapat lintasan terpendek dari A ke E => A – B – C – E => 4 + 3 + 7 = 14.

2.7 Google Maps

Google Maps merupakan layanan gratis Google yang cukup popular. Kita dapat menambahkan fitur *Google Maps* dalam web kita sendiri dengan *Google Maps API*. *Google Maps API* merupakan *library JavaScript*. Untuk melakukan pemrograman *Google Maps API* dapat dibilang mudah. Yang kita butuhkan adalah pengetahuan tentang HTML dan JavaScript, serta koneksi Internet. Dengan menggunakan *Google Maps API,* kita dapat menghemat waktu dan biaya untuk membangun aplikasi peta digital yang handal, sehingga kita dapat fokus hanya pada data-data yang diperlukan. Data peta-peta dunia menjadi urusan Google. (Yuhana,2010).

Google Maps sebuah jasa peta global virtual gratis dan online yang disediakan oleh Google dapat ditemuakan di maps.google.com yang menawar peta dapat diseret dan gambar satelit untuk seluruh dunia. *Google Maps API* merupakan aplikasi *interface* yang dapat diakses lewat *javascript* dapat ditampilkn pada halaman web yang sedang dibangun. *Google Maps* mempunyai banyak kegunaan untuk menampilkan lokasi, lokasi kegiatan even atau dapat juga digunakan untuk aplikasi GIS (Gufroni, 2013).

2.8 Penelitian Terdahulu

Penelitian tentang pencarian jalur terpendek sudah banyak dilakukan dengan beberapa metode. Pemberian informasi terhahap layanan di bidang sistem informasi geografis dalam penentuan rute juga sudah pernah dilakukan dengan berbagai metode.

Pada tahun 2016, Vandia, T.A melakukan penelitian dengan judul penelitian perencanaan perjalanan wisata dengan metode *Greedy*. Penelitian ini menunjukkan metode *Greedy* dapat mempermudah menyusun sebuah rencana perjalanan secara otomatis, dengan cara mencari solusi *optimum local* atau solusi pilihan rute dan menjadikan solusi *optimum local* menjadi solusi *optimum global* atau solusi perjalanan yang menyeluruh dalam rute perjalanan wisata.

Selanjutnya pada tahun 2009, Lubis, H.N melakukan penelitian dengan judul penelitian perbandingan algoritma *Greedy* dan Dijkstra untuk menentukan lintasan terpendek. Penelitian menunjukkan perbandingan algoritma *Greedy* dan Dijkstra berdasarkan jarak lintasannya, algoritma *Greedy* menghasilkan jarak yang lebih besar dan juga algoritma *Greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif fungsi yang ada, sehingga lintasan terpendek hanya diperoleh dari verteks asal hingga verteks tujuan, sedangkan algoritma Dijkstra beroperasi secara menyeluruh terhadap semua alternative yang ada, sehingga lintasan terpendek tidak hanya diperoleh dari vertex sumber ke vertex tujuan saja, akan tetapi lintasan terpendek dapat diperoleh dari semua vertex.

Selanjutnya pada tahun 2016, Masyunita melakukan penelitian dengan judul penelitian aplikasi pelayanan sistem informasi geografis di Universitas Sumatera Utara (USU) berbasis android menggunakan algoritma *Bellman-Ford.* Penelitian menunjukkan sistem dapat memudahkan user untuk mendapat informasi dan pemetaan alur menuju titik lokasi dan hasil penelitian yang diperoleh dari aplikasi

yang dibangun menunjukkan bahwa algoritma Bellman-ford mampu memberikan solusi dalam pencarian jarak terpendek di USU dengan tingkat keberhasilan 84%.

Selanjutnya pada tahun 2013, Zainuddin melakukan penelitian dengan judul penelitian perancangan SIG berbasis web objek wisata kota binjai dengan algoritma A*. Penelitian menunjukkan Sistem dapat menunjukkan rute terpendek dari suatu titik dengan ke titik lainnya dengan mengimplementasikan algoritma A* pada WEB GIS. Dengan menggunakan sistem ini pengguna dapat lebih efektif dan efisien dalam menentukan rute terpendek objek wisata yang diinginkan.

Selanjutnya pada tahun 2016, Natasha M Siregar dalam penelitiannya yang berjudul Analisis dan Perbandingan Algoritma L-Deque dan Algoritma Bellman-Ford dalam Mencari Jarak Terpendek menunjukkan algoritma Bellman-Ford menghasilkan bobot paling minimum.

Tabel 2.1 Penelitian Terdahulu

No	Judul	Peneliti	Metode	Keterangan
1	Perencanaan Perjalanan Wisata dengan Metode Greedy	Vandia (2016)	Greedy	Metode Greedy dapat mempermudah menyusun sebuah rencana perjalanan secara otomatis dalam rute perjalanan wisata.
2	Perbandingan Algoritma *Greedy* dan Dijkstra untuk Menentukan Lintasan Terpendek	H.N Lubis (2009)	*Greedy* dan Dijkstra	Algoritma *Greedy* menghasilkan jarak yang lebih besar dan algoritma Dijkstra beroperasi secara menyeluruh terhadap semua alternative yang ada.

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

3	Aplikasi Pelayanan Sistem Informasi Geografis di Universitas Sumatera Utara (USU) berbasis Android menggunakan algoritma Bellman-Ford	Masyunita (2016)	Bellman-Ford	Bellman-Ford dapat menemukan jalur terpendek untuk sampai ke tujuan tempat lokasitingkat keberhasilan 84%.
4	Perancangan SIG Berbasis Web Objek Wisata Kota Binjai dengan Algoritma A*.	Zainuddin (2013)	A*	Sistem lebih efektif dan efisien dalam menentukan rute terpendek objek wisata yang diinginkan menggunakan algoritma A*.
5	Analisis dan Perbandingan Algoritma L-Deque dan Algoritma Bellman-Ford dalam Mencari Jarak Terpendek	Natasha M Siregar (2016)	L-Deque dan Bellman-Ford	Dalam mencari jarak terpendek menunjukkan algoritma Bellman-Ford menghasilkan bobot paling minimum.

BAB 3

ANALISIS DAN PERANCANGAN SISTEM

Analisis sistem merupakan teknik pemecahan masalah yang bertujuan untuk menguraikan sebuah sistem menjadi subsistem untuk mengetahui kinerja, hubungan dan interaksi tiap komponen yang berada di sistem guna mencapai suatu tujuan (Whitten, 2007).

Pada bab ini, akan dibahas mengenai analisis perancangan sistem yang akan dibangun dan penerapan algoritma *L-Deque* dalam penentuan rute destinasi objek wisata di kawasan Danau Toba.

3.1 Analisis Sistem

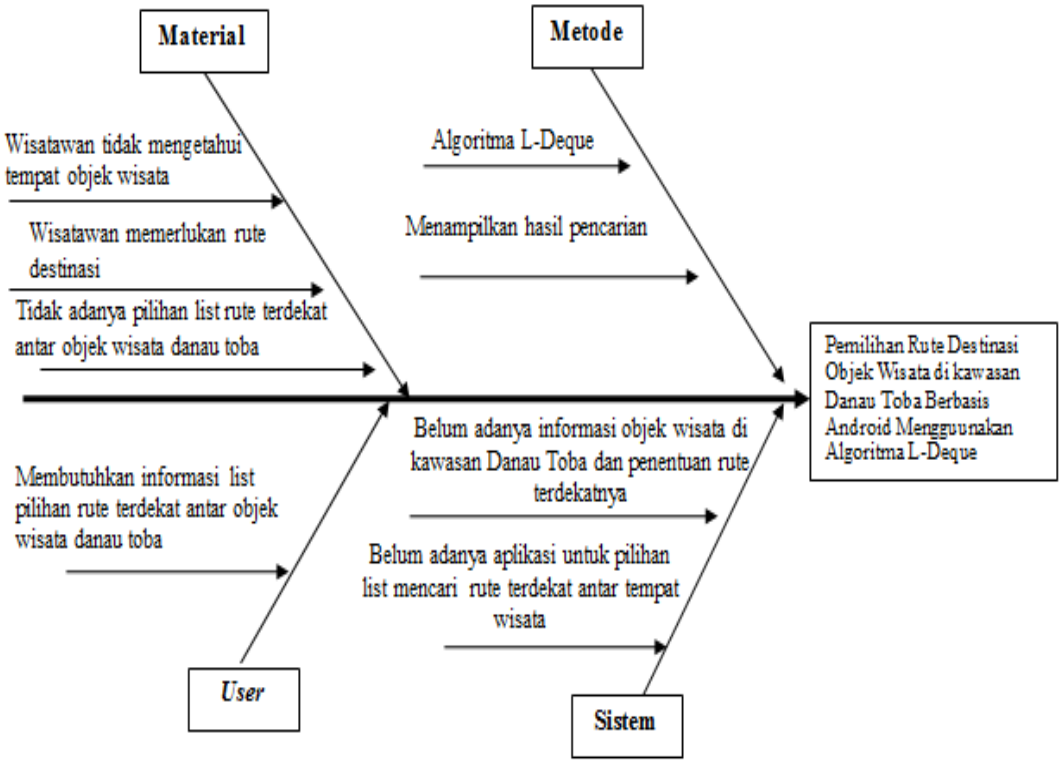
Pada analisis sistem akan dilakukan analisis terhadap sistem untuk melakukan penentuanrute terdekat yang efisien untuk destinasi objek wisata di kawasan Danau Toba dengan menggunakan algoritma *L-Deque.*

3.1.1 Analisis Masalah

Permasalahan yang akan diselesaikan dengan sistem ini adalah bagaimana cara menentukan rute terdekat antar objek dikawasan Danau Toba. Namun, adapun masalah utama didalam penelitian ini ialah bagaimana menerapkan algoritma *L-Deque* untuk menentukan rute dengan rute yang efektif dan efisien dengan memberikan pilihan *list* rute destinasi yang akan dilalui sesuai titik awal lokasi yang diinginkan.

Pada penelitian ini, penulis membatasi masalah hanya dengan 17 buah *vertex*. Dalam representasi *graph*, penentuan hubungan antar *vertex* dan *edge* pada *graph* objek wisata di kawasan Danau Toba sangat diperlukan, agar dapat dilakukan proses yang lebih optimal. *Graph* yang digunakan merupakan *graph* berarah terhubung, dimana suatu *graph* berarah terhubung disebut jika ada *walk* yang menghubungkan setiap dua *vertex.* Setelah itu sistem akan melakukan percarian dengan menggunakan *graph* terhadap algoritma *L-Deque*. Analisis masalah dapat diidentifikasi dengan

menggunakan diagram *Ishikawa* (*fishbone diagram*). Maka dari itu, untuk mengetahui penyebab permasalahan dan penyelesaian, digambarkan dalam bentuk diagram *Ishikawa* (*fishbone diagram*) agar diketahui penyebab permasalahan yang saling berkaitan, sehingga akan lebih mudah dalam memahami masalah yang akan di analisa terhadap sistem. Diagram tersebut dapat kita lihat pada Gambar 3.1.



Gambar 3.1 Diagram *Ishikawa* (*fishbone diagram*)

Diagram *Ishikawa* (*fishbone diagram*) diatas dibagi dalam 4 aspek yaitu material, metode, *user* dan sistem. Pada bagian material terdapat tiga buah masalah dimana masalahnya tersebut adalah wisatawan tidak mengetahui tempat objek wisata sehingga wisatawan memerlukan rute destinasi dan belum adanya sistem menampilkan pilihan *list* rute terdekat antar objek wisata di kawasan Danau Toba. Metode adalah kebutuhan yang spesifik dalam proses yaitu dibutuhkannya hasil pencarian dalam penerapan algoritma *L-Deque*. *User* adalah apa saja yang akan diperoleh pengguna aplikasi dimana *user* membutuhkan informasi *list* pilihan rute terdekat antar objek wisata di kawasan Danau Toba. Sistem adalah hal yang akan dibuat dengan membuat aplikasi penentuan dan pemilihan list rute terdekat antar objek wisata dengan menggunakan algoritma *L-Deque*.

Hasil kerja dari algoritma *L-Deque* diaplikasikan dalam bentuk *graph* dengan mengikuti peta di kawasan Danau Toba, lalu *vertex* ditentukan berdasarkan nama-nama objek wisata yang telah dipilih. Dan juga *edge* yang merupakan panjang jarak yang akan dilalui dari satu objek ke objek lainnya. Terdapat 17 buah objek wisata yang telah dipilih yang nantinya akan diterapkan ke dalam *graph* pada aplikasi yang akan dibuat. Nama-nama objek wisata yang digunakan dapat dilihat pada Tabel 3.1.

Tabel 3.1 Daftar Objek Wisata yang menjadi verteks

No	Nama objek wisata
1	Pulau samosir
2	Parapat
3	Pantai bulbul
4	Tongging
5	Pusuk buhit
6	Sipinsur
7	Ambarita
8	Simanindo
9	Pangguruan
10	Tuktuk
11	Aek sipitu dai
12	Danau sidihoni
13	Sialanguan
14	Rumah pengasingan mantan Presiden RI Soekarno
15	Balige cultural center
16	Haranggaol
17	Paropo

Pada penelitian ini, penulis membatasi masalah hanya dengan 17 buah *vertex*, diharapkan akan ada penelitian lebih lanjut untuk menyelesaikan masalah yang serupa dengan memperbanyak jumlah simpul untuk objek wisata yang ada di kawasan Danau Toba.

3.1.2 Analisis Persyaratan

Terdapat dua jenis analisis persyaratan, yaitu analisis persyaratan fungsional dan analisis persyaratan non fungsional. Dimana, persyaratan fungsional adalah aktifitas yang harus dipenuhi dari suatu sistem dan persyaratan non fungsional adalah hal yang digunakan sebagai pelengkap dari suatu sistem.

3.1.2.1 Persyaratan fungsional

Persyaratan fungsional adalah aktifitas yang harus dipenuhi dari suatu sistem, yang jenis kebutuhannya berisi proses-proses apa saja yang akan dilakukan oleh sistem. Pada penelitian ini persyaratan fungsional sistem adalah:

1. Sistem mampu menentukan dan menampilkan hasil rute mana saja yang akan dilalui dengan pilihan *list* rute terdekat antar objek wisata menggunakan algoritma *L-Deque*.
2. Sistem harus mampu menampilkan jarak dan perkiraan waktu yang ditempuh.
3. Sistem ini menggunakan *graph* objek wisata di kawasan Danau Toba.
4. Sistem ini dapat menggunakan *vertex* awal yang diinginkan sebagai *vertex* awal.

3.1.2.2 Persyaratan non fungsional

persyaratan non fungsional adalah hal yang digunakan sebagai pelengkap dari suatu sistem. Pada penelitian ini persyaratan fungsional sistem adalah:

1. Ekonomi

   Sistem yang akan dibangun tidak perlu mengeluarkan biaya tambahan dalam penggunaannya.

2. Kinerja

   Sistem yang akan dibangun dapat menampilkan rute yang akan dilalui.

3. *User friendly*

   Sistem yang akan dibangun mudah untuk digunakan dan dipahami.

4. Kualitas

   Sistem yang akan dibangun menghasilkan *output* yang akurat dan efisien.

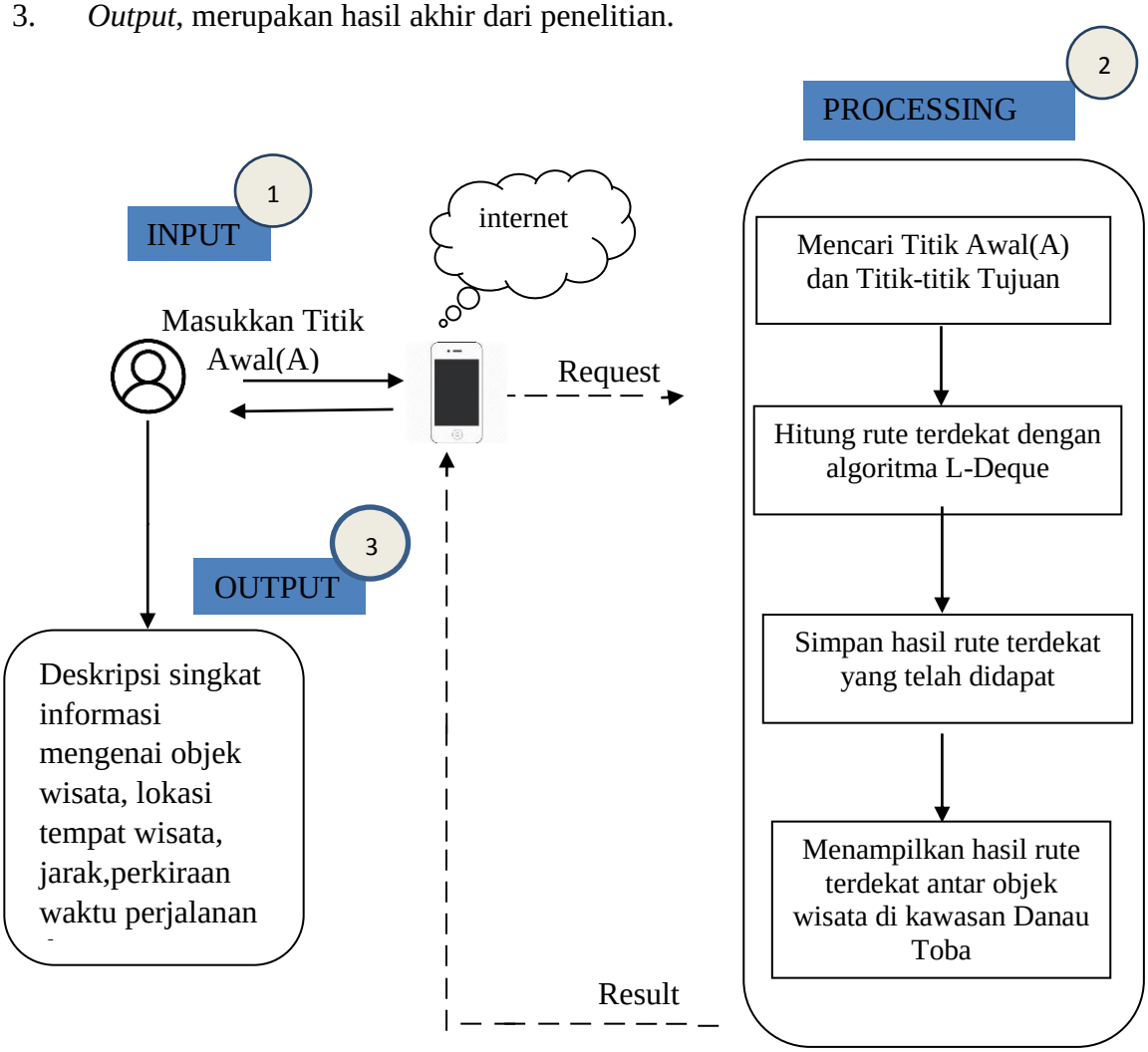
3.1.3 Analisis Proses

Pada penelitian ini sistem yang akan dibangun menggunakan algoritma *L-Deque* untuk menentukan rute terdekat. Hal yang pertama kali dilakukan adalah menentukan titik awal Kemudian, algoritma *L-Deque* akan bekerja dalam menghitung dan menentukan rute terdekat menuju titik destinasi tersebut. Lalu sistem akan menampilkan rute terdekat yang dihasilkan oleh algoritma *L-Deque*.

3.1.4 Arsitektur umum

Adapun arsitektur umum pada penelitian ini terdiri dari beberapa tahap, yaitu:

1. *Input*, merupakan tahap bagaimana mendapatkan titik awal.
2. *Processing*, merupakan tahap untuk mencari titik awal dan titik - titik tujuan sehingga mempermudah mempresentasikan *graph* berbobot dan selanjutnya menghitung rute terdekat menggunakan algoritma *L-Deque*. Setelah dilakukan penentuan rute terdekat maka langkah selanjutnya yaitu mengevaluasi rute yang dilalui sehingga dapat menampilkan hasil rute terdekat antar objek wisata di kawasan Danau Toba.
3. *Output*, merupakan hasil akhir dari penelitian.



Gambar 3.2 Arsitektur Umum Perancangan Sistem

Adapun Arsitektur umum dari perancangan yang akan dibangun dapat dilihat pada Gambar 3.2. Penjelasan dari gambar Arsitektur Umum adalah untuk melakukan pencarian titik-titik lokasi yang dituju oleh *user* atau pengguna aplikasi. Adapun rancangan tampilan yang dihasilkan di antaranya deskripsi singkat dari setiap lokasi

objek wisata yang akan dituju, jarak, perkiraan waktu danrute yang dipilih oleh *user* berupa tampilan peta sebagai petunjuk arah untuk sampai ke lokasi tujuan objek wisata.

1. Input

Input pertama yang dimasukkan oleh user melalui *mobile phone* setelah membuka aplikasi yang telah terhubung dengan internet yaitu menginput titik awal objek wisata di kawasan danau toba yang sudah di simpan dalam server.

2. Proses

Pada bagian proses ini, setelah user menentukan titik awal, maka sistem akan mengirim perintah pada server. Server akan menyimpan koordinat *user* dan mencari koordinat titik – titik lokasi tujuan objek wisata. Setelah didapatkan arah lintasan setiap step menuju lokasi maka algoritma *L-Deque* akan bekerja untuk menghitung rute terdekat dan mengevalusi rute yang dilalui dari objek wisata yang satu dengan objek wisata lainnya. Setelah itu server akan menyimpan hasil rute terdekat yang telah didapat antar objek wisata. Kemudian menampilkan pilihan *list* hasil rute terdekat antar objek wisata sehingga *user* bisa memilih rute yang akan dilalui sesuai keinginan.

3. Output

Output yang dihasilkan merupakan tampilan deskripsi singkat informasi mengenai objek wisata, lokasi tempat wisata, jarak, perkiraan waktu dan rute destinasi yang dipilih oleh *user.*

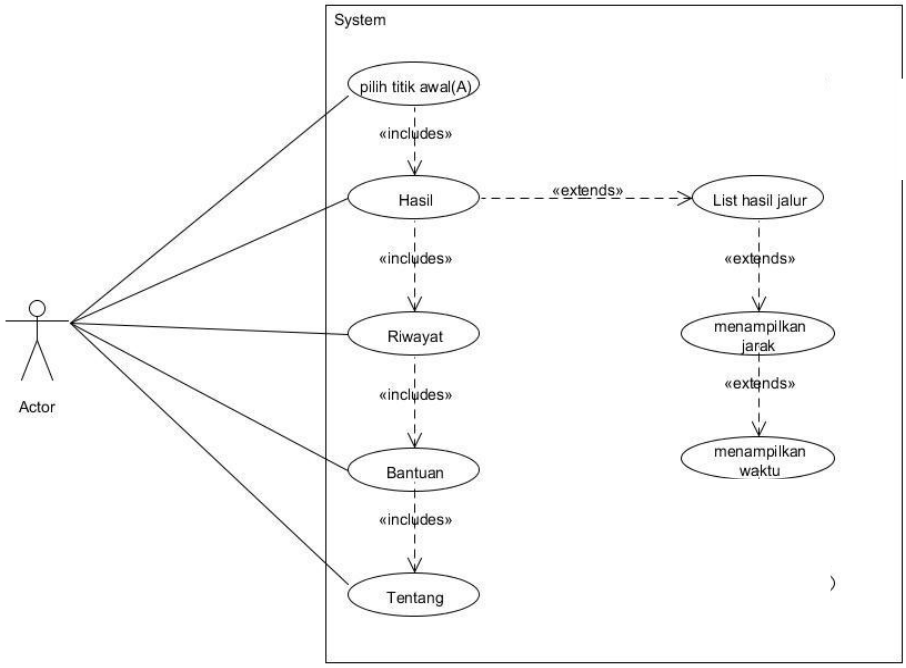
3.2 Pemodelan

Pada perancangan sistem ini menggunakan diagram UML *(Unified Modelling Languange)* untuk menggambarkan bagaimana sistem akan bekerja khususnya sistem yang berorientasi objek. Diagram UML yang digunakan adalah *use case* diagram, *activity* diagram dan *sequence* diagram.

3.2.1 *Use case* diagram

Use case adalah gambaran fungsionalitas dari suatu sistem, sehingga *user* paham dan mengerti mengenai kegunaan sistem yang dibangun, dimana penggambaran sistem dari sudut pandang *user* itu sendiri sehingga *use case* lebih dititikberatkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian. *Use case* diagram menggambarkan siapa saja yang berinteraksi dengan sistem dan apa saja

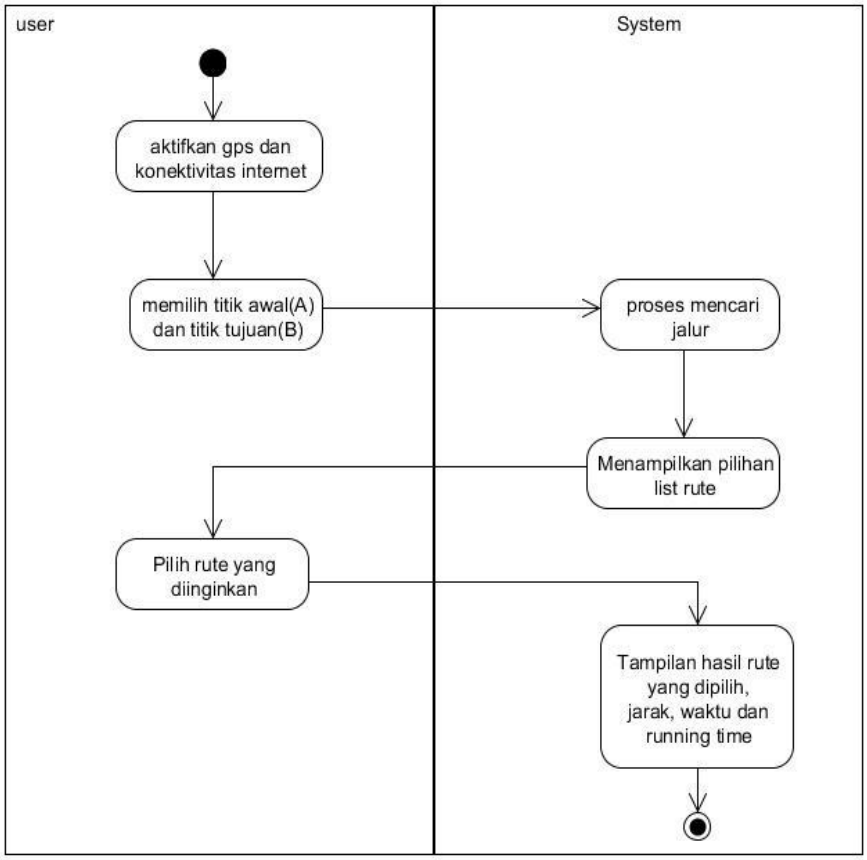
yang akan dilakukan dengan sistem bisa digambarkan dengan jelas berdasarkan analisis kebutuhan sistem. Adapun *usecase* dari sistem yang akan dibangun pada penelitian ini dapat dilihat pada Gambar 3.3.



Gambar 3.3*Use Case* Diagram

3.2.2 *Activity* diagram

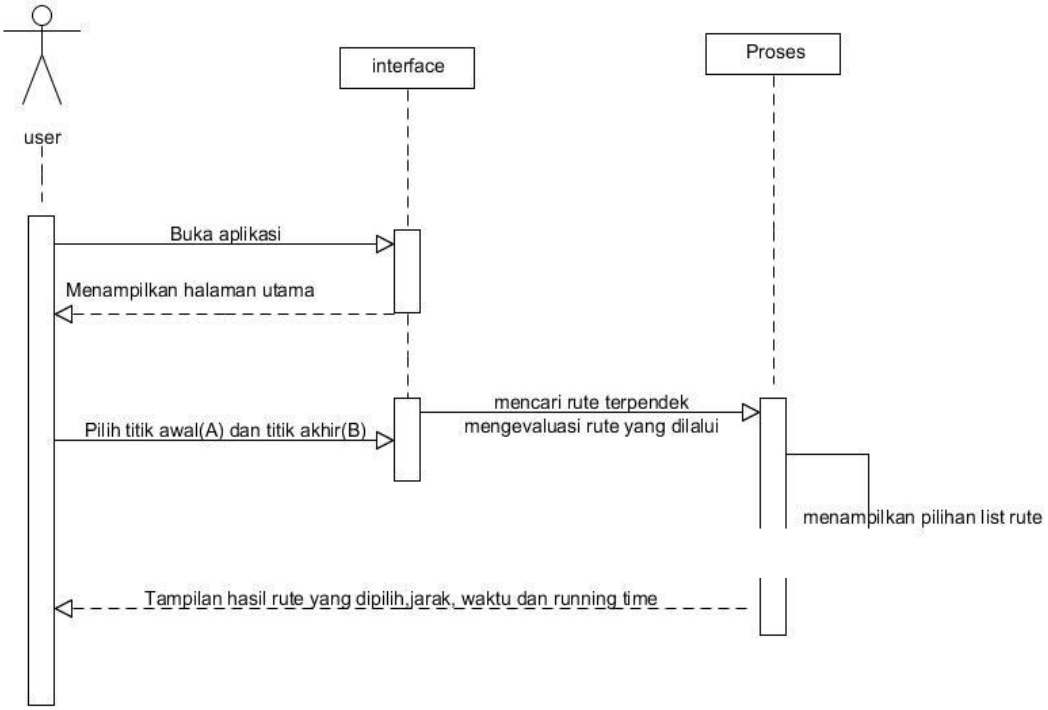
Activity diagram dapat menggambarkan urutan aktivitas antara pengguna sistem yang dibuat secara berurut dalam sebuah proses, serta dapat menggambarkan alur aktivitas kerja pada sistem yang sedang dirancang ataupun berjalan, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana berakhir. *Activity* diagram sistem pada aplikasi yang dibuat dapat dilihat pada Gambar 3.4.



Gambar 3.4 *Activity* Diagram

3.2.3 *Sequence* Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Aksi pengguna terhadap sistem ditunjukkan dengan tanda panah garis, sedangkan respon terhadap pengguna ditunjukkan dengan tanda panah garis putus-putus. Berikut *sequence* diagram pada Gambar 3.5.



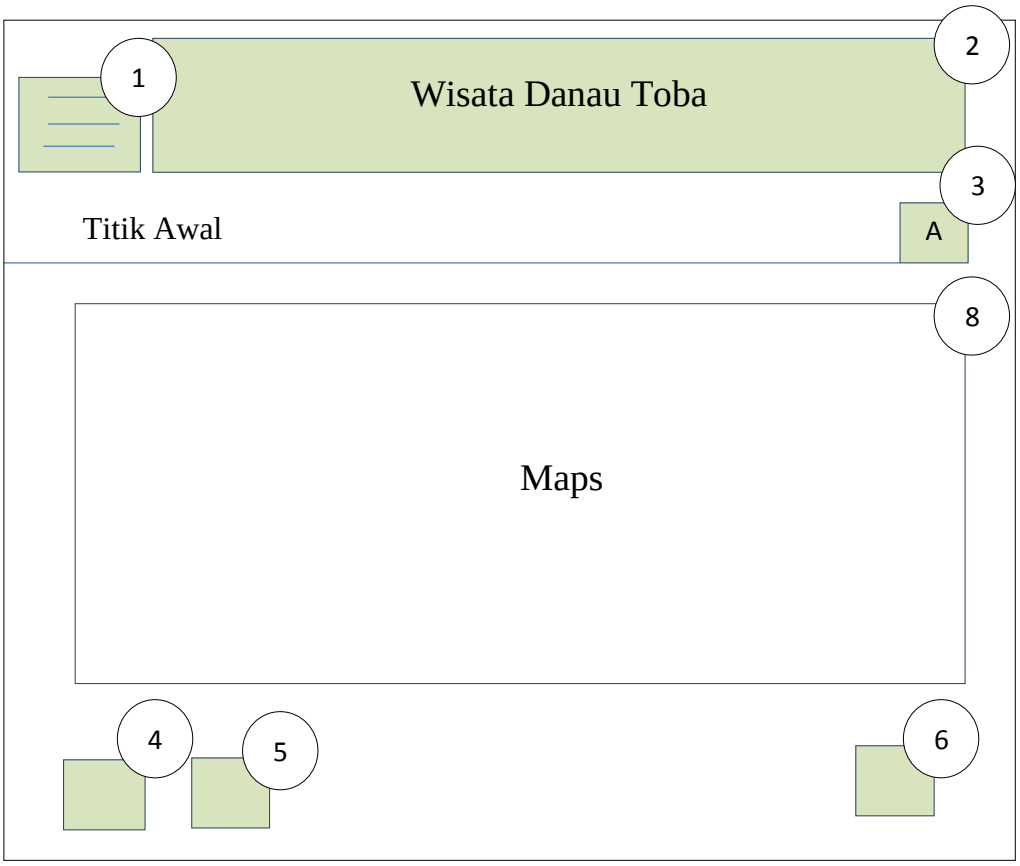
Gambar 3.5 *Sequence* diagram

3.3 Perancangan Sistem

Perancangan *interface*merupakan salah satu bagian penting dalam proses membangun sebuah sistem. Dimana, perancangan *interface* yang baik perlu memperhatikan interaksi antara sistem dengan *user*, selain untuk mempermudah pengguna dalam menggunakan sistem yang dibangun juga perlu diperhatikan kenyamanan dari pengguna dalam menggunakan sistem. Adapun *interface* yang dibuat dalam sistem ini adalah Halaman utama, Halaman Menu, Halaman lokasi wisata, Halaman riwayat, Halaman bantuan dan Halaman tentang.

3.3.1 Halaman Utama

Rancangan *interface* halaman utama dapat dilihat pada Gambar 3.6. Halaman utama adalah tampilan awal dari sistem ketika sistem dijalankan.



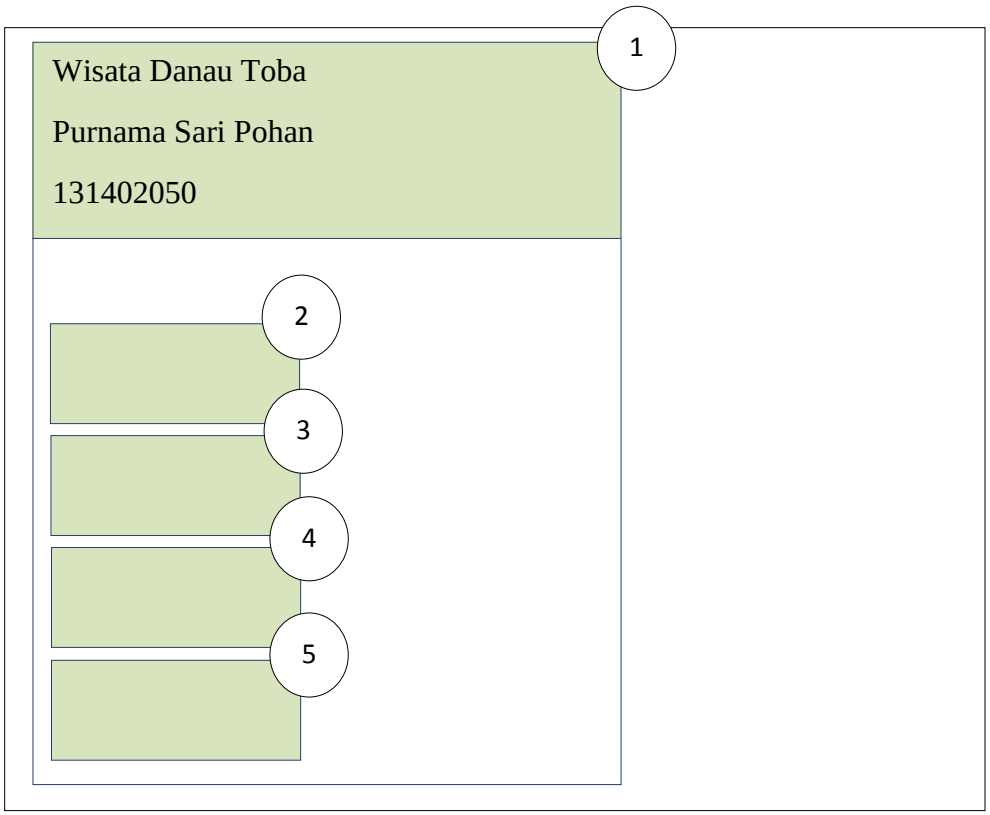
Gambar 3.6 Rancangan *Interface* Halaman utama

Keterangan Gambar 3.6 :

1. Tampilan yang berisi menu halaman-halaman aplikasi yang tersedia berupa halaman lokasi wisata, halaman riwayat, halaman bantuan dan halaman tentang.
2. Tampilan yang berisi nama aplikasi.
3. Tampilan yang berisi tentang input data untuk pemilihan *vertex* titik awal(A).
4. Tampilan yang berisi informasi objek wisata terdekat berupa jarak.
5. Tampilan yang berisi informasi objek wisata terdekat berupa waktu.
6. Tampilan yang berisi untuk memulai proses pencarian rute.
7. Tampilan yang berisi berupa rute *(maps*) yang dipilih oleh pengguna.

3.3.2 Halaman Menu

Rancangan *interface* halaman menu dapat dilihat pada Gambar 3.7. Halaman menu adalah tampilan yang berisi menu – menu halaman yang tersedia pada sistem.



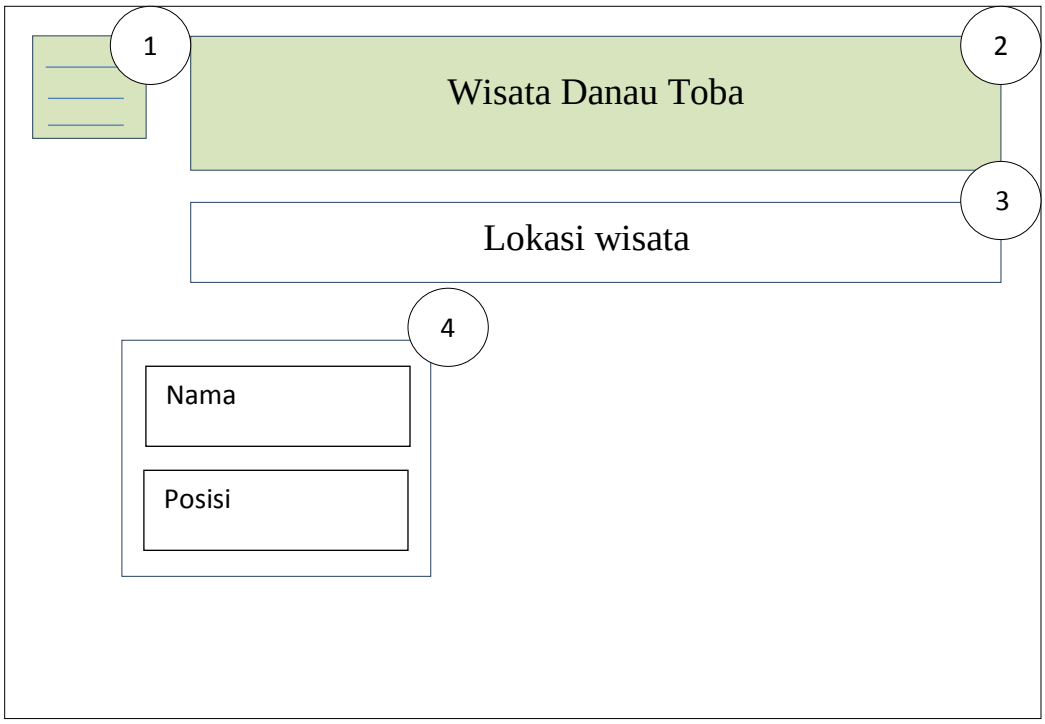
Gambar 3.7 Rancangan *Interface* Halaman Menu

Keterangan Gambar 3.7 :

1. Tampilan yang berisi nama aplikasi.
2. Tampilan halaman menu lokasi wisata
3. Tampilan halaman menu riwayat.
4. Tampilan halaman menu bantuan.
5. Tampilan halaman menu tentang.

3.3.3 Halaman Lokasi Wisata

Rancangan *interface* halaman lokasi wisata dapat dilihat pada Gambar 3.8. Halaman lokasi wisata adalah tampilan yang berisi tentang titik koordinat dari objek wisata.



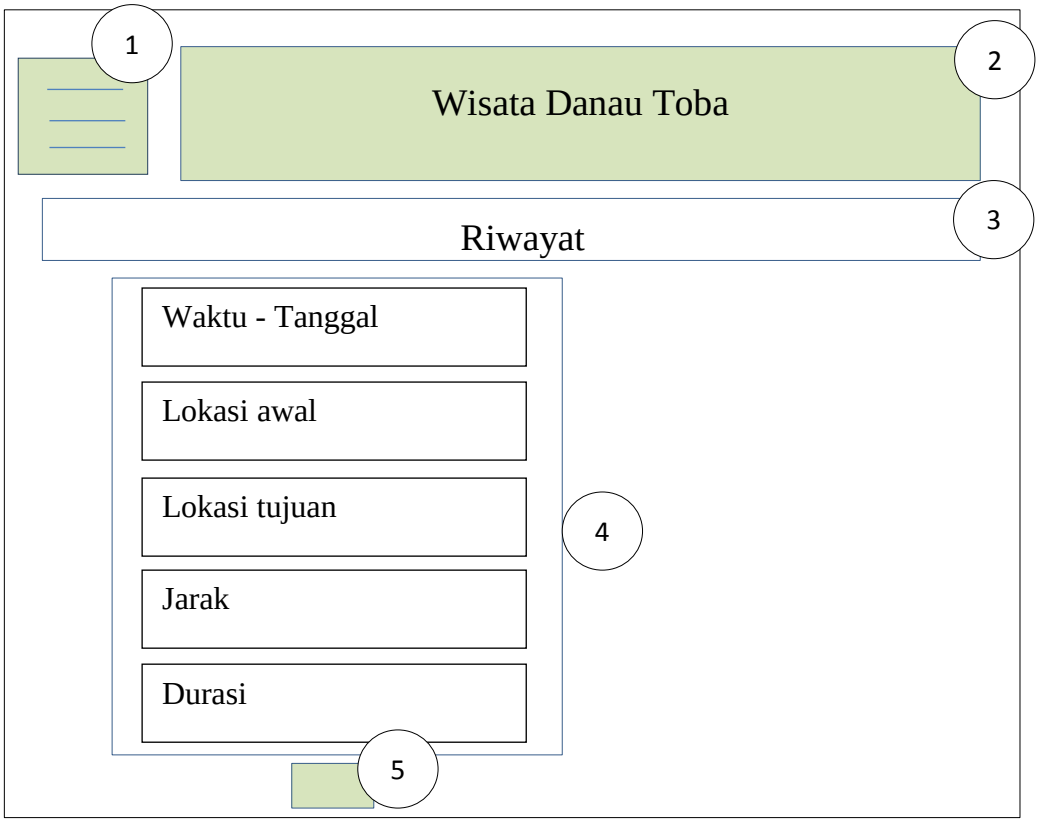
Gambar 3.8 Rancangan *Interface* Halaman lokasi wisata

Keterangan Gambar 3.8 :

1. Tampilan yang berisi halaman-halaman aplikasi yang tersedia berupa : halaman lokasi wisata, halaman riwayat, halaman bantuan dan halaman tentang.
2. Tampilan yang berisi nama aplikasi.
3. Tampilan untuk menunjukkan halaman yang tersedia pada sistem.
4. Tampilan untuk menampilkan berupa informasi nama tempat dan posisi objek wisata yang ada di kawasan Danau Toba.

3.3.4 Halaman Riwayat

Rancangan *interface* halaman riwayat dapat dilihat pada Gambar 3.9. Halaman riwayat merupakan tampilan yang berisi tentang riwayat-riwayat dalam pencarian rute terdekat antar objek wisata dikawasan Danau Toba yang telah diakses oleh *user* atau pengguna aplikasi.



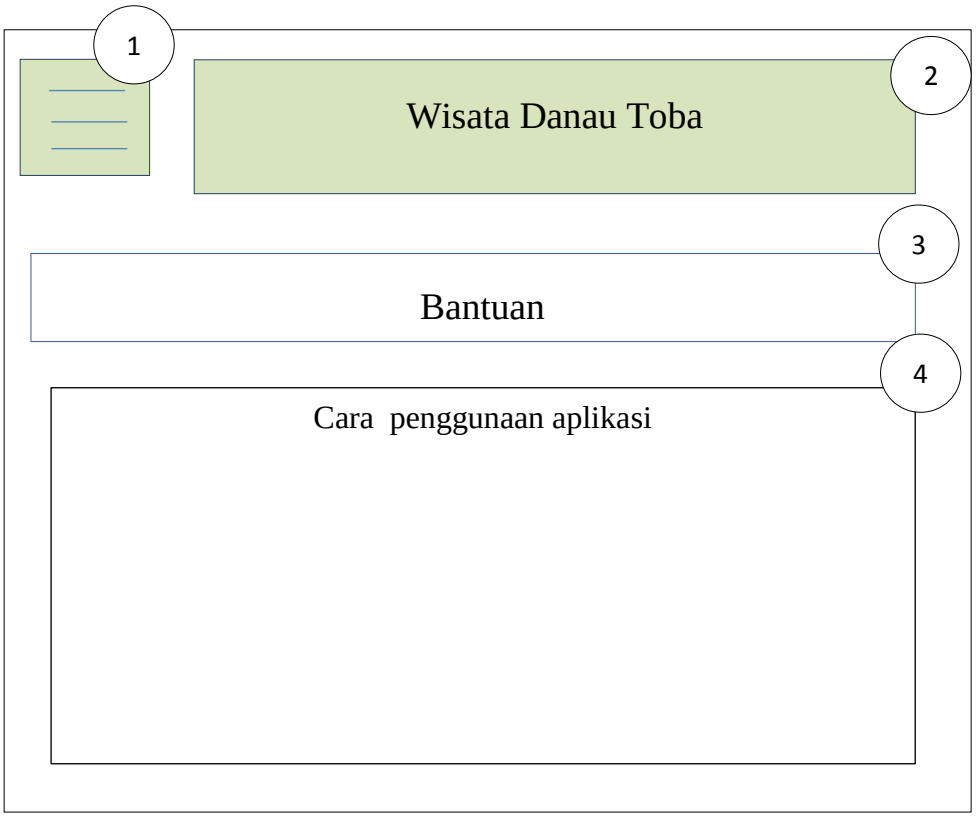
Gambar 3.9 Rancangan *Interface* Halaman riwayat

Keterangan gambar 3.9 :

1. Tampilan yang berisi halaman-halaman aplikasi yang tersedia berupa : halaman lokasi wisata, halaman riwayat, halaman bantuan dan halaman tentang.
2. Tampilan yang berisi nama aplikasi.
3. Tampilan untuk menunjukkan halaman yang tersedia pada sistem.
4. Tampilan untuk menampilkan informasi riwayat hasil yang telah diproses berupa waktu-tanggal, lokasi awal, lokasi tujuan, jarak dan durasi rute perjalanan.
5. Tampilan untuk *icon* hapus riwayat

3.3.5 *Halaman Bantuan*

Rancangan *interface* halaman bantuan dapat dilihat pada Gambar 3.10. Halaman bantuan merupakan halaman yang akan tampil saat pengguna memilih halaman pada *menu*. Yang mana berfungsi sebagai petunjuk cara penggunaan sistem.



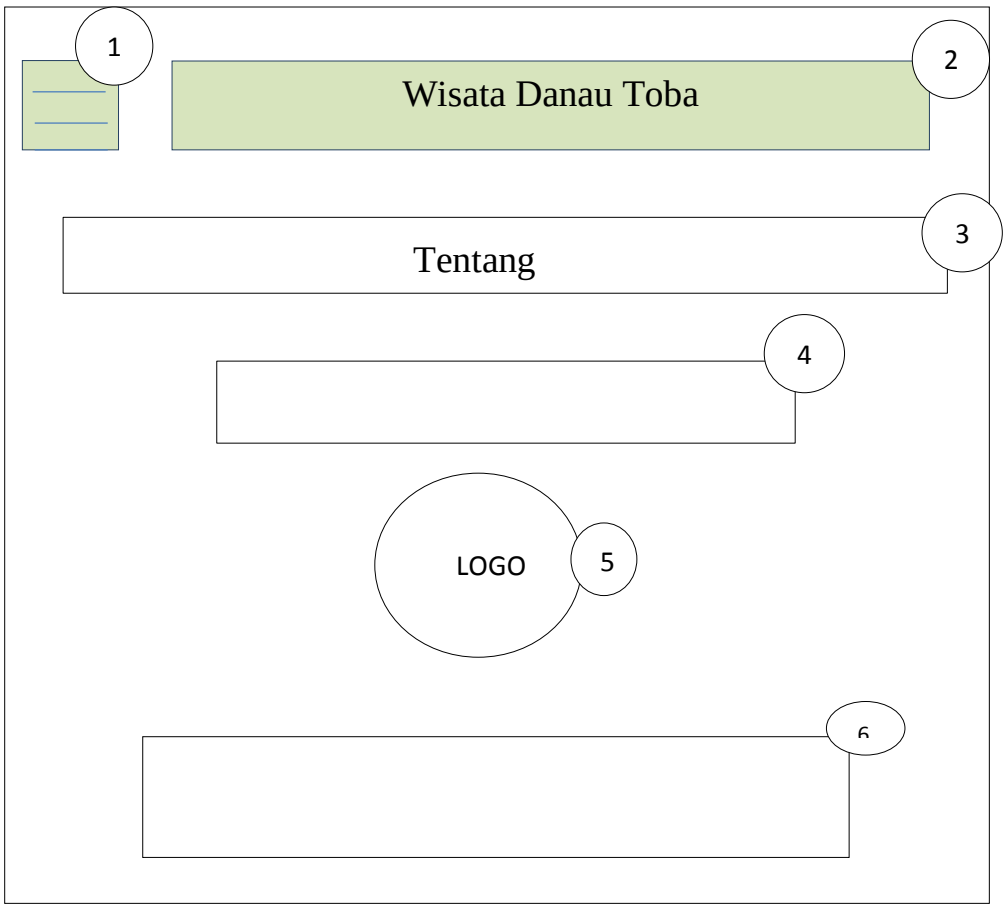
Gambar 3.10 Rancangan *Interface* Halaman bantuan

Keterangan Gambar 3.10 :

1. Tampilan yang berisi halaman-halaman aplikasi yang tersedia berupa : halaman lokasi wisata, halaman riwayat, halaman bantuan dan halaman tentang.
2. Tampilan yang berisi nama aplikasi.
3. Tampilan untuk menunjukkan halaman yang tersedia pada sistem.
4. Tampilan untuk menampilkan urutan cara penggunaan aplikasi.

3.3.6 Halaman Tentang

Rancangan *interface* halaman tentang dapat dilihat pada Gambar 3.11.Halaman tentang adalah tampilan yang berisi tentang biodata dari penulis.



Gambar 3.11 Rancangan *Interface* Halaman tentang

Keterangan Gambar 3.11 :

1. Tampilan yang berisi halaman-halaman aplikasi yang tersedia berupa : halaman lokasi wisata, halaman riwayat, halaman bantuan dan halaman tentang.
2. Tampilan yang berisi nama aplikasi.
3. Tampilan untuk menunjukkan halaman yang tersedia pada sistem.
4. Tampilan yang berisi judul skripsi, nama dan NIM.
5. Tampilan untuk menampilkan logo Universitas Sumatera Utara.
6. Tampilan untuk menampilkan Program Studi.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dijelaskan tentang proses implementasi dan pengujian terhadap aplikasi, sesuai dengan perancangan pada Bab 3 dan melakukan pengujian aplikasi yang telah dibangun.

4.1 Implementasi

Tahap implementasi merupakan tahapan lanjutan dari tahap perancangan dan analisis. Dalam penelitian ini, bahasa pemrograman yang akan digunakan untuk membangun sistem adalah *Java mobile programming* dengan menggunakan *Software Android Studio*. Proses pengimplementasian menggunakan *hardware* dan *software* untuk menjalankan aplikasi.

4.1.1 Spesifikasi Perangkat Keras (Hardware)

Spesifikasi perangkat keras yang digunakan dalam implementasi aplikasi ini adalah sebagai berikut :

1. Prosessor Intel® CoreTM i3- 2348M CPU @ 2.30GHz 2.30GHz
2. Memory (RAM): 4,00 GB
3. Monitor 14.0”
4. Samsung T231 Galaxy Tab 4.7.0 “ 3G (SM-T231)

4.1.2 Spesifikasi Perangkat Lunak (Software)

1. Windows 10
2. Android Studio 2015
3. SQLite Android

4.2 Evaluasi Pengujian Sistem

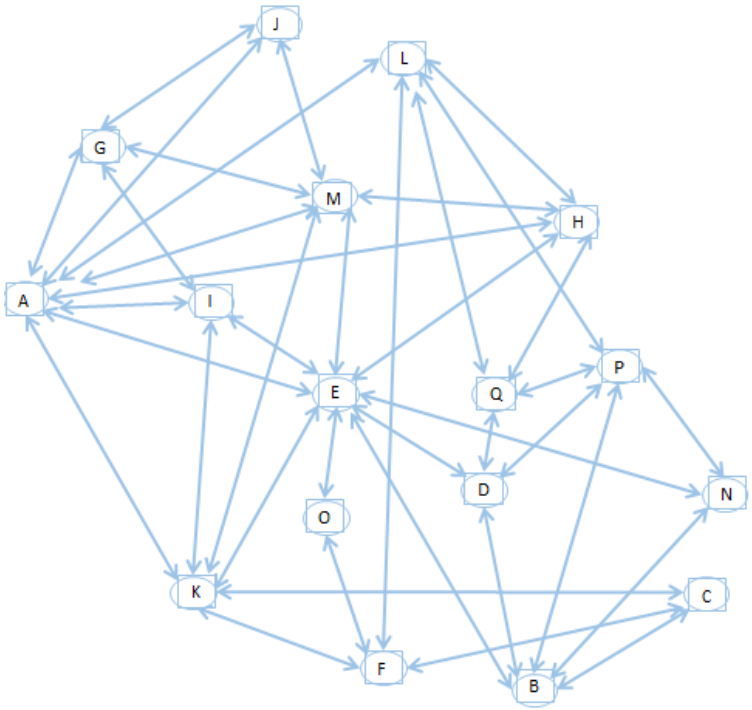
Pada evaluasi pengujian sistem ini dilakukan untuk melihat aplikasi apakah dapat berjalan dan berfungsi sesuai dengan perancangan sistem yang di telah dibuat. Pada bab ini akan dijelaskan tentang pengujian aplikasi yang telah dibuat, penjelasan dibagi atas uji metode, uji *interface* dan uji sistem.

4.2.1 Uji Metode

Pada tahap pengujian metode perlu dilakukan seberapa baik kerja metode dalam suatu sistem, jika suatu metode tidak dapat bekerja maka solusi yang digunakan adalah perlu di selidiki adanya faktor kendala. Metode yang dipakai dalam tahap uji kali ini adalah metode *L-Deque* dapat menemukan rute terdekat dari suatu titik awal yang diinginkan menuju titik destinasi antar objek wisata.

4.2.1.1 Perhitungan Manual Algoritma L-Deque

Berdasarkan data lokasi objek wisata di kawasan Danau Toba yang telah dikumpulkan dan kemudian dihitung jarak antar titik lokasi, lalu digambarkan ke dalam sebuah *graph* yang berupa *vertex* (simpul) dan *edges* (jarak). Pada contoh penentuan rute terdekat dengan menggunakan algortima L-Deque dapat dilihat pada Gambar 4.1.



Gambar 4.1 Notasi *Graph* Pencarian Rute Terdekat

Dari gambar 4.9 diatas terdapat 17 buah titik atau node yakni A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P dan Q. Adapan keterangan titik atau node pada *graph* dapat dilihat pada Tabel 4.1.

Tabel 4.1 Keterangan Node pada *Graph*

NO	*CODE*	*KETERANGAN*
1	*A*	Pulau samosir
2	*B*	Parapat
3	*C*	Pantai bulbul
4	*D*	Tongging
5	*E*	Pusuk buhit
6	*F*	Sipinsur
7	*G*	Ambarita
8	*H*	Simanindo
9	*I*	Pangguruan
10	*J*	Tuktuk
11	*K*	Aek sipitu dai
12	*L*	Danau sidihoni
13	*M*	Sialanguan
14	*N*	Rumah pengasingan mantan Presiden RI Soekarno
15	*O*	Balige cultural center
16	*P*	Haranggaol
17	*Q*	Paropo

Data yang akan dianalisis pada penelitian ini adalah jarak antar titik, adapun jarak antara node asal ke node tujuan dapat dilihat pada Tabel 4.2.

Tabel 4.2 Jarak Antar Titik

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
A	0	192	134	128	11	105	37	22	27	38	32	14	26	193	137	150	113
B	192	0	61	95	178	96	207	194	176	211	165	180	185	18	65	80	123
C	134	61	0	139	120	38	149	136	118	153	107	122	127	62	8,5	127	160
D	128	95	139	0	115	147	143	131	112	147	101	116	121	83	147	26	34
E	11	178	120	115	0	91	37	24	9,8	41	18	17	15	179	124	137	99
F	105	96	38	147	91	0	120	107	89	124	78	92	98	97	40	162	131
G	37	207	149	143	37	120	0	43	38	4,6	47	45	26	208	152	165	128
H	22	194	136	131	24	107	43	0	26	47	34	23	21	195	136	153	115
I	27	176	118	112	9,8	89	38	26	0	42	16	15	16	177	121	134	97
J	38	211	153	147	41	124	4,6	47	42	0	51	49	31	211	157	169	132
K	32	165	107	101	18	78	47	34	16	51	0	19	25	165	111	123	86
L	14	180	122	116	17	92	45	23	15	49	19	0	23	177	122	121	84
M	26	185	127	121	15	98	26	21	16	31	25	23	0	183	128	127	90
N	193	18	62	83	179	97	208	195	177	211	165	177	183	0	66	54	104
O	137	65	8,5	147	124	40	152	136	121	157	111	122	128	66	0	117	145
P	150	80	127	26	137	162	165	153	134	169	123	121	127	54	117	0	57
Q	113	123	160	34	99	131	128	115	97	132	86	84	90	140	145	57	0

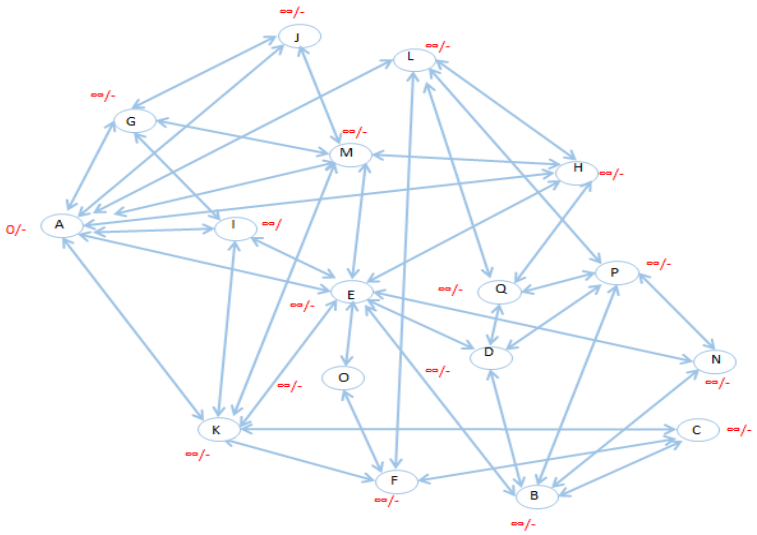
Adapun langkah – langkah untuk mencari rute terdekat menggunakan algoritma L-Deque adalah sebagai berikut:

Langkah 1:

Masukkan seluruh *vertex* yang ada pada *graph* ke daftar data deque. Daftar deque [A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q].

Langkah 2:

Vertex asal diatas pada *graph* ialah *vertex* A, tandai *vertex* tersebut dengan nilai 0 dan beri nilai *infinite*(∞) pada *vertex* lainnya.



Gambar 4.2 *Graph* Penentuan *Vertex* Asal

Langkah 3:

Hitung *vertex* yang bisa dikunjungi *vertex* A, yaitu AE dengan bobot jarak 11, AG dengan bobot jarak 37, AH dengan bobot jarak 22, AI dengan bobot jarak 27, AJ dengan bobot jarak 38, AK dengan bobot jarak 32, AL dengan bobot jarak 14, dan AM dengan bobot jarak 26.

Hitung d[v] untuk *vertex* E :

d[v] > d[u] + *edges* [u][v]

d[E] > d[A] + *edges* [A][E]

d[E] > 0 + 11

inf >11

Hitung d[v] untuk *vertex* G :

d[v] > d[u] + *edges* [u][v]

d[G] > d[A] + *edges* [A][G]

d[G] > 0 + 37

inf >37

Hitung d[v] untuk *vertex* H :

d[v] > d[u] + *edges* [u][v]

d[H] > d[A] + *edges* [A][H]

d[H] > 0 + 22

inf >22

Hitung d[v] untuk *vertex* I :

d[v] > d[u] + *edges* [u][v]

d[I] > d[A] + *edges* [A][I]

d[I] > 0 + 27

inf >27

Hitung d[v] untuk *vertex* J :

d[v] > d[u] + *edges* [u][v]

d[J] > d[A] + *edges* [A][J]

d[J] > 0 + 38

inf >38

Hitung d[v] untuk *vertex* K :

d[v] > d[u] + *edges* [u][v]

d[K] > d[A] + *edges* [A][K]

d[K] > 0 + 32

inf >32

Hitung d[v] untuk *vertex* L :

d[v] > d[u] + *edges* [u][v]

d[L] > d[A] + *edges* [A][L]

d[L] > 0 + 14

inf >14

Hitung d[v] untuk *vertex* M :

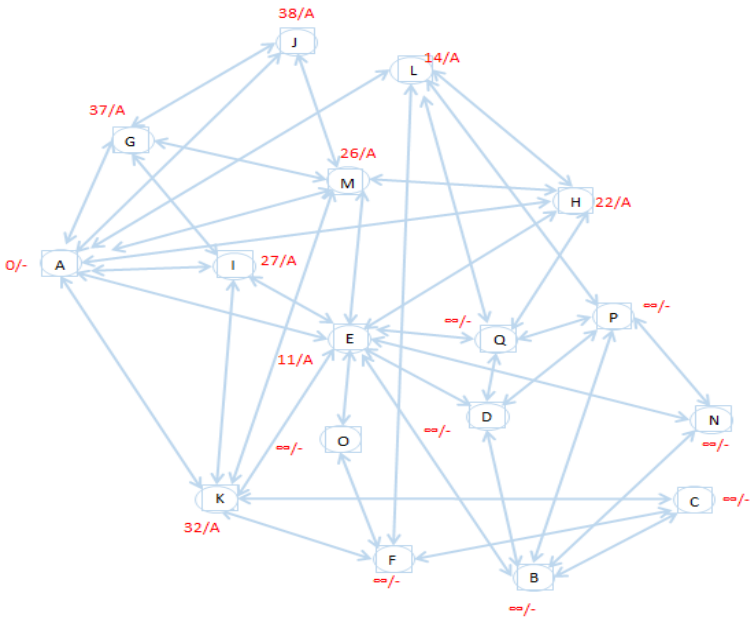
d[v] > d[u] + *edges* [u][v]

d[M] > d[A] + *edges* [A][M]

d[M] > 0 + 26

inf >26

Jadi diperoleh rute terpendek d[E]= 11, d[G]=37, d[H]=22, d[I]=27, d[J]=38, d[K]=32, d[L]=14 dan d[M] = 26 melalui *vertex* A.



Gambar 4.3 Perhitungan *Graph* dari *Vertex* A

Kemudian hitung *vertex* yang bisa dikunjungi *vertex* E, yaitu EH dengan bobot jarak 24, EI dengan bobot jarak 9,8, EK dengan bobot jarak 18, EM dengan bobot

jarak 15, ED dengan bobot jarak 115, EO dengan bobot jarak 124, EB dengan bobot jarak 178 dan EN dengan bobot jarak 179.

Hitung d[v] untuk *vertex* H :

d[v] > d[u] + *edges* [u][v]

d[H] > d[E] + *edges* [E][H]

22>11 + 24

22>35 (‘Tidak’)

Hitung d[v] untuk *vertex* I :

d[v] > d[u] + *edges* [u][v]

d[I] > d[E] + *edges* [E][I]

27>11 + 9,8

27>20,8(‘Ya’)

Hitung d[v] untuk *vertex* D :

d[v] > d[u] + *edges* [u][v]

d[D] > d[E] + *edges* [E][D]

d[D]>11 + 115

inf>126

Hitung d[v] untuk *vertex* O :

d[v] > d[u] + *edges* [u][v]

d[O] > d[E] + *edges* [E][O]

d[O]>11 + 124

inf>135

Hitung d[v] untuk *vertex* K :

d[v] > d[u] + *edges* [u][v]

d[K] > d[E] + *edges* [E][K]

32>11 + 18

37>29 (‘Ya’)

Hitung d[v] untuk *vertex* M :

d[v] > d[u] + *edges* [u][v]

d[M] > d[E] + *edges* [E][M]

26>11 + 15

26>26 (‘Sama dengan’)

Hitung d[v] untuk *vertex* Q :

d[v] > d[u] + *edges* [u][v]

d[Q] > d[E] + *edges* [E][Q]

d[Q]>11 + 99

inf>110

Hitung d[v] untuk *vertex* B :

d[v] > d[u] + *edges* [u][v]

d[B] > d[E] + *edges* [E][B]

d[B]>11 + 178

inf>189

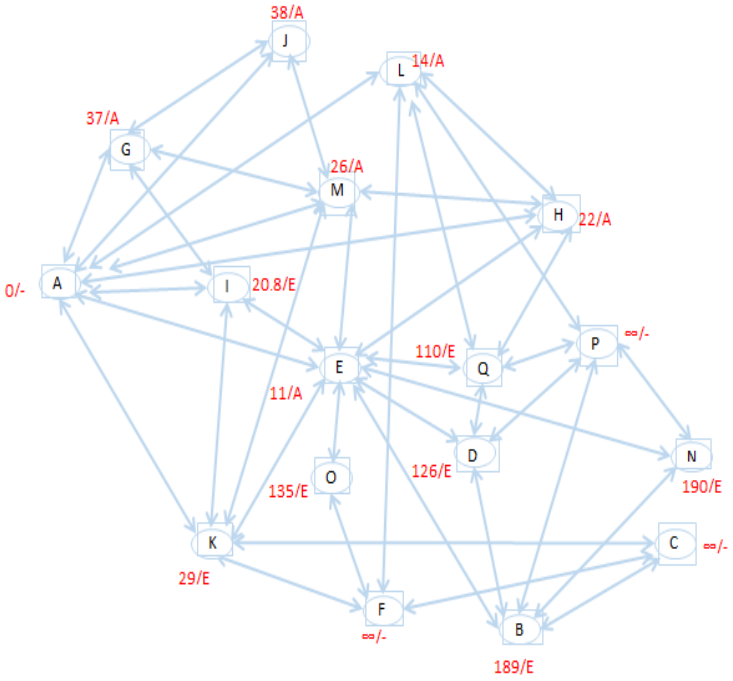
Hitung d[v] untuk *vertex* N :

d[v] > d[u] + *edges* [u][v]

d[N] > d[E] + *edges* [E][N]

d[N]>11 + 179

inf>190



Gambar 4.4 Perhitungan *Graph* dari *Vertex* E

Kemudian hitung yang bisa dilalui *vertex* M, yaitu, ME dengan bobot jarak 15, MG dengan bobot jarak 26, MH dengan bobot jarak 21, MJ dengan bobot jarak 31 dan MK dengan bobot jarak 25 .

Hitung d[v] untuk *vertex* E :

d[v] > d[u] + *edges* [u][v]

d[E] > d[M] + *edges* [M][E]

11>26 + 15

11>41 (‘Tidak’)

Hitung d[v] untuk *vertex* G :

d[v] > d[u] + *edges* [u][v]

d[G] > d[M] + *edges* [M][G]

37>26 + 26

37>52 (‘Tidak’)

Hitung d[v] untuk *vertex* H :

d[v] > d[u] + *edges* [u][v]

d[H] > d[M] + *edges* [M][H]

22>26 + 21

22>47 (‘Tidak’)

Hitung d[v] untuk *vertex* K :

d[v] > d[u] + *edges* [u][v]

d[K] > d[M] + *edges* [M][K]

29>26 + 25

29>51 (‘Tidak’)

Hitung d[v] untuk *vertex* J :

d[v] > d[u] + *edges* [u][v]

d[J] > d[M] + *edges* [M][J]

38>26 + 31

38>57 (‘Tidak’)

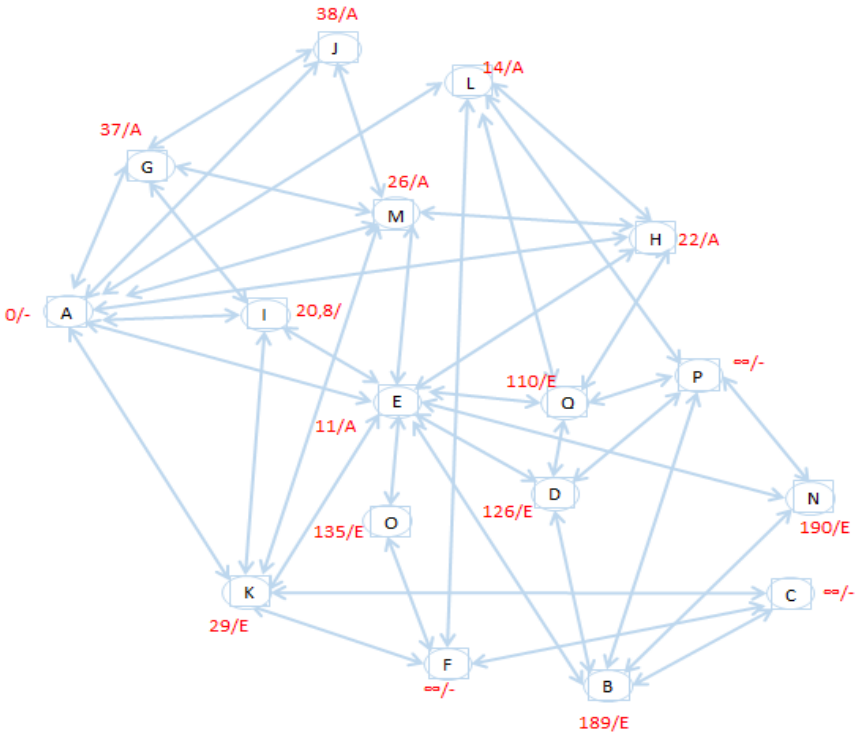
Hitung d[v] untuk *vertex* L :

d[v] > d[u] + *edges* [u][v]

d[L] > d[M] + *edges* [M][L]

14>26 + 23

14>49 (‘Tidak’)



Gambar 4.5 Perhitungan *Graph* dari *Vertex* M

Kemudian hitung yang bisa dilalui *vertex L*, yaitu LH dengan bobot jarak 23, LQ dengan bobot jarak 84, LF dengan bobot jarak 92 dan LP dengan bobot jarak 121.

Hitung d[v] untuk *vertex* H :

d[v] > d[u] + *edges* [u][v]

d[H] > d[L] + *edges* [L][H]

22>14 + 23

22>37 (‘Tidak’)

Hitung d[v] untuk *vertex* F :

d[v] > d[u] + *edges* [u][v]

d[F] > d[L] + *edges* [L][F]

d[F]>14 + 92

inf>106

Hitung d[v] untuk *vertex* Q :

d[v] > d[u] + *edges* [u][v]

d[Q] > d[L] + *edges* [L][Q]

110>14 + 84

110 >98 (‘Ya’)

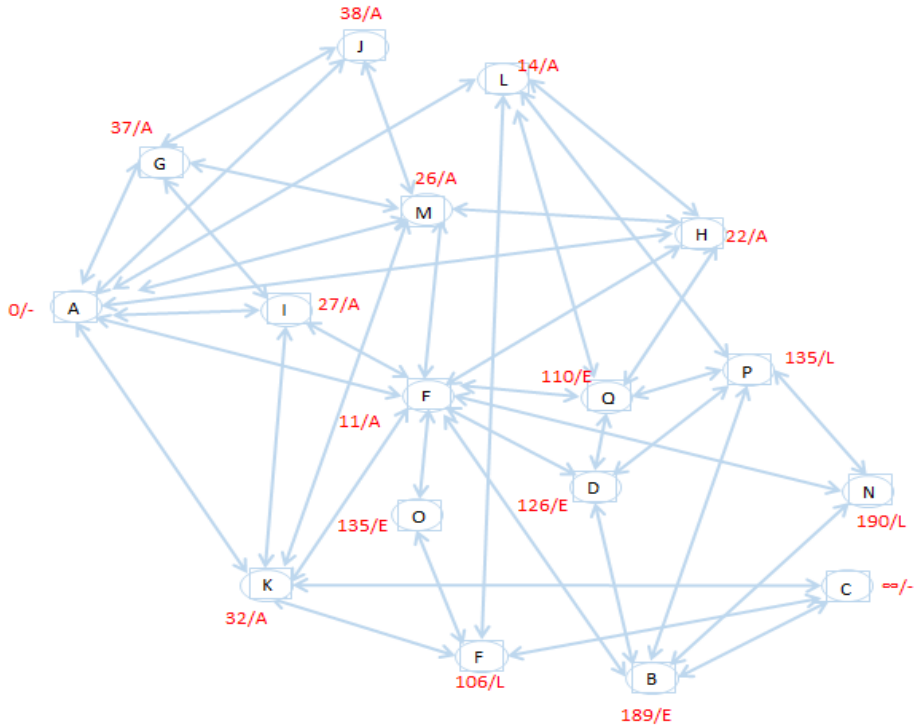
Hitung d[v] untuk *vertex* P :

d[v] > d[u] + *edges* [u][v]

d[P] > d[L] + *edges* [L][P]

d[P]>14 + 121

inf>135



Gambar 4.6 Perhitungan *Graph* dari *Vertex* L

Kemudian hitung *vertex* yang bisa dilalui *vertex* K, yaituKE dengan bobot jarak 18, KF dengan bobot jarak 78, KI dengan bobot jarak 16, KM dengan bobot jarak 25 dan KC dengan bobot jarak 107.

Hitung d[v] untuk *vertex* E :

d[v] > d[u] + *edges* [u][v]

d[E] > d[K] + *edges* [K][E]

11>29 + 18

11>47(‘Tidak’)

Hitung d[v] untuk *vertex* H :

d[v] > d[u] + *edges* [u][v]

d[H] > d[K] + *edges* [K][H]

22>26 + 34

22>60 (‘Tidak’)

Hitung d[v] untuk *vertex* M :

d[v] > d[u] + *edges* [u][v]

d[M] > d[K] + *edges* [K][M]

26>29 + 25

26>54 (‘Tidak’)

Hitung d[v] untuk *vertex* F :

d[v] > d[u] + *edges* [u][v]

d[F] > d[K] + *edges* [K][F]

106>29 + 78

106>107(‘Tidak’)

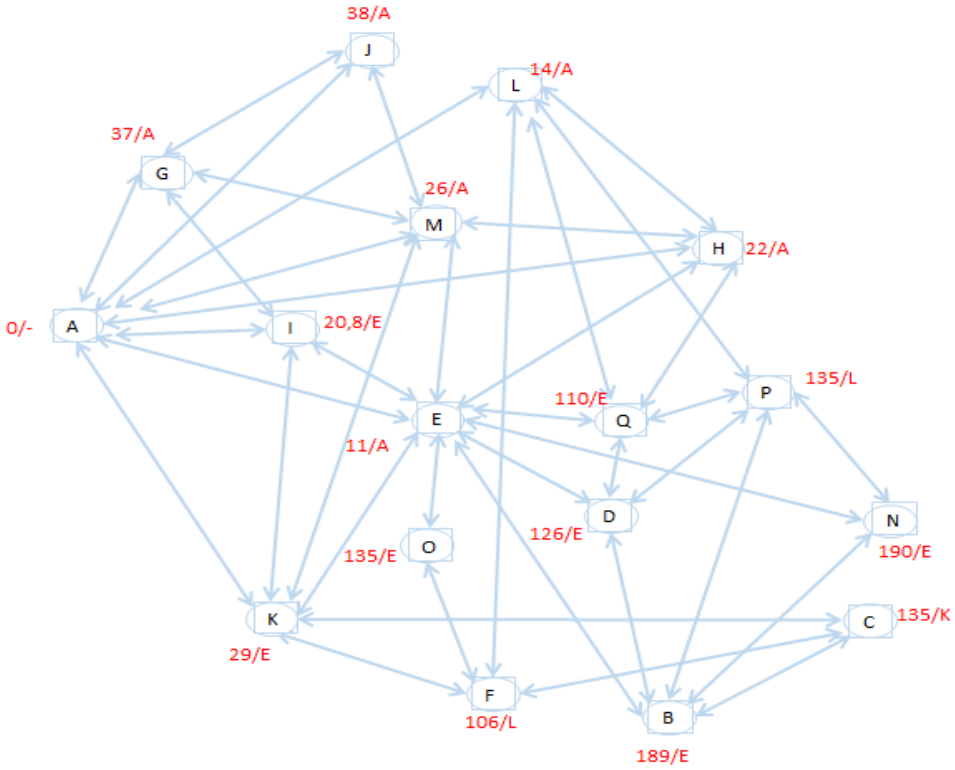
Hitung d[v] untuk *vertex* C :

d[v] > d[u] + *edges* [u][v]

d[C] > d[K] + *edges* [K][C]

d[C]>29 + 107

inf>135



Gambar 4.7 Perhitungan *Graph* dari *Vertex* K

Kemudian hitung ang bisa dilalui *vertex* G, yaitu GI dengan bobot jarak 38, GJ dengan bobot jarak 4,6 dan GM dengan bobot jarak 26.

Hitung d[v] untuk *vertex* I :

d[v] > d[u] + *edges* [u][v]

d[I] > d[G] + *edges* [G][I]

20,8>37 + 38

20,8>75 (‘Tidak’)

Hitung d[v] untuk *vertex* M :

d[v] > d[u] + *edges* [u][v]

d[M] > d[G] + *edges* [G][M]

26>37 + 26

26> 63 (‘Tidak’)

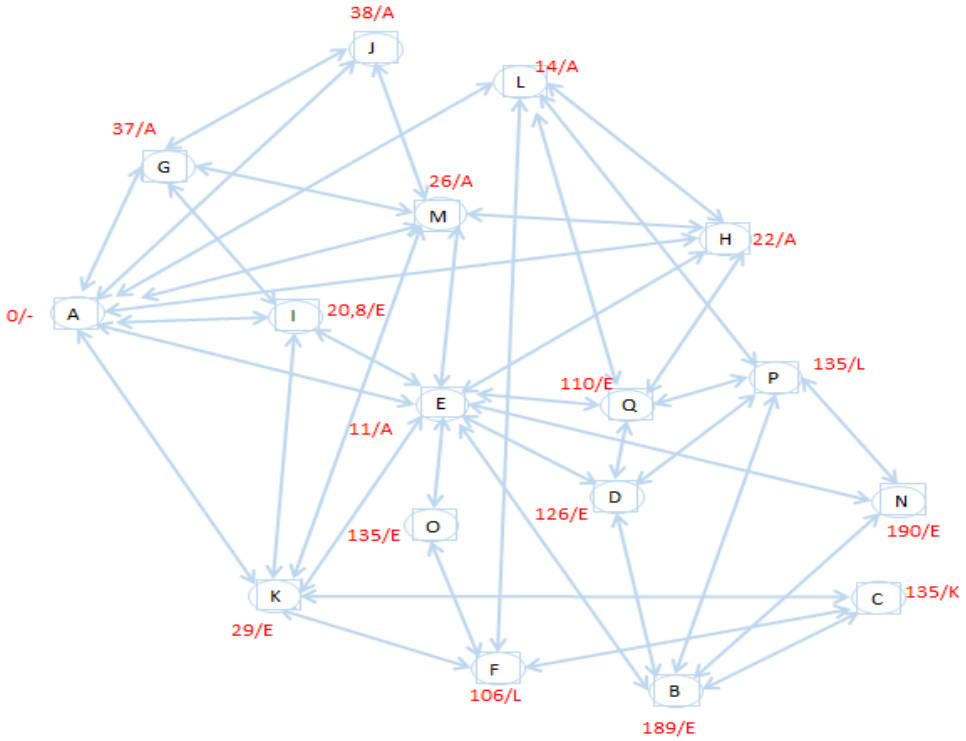
Hitung d[v] untuk *vertex* J :

d[v] > d[u] + *edges* [u][v]

d[J] > d[G] + *edges* [G][J]

38>37 + 4,6

38>41,6 (‘Tidak’)



Gambar 4.8 Perhitungan *Graph* dari*vertex* G

Kemudian hitung yang bisa dilalui *vertex* H, yaitu HE dengan bobot jarak 24 ,HL dengan bobot jarak 23, HM dengan bobor jarak 21 dan HQ dengan bobot jarak 115.

Hitung d[v] untuk*vertex* E:

d[v] > d[u] + *edges* [u][v]

d[E] > d[H] + *edges* [H][E]

11>22 + 24

11> 46 (‘Tidak’)

Hitung d[v] untuk *vertex* M:

d[v] > d[u] + *edges* [u][v]

d[M] > d[H] + *edges* [H][M]

26>22 + 21

26>43 (‘Tidak’)

Hitung d[v] untuk *vertex* L:

d[v] > d[u] + *edges* [u][v]

d[L] > d[H] + *edges* [H][L]

14>22 + 23

14>45(‘Tidak’)

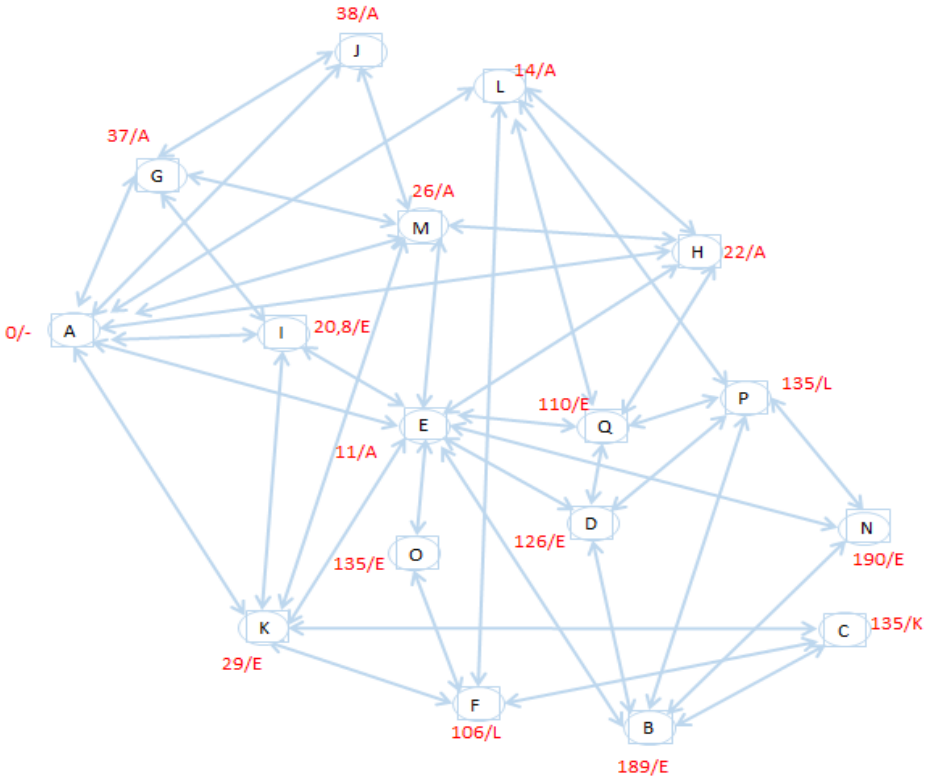
Hitung d[v] untuk *vertex* Q:

d[v] > d[u] + *edges* [u][v]

d[Q] > d[H] + *edges* [H][Q]

98>22 + 115

14>137 (‘Tidak’)



Gambar 4.9 Perhitungan *Graph* dari *Vertex* H

Kemudian hitung yang bisa dilalui *vertex* I, yaitu IE dengan bobot jarak 9,8 , IG dengan bobot jarak 38 dan IK dengan bobot jarak 16

Hitung d[v] untuk *vertex* E:

d[v] > d[u] + *edges* [u][v]

d[E] > d[I] + *edges* [I][E]

11>20,8 + 9,8

11>30,6 (‘Tidak’)

Hitung d[v] untuk *vertex* G:

d[v] > d[u] + *edges* [u][v]

d[G] > d[I] + *edges* [I][G]

37>20,8 + 38

37>58,8 (‘Tidak’)

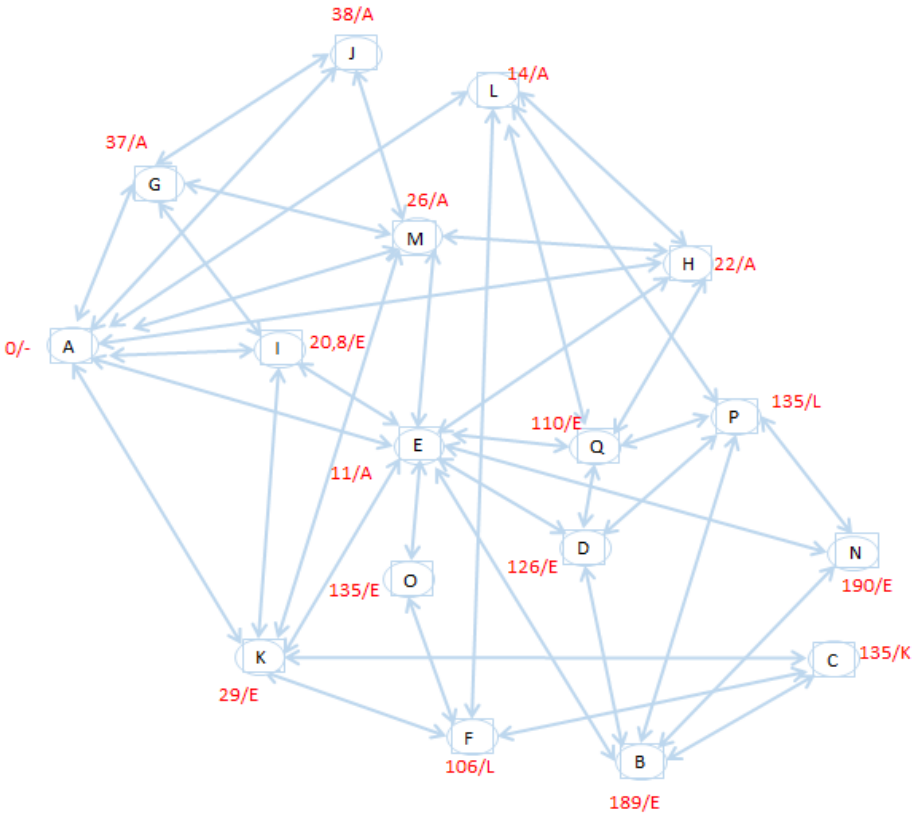
Hitung d[v] untuk *vertex* K:

d[v] > d[u] + *edges* [u][v]

d[K] > d[I] + *edges* [I][K]

29>20,8 + 16

29>36,8 (‘Tidak’)



Gambar 4.10 Perhitungan *Graph* dari *Vertex* I

Kemudian hitung yang bisa dilalui *vertex* J, yaitu JG dengan bobot jarak 4,6 dan JM dengan bobot jarak 31.

Hitung d[v] untuk *vertex* G:

d[v] > d[u] + *edges* [u][v]

d[G] > d[J] + *edges* [J][G]

37>38 + 4,6

37>42,6 (‘Tidak’)

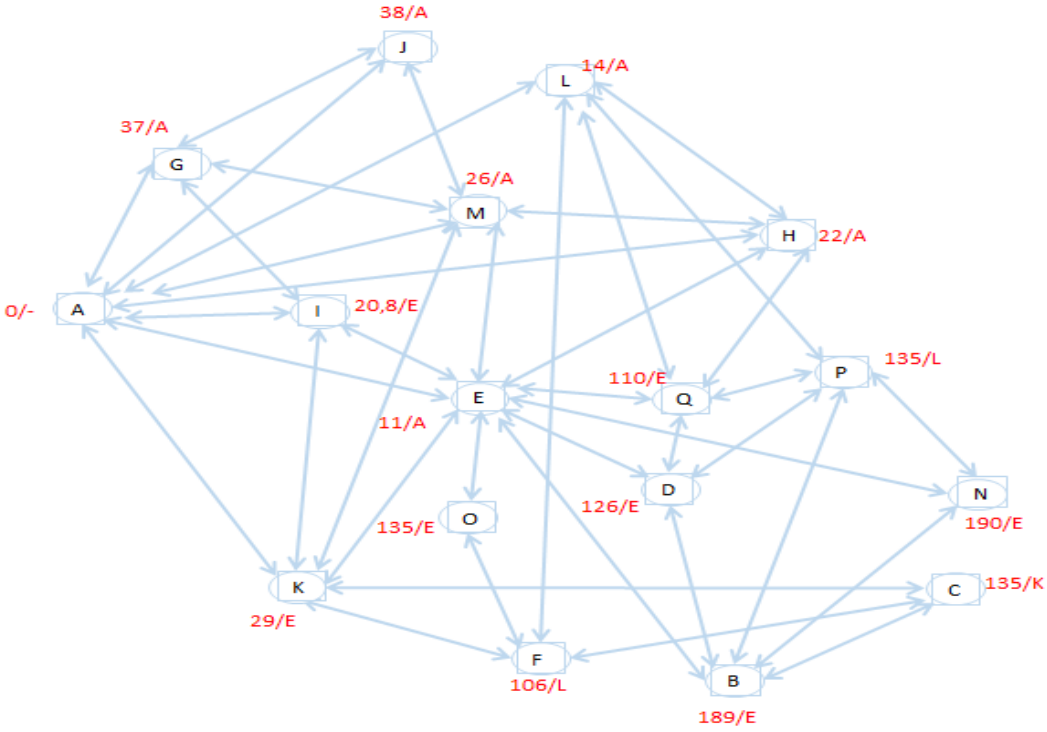
Hitung d[v] untuk *vertex* M:

d[v] > d[u] + *edges* [u][v]

d[M] > d[J] + *edges* [J][M]

26>38 + 31

26>69 (‘Tidak’)



Gambar 4.11 Perhitungan *Graph* dari *Vertex* J

Kemudian hitung yang bisa dilalui *vertex* F, yaitu FC dengan bobot jarak 38, FO dengan bobot jarak 40, FK dengan bobot jarak 78 dan FL dengan bobot jarak 92.

Hitung d[v] untuk *vertex* C:

d[v] > d[u] + *edges* [u][v]

d[C] > d[F] + *edges* [F][C]

135>106 + 38

135>144 (‘Tidak)

Hitung d[v] untuk *vertex* O:

d[v] > d[u] + *edges* [u][v]

d[O] > d[F] + *edges* [F][O]

135>106 + 40

135>146(‘Tidak’)

Hitung d[v] untuk *vertex* K:

d[v] > d[u] + *edges* [u][v]

d[K] > d[F] + *edges* [F][K]

29>106 + 78

29>184 (‘Tidak’)

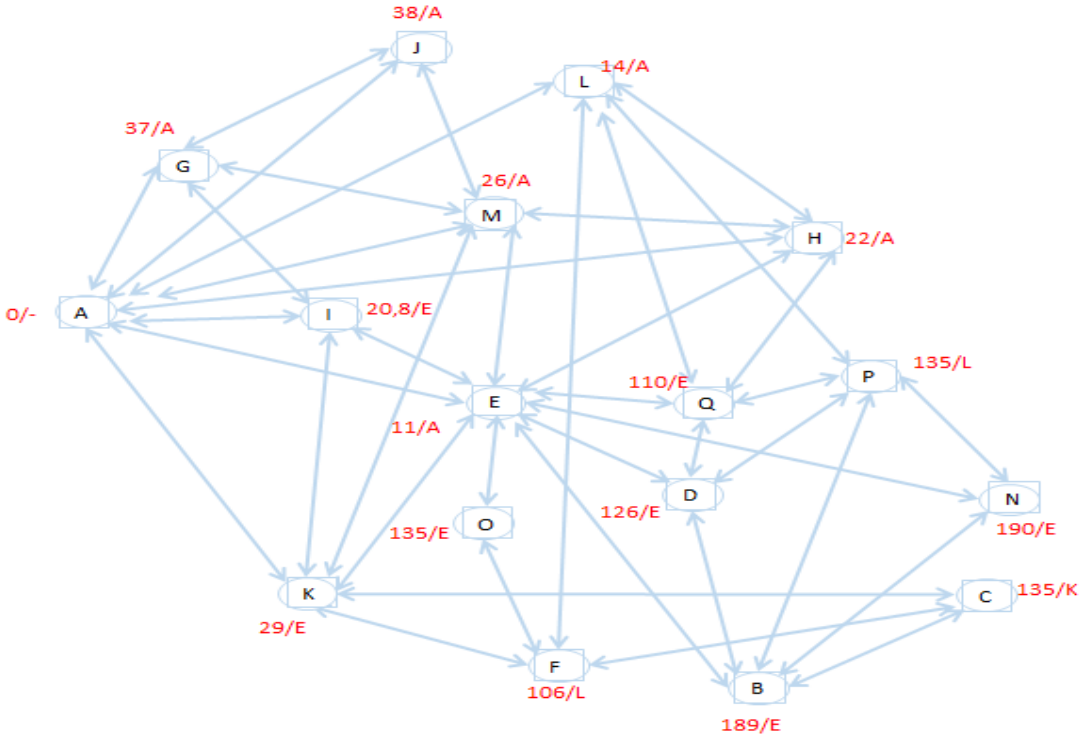
Hitung d[v] untuk *vertex* L:

d[v] > d[u] + *edges* [u][v]

d[L] > d[F] + *edges* [F][L]

14 >106 + 92

14>189 (‘Tidak’)



Gambar 4.12 Perhitungan *Graph* dari *Vertex* F

Kemudian hitung yang bisa dilalui *vertex* C, yaitu CB dengan bobot jarak 61, CF dengan bobot jarak 38, CK dengan bobot jarak 107 dan CO dengan bobot jarak 8,5.

Hitung d[v] untuk *vertex* B:

d[v] > d[u] + *edges* [u][v]

d[B] > d[C] + *edges* [B][C]

189>135 + 61

189 >196 (‘Tidak’)

Hitung d[v] untuk *vertex*K:

d[v] > d[u] + *edges* [u][v]

d[K] > d[C] + *edges* [C][K]

29>135 + 107

29>242 (‘Tidak’)

Hitung d[v] untuk *vertex* F:

d[v] > d[u] + *edges* [u][v]

d[F] > d[C] + *edges* [C][F]

106>135 + 38

106>172 (‘Tidak’)

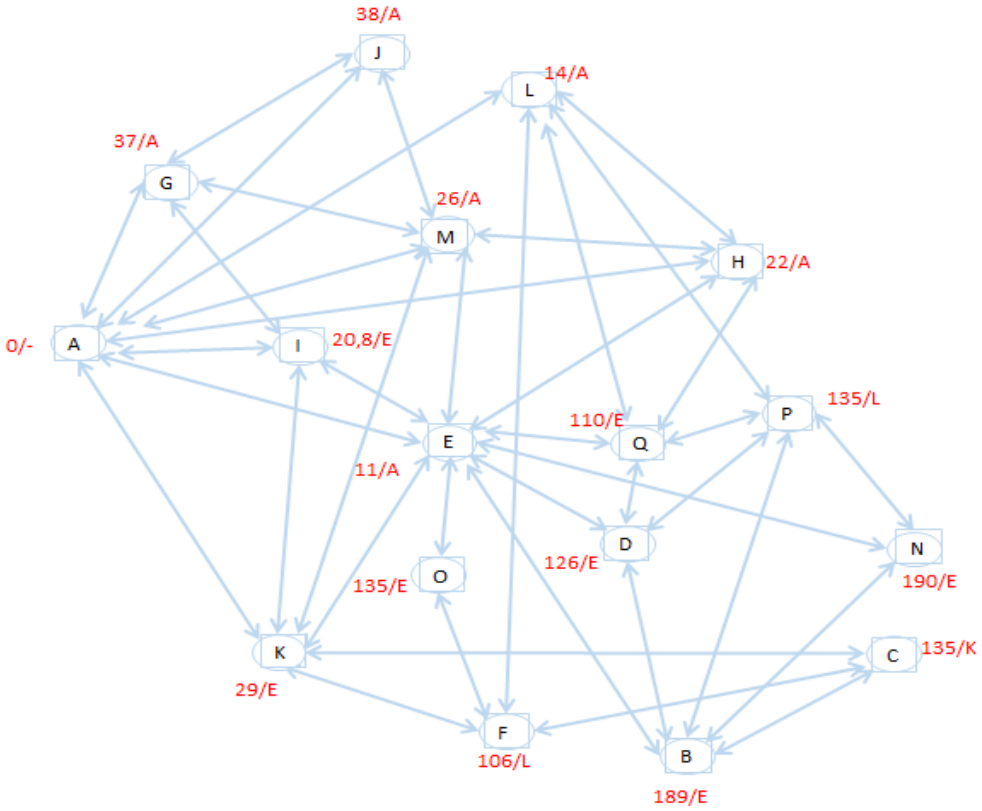
Hitung d[v] untuk *vertex* O:

d[v] > d[u] + *edges* [u][v]

d[O] > d[C] + *edges* [C][O]

135>135 + 8,5

135>143,5 (‘Tidak’)



Gambar 4.13 Perhitungan *Graph* dari *Vertex* C

Kemudian hitung yang bisa dilalui *vertex* N, yaitu NB dengan bobot jarak 18, NP dengan bobot jarak 54 dan NE dengan bobot jarak 179.

Hitung d[v] untuk *vertex* B:

d[v] > d[u] + *edges* [u][v]

d[B] > d[N] + *edges* [N][B]

189>190 + 18

189 >208 (‘Tidak’)

Hitung d[v] untuk *vertex* E:

d[v] > d[u] + *edges* [u][v]

d[E] > d[N] + *edges* [N][E]

11>190 + 369

11>219 (‘Tidak’)

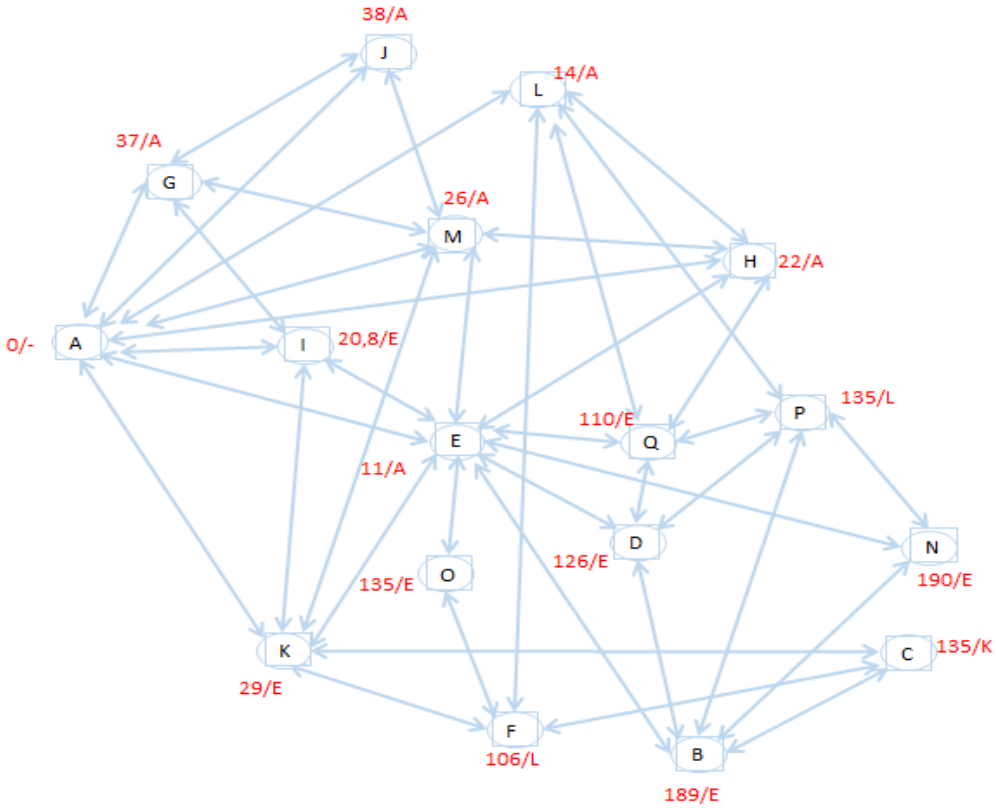
Hitung d[v] untuk *vertex* P:

d[v] > d[u] + *edges* [u][v]

135> d[N] + *edges* [N][P]

135>190 + 54

135 >244 (‘Tidak’)



Gambar 4.14 Perhitungan *Graph* dari *Vertex* N

Kemudian hitung yang bisa dilalui *vertex* B, yaitu BC dengan bobot jarak 61, BN dengan bobot jarak 18, BP dengan bobot jarak 80, BE dengan bobot jarak 178.

Hitung d[v] untuk *vertex* C:

d[v] > d[u] + *edges* [u][v]

d[C] > d[B] + *edges* [B][C]

135>189 + 61

135>150 (‘Tidak’)

Hitung d[v] untuk *vertex* N:

d[v] > d[u] + *edges* [u][v]

d[N] > d[B] + *edges* [B][N]

190>189 + 18

190>107(‘Ya’)

Hitung d[v] untuk *vertex* P:

d[v] > d[u] + *edges* [u][v]

d[P] > d[B] + *edges* [B][P]

135>189 + 80

135>269 (‘Tidak’)

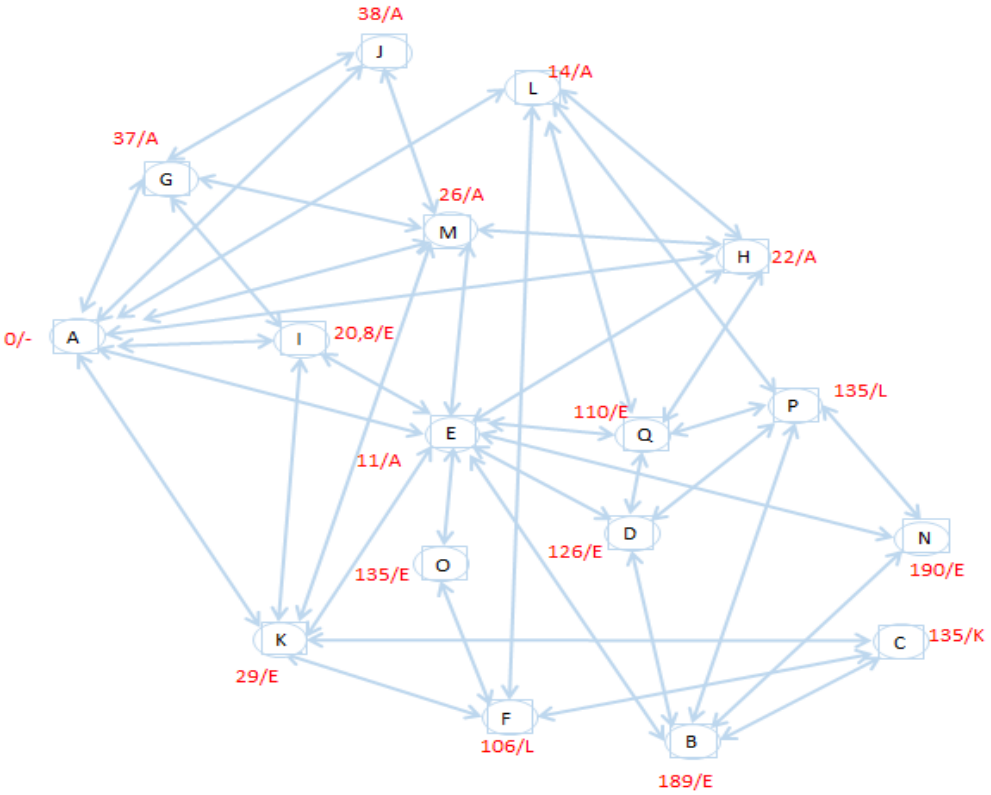
Hitung d[v] untuk *vertex* E:

d[v] > d[u] + *edges* [u][v]

d[E] > d[B] + *edges* [B][E]

11>189 + 178

11>268(‘Tidak’)



Gambar 4.15 Perhitungan *Graph* dari *Vertex* B

Kemudian hitung yang bisa dilalui *vertex* P, yaitu PB dengan bobot jarak 80, PN dengan bobot jarak 54, PQ dengan bobot jarak 57 dan PL dengan bobot jarak 121.

Hitung d[v] untuk *vertex* B:

d[v] > d[u] + *edges* [u][v]

d[B] > d[P] + *edges* [P][B]

125>135 + 80

125>215 (‘Tidak’)

Hitung d[v] untuk *vertex* Q:

d[v] > d[u] + *edges* [u][v]

d[Q] > d[P] + *edges* [P][Q]

98>135 + 57

98>192 (‘Tidak’)

Hitung d[v] untuk *vertex* N:

d[v] > d[u] + *edges* [u][v]

d[N] > d[P] + *edges* [P][N]

107>135 + 54

107>189 (‘Tidak’)

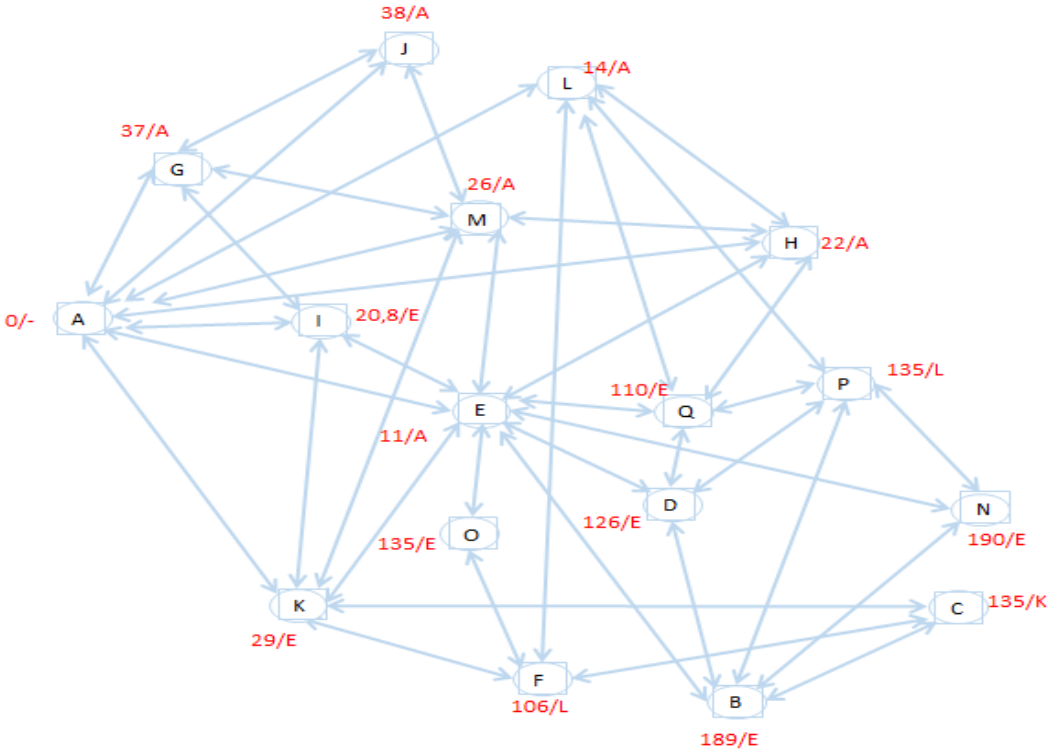
Hitung d[v] untuk *vertex* L:

d[v] > d[u] + *edges* [u][v]

d[L] > d[P] + *edges* [P][L]

14 >135 + 121

14>156 (‘Tidak’)



Gambar 4.16 Perhitungan *Graph* dari *Vertex* P

Kemudian hitung yang bisa dilalui *vertex* D, yaitu DP dengan bobot jarak 26 dan DQ dengan bobot jarak 34, DE dengan bobot jarak 115, dan DB dengan bobot jarak 95.

Hitung d[v] untuk *vertex* P:

d[v] > d[u] + *edges* [u][v]

d[P] > d[D] + *edges* [D][P]

135>126+ 26

135>152(‘Tidak’)

Hitung d[v] untuk *vertex* Q:

d[v] > d[u] + *edges* [u][v]

d[Q] > d[D] + *edges* [D][Q]

98>126 + 34

98>160(‘Tidak’)

Hitung d[v] untuk *vertex* E:

d[v] > d[u] + *edges* [u][v]

d[E] > d[D] + *edges* [D][E]

11>126 + 115

11>241 (‘Tidak’)

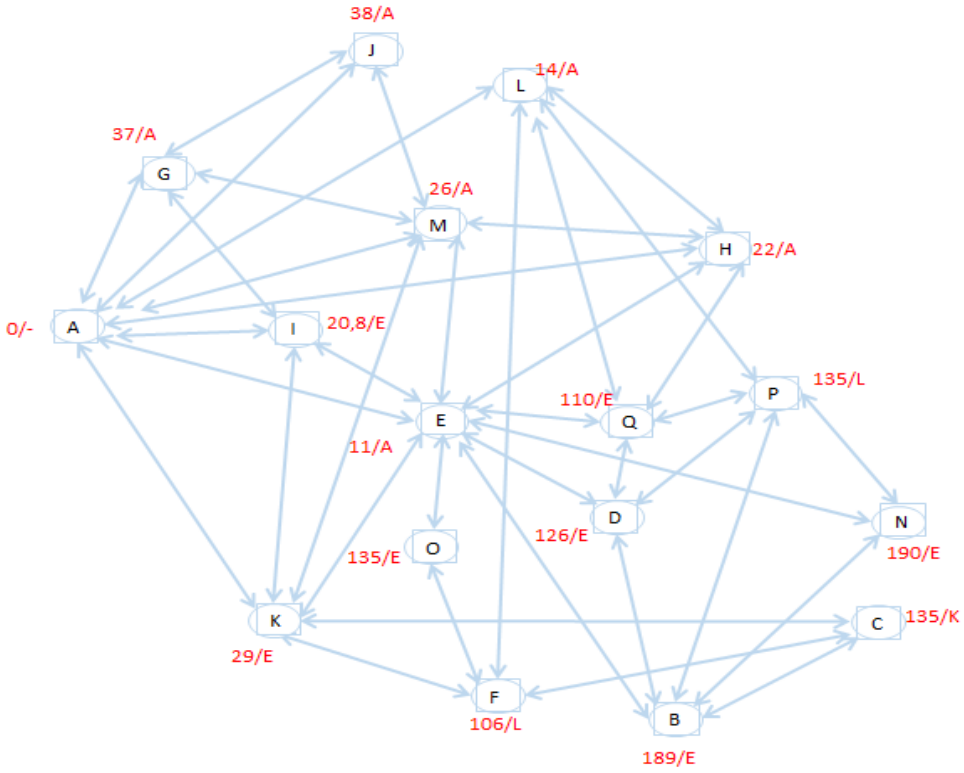
Hitung d[v] untuk *vertex* B:

d[v] > d[u] + *edges* [u][v]

d[B] > d[D] + *edges* [D][B]

125>126 + 95

125>221 (‘Tidak’)



Gambar 4.17 Perhitungan *Graph* dari *Vertex* D

Kemudian hitung yang bisa dilalui *vertex* Q, yaitu QD dengan bobot jarak 34 dan QP dengan bobot jarak 57, QE denganbobot jarak 99, QL dengan bobot jarak 84, QH dengan bobot jarak 115.

Hitung d[v] untuk *vertex* D:

d[v] > d[u] + *edges* [u][v]

d[D] > d[Q] + *edges* [Q][D]

126>98 + 34

126>132 (‘Tidak’)

Hitung d[v] untuk *vertex* E:

d[v] > d[u] + *edges* [u][v]

d[E] > d[Q] + *edges* [Q][E]

11>98 + 99

11>197 (‘Tidak’)

Hitung d[v] untuk *vertex* P:

d[v] > d[u] + *edges* [u][v]

d[P] > d[Q] + *edges* [Q][P]

135>98 + 57

135>155 (‘Tidak’)

Hitung d[v] untuk *vertex* L:

d[v] > d[u] + *edges* [u][v]

d[L] > d[Q] + *edges* [Q][L]

14>98 + 84

14>182(‘Tidak’)

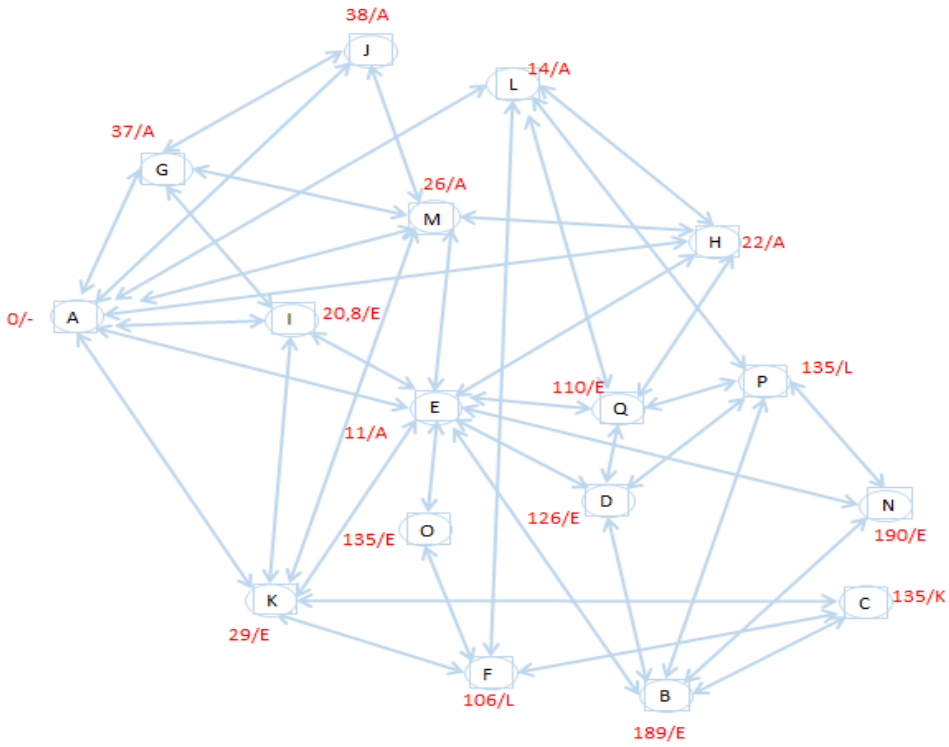
Hitung d[v] untuk *vertex* H:

d[v] > d[u] + *edges* [u][v]

d[H] > d[Q] + *edges* [Q][H]

22>98 + 115

22>213 (‘Tidak’)



Gambar 4.18 Perhitungan *Graph* dari *Vertex* Q

Kemudian hitung yang bisa dilalui *vertex* O, yaitu OE dengan bobot jarak 124, OC dengan bobot jarak 8,5 dan OF dengan bobot jarak 40.

Hitung d[v] untuk *vertex* E:

d[v] > d[u] + *edges* [u][v]

d[E] > d[O] + *edges* [O][E]

11>135 + 124

11>259 (‘Tidak’)

Hitung d[v] untuk *vertex* C:

d[v] > d[u] + *edges* [u][v]

d[C] > d[O] + *edges* [O][C]

135>135 + 8,5

135>143,5(‘Tidak’)

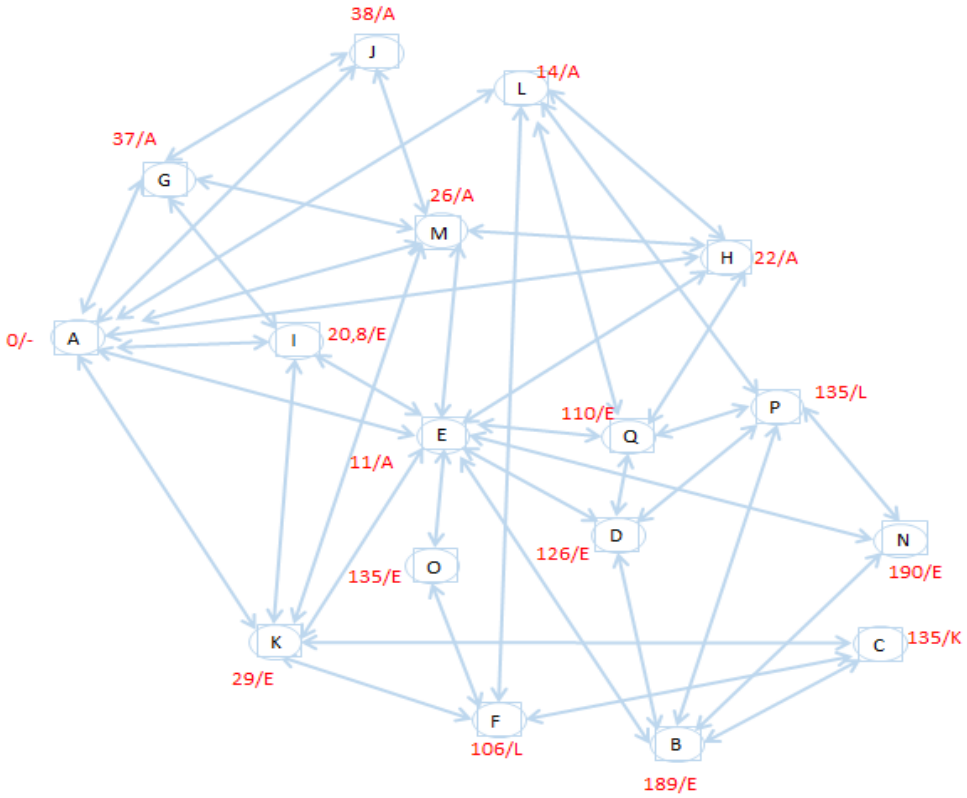
Hitung d[v] untuk *vertex* F:

d[v] > d[u] + *edges* [u][v]

d[F] > d[O] + *edges* [O][F]

106>135 + 40

106>175(‘Tidak’)



Gambar 4.19 Perhitungan *Graph d*ari *Vertex* O

Hasil yang diperoleh dari *graph* menggunakan Algoritma L-Deque ialah :

AB = AE - EB = 189

AC = AE - EK - KC = 135

AD = AE – ED = 126

AH = 22(Direct)

AI = AE – EI = 20,8

AJ = 38

AE = 11 (Direct)

AF = 106

AG = 37 (Direct)

AK = AE – EK = 29

AL = 14

AM = 26

AN = AE – EB – BN = 107 AP = AL - LP = 135

AO = AE - EO = 135 AQ = AE - EQ = 110

4.2.2 Uji Interface

Interface yang baik perlu memperhatikan faktor pengguna dalam menggunakan sistem, selain untuk mempermudah pengguna dalam mengguunakan sistem yang dibangun juga perlu diperhatikan kenyamanan dari pengguna dalam menggunakan sistem tersebut. *Interface* yang terdapat pada sistem ini memiliki 6 halaman yaitu halaman utama, halaman menu, halaman lokasi wisata, halaman riwayat, halaman bantuan dan halaman tentang.

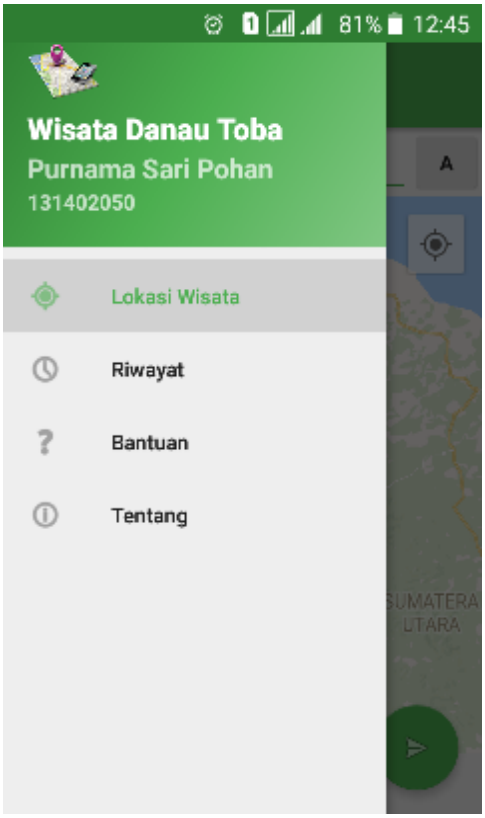
4.2.2.1 Tampilan Halaman Utama

Tampilan halaman utama merupakan tampilan halaman yang terlihat pertama sekali pada saat sistem dijalankan. Halaman ini berisi menu halaman-halaman yang tersedia, input data untuk pemilihan *verteks* titik awal(A), proses mencari rute, hasil pilihan *list* rute dan pilihan destinasi yang dipilih oleh *user*. Tampilan halaman utama dapat dilihat pada Gambar 4.20.

Gambar 4.20 Tampilan Halaman Utama

4.2.2.2 Tampilan Halaman Menu

Tampilan halaman menu merupakan tampilan halaman menu – menu yang tersedia pada sistem. Tampilan halaman menu dapat dilihat pada Gambar 4.21.



Gambar 4.21 Tampilan Halaman Menu

4.2.2.3 Tampilan Halaman Lokasi Wisata

Halaman lokasi wisata adalah tampilan yang berisi tentang titik koordinat dari objek wisata yang tersedia pada sistem. Tampilan halaman lokasi wisata dapat dilihat pada gambar 4.22.

Gambar 4.22 Tampilan Halaman Lokasi Wisata

4.2.2.4 Tampilan Halaman Riwayat

Halaman riwayat adalah merupakan tampilan yang berisi tentang riwayat – riwayat dalam penentuan rute objek wisata. Tampilan halaman lokasi wisata dapat dilihat pada gambar 4.23.

Gambar 4.23 Tampilan Halaman Riwayat

4.2.2.5 Tampilan Halaman Bantuan

Tampilan halaman bantuan dapat dilihat pada gambar 4.24 berisi tentang petunjuk cara penggunaan sistem, yang bertujuan untuk membantu penggunadalam menggunakan sistem.

Gambar 4.24 Tampilan Halaman Bantuan

4.2.2.6 Tampilan Halaman Tentang

Halaman tentang adalah tampilan yang berisi tentang biodata dari penulis. Tampilan halaman lokasi wisata dapat dilihat pada Gambar 4.25.

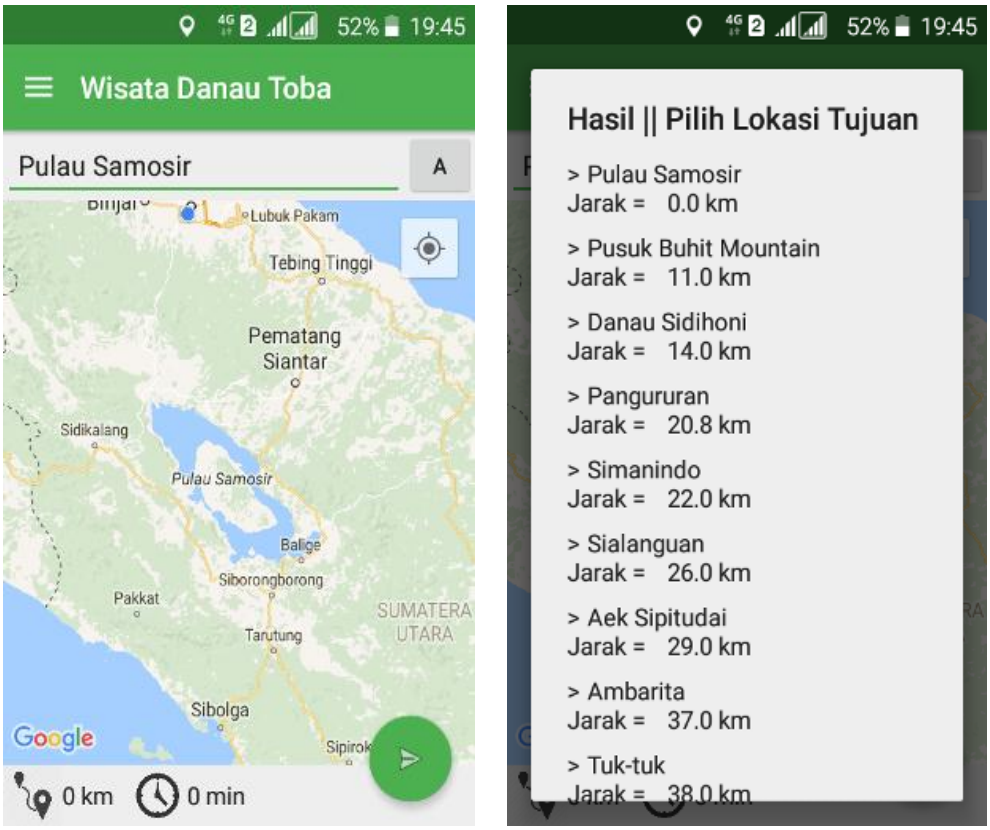
Gambar 4.25 Halaman Tentang

4.3 Pengujian Sistem

Pengujian sistem merupakan tahap lanjutan setelah implementasi sistem. Pengujian sistem bertujuan untuk membuktikan apakah sistem yang dibangun telah berjalan dengan baik dan sesuai dengan apa yang diharapkan dalam proses penentuan rute terdekat.

4.3.1 Tampilan Hasil Pencarian Lokasi

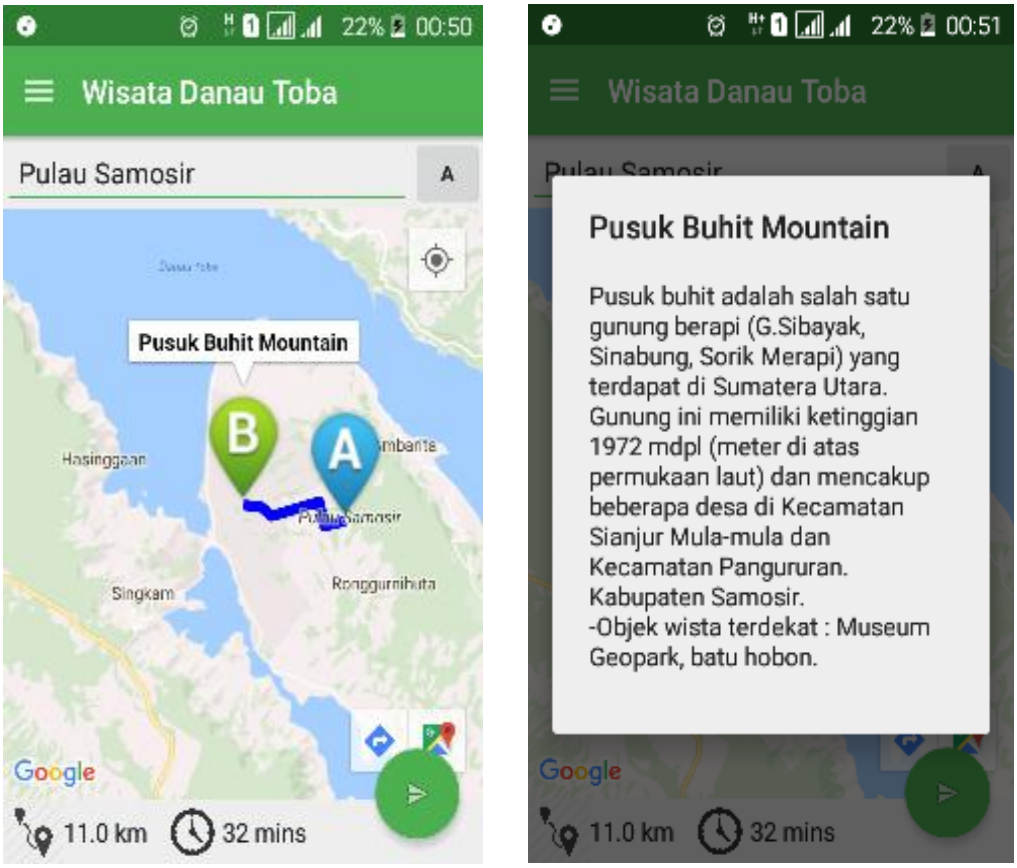
Hasil perhitungan dari algoritma *L-Deque* telah diimplementasikan kedalam sistem pada penelitian ini. Hasil yang ditampilkan sistem telah sesuai dengan perhitungan manual algoritma *L-Deque*, contohnya titik awal 'Pulau Samosir' kemudian klik *button* proses dimana nanti hasil akan ditampilkan berupa pilihan *list* semua rute terdekat objek wisata dari posisi titik awal 'Pulau Samosir', dapat dilihat pada Gambar 4.26.



Gambar 4.26 Tampilan Hasil Pengujian Menggunakan Algoritma L-Deque

4.3.2 Tampilan Pengujian sistem dari titik awal menuju destinasi objek wisata

Pengujian sistem ini dilakukan untuk membuktikan apakah dari titik awal lokasi ke tujuan destinasi objek wisata dapat dijalankan. Dari hasil pengujian yang sudah dilakukan maka titik koordinat yang sudah disimpan dapat bekerja dan bisa menuju lokasi yang akan dituju, dapat di lihat pada Gambar 4.27.



Gambar 4.27 Tampilan Pengujian Sistem

Pada gambar terlihat hasil rute untuk sampai ke tujuan. Dimana marker yang berwarna biru merupakan titik awal dan marker bewarna hijau merupakan titik yang akan dituju serta deskripsi singkat mengenai destinasi objek wisata.

4.3.4 Hasil Pengujian sistem

Hasil pengujian yang dilakukan berdasarkan perbandingan total jarak sebelum dan sesudah menggunakan algoritma *L-Deque* pada penentuan rute terdekat. Posisi titik awal yang digunakan adalah ‘Pulau Samosir’ atau *vertex* A sehingga total keseluruhan posisi titik tujuan adalah 16, dapat dilihat pada Tabel 4.3.

Tabel 4.3 Tabel Pengujian

Posisi Awal	Rute Terdekat Algoritma L-Deque	Jarak	Rute Awal	Jarak *(edges)*
Pulau Samosir	A – E – B	189	A – B	192
	A – E – K - C	131	A – C	134
	A – E - D	126	A – D	128
	A – E	11	A – E	11
	A – F	105	A – F	105
	A – G	37	A – G	37
	A – H	22	A – H	22
	A – E – I	20,8	A – I	27
	A – J	38	A – J	38
	A – E – K	29	A – K	32
	A – L	14	A – L	14
	A – M	26	A – M	26
	A – E – B – N	107	A – N	193
	A – E – O	135	A – O	137
	A – L – P	135	A – P	150
	A – E – Q	110	A – Q	113
Rata-rata		**77**		**84**

Berdasarkan hasil pada Tabel 4.3. maka tingkat efisiensi dalam proses penentuan rute terdekat dengan rata-rata 100%. Hasil dari nilai akurasi dapat dilihat pada persamaan 4.1.

$$Efisiensi = \frac{Total\ Jarak\ Rute\ Awal - Total\ Jarak\ \ Rute\ Algoritma\ L\text{-}Deque}{Total\ Jarak\ Rute\ Awal} \ x\ 100\% \qquad 4.1$$

$$= \frac{84 - 77}{84} \ x\ 100\ \%$$

$$= 8{,}33\ \%$$

Pada hasil persentase di atas diperoleh tingkat efisiensi sebesar 8,33% dalam penentuan rute terdekat dari posisi titik awal yang telah diuji.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan analisis, perancangan dan hasil implementasi algoritma *L-Deque* yang telah dilakukan penulis, maka dapat disimpulkan bahwa:

1. Algoritma *L-Deque* mampu menyelesaikan permasalahan penentuan rute objek wisata terdekat di kawasan Danau Toba.
2. Pada penerapan algoritma L-Deque mempermudah dalam penentuan rute terdekat karena ketika dari satu titik awal yang ditentukan bisa diperoleh semua titik tujuan yang ada.
3. Berdasarkan hasil pengujian dari algoritma *L-Deque* diperoleh tingkat efisiensi sistem sebesar 8,33 %.

5.2 Saran

Berikut ini beberapa saran yang dapat dipertimbangkan untuk memperbaiki penelitian ini dan untuk penelitian kedepannya, yaitu:

1. Penelitian selanjutnya diharapkan dapat menggunakan *graph* bersifat dinamis agar bisa menambahkan *vertex* baru.
2. Sistem ini sebaiknya diperluas untuk mencakup semua objek wisata yang berada di kawasan Danau Toba.
3. Penelitian selanjutnya sebaiknya mengkombinasikan dua atau lebih algoritma yang lain untuk lebih mengoptimalkan penentuan rute terdekat.

Daftar Pustaka

Chang, Kang Tsung. 2004, Introduction To Geographic Information System Second Edition, McGraw Hill, New York.

Gallo, G & Pallotino, S. 1986. Shortest Path Methods : A Unifying Approach. *Mathematical Programming Study* 26: 38-64.

Gufroni, A I. 2013. Implementasi Google Maps API Dalam Aplikasi Mobile Penghitung Jarak Aman Dari Dampak Kemungkinan Letusan Gunung Galunggung. Seminar Nasional Aplikasi Teknologi Informasi Yogyakarta pp 12-16.

Hayati, E.N. & Yohanes, A. 2014. Pencarian Rute Terpendek Menggunakan Algoritma *Greedy***.** *Prosiding Seminar Nasional IENACO 2014*, pp. 391-397.

Kundyanirum, Ambrina. 2013. Sistem Informasi Geografis Pariwisata Kota Semarang, Skripsi S-1 Universitas Diponegoro, Semarang.

Hadinoto, Kusudianto. 1996. Perencanaan Pengembangan Destinasi Pariwisata.Jakarta : UI Press.

Knuth, D.E., 1968. Fundamental algorithms, vol. 1. The Art of Computer. Vancouver.

Lubis, H. N. 2009. Perbandingan Algoritma *Greedy* dengan Algoritma *Dijkstra* untuk Menentukan Lintasan Terpendek. Skripsi. Universitas Sumatera Utara.

Masyunita. 2016. Aplikasi Pelayanan Sistem Informasi di Universitas Sumatera Utara (USU) berbasis Android menggunakan algoritma Bellman-Ford. Skripsi. Universitas Sumatera Utara.

Munir, R. 2014. *Matematika Diskrit*. Edisi Revisi Kelima. Informatika Bandung: Bandung.

Siregar, Natasha M. 2016. Analisis dan Perbandingan Algoritma L-Deque dan Algoritma Bellman-Ford dalam Mencari Jarak Terpendek. Skripsi. Universitas Sumatera Utara.

Pinem, A. 2014. Analisis dan Perbandingan Algoritma A* Dan Kombinasi Floyd-Warshall Dengan Bryte Force Pada Penentuan Jalur Monitoring Di Badan Penanaman Modal Kota Medan. Skripsi. Universitas Sumatera Utara.

Sumaja, L G. 2013. Lembaga Pengesahan Sistem Informasi Geografis (SIG) Pencarian Letak posisi Ruang Perkuliahan di Universitas Widyatama. Teknik Sistem informasi Universitas Widyatama.

Utami, E. dan R. Suwanto, 2004. Logika Algoritma dan Implementasi Dalam Bahasa Python Di GNU/Linux. Yogyakarta: Andi.

Vandia, T. A.2016. Perencanaan Perjalanan Wisata dengan Metode *Greedy*. Skripsi. Universitas Sumatera Utara.

Whitten, J.L. & Bentley, L.D. 2007. System Analysis and Design Methods. 7th Edition. McGraw-Hill/Irwin: New York.

Yuhana, U L. 2010. Pemanfaatan Google Maps Untuk Pemetaan Dan Pencarian Data Perguruan Tinggi Negeri Di Indonesia. Teknik Informatika Institut Teknologi Sepuluh November.

Zainuddin, 2013. Perancangan SIG berbasis Web Objek Wisata Kota Binjai dengan Algoritma A*.

Ziad, Ibnu. 2013. Rancang Bangun Pelacak Lokasi Dengan Teknologi GPS. Jurnal Teknologi Dan Informatika vol 3 no 1.