Pencarian Pom Bensin Terdekat di Denpasar Menggunakan Algoritma Djikstra Berbasis Web Mobile

Putu Andhika Kurniawijaya
Teknik Informatika, Fakultas Teknik, Universitas Kristen Duta Wacana, Yogyakarta
andhika_mylo@yahoo.com

ABSTRACT

Denpasar as one of travel destination in Indonesia, is often visited by local and foreign tourists. Even some of those tourist who traveled in Denpasar is unaccompanied by a guide. Regional and environmental differences could be problems for tourists who do not recognize Denpasar areas very well. They are confused to find gas stations when needed.
The author uses Dijkstra's algorithm method,which purpose is to determine nearest gas station to the tourists. The initial step in this method is to give the weight value (distance) from one point to another point, then gives a value of 0 at the starting point and infinite value to the other point.
From the point of departure, the algorithm compare the unidentified neighboring point and count the distance from the point of departure.The smallest value of the destination point is the smallest weight of the starting point to the point of destination. By using J2ME search results layout nearest gas station can be displayed on a mobile traveler.

Based on the analysis of several trials of the method of Dijkstra's algorithm, it can be determined that the method can provide the right solution in the search for the nearest gas station because it can provide quick results according to the needs of travelers. In addition, the search results can be displayed on mobile travelers, making it easier to search.
***Keywords** : Djikstra's Algorythm, Web Mobile.*

ABSTRAK

Denpasar sebagai salah satu daerah wisata di Indonesia, tentu sering dikunjungi oleh wisatawan lokal maupun asing. Bahkan tidak sedikit dari mereka yang berkeliling tanpa ditemani oleh seorang pemandu. Perbedaan daerah dan lingkungan dapat menjadi hambatan bagi wisatawan yang belum mengenali daerah Denpasar dengan baik. Seringkali mereka kebingungan untuk mencari tempat pengisian bensin pada saat diperlukan.
Penulis menggunakan Metode Algoritma Dijkstra yang bertujuan untuk menentukan pom bensin yang berada paling dekat dengan wisatawan.Langkah awal dalam metode ini adalah memberi nilai bobot (jarak) dari setiap titik ke titik lainnya, kemudianmemberi nilai 0 pada titik awal dan nilai tak hingga terhadap titik lain.

Nilai terkecil dari titik tujuan tersebut merupakan bobot terkecil dari titik awal menuju titik tujuan.Dengan menggunakan J2ME hasil pencarian letak pom bensin terdekat tersebut dapat ditampilkan pada *handphone* wisatawan. Berdasarkan hasil analisis dari beberapa percobaan terhadap Metode Algoritma Dijkstra, dapat diten-tukan bahwa metode tersebut dapat memberikan solusi yang tepat dalam pencarian pom bensin ter-dekat karena dapat memberikan hasil yang cepat sesuai dengan kebutuhan wisatawan. Selain itu, hasil pencarian dapat ditampilkan pada *handphone* wisatawan, sehingga mempermudah dalam pencarian.
Kata Kunci : Algoritma Djikstra***, Web Mobile.***

PENDAHULUAN

Denpasar merupakan daerah wisata yang sering dikunjungi oleh para wisatawan lokal maupun wisatawan asing. Sering kali wisa-tawan menggunakan transportasi sendiri untuk berwisata ke tempat-tempat wisata yang ada di Denpasar. Wisatawan yang membawa transportasi sendiri ini seringkali kebingungan bila ingin mengisi bahan bakar bila bahan bakar telah habis. Untuk meng-hemat waktu agar tidak tersesat se-waktu mereka mencari pom bensin maka dapat meng-gunakan sistem pencarian pom bensin terdekat.

Pencarian pom bensin terdekat adalah pencarian lokasi pom bensin yang terdekat dari suatu titik awal yang telah dimasukkan oleh *user*. Dalam penelitian ini, sistem akan memberikan informasi tentang lokasi pom bensin yang terdekat dari data yang telah di-inputkan *user* yang mempunyai sekian ba-nyak jalur yang dapat ditempuh oleh *user* dan berbasis *web mobile*. Agar sistem dapat be-kerja, sistem ini harus diberikan kecer-dasan untuk dapat mencari posisi pom ben-sin yang dituju. Penyelesaikan kasus ini tidak hanya dibutuhkan kecerdasan yang digunakan untuk mencari posisi dari obyek tujuan namun ha-rus diperhatikan juga langkah yang digu-nakan apakah sudah optimal atau belum.

Permasalahan mendorong penulis un-tuk melakukan penelitian mengenai imple-mentasi metode Dijkstra sebagai sebuah me-tode pencarian jalur terpendek untuk mencari so-lusi dari permasalahan diatas. Pemilihan metode Dijkstra ini, disebabkan karena wak-tu yang diperlukan tidak terlalu banyak dan hanya menghasilkan satu output yang dapat menyelesaikan masalah di atas.

TINJAUAN PUSTAKA

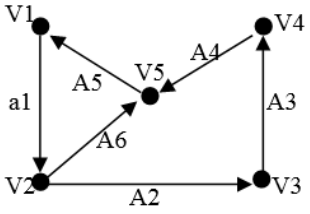
Teori Graf

Teori graf pertama kali muncul dan di-ilhami oleh penemuan Leonhard Euler dalam tulisannya yang membahas mengenai *Seven Bridges of Koningsberg*. Dan sampai saat ini penemuannya lebih dikenal sebagai graf euler. Graf merupakan kumpulan sejumlah *vertex* (node) dan *edges* (garis) yang saling ber-hubungan. Dalam graf dikenal 2 buah jenis graf yaitu graf berarah (*Directed Graph*) dan graf tidak berarah (*Undirected Graph*).

Secara nyata graf dapat di imple-mentasikan ke berbagai macam perma-salahan. Seperti contoh penggambaran hubu-ngan relasi, pencocokan data, pencarian jalur terpendek, penjadwalan dan masih banyak lagi peman-faatan dari teori graf yang dapat dikem-bangkan berdasarkan masalah yang diha-dapi**.**

Graph Berarah (*directed graph* atau *digraph*)

Pada *Graph* tak berarah (*undirected graph*) elemen dari E disebut dengan *edge*, sedang-kan pada *graph* berarah (*directed graph*) ele-men dari E(A) disebut dengan *arc*. *Graph* berarah G terdiri dari suatu himpunan V dari verteks – verteks dan suatu himpunan E(A) dari *arc* sedemikian rupa sehingga setiap *arc a* ϵ A menghubungkan pasangan verteks terurut. Jika terdapat sebuah *arc a* yang menghubungkan pasangan terurut (*v,w*) dari verteks – verteks, maka dapat ditulis dengan a=(*v,w*), yang menyatakan sebuah *arc* dari *v* ke *w*.



Gambar 1. Graph Berarah 1

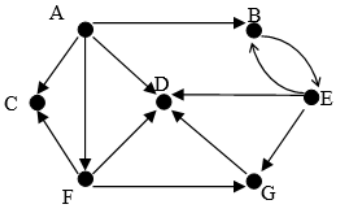
Graph pada Gambar 2.1 adalah *graph* berarah dengan himpunan verteks – verteks sebagai berikut V(G)={v_1,v_2,v_3.v_4.v_5} dan himpunan *arc* – *arc* sebagai berikut A(G) = {a_1 ,a_2 ,a_3 ,a_4 ,a_5 ,a_6} yaitu pasangan terurut dari {(v_1,v_2), (v_2,v_3), (v_3,v_4), (v_4,v_5), (v_5,v_1), (v_2,v_5)}.

Pada suatu *graph* jika dua buah verteks v_1 dan v_2 dihubungkan oleh suatu *edge (arc)*, maka kedua verteks tersebut dikatakan *adjacent*. Pada Gambar 1 verteks v_1 *adjacent* (bertangga) dengan verteks v_2. Sementara itu, *arc* a_1 dikatakan *incident* (bersisian) de-ngan verteks v_1 dan verteks v_2. Matriks yang bersesuaian dengan *graph* berlabel G adalah matriks *adjacency* A=(a_{ij}), dengan a_{ij} = nilai *arc* yang menghubungkan verteks v_i dengan verteks v_j. Jika titik v_i tidak berhubungan langsung dengan titik v_j, maka $a_{ij} = \infty$, dan jika i = j, maka $a_{ij} = 0$. Misalkan *G* adalah sebuah *graph* berarah. Sebuah *arc* berarah *a* = [*u,v*] dikatakan mulai pada verteks awal *u* dan berakhir di verteks akhir *v*,*u* dan *v* dikatakan *adjacent*. Derajat luar dari *v*, (*outdeg* (*v*)) adalah jumlah *arc* yang berawal pada *v*, dan derajat dalam dari *v* (*indeg* (*v*)) adalah jumlah *arc* yang berakhir

di *v*. Karena setiap *arc* mulai dan berakhir pada suatu verteks, maka jumlah derajat dalam dan jumlah derajat – luar harus sama dengan *n*, yaitu jumlah *arc* pada G. Sumber (*source*) adalah sebuah verteks v di G yang mem-punyai derajat-luar positif dan derajat-dalam nol. Sedangkan, tujuan (*sink*) adalah verteks v di G yang mempunyai derajat-dalam positif tetapi derajat-luar nol.

Verteks	A	B	C	D	E	F	G
Derajat-dalam (indegree)	0	2	2	4	1	1	2
Derajat-luar (outdegree)	4	1	0	0	3	3	1

Tabel 1. Derajat Positif Luar Nol

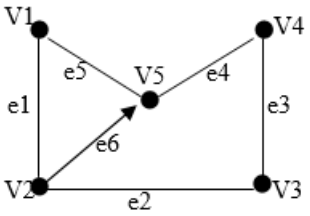


Gambar 2. Graph Berarah 2

Gambar 2.2 terdiri dari : Jumlah derajat dalam dan jumlah derajat luar sama dengan 12 yaitu jumlah busur.
Graph pada Gambar 2. verteks *A* adalah sumber (*source*) karena *arc*-nya berawal pada *A* tetapi tidak berakhir di *A*. Sedangkan *C* dan *D* adalah verteks tujuan (*sink*) karena arc-nya berakhir di *C* dan di *D* tetapi tidak berawal di verteks itu.

Graph Tak Berarah (*Undirected Graph*)

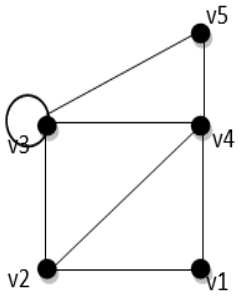
Graph tak berarah G terdir dari suatu himpu-nan V dari verteks – verteks dan suatu himpunan E dari *edge* – *edge* sedemikian rupa sehingga setiap sisi $e \in E$ dikaitkan dengan pasangan verteks tak terurut.
Jika terdapat sebuah *edge e* yang meng-hubungkan verteks *v* dan *w*, maka dapat dituliskan dengan *e* (*v*,*w*) atau *e* = (*w*,*v*) yang menyatakan sebuah *edge* antara *v* dan *w*.
Graph pada Gambar adalah *graph* tak ber-arah dengan himpunan verteks – verteks V(G) = { v_1 ,v_2 ,v_3 ,v_4 ,v_5} dan himpunan sisi E(G) = { e_1 ,e_2 ,e_3 ,e_4, e_5 ,e_6} yaitu pasangan tak terurut dari { (v_1,v_2), (v_2,v_3), (v_3,v_4), (v_4,v_5), (v_5,v_2) }.



Gambar 3. Graph Tak Berarah 1

Pengenalan *Walk* (jalan), *Path* (lintasan) dan *Cycle* (Untai)

Walk (jalan) didefinisikan sebagai urutan bolak–balik dari *verteks* dan *edge* yang di-mulai dan diakhiri *verteks* sedemikian rupa, sehingga setiap *edge* berpengaruh dengan terhadap *verteks* yang mendahuluinya dan mengikuti. Tidak ada *edge* yang muncul (dilingkupi dan dilalui) lebih dari sekali jalan, sedang *verteks* dapat dilalui lebih dari satu kali. Contoh : v1 a v2 b v3 c v3 d v4 e v2 f v5 merupakan jalan (*walk*) dari graph berikut ini :



Gambar 4. Graph Tak Berarah 2

Jika semua *edge* dari jalan tersebut berbeda maka jalan tersebut dinamakan trail. Bila terdapat tambahan syarat bahwa semua *ver-teks* berbeda maka trail tersebut disebut *path* (lintasan). Dari gambar di atas dapat diten-tukan path = *v1 a v2 b v3 d v4 h v5*, sedangkan *v1 a v2 b v3 c v3 d v4 f v5* merupakan bukan path karena *verteks v3* muncul dua kali.

Jadi suatu lintasan (*path* atau *simple path* atau *elemen-tary path*) merupakan jalan terbuka dimana semua *verteks* dan *edge*nya berbeda juga tidak ada *verteks* yang muncul lebih dari satu kali. Sebagai catatan jalan (walk) boleh memakai selfloop, tapi tidak untuk path. Penomoran dari *edge* dalam path dinamakan panjang dari path. Jalan tertutup dengan semua *edge*_nya berbeda disebut trail tertutup, jika ditambahkan syarat bahwa semua *verteks* berbeda maka disebut untai (*cycle*).

Graph Berbobot (*weight graph*)

Graph G dikatakan berbobot (*weigthed*) bila terdapat bilangan riil yang berasosiasi dengan masing – masing *edge* G. Bobot – bobot itu sendiri biasanya berfungsi pada saat dibuat suatu lintasan antar *verteks* – *verteks* pada graph tersebut. Misalnya, jika suatu graph G menggambarkan peta jalan kota Denpasar maka *verteks* dari graph G menggambarkan kota tersebut, *edge* menggambarkan jalan, dan bobot menggambarkan jarak/panjang jalan tersebut. Jika dibuat suatu lintasan dari satu titik ke titik lainnya, baik melewati titik – titik lainnya maupun tidak, maka jumlah bobot dari semua *edge* yang dilewati lintasan tersebut dari titik asal ke titik tujuan.

Pohon Bentangan Jarak Terpendek Pada Suatu Graph Berbobot (*Minimum Distance Spanning Tree*)

Bobot pohon bentangan T dari G didefinisikan sebagai jumlah bobot dari semua cabang di T. Diantaranya semua pohon bentangan di G, pohon bentangan dengan bobot terkecil disebut pohon bentangan terpendek (*shortest spanning tree* atau minimal *distance spanning tree*).

Masalah Lintasan Terpendek (Shortest Path)

Suatu metode adalah pada dasarnya suatu resep atau cara untuk menyelesaikan ma-salah matematis tertentu. Terdiri dari bebe- rapa instruksi yang diikuti langkah demi langkah akan membawa pada solusi ma-salah. Setiap langkah pada suatu metode harus didefinisikan secara tepat dan jelas, metode harus berakhir setelah menyelesaikan masalah yang diberikan dalam sejumlah langkah yang terbatas. Ada beberapa macam tipe masalah pencarian lintasan terpendek yang paling sering ditemukan adalah kelima masalah dibawah ini :

(1) Lintasan terpendek antara dua pasang *verteks* terspesifikasi. (2) Lintasan terpendek antara semua pasa-ngan *verteks*. (3) Lintasan terpendek dari suatu *verteks* terspesifikasi ke semua *verteks* lainnya. (4) Lintasan terpendek antara *verteks* terspesifikasi yang melewati *verteks* – *verteks* tertentu. (5) Lintasan terpendek ke-dua, ketiga, dan seterusnya dari masalah lintasan terpendek. Pada kasus terburuk, masalah tipe satu bisa menjadi sama dengan masalah tipe tiga, karena mungkin saja dalam mencari lintasan ter-pendek antara dua *verteks* tertentu mungkin saja terlewati dan ditemukan lintasan ter-pendek ke semua *verteks* lainnya. Dari kelima macam tipe masalah pencarian lin-tasan terpendek diatas, yang paling dekat dengan masalah yang ditemui, yaitu mencari jalur terpendek antara *verteks* – *verteks* ter-spesifikasi yang melalui *verteks* lainnya adalah tipe masalah keempat.

Metode Dijkstra

Metode Dijkstra ditemukan oleh Edsger W. Dijkstra yang merupakan salah satu varian bentuk metode popular dalam pemecahan persoalan yang terkait dengan masalah optimasi dan bersifat sederhana. Metode ini menyelesaikan masalah mencari sebuah lintasan terpendek dari verteks a ke verteks z dalam *graph* berbobot, bobot tersebut adalah bilangan positif jadi tidak dapat dilalui oleh node negatis, namun jika terjadi demikian, maka penyelesaian yang diberikan adalah *infiniti*.

Metode Dijkstra melibatkan pemasangan label pada verteks. Misalkan L(v) menya-takan label dari vertex v. Pada setiap pem-bahasan, beberapa verteks mempunyai label sementara dan yang lain mempunyai label tetap. Misalkan T menyatakan himpunan verteks yang mempunyai label sementara. Dalam menggambarkan Metode tersebut verteks – verteks yang mempunyai label tetap akan dilingkari. Selanjutnya, jika L(v) adalah label tetap dari verteks v, maka L(v) merupakan panjang lintasan terpendek dari a ke v. Sebelumnya semua verteks mempunyai label sementara. Setiap iterasi dari Metode tersebut mengubah status satu tabel dari sementara ke tetap, sehingga Me-tode dapat berakhir ketika z menerima se-buah label tetap. Pada bagian ini L(z) merupakan panjang lintasan terpendek dari a ke z. Pada metode Dijkstra node digunakan, karena Metode Dijkstra menggunakan diagram pohon (*tree*) untuk penentuan jalur lintasan terpendek dan menggunakan *graph* yang berarah.

Cara Kerja Metode Dijkstra

Pencarian jalur terpendek dengan menggunakan metode Dijkstra ini merupakan pencarian jarak terpendek diantara 2 *verteks* dari sebuah *graph* berbobot. Dimana jarak terpendek dihitung berdasar-kan kedekatan setiap *verteks* dengan *verteks* yang lainnya. Program dibuat menyerupai rencana yang dibuat untuk memadu pengemudi mobil untuk mencapai tujuannya yaitu pom bensin.

Sebuah kota disimbulkan sebagai titik (*verteks*) dan tiap–tiap jarak antar *verteks* digambarkan sebagai garis (*edge*) yang mempunyai nilai. Sehingga jika antar *verteks* dihubungkan akan terbentuk suatu graph, dan dari graph itu ditentukan jalur (*Path*) yang terpendek. Untuk memulai mencari ja-lur terpendek harus di-tentukan *verteks* awal (asal) dan *verteks* akhir (tujuan). Secara singkat metode tersebut dapat dijelaskan sebagai berikut : (1) Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada titik awal dan nilai tak hingga terhadap titik lain (belum terisi) ubah semua titik "Belum terjamah" dan ubah titik awal sebagai "Titik keberangkatan". (2) Dari titik keberangkatan, pertimbangkan titik tetangga yang belum terjamah dan hitung jaraknya dari titik keberangkatan. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru. (3) Saat kita selesai mempertimbangkan setiap jarak terhadap titik tetangga, tandai titik yang telah terjamah sebagai "Ti-tik terjamah". Titik terjamah tidak akan per-nah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling mi-nimal bobotnya. (4) Set "Titik belum ter-jamah" dengan jarak terkecil (dari titik keberangkatan) sebagai "Titik Keberangkatan" selanjutnya dan lanjutkan dengan kembali ke step 3.

Java

Java merupakan bahasa pemrograman yang dikembangkan oleh Sun Microsystem. Dari pertama diluncurkannya, java memiliki sem-boyan *write one run everywhere*. Java mem-punyai beberapa keunggulan, antara lain (1) Sederhana, Bahasa pemrograman Java mudah dipahami dan dipelajari. (2) Berorientasi Obyek, Java berbasis pada pemrograman berorientasi obyek (3) Kuat dan Aman, Java banyak menekankan pada pengecekan awal untuk kemungkinan terjadinya masalah, pengecekan pada saat runtime dan mengurangi timbulnya kesalahan. Java memiliki pengaturan yang baik khususnya pada masalah *buffer overflow* yang umumnya menjadi lubang keamanan. (4) Arsitektur Netral dan Portabel, Java dirancang untuk dapat dijalankan pada semua platform. Kompiler Java yang digu-nakan untuk meng-kompilasi kode pro-gram Java dirancang untuk menghasilkan kode yang netral terhadap semua arsitektur pe-rangkat keras yang disebut dengan Java *bytecode*.

Bahasa Java merupakan bahasa yang cocok untuk *wireless device*. Hal ini disebabkan adanya keuntungan yang disediakan oleh Java platform antara lain, (1) Security: ada-nya verifikasi file *Class* dan interface program aplikasi yang terdefinisi menye-babkan *third party application* tahan uji dan tidak membahayakan *device* ataupun jari-ngan. (2) *Cross platform compatibility*: apli-kasi dapat ditransfer secara fleksibel antara *device* yang berbeda. (3) *Offline acces* : apli-kasi tetap dapat digunakan walau tanpa koneksi. Hal ini menurunkan biaya dan mengurangi kesalahan jaringan. (4) *Large developer community* : saat ini diperkirakan lebih dari 2,5 juta pengembang *software* Java di dunia.

Pada perkembangan selanjutnya Sun Micro system memperkenalkan Java versi 1.2 atau lebih dikenal dengan Java2. Java2 terdiri atas JDK dan JRE. Java2 ini dibagi menjadi 3 kategori yaitu: (1) J2SE (*Java 2 Standard Edition*) Java kategori ini digunakan untuk menjalankan dan mengembangkan aplikasi – aplikasi java pada level PC (*Personal Computer*). (2) J2EE (*Java 2 Enterprise Edition*) Java kategori ini digunakan untuk menjalankan dan mengembangkan aplikasi – aplikasi Java pada lingkungan enterprise, dengan menambah fitur-fitur Java seperti EJB (*Enterprise Java Bean*), Java CORBA, Servlet dan JSP, serta Java XML (*Extensible Markup Language*). (3) J2ME (*Java 2 Micro Editon*) Pada kategori ini Java digunakan untuk menjalankan dan mengembangkan aplikasi – aplikasi Java pada *mobile device* semacam *handphone*, Palm, PDA, dan *Pocket* PC.

J2ME

Java Micro Edition (j2me) merupakan salah satu bagian dari Java yang dikembangkan untuk pembuatan program yang dapat di-jalankan pada perangkat – perangkat *mobile* seperti Palm, Pocket PC, *Handphone* yang mendukung Java. Pada J2ME dibagi menjadi 2 bagian yaitu J2ME *Configuration* dan J2ME *Profile*.

J2ME Configuration

Karena banyaknya jenis perangkat mobile dan memiliki fitur–fitur yang berbeda–beda, maka J2ME Configuration menyediakan fungsi standar yang mengimplementasikan fitur standar yang dimiliki oleh sebuah pe-rangkat *mobile*. J2ME Configuration memastikan terdapat keseragaman karakteristik Java development en-

vironment yang ter-dapat pada semua jenis Java enabled devices yang memiliki kesa-maan dalam spesifikasi sistem, meliputi kemampuan prosesor dan kapasitas memori.

Ada 2 kategori pada J2ME configuration berdasarkan dari kapasitas memori dan prosesor, yaitu : (1) CDC (*Connected Device Configuration*) Kategori ini digunakan untuk aplikasi Java pada piranti *handled devices* yang memiliki kapasitas memori minimal 2 *megabytes*. Contohnya Internet TV, Car Television dan Nokia *Communicator*. (2) CLDC (*Connected Limited Divice Configuration*) Kategori ini digunakan untuk aplikasi Java pada PDA dan *handphone* yang mendukung teknologi java, seperti Nokia, Siemens, Samsung. Pada umumnya, semua perangkat tersebut memiliki kapasitas memori 160–512 Kilo-bytes.

J2ME Profile

J2ME Profile mendefinisikan tambahan kumpulan fungsi yang bersifat lebih spesifik dari sebuah perangkat *handled devices*. Sebagai contoh sebuah handphone memiliki kemampuan untuk menelpon ke nomor handphone lainnya, ini merupakan fungsi standar yang dimiliki oleh handphone yang diimplementasikan oleh J2ME Configuration. Selain memiliki fungsi–fungsi standar, handphone memiliki fungsi khusus yang hanya dimiliki oleh handphone tertentu, sebagai contoh Siemens SL45 dapat men-jalankan file MP3. Nokia dengan game snake yang menjadi trademarknya. Fitur– fitur tersebut tergantung pada jenis handheld devices dan akan dimplementasikan oleh J2ME Profile.
Ada 5 kategori pada J2ME Profile, yaitu : (1) Mobile Information Device Profile (MIDP) ; *CLDC Profilie MIDP* yang di-rancang khusus untuk *wireless phone* dan *two way pager.*MIDP menyediakan librari– librari Java : Antarmuka (GUI) Librari ini menye-diakan fungsi–fungsi yang berhu-bungan dengan antarmuka dari handheld device seperti Screen, Item, Canvas dan lain–lain; Jaringan (Networking), Librari ini menye-diakan fungsi database sederhana yang dapat melakukan penambahan data, peng-hapusan data serta pengeditan data; Record (Records Management System), Librari ini menye-diakan fungsi database sederhana yang dapat melakukan penambahan data, penghapusan data serta pengeditan; Data Timer *Library* ini menye-diakan fungsi waktu. (2) Foundation Profile (FP); CDC profile yang dikhususkan untuk digunakan oleh device yang membutuhkan implement-tasi yang menye-luruh dari Java virtual machine dan termasuk di dalamnya API dan J2SE. (3) Personel Profile, CDC profile yang dikembangkan oleh Sun's Personel Java Environtment. Profile ini menye-diakan kompatibilitas bagi perkem-bangan aplikasi untuk versi 1.1.x dan versi 1.2.x. (4) RMI Profile, CDC profile yang mendefinisikan API dari J2SE RMI. (5) Personel Digital Assitance Profile (6) CLDC profile yang didesain khusus untuk handheld device yang dikeluarkan oleh Palm OS seperti Palm Pilots dan Visor.

PERANCANGAN SISTEM

Pencarian pom bensin terdekat di Denpasar menggunakan algoritma Dijkstra berbasis *web mobile* adalah sebuah aplikasi dimana aplikasi akan mencari posisi pom bensin yang terdekat dengan posisi *user* yang telah di-*input*kan *user*. Langkah awal program adalah user memasukkan nama jalan dimana *user* berada dan memilih sub jalan untuk menentukan titik dari sub jalan yang sebelumnya telah dipilih oleh *user*.

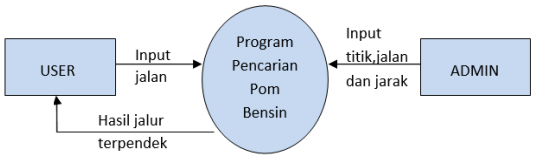
Selanjutnya, program akan memproses menggunakan metode Dijkstra dimana semua titik pom bensin akan dicari jalur terpendek dari masing–masing pom bensin. Hasil dari masing–masing jalur pom bensin akan dibandingkan dan hasil jalur yang terkecil yang akan dijadikan jalur terpendek referensi pom bensin. Untuk mempermudah *user*, aplikasi akan memberikan *highlight* berwarna merah untuk jalur–jalur yang akan dilewati oleh *user*.

Perancangan Basis Data

***Data Flow Diagram* (DFD)**

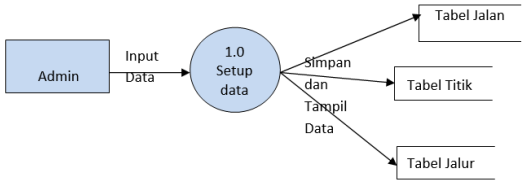
DFD digunakan untuk menggambarkan asal aliran informasi yang terlibat dalam suatu *link* sistem yang sering disebut sebagai *event*, kemana tujuan data ke-luaran dan dimana data disimpan. DFD terbagi menjadi beberapa level. Level 0 menggambarkan sistem secara umum. Level 1 merupakan penurunan dari Level 0. Level 1 menggambarkan proses-proses yang terjadi dalam sistem secara lebih detail. Semakin tinggi level DFD, maka proses yang digambarkan semakin detail. Pada penulisan tugas akhir ini, penulis menggambarkan sistem sampai level 2, yaitu Level 0 yang terlihat pada Gambar 3.1, Level 1 yang terlihat pada Gambar 3.2 dan Level 2 yang terlihat pada Gambar 3.3. Kedua *DFD leveled* tersebut adalah sebagai berikut:

- **DFD level 0**



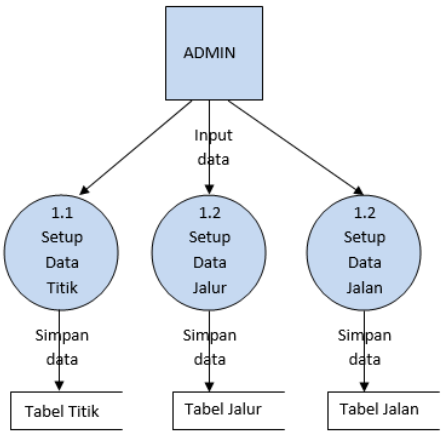
Gambar 5. DFD level

- **DFD level 1**



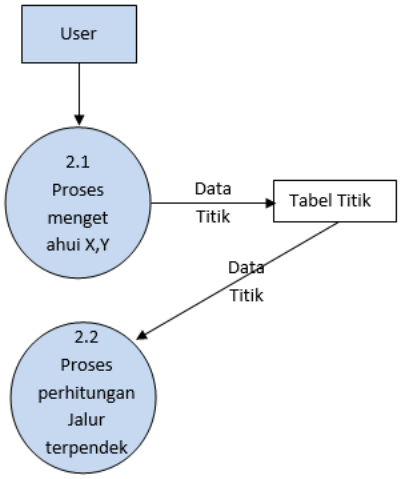
Gambar 6. DFD level 1

- **DFD level 2.1**



Gambar 7. DFD level 2.1

- **DFD level 2.2**



Gambar 8. DFD level 2.1

Tabel 3.1 Tabel titik

Nama Field	Tipe	Ukuran	Keterangan
Id_titik*	Int	11	Nomer ID
Titik	Char	255	Nama titik
Sumbux	Int	11	Koordinat X
Sumbuy	Int	11	Koordinat Y
Status	Boolean		Status Pom Bensin

Tabel 3.2 Tabel jalur

Nama Field	Tipe	Ukuran	Keterangan
Id_jalur*	Int	11	Nomer jalur
Id_titik1	Int	11	Koordinat titik1
Id_titik2	Int	11	Koordinat titik2
jarak	Int	11	Jarak
Keterangan	Char	255	Keterangan pom bensin

Tabel 3.3 Tabel Jalan

Nama Field	Tipe	Ukuran	Keterangan
Id_jalan	Int	11	Id jalan
Nama_jalan	varchar	40	Nama jalan

Perancangan Database

Setelah menyusun DFD, struktur *database* yang bagaimana yang akan dipakai se-hubungan dengan rancangan DFD yang telah dibuat. Dalam hal merancang struktur tabel, yang pertama harus dilakukan adalah menentukan field–field yang akan digunakan pada masing–masing tabel yang telah di-rancang pada DFD sebelumnya dan se-lanjutnya harus menentukan field–field apa yang merupakan primary key.

Perancangan Antar Muka

Perancangan antar muka yang akan dibuat pada sistem ini dibagi menjadi 2 yaitu perancangan *input* dan perancangan *output.*

Perancangan Input

Perancangan antarmuka pengguna be-risi rancangan form yang akan digunakan sebagai Penghubung antara program bantu dengan pemakai.

Form Menu Pada Dekstop

Form ini hanya dapat di akses oleh admin, di form ini admin dapat memasukkan semua data yaitu menginputkan sumbu x dan sum-bu y jalan, memasukkan besaran jarak antar titik dan menempatkan posisi pom bensin. Tujuan pembuatan form ini agar memper-mudah admin un-tuk menginputkan data dan melakukan perubahan dari tiap data yang ada. Pada form ini terdiri atas titik,jalur dan peta.

Form Titik Pada Dekstop

Form ini digunakan untuk memasuk-kan sumbu x dan sumbu y. Admin juga dapat mengubah maupun menghapus data yang sudah ada pada form ini. Form titik pada menu admin dapat dilihat pada gambar berikut :

Gambar 9. Form titik pada menu titik

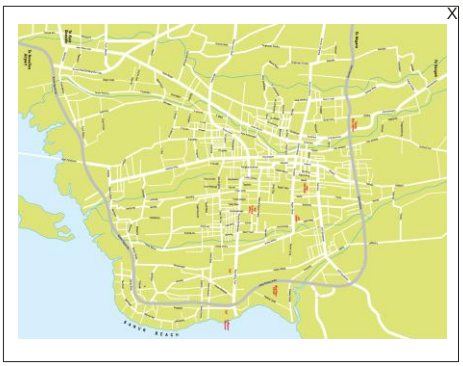
Form Jalan Pada Dekstop

Form ini digunakan untuk mencatat masukan nama – nama jalan yang ada di peta. Admin juga dapat mengubah dan menghapus data yang sudah ada pada form ini. Form jalur pada menu admin dapat dilihat pada gambar berikut ini :

Gambar 10. Form titik pada menu titik

Form Peta Pada Dekstop

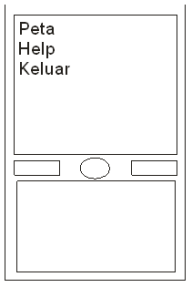
Form ini digunakan untuk mempermu-dah admin dalam menentukan titik – titik pa-da jalan yang ada dalam peta. Form peta pa-da menu admin dapat dilihat pada gambar berikut ini :



Gambar 11. Form Peta Pada Menu Admin

Form Menu Pada Handphone

Form utama adalah form yang akan muncul pada saat *user* pertama kali men-jalankan program. Di dalam form ini ter-dapat menu berupa peta, bantuan dan keluar.



Gambar 12.
Form Menu Pada Handphone

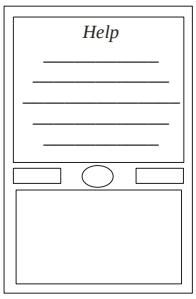
Form Peta Pada Handphone

Form peta adalah form yang berisi gambar peta dimana terdapat 2 pilihan yaitu Pilih Jalan dan Pilih Sub Jalan.

Gambar 13.
Form Peta Pada Handphone

Form Bantuan Pada Handphone

Form bantuan adalah form yang menuliskan langkah-langkah menggunakan program bantu ini untuk mempermudah pengguna menggunakan sistem tersebut.



Gambar 14. Form Bantuan

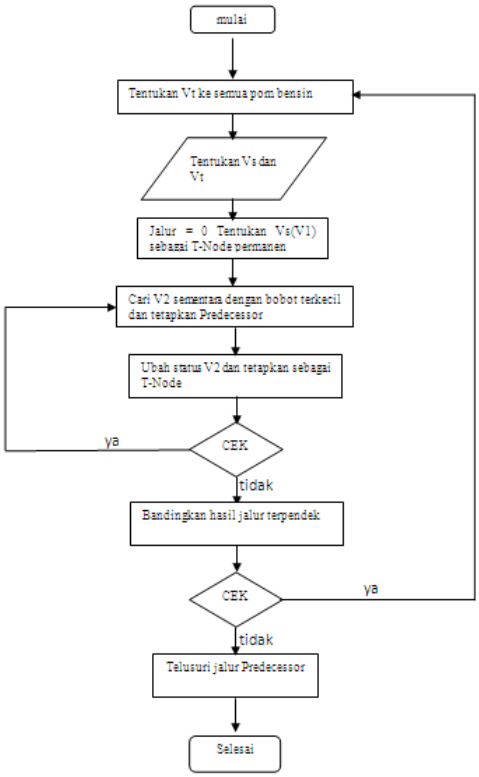
Perancangan Output

Output yang dikeluarkan oleh sistem berupa tampilan jalur pada peta yang telah di *highlight* untuk mempermudah user dalam melihat jalur yang pom bensin yang terdekat.

Gambar 15. Form Output

Perancangan Proses

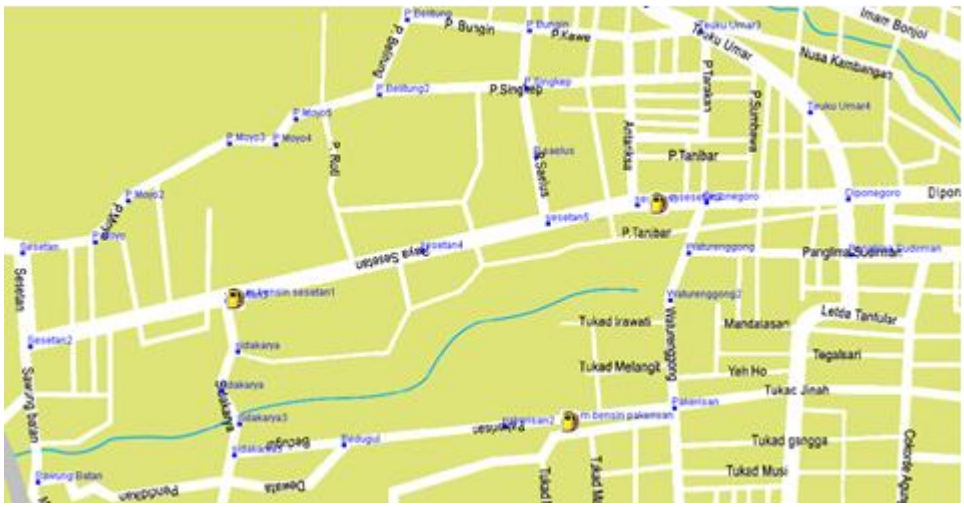
Diagram alir pada Gambar 16 menjelaskan alur dari cara kerja sistem yang akan dibuat. Setelah *user* memilih titik subjalan sesuai dengan posisi saat itu, sistem akan menyimpan titik–titik koordinat pilihan *user*. Kemudian akan diproses sebagai berikut :



Gambar 16. Diagram Alir
Metode Djikstra

HASIL UJI COBA

Contoh Penerapan Metode DIJKSTRA



Gambar 13 Gambar Contoh Graph untuk Kasus Perhitungan

Seorang wisatawan yang sedang berlibur di Bali kebingungan untuk mengisi bensin ken-daraan mereka dalam keadaan dan situasi yang cukup genting karena bensin dalam mobil akan segera habis. Diketahui suatu graph jalan yang akan dilalui wisatawan tersebut sebagai berikut :

Dari Graph diatas, misalkan wisatawan tersebut sedang berada di titik P.Bungin maka sesuai dengan metode yang ada dapat dijelaskan proses pencarian dan perhitungan jalur terpendek ter-sebut berikut ini : (1) Inisialisasi *verteks* : verteks awal = P.Bungin; (2) Inisialisasi *verteks* awal = P.Bungin dan *verteks* tujuan ke 42 *verteks* pom bensin. (3) Sistem akan melakukan pengecekan pada pom bensin pertama yaitu pom sesetan2 dan akan dilanjutkan ke pom bensin berikutnya. (4) Berikan label permanen untuk *verteks* awal (P.Bungin) = 0 dan label sementara ∞ ke *verteks* – *verteks* lainnya; 5) Titik awal mulai adalah P.Bungin yang pertama ber-nilai 0, kemudian dijkstra melakukan kal-kulasi terhadap titik tetangga yang ter-hubung langsung dengan titik P.Bungin dan hasil yang didapat adalah titik P.Belitung; (6) L(P.Belitung) = min {∞,(0+1254)} = 1254; (7) L(P.Singkep) = min{∞, (0+671)} = 671.

(8) Titik P.Singkep diset sebagai titik keberangkatan karena mem-punyai nilai terkecil. Dijkstra melakukan kal-kulasi kembali terhadap titik tetangga yang terhubung dengan titik P.Singkep. Dan kalkulasi dijkstra menemukan titik P.Belitung2 dan P.Saelus. (9) L(P.Belitung2) = min{∞, (671+1371)} = 2042; (10) L(P.Saelus) = min {∞, (671 + 502)} = 1173; (11) Titik P.saelus diset menjadi titik keberangkatan berikutnya karena memiliki nilai terkecil. Dijsktra kembali melakukan kalkulasi terhadap titik tetangga terdekat yang terhubung dengan titik P.Saelus dan menemukan titik sesetan5; (12) L (sesetan5) = min {∞, (1173+512)} = 1685; (13) Dari titik sesetan5, dijkstra melakukan kalkulasi kembali dan menemukan titik sesetan4 dan sesetan6; (14) L(sesetan4) = min {∞,(1675+1112)}=2787;(15) L(sesetan6) = min{∞,(1675 + 685) = 2360.

Titik sesetan6 diset menjadi titik keberangkatan berikutnya karena memiliki nilai terkecil. Dijkstra kembali melakukan kalkulasi dan menemukan titik pom bensin tujuan se-bagai titik keberangkatan berikutnya. L (pomsesetan2) = min{∞,(2360 + 100)} = 2460.

Karena pomsesetan2 merupakan titik tujuan maka dijkstra tidak melakukan iterasi kembali. (a) Jalankan langkah (5) dan (6) ke semua *verteks* pom bensin. (b) Didapatkan jalur terpendek = 2460 yaitu dari P.Bungin– P.Singkep – P.Saelus – sesetan5 – sesetan6 – pomsesetan2 = 671 + 502 + 512 + 685 + 100 = 2460.

Metode Dijkstra Pada Contoh Kasus

Untuk lebih memperjelas proses penentuan jalur terpendek dengan perhitungan metode Dijkstra, maka akan ditujukkan salah satu contok kasus sebagai berikut :

Seorang wisatawan asing sedang berlibur ke Bali dan saat ini ia berada di Denpasar. Wisatawan ini kebingungan untuk mencari pom bensin karena mobil yang ia kendarai telah kehabisan bahan bakar. Posisi wisatawan berada di jalan diponegoro.

Dari contoh kasus tersebut, penyelesaian yang dilakukan adalah *user* memasukkan nama jalan pada sistem, sehingga berdasarkan kondisi yang diinputkan *user* dapat dila-kukan perhitungan metode Dijkstra dengan tahapan-tahapan sebagai berikut:Saat sesi pilih jalan dijalankan *user* akan memasukkan nama jalan dimana ia berada. Setelah memilih jalan, *user* akan memilih sub jalan sebagai titik – titik jalan yang dipilih sebelumnya. Inputan yang telah dipilih oleh *user* akan diproses menggunakan metode Dijkstra.

Proses ini akan membandingkan seluruh pom bensin yang telah ter-simpan. Setelah diproses dengan metode Dijkstra maka sistem akan memberikan tam-pilan berupa garis untuk menunjukkan hasil jalur pom bensin terpendek yang dihasilkan oleh sistem.

Gambar 14
Input Pilih Jalan

Hasil perhitungan yang dihasilkan dapat dilihat pada gambar berikut ini :

Gambar 15
Input Pilih Subjalan

Gambar 16
Hasil jalur terpendek

Aplikasi Handphone

Form Utama

Form utama merupakan tampilan awal ke-tika aplikasi dijalankan. Form utama berisi menu berupa peta, batuan dan keluar. Form utama dapat dilihat pada Gambar 4.5.

Gambar 17. Form Utama

Form Peta

Form peta merupakan form dimana user dapat melihat peta kota Denpasar dan dapat menentukan posisi user berada sekarang. Form ini memiliki 2 tombol pilihan yaitu pilih jalan dan menu. Form peta dapat dilihat pada Gambar 18.

Gambar 18. Form Peta

Form Pilih Jalan

Form pilih jalan merupakan form untuk user dapat memasukkan jalan dimana ia berada sekarang dan mengetahui letak posisi jalan tersebut. User akan diberikan *combobox* yang berisi nama jalan – jalan yang ada di kota Denpasar dan terdapat *textfield* cari yang digunakan untuk mempercepat proses pencarian jalan. Form pilih ja-lan dapat dilihat pada Gambar 19.

Gambar 19.
Form Pilih Jalan

Form Pilih SubJalan

Form pilih subjalan merupakan form untuk user dapat memilih subjalan yang telah dipilih sebelummya pada form jalan. User akan diberikan *combobox* yang berisi nama sub – sub jalan berdasarkan masukkan jalan pada form jalan. Form pilih subjalan dapat dilihat pada Gambar 20.

Gambar 20.
Form Pilih SubJalan

SIMPULAN

Dari penelitian mengenai pencarian pom bensin terdekat di Denpasar, maka kesim-pulan yang didapat adalah, Aplikasi pencarian pom bensin terdekat menggu-nakan algoritma Dijkstra berbasis *web mo-bile* dapat memberikan solusi untuk memper-mudah *user* dalam pencarian pom bensin terdekat.

Metode Dijkstra dalam proses pen-cariannya tidak optimal apabila tujuan yang ingin dicari lebih dari 2 buah karena harus melakukan proses pencarian berulang–ulang ke semua tujuan. Waktu proses yang diper-lukan untuk menghitung nilai jarak terpen-dek dari suatu peta, dipengaruhi oleh ba-nyaknya titik dan juga banyak tujuan yang ingin dicapai.

DAFTAR PUSTAKA

[1] Agnarsson,Geir., dan Raymond Greenlaw.(2007).***Graph Teory : Modelling, Applications, and Algorithms.*** Pearson Prentice Hall

[2] Feng YU, Zhu Jun.(2001). ***Wireless Java™ Programming with Java™2 Micro Edition***. Indianapolis : Sams Publishing

[3] Hartanto, Antonius A.(2004). **Pemrograman Mobile Java Dengan MIDP 2.0.** Yogyakarta : Andi Offset

[4] Irawan. (2009). ***12 Aplikasi Java Mobile*** , Palembang : Maxikom

[5] Rosen, Kenneth H. (1995). ***Discrete Mathematics and Its Applications*, 3rd ed**. New York : Mc Graw Hill, Inc

[6] Tremblett, Paul. (2002). ***Instant Wireless Java™ with J2ME™***. McGraw-Hill Companies

[7] Utomo, Eko Priyo. (2009). ***Panduan Mudah Mengenal Bahasa Java***. Bandung : Yrama Widya