

ABSTRAK

Google Maps adalah salah satu aplikasi yang dapat mengetahui pemetaan jalan, kondisi lalu lintas, dan penelusuran rute, jarak tempuh dan waktu tempuh ke tempat yang hendak kita tuju. Namun dengan adanya *Google Maps*, masih dihadapi beberapa persoalan, salah satunya mengenai pencarian rute efektif, karena *Google Maps* tidak dapat mencari rute perjalanan yang efektif yang melalui beberapa tempat dalam satu kali perjalanan. Dengan adanya Sistem Pencarian Rute Menggunakan Algoritma *Branch and Bound* dan Dijkstra ini, khususnya bagi para pengguna jalan, dapat menghitung jarak tempuh rute yang mendekati optimal sehingga para pengguna dapat memperoleh rute perjalanan yang harus ditempuhnya. Untuk mencari rute perjalanan pada permasalahan *travelling salesman problem* jarak masing-masing dari titik tempat awal ke titik tempat lainnya yang akan pengguna kunjungi akan dihitung dengan menerapkan algoritma *Branch and Bound* sehingga hasil perhitungan total jarak tempuh dari rute yang dihasilkan dapat diperoleh serta rute yang mendekati optimal yang dihasilkan dapat ditampilkan pada peta yang tersedia pada aplikasi.

Kata kunci: *Google Maps*, *travelling salesman problem*, pencarian rute, *Branch and Bound*.

ABSTRACT

Google Maps is an application to know street mapping, detects traffic conditions, and routing, counts mileage and travel time to our destination. But although Google Maps exists, several problem were still faced, one of them is about the search for effective route, because Google Maps cannot find an effective route that goes through several places in one way trip. With this application called Route Searching System with Branch and Bound and Dijkstra's Algorithm, especially for road users, can count the mileage of the nearly optimal route and therefore it can obtain the direction of the route. To search the travel route on the travelling salesman problem, the mileage for each location from the starting location to the other locations that the user visit will be counted by applying Branch and Bound algorithm so therefore the total mileage of the route can be obtained and also the direction of the optimal route can be showed in the map that available on the application.

Keywords: Google Maps, travelling salesman problem, route searching, Branch and Bound.

DAFTAR ISI

LEMBAR PENGESAHAN	i
PERNYATAAN ORISINALITAS LAPORAN PENELITIAN	ii
PERNYATAAN PUBLIKASI LAPORAN PENELITIAN.....	iii
PRAKATA.....	iv
ABSTRAK	vi
ABSTRACT	vii
DAFTAR ISI.....	viii
DAFTAR GAMBAR	xi
DAFTAR TABEL.....	xiv
DAFTAR NOTASI/ LAMBANG.....	xv
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan Pembahasan	2
1.4 Ruang Lingkup.....	3
1.5 Sumber Data.....	4
1.6 Sistematika Penyajian	4
BAB 2 KAJIAN TEORI	6
2.1 <i>Branch and Bound</i>	6
2.2 <i>Travelling Salesman Problem (TSP)</i>	7
2.3 Pemecahan Masalah TSP dengan Algoritma <i>Branch and Bound</i>	7
2.4 Algoritma Dijkstra dalam Memecahkan Permasalahan Pencarian Rute Terpendek.....	12
2.5 Algoritma <i>Brute Force</i> dalam Pemecahan Permasalahan TSP dan Pencarian Rute Terpendek.....	13

2.6 Google Maps API.....	14
2.7 Global Positioning System (GPS)	16
BAB 3 ANALISIS DAN RANCANGAN SISTEM.....	17
3.1 Analisis Sistem.....	17
3.2 Rancangan Sistem	23
3.2.1 Use Case Diagram	23
3.2.2 Activity Diagram	24
3.2.2.1 Activity Diagram Mencari Rute Perjalanan	24
3.2.2.2 Activity Diagram Menghitung TSP dengan Algoritma <i>Branch and Bound</i>	25
3.2.2.3 Activity Diagram Menghitung Rute Terpendek dengan Algoritma Dijkstra.....	27
3.2.2.4 Activity Diagram Mengetahui Lokasi	28
3.2.2.5 Activity Diagram Memasukkan Lokasi Baru.....	28
3.2.2.6 Activity Diagram Mengupdate Data Lokasi.....	29
3.2.2.7 Activity Diagram Memasukkan Tipe Lokasi Baru.....	30
3.2.3 Rancangan Penyimpanan Data.....	31
BAB 4 IMPLEMENTASI.....	33
4.1 Implementasi Antar Muka Aplikasi	33
4.1.1 Tampilan Mencari Rute Perjalanan.....	33
4.1.2 Tampilan Mengetahui Lokasi	36
4.1.3 Tampilan Memasukkan Lokasi Baru	37
4.1.4 Tampilan <i>Login</i> untuk <i>Administrator</i>	38
4.1.5 Tampilan Mengupdate Data Lokasi untuk <i>Administrator</i>	39
4.1.6 Tampilan Memasukkan Tipe Lokasi Baru untuk <i>Administrator</i>	41
4.1.7 Tampilan Aplikasi Pada <i>Smartphone</i>	43

4.2 Implementasi Metode.....	44
4.2.1 Implementasi <i>Branch and Bound</i>	44
4.2.1.1 Kelas Simpul (<i>Node</i>).....	45
4.2.1.2 Fungsi TSP dengan <i>Branch and Bound</i>	47
4.2.1.3 Implementasi <i>Branch and Bound</i> Dalam Program Utama.....	48
4.2.2 Implementasi Dijkstra	49
4.2.2.1 Fungsi Pencarian Rute Terpendek dengan Algoritma Dijkstra.....	50
4.2.2.2 Implementasi Algoritma Dijkstra Dalam Program Utama.....	51
4.2.3 Implementasi Penyimpanan Data.....	52
4.2.3.1 Implementasi Menyimpan Lokasi.....	52
4.2.3.2 Implementasi Menghapus Lokasi	55
4.2.3.3 Implementasi Mengupdate Lokasi	55
4.2.3.4 Implementasi Memasukkan Tipe Lokasi Baru	57
BAB 5 PENGUJIAN.....	59
5.1 Hasil Pengujian	59
5.2 Pengujian dengan Algoritma <i>Brute Force</i>	78
5.3 Pembahasan Hasil Pengujian	83
BAB 6 SIMPULAN DAN SARAN.....	89
6.1 Simpulan	89
6.2 Saran.....	90
DAFTAR PUSTAKA	91

DAFTAR GAMBAR

Gambar 2.1 Contoh persoalan TSP dengan penyelesaian <i>Brute Force</i>	7
Gambar 2.2 Langkah Mereduksi Matriks	8
Gambar 2.3 Mendapatkan Matriks Bobot Tereduksi untuk Simpul S	9
Gambar 2.4 Mendapatkan Nilai Batas pada Sebuah Simpul	10
Gambar 2.5 Solusi Permasalahan TSP dengan Algoritma <i>Branch and Bound</i> (Bagian 1).....	10
Gambar 2.6 Solusi Permasalahan TSP dengan Algoritma <i>Branch and Bound</i> (Bagian 2).....	11
Gambar 2.7 Solusi Permasalahan TSP dengan Algoritma <i>Branch and Bound</i> (Bagian 3).....	11
Gambar 2.8 Contoh Penerapan Algoritma Dijkstra	13
Gambar 2.9 Bagian-Bagian GPS (GPS Segments)	16
Gambar 3.1 Graf Jarak Antar Lokasi	18
Gambar 3.2 Matriks Jarak Antar Lokasi	18
Gambar 3.3 Proses Reduksi Matriks Akar	19
Gambar 3.4 Contoh Matriks Reduksi AB ($x = 2$)	19
Gambar 3.5 Pohon Pencarian (Bagian 1)	20
Gambar 3.6 Pohon Pencarian (Bagian 2)	20
Gambar 3.7 Pohon Pencarian (Bagian 3)	21
Gambar 3.8 <i>Use Case Diagram</i>	24
Gambar 3.9 <i>Activity Diagram</i> Mencari Rute Perjalanan Antar Lokasi	25
Gambar 3.10 <i>Activity Diagram</i> Menghitung TSP dengan Algoritma <i>Branch and Bound</i>	26
Gambar 3.11 <i>Activity Diagram</i> Menghitung Rute Terpendek dengan Algoritma Dijkstra.....	27
Gambar 3.12 <i>Activity Diagram</i> Mengetahui Lokasi	28
Gambar 3.13 <i>Activity Diagram</i> Memasukkan Lokasi Baru	29
Gambar 3.14 <i>Activity Diagram</i> Mengupdate Data Lokasi	30
Gambar 3.15 <i>Activity Diagram</i> Menambahkan Tipe Lokasi Baru	31
Gambar 3.16 Daftar Tipe Data.....	32

Gambar 4.1 Tampilan Mencari Rute Perjalanan.....	33
Gambar 4.2 Tampilan Masukkan Tempat.....	34
Gambar 4.3 Tampilan Rute Perjalanan TSP	35
Gambar 4.4 Tampilan Rute Perjalanan A-Z	35
Gambar 4.5 Tampilan Mengetahui Lokasi.....	36
Gambar 4.6 Tampilan Rute Lokasi	37
Gambar 4.7 Tampilan Memasukkan Lokasi Baru	37
Gambar 4.8 Tampilan Memasukkan Data Lokasi Baru.....	38
Gambar 4.9 Tampilan <i>Login</i> untuk <i>Administrator</i>	39
Gambar 4.10 Tampilan Mengupdate Data Lokasi untuk <i>Administrator</i>	39
Gambar 4.11 Tampilan Daftar Lokasi	40
Gambar 4.12 Tampilan Mengupdate Lokasi yang Berstatus <i>False</i>	41
Gambar 4.13 Tampilan Peta Setelah Status Lokasi <i>True</i>	41
Gambar 4.14 Tampilan Memasukkan Tipe Lokasi Baru	42
Gambar 4.15 Tampilan Aplikasi Pada <i>Smartphone</i> (Bagian 1).....	43
Gambar 4.16 Tampilan Aplikasi Pada <i>Smartphone</i> (Bagian 2).....	43
Gambar 4.17 Tampilan Aplikasi Pada <i>Smartphone</i> (Bagian 3).....	44
Gambar 4.18 Kode Program untuk <i>Class Node</i>	46
Gambar 4.19 Kode Program untuk Fungsi TSP (<i>Branch and Bound</i>).....	48
Gambar 4.20 Kode Program untuk Implementasi <i>Branch and Bound</i> pada Program Utama	49
Gambar 4.21 Kode Program untuk Fungsi <i>shortestPath</i> (Algoritma Dijkstra)	51
Gambar 4.22 Kode Program untuk Implementasi Algoritma Dijkstra pada Program Utama	52
Gambar 4.23 Keterangan Tabel <i>Marker</i> pada Basis Data.....	52
Gambar 4.24 Kode Program untuk Fungsi <i>data</i>	53
Gambar 4.25 Kode Program untuk <i>addNewMarkerAdmin</i>	53
Gambar 4.26 Kode Program untuk Fungsi <i>Input</i>	54
Gambar 4.27 Kode Program untuk Fungsi <i>addNewMarkerUser</i>	54
Gambar 4.28 Kode Program untuk Menghapus Lokasi (<i>data.js</i>)	55
Gambar 4.29 Kode Program untuk Menghapus Lokasi (<i>deleteMarker.php</i>).....	55
Gambar 4.30 Kode Program untuk Mengupdate Lokasi (<i>data.js</i>)	56

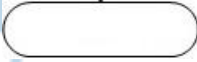
Gambar 4.31 Kode Program untuk Mengupdate Lokasi (<i>updateMarker.php</i>).....	57
Gambar 4.32 Kode Program untuk Fungsi <i>tipe</i>	58
Gambar 4.33 Kode Program untuk Fungsi <i>addTipe</i>	58



DAFTAR TABEL

Tabel 3.1 Tabel Jarak dengan Rute Tercepat Antar Lokasi	18
Tabel 3.2 Tabel Penjabaran Contoh Kasus dengan Algoritma Dijkstra	22
Tabel 5.1 Tabel Hasil Pengujian Rute TSP (Rute Memutar) dengan Algoritma <i>Branch and Bound</i>	60
Tabel 5.2 Tabel Hasil Pengujian Rute A-Z (Rute Satu Arah) dengan Algoritma Dijkstra.....	70
Tabel 5.3 Tabel Hasil Pengujian Rute TSP (Rute Memutar) dengan Algoritma <i>Brute Force</i>	79
Tabel 5.4 Tabel Hasil Pengujian Rute A-Z (Rute Satu Arah) dengan Algoritma <i>Brute Force</i>	81
Tabel 5.5 Tabel Perhitungan Perbandingan Algoritma <i>Branch and Bound</i> dan <i>Brute Force</i> untuk Permasalahan TSP	84
Tabel 5.6 Tabel Perhitungan Perbandingan Algoritma <i>Branch and Bound</i> dan <i>Brute Force</i> untuk Permasalahan TSP Khusus Matriks Berukuran Kecil.....	86
Tabel 5.7 Tabel Perhitungan Perbandingan Algoritma Dijkstra dan <i>Brute Force</i> untuk Permasalahan Pencarian Rute Terpendek.....	87

DAFTAR NOTASI/ LAMBANG

Jenis	Notasi/ Lambang	Nama	Arti
<i>Use Case Diagram</i>		<i>Actor</i>	Orang/sistem yang berkaitan dan berinteraksi dengan program
		<i>Association line</i>	Fungsi yang hanya menggambarkan keterlibatan <i>actor</i> dengan <i>use case</i>
		<i>System Boundary</i>	Batas gambaran antara sistem dan <i>actor</i>
<i>Activity Diagram</i>		<i>Initial State</i>	Kondisi awal sebuah objek sebelum ada perubahan keadaan
		<i>Final State</i>	Kondisi akhir ketika objek berhenti memberikan respon
		<i>State</i>	Kondisi sebuah entitas
		<i>Transition</i>	Perubahan kondisi suatu objek yang disebabkan oleh suatu event
		<i>Decision</i>	Kondisi percabangan yang diharuskan memilih satu kondisi

Referensi:

Notasi / Lambang *Use Case Diagram* dari UML *Distilled*. [1]

Notasi / Lambang *Activity Diagram* dari Object-Oriented Software Engineering: Practical Software Development using UML and Java. [2]