



ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IF184802

**PEMANFAATAN LOCATION-AWARE REACHABILITY
QUERIES PADA DATA GEOSOSIAL UNTUK PEMETAAN
POTENSI PENYERANG TERHADAP KEAMANAN JARINGAN**

ROHANA QUDUS
NRP 05111540000045

Dosen Pembimbing I
Bagus Jati Santoso, S.Kom., Ph.D.

Dosen Pembimbing II
Henning Titi Ciptaningtyas, S.Kom., M.Kom

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya, 2020

(Halaman ini sengaja dikosongkan)



TUGAS AKHIR - IF184802

**PEMANFAATAN LOCATION-AWARE REACHABILITY
QUERIES PADA DATA GEOSOSIAL UNTUK PEMETAAN
POTENSI PENYERANG TERHADAP KEAMANAN JARINGAN**

ROHANA QUDUS
NRP 0511154000045

Dosen Pembimbing I
Bagus Jati Santoso, S.Kom., Ph.D.

Dosen Pembimbing II
Henning Titi Ciptaningtyas, S.Kom., M.Kom

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya, 2020

(Halaman ini sengaja dikosongkan)



UNDERGRADUATE THESIS - IF184802

**LOCATION-AWARE REACHABILITY QUERIES UTILIZATION
ON GEOSOCIAL DATA FOR THE MAPPING OF POTENTIAL
OF ATTACKERS AGAINST NETWORK SECURITY**

ROHANA QUDUS
NRP 05111540000045

Supervisor I
Bagus Jati Santoso, S.Kom., Ph.D.

Supervisor II
Henning Titi Ciptaningtyas, S.Kom., M.Kom

DEPARTMENT OF INFORMATICS ENGINEERING
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2020

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

**PEMANFAATAN LOCATION-AWARE REACHABILITY
QUERIES PADA DATA GEOSOSIAL UNTUK
PEMETAAN POTENSI PENYERANG TERHADAP
KEAMANAN JARINGAN**

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S1 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh :

ROHANA QUDUS
NRP: 0511154000045

Disetujui oleh Dosen Pembimbing Tugas Akhir

Bagus Jati Santoso, S.Kom., Ph.D.

NIP: 198611252018031001

(Pembimbing 1)

Henning Titi Ciptaningtyas, S.Kom., M.Kom.

NIP: 198407082010122004

(Pembimbing 2)



SURABAYA
Januari 2020

(Halaman ini sengaja dikosongkan)

**PEMANFAATAN LOCATION-AWARE REACHABILITY
QUERIES PADA DATA GEOSOSIAL UNTUK
PEMETAAN POTENSI PENYERANG TERHADAP
KEAMANAN JARINGAN**

Nama : ROHANA QUDUS
NRP : 0511154000045
Departemen : Teknik Informatika FTEIC
Pembimbing I : Bagus Jati Santoso, S.Kom., Ph.D.
Pembimbing II : Henning Titi Ciptaningtyas, S.Kom.,
M.Kom

Abstrak

Penyerangan yang terjadi pada kehidupan sehari-hari dapat terjadi di mana saja. Dari lokasi kejadian dibutuhkan algoritma untuk memetakan pengguna yang berpotensi dapat melakukan penyerangan berdasarkan informasi yang ada. Tujuan dari pembuatan tugas akhir ini adalah mendesain algoritma yang dapat menyelesaikan permasalahan tersebut dengan memanfaatkan location-aware reachability query pada geosocial graph.

Geosocial graph adalah graf yang menyatukan informasi lokasi dan hubungan sosial. Reachability query merupakan salah satu kueri dasar untuk menguji apakah terdapat jalur dari satu node ke node lainnya. Dengan memanfaatkan reachability query pada geosocial graph, permasalahan seperti kasus penyerangan dapat diselesaikan.

Terdapat dua penyelesaian yang diusulkan untuk menyelesaikan permasalahan pemetaan pengguna, yaitu dengan menggunakan RMBR dan graf traversal. Pemetaan dengan menggunakan RMBR dilakukan dengan mencari nilai RMBR yang beririsan sedangkan pemetaan dengan menggunakan graf

traversal dilakukan dengan cara menelusuri graf. Berdasarkan hasil uji coba skenario yang dilakukan, performa algoritma dengan menggunakan RMBR jauh lebih unggul dibandingkan dengan menggunakan graf traversal.

Kata-Kunci: *Geosocial Graph, Reachability Query*

**LOCATION-AWARE REACHABILITY QUERIES
UTILIZATION ON GEOSOCIAL DATA FOR THE
MAPPING OF POTENTIAL OF ATTACKERS AGAINST
NETWORK SECURITY**

Name : ROHANA QUDUS
NRP : 0511154000045
Department : Informatics Engineering ELECTICS
Supervisor I : Bagus Jati Santoso, S.Kom., Ph.D.
Supervisor II : Henning Titi Ciptaningtyas, S.Kom.,
M.Kom

Abstract

Attacks that occur in everyday life can happen anywhere. An algorithm is needed to map users who have the potential to attack based on the given information of the location listed in the crime scene. The purpose of this research is to design an algorithm to solve the aforementioned problem by using location-aware reachability query on a geosocial graph.

Geosocial graph is a graph that combined location information and social relation. Reachability query is a basic query for graph data to verify whether one node is reachable from another node. By using reachability query on a geosocial graph, problems such as attacks that occur in everyday life can be solved.

There are two solutions proposed for this user mapping problem, the first one is RMBR-Based solution and the second one is Graph Traversal-Based solution. User mapping with RMBR-Based solution is done by looking for intersecting RMBR values while user mapping with Graph Traversal-Based is done by traversing the graph. Based on the result of the performed scenario tests, the performance of RMBR-Based solution

algorithm have much better performance compared to the Graph Traversal-Based.

Keywords: *Geosocial Graph, Reachability Query*

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **Pemanfaatan *Location-Aware Reachability Queries* pada Data Geososial untuk Pemetaan Potensi Penyerang Terhadap Keamanan Jaringan**. Pengerjaan tugas akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan tugas akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Departemen Informatika ITS. Dengan tugas akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari. Selesainya tugas akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Keluarga penulis yang selalu menyemangati.
3. Bapak Bagus Jati Santoso, S.Kom., Ph.D selaku pembimbing I yang telah membantu, membimbing dan memotivasi penulis mulai dari pengerjaan proposal hingga terselesaikannya tugas akhir ini.
4. Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom selaku pembimbing II yang juga telah membantu, membimbing dan memotivasi penulis mulai dari pengerjaan proposal hingga terselesaikannya tugas akhir ini.
5. Ibu Dr.Eng. Chastine Fatichah, S.Kom., M.Kom selaku Kepala Departemen Informatika ITS pada masa pengerjaan tugas akhir.
6. Bapak Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc.,

selaku Koordinator TA dan segenap dosen Departemen Informatika yang telah memberikan ilmu dan pengalamannya.

7. Baby, Elga, Nadia, dan Kevin yang telah menjadi tempat berkeluh kesah bagi penulis selama pengerjaan tugas akhir ini.
8. Mas Syukron yang telah membantu penulis dalam menyelesaikan tugas akhir ini.
9. Teman-teman Administrator Laboratorium Arsitektur dan Jaringan Komputer.
10. Serta semua pihak yang telah turut membantu penulis dalam menyelesaikan tugas akhir ini.

Penulis menyadari bahwa tugas akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya 2020

Rohana Qudus

DAFTAR ISI

ABSTRAK	vii
ABSTRACT	ix
KATA PENGANTAR	xi
DAFTAR ISI	xiii
DAFTAR TABEL	xvii
DAFTAR GAMBAR	xix
DAFTAR KODE SUMBER	xxi
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan	3
1.5 Manfaat	3
1.6 Metodologi	3
1.7 Sistematika Penulisan	6
BAB II TINJAUAN PUSTAKA	9
2.1 Teori Graf	10
2.2 <i>Geosocial Graph</i>	11
2.3 <i>Reachability Query</i>	12
2.4 <i>Reachability Minimum Bounding Rectangle</i>	13
2.5 <i>Graph Traversal</i>	14
2.6 <i>Depth-First Search</i>	14
2.7 <i>Breadth-First Search</i>	14
2.8 Struktur Data	15
2.9 Python	15

BAB III ANALISIS DAN PERANCANGAN	17
3.1 Analisis Sistem	17
3.1.1 Analisis Permasalahan	17
3.1.2 Deskripsi Umum Sistem	17
3.2 Perancangan Sistem	19
3.2.1 <i>Preprocessing</i> Data	19
3.2.2 Proses Utama	27
BAB IV IMPLEMENTASI	35
4.1 Lingkungan Implementasi Perangkat Lunak	35
4.1.1 Perangkat Keras	35
4.1.2 Perangkat Lunak	36
4.2 Implementasi Program <i>Preprocessing</i> Data	36
4.2.1 Pembuatan Graf	36
4.2.2 <i>Update</i> RMBR	37
4.3 Implementasi Program Utama	39
4.3.1 Pemetaan Pengguna dengan Menggunakan RMBR	39
4.3.2 Pemetaan Pengguna dengan Menggunakan Graf Traversal	40
BAB V PENGUJIAN DAN EVALUASI	43
5.1 Lingkungan Pengujian	43
5.2 Jenis Data Pengujian	43
5.3 Skenario Pengujian	44
5.3.1 Uji Coba Fungsionalitas	44
5.3.2 Uji Coba Performa	45
5.4 Analisis Hasil Uji Coba	46
5.4.1 Hasil Uji Coba Fungsionalitas	47
5.4.2 Hasil Uji Coba Performa	49
BAB VI KESIMPULAN DAN SARAN	61
6.1 Kesimpulan	61
6.2 Saran	62

DAFTAR PUSTAKA	63
BAB A Kode Sumber	65
BIODATA PENULIS	81

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

2.1	Daftar Simbol dan Keterangan Bab II	9
3.1	Atribut Data Pengguna	20
3.2	Atribut Data Lokasi	20
3.3	Data Relasi Pengguna	21
3.4	Data Relasi Lokasi	21
3.5	Data <i>Check-in</i>	22
3.6	Nilai Awal RMBR untuk Tiap Lokasi	24
3.7	Nilai RMBR Lokasi Setelah Diperbarui	25
3.8	Nilai RMBR Pengguna Setelah Diperbarui	26
3.9	Hasil Pemetaan dengan Menggunakan Algoritma Graf Traversal	32
4.1	Tabel Perangkat Keras Implementasi Sistem	35
4.2	Tabel Perangkat Lunak Implementasi Sistem	36
5.1	Lingkungan Pengujian	43
5.2	Kebutuhan Fungsional	44
5.3	Graf Pengujian	45
5.4	Skenario Variasi Jumlah <i>Node</i>	45
5.5	Skenario Rasio Jumlah Pengguna dan Lokasi	46
5.6	Skenario Jumlah <i>Edge</i> Pengguna	46
5.7	Hasil Uji Coba Kebutuhan Fungsional	47
5.8	Rata-rata Waktu <i>Preprocessing</i>	49
5.9	Perbandingan Waktu Eksekusi dengan Jumlah <i>Node</i>	51
5.10	Perbandingan Penggunaan Memori dengan Jumlah <i>Node</i>	51
5.11	Perbandingan Jumlah Pengguna pada Hasil Tiap Skenario	51
5.12	Perbandingan Jumlah Lokasi pada Hasil Tiap Skenario dengan Menggunakan Algoritma Graf Traversal	52
5.13	Perbandingan Waktu Eksekusi dengan Rasio Jumlah Pengguna dan Lokasi	54

5.14	Perbandingan Penggunaan Memori dengan Rasio Jumlah Pengguna dan Lokasi	54
5.15	Perbandingan Jumlah Pengguna pada Hasil Tiap Skenario	54
5.16	Perbandingan Jumlah Lokasi pada Hasil Tiap Skenario dengan Menggunakan Algoritma Graf Traversal	55
5.17	Perbandingan Waktu Eksekusi dengan Jumlah <i>Edge</i> Pengguna	57
5.18	Perbandingan Penggunaan Memori dengan Jumlah <i>Edge</i> Pengguna	57
5.19	Perbandingan Jumlah Pengguna pada Hasil Tiap Skenario	58
5.20	Perbandingan Jumlah Lokasi pada Hasil Tiap Skenario dengan Menggunakan Algoritma Graf Traversal	58

DAFTAR GAMBAR

2.1	Representasi Teori Graf	10
2.2	Struktur Sederhana <i>Geosocial Graph</i>	11
2.3	Contoh <i>Reachability Query</i>	12
2.4	Contoh <i>Reachability Minimum Bounding Rectangle</i>	13
3.1	Diagram Alur Deskripsi Umum Sistem	18
3.2	Diagram Alur <i>Preprocessing</i>	19
3.3	Contoh Grafik	23
3.4	Diagram Alur Proses Utama	27
3.5	Diagram Alur Penyelesaian dengan Menggunakan RMBR	29
3.6	Ilustrasi RMBR Pengguna yang Beririsan	30
3.7	Diagram Alur Penyelesaian dengan Menggunakan Graf Traversal	31
5.1	Hasil Uji Coba F01	47
5.2	Hasil Uji Coba F02	48
5.3	Hasil Uji Coba F03	48
5.4	Hasil Uji Coba F03	49
5.5	Grafik Perbandingan Waktu Eksekusi dengan Jumlah <i>Node</i>	50
5.6	Grafik Perbandingan Penggunaan Memori dengan Jumlah <i>Node</i>	50
5.7	Grafik Perbandingan Waktu Eksekusi dengan Rasio Jumlah Pengguna dan Lokasi	53
5.8	Grafik Penggunaan Memori dengan Rasio Jumlah Pengguna dan Lokasi	53
5.9	Grafik Perbandingan Waktu Eksekusi dengan Jumlah <i>Edge</i> Pengguna	56
5.10	Grafik Penggunaan Memori dengan Jumlah <i>Edge</i> Pengguna	56

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

4.1	Algoritma Pembuatan Graf	36
4.2	Algoritma Traversal Graf Lokasi	37
4.3	Algoritma <i>Update</i> RMBR	38
4.4	Algoritma Pemetaan Pengguna dengan Menggunakan RMBR	39
4.5	Algoritma untuk Mendapatkan Daftar Lokasi Kejadian	40
4.6	Algoritma Pemetaan Pengguna dengan Menggunakan Graf Traversal	41
1.1	Kode Sumber Generate Data	65
1.2	Kode Sumber Program Utama	68

(Halaman ini sengaja dikosongkan)

BAB I

PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari tugas akhir yang dikerjakan.

1.1 Latar Belakang

Bekembangnya perangkat seluler yang mendukung GPS dan semakin populernya penggunaan jejaring sosial menyebabkan pertumbuhan jaringan geososial yang cepat. Hal ini bisa dilihat dengan pembagian informasi terkait lokasi seperti *check-in* yang kini menjadi hal yang umum ditemui. Dengan ditambahkannya atribut spasial berupa informasi lokasi pada data jejaring sosial, maka akan terbentuk apa yang dikenal sebagai *geosocial graph*. Graf ini menyatukan informasi lokasi dengan hubungan sosial sehingga permasalahan yang terkait dengan geososial dapat diselesaikan.

Dari *geosocial graph* yang terbentuk tadi, diperlukan kueri pemrosesan untuk mendapatkan informasi dari data yang ada. *Reachability Queries* merupakan salah satu kueri dasar pada data graf yang digunakan untuk menguji apakah terdapat jalur dari *node u* ke *node v*. Pada analisis jejaring sosial misalnya, sekumpulan data memiliki *node* yang merepresentasikan orang dan *edge* (sisi) yang merepresentasikan relasi dari tiap orang, perlu untuk mengetahui apakah ada hubungan antara dua entitas untuk alasan keamanan.

Dari beberapa permasalahan di atas dibutuhkan struktur data dan algoritma yang tepat untuk pemetaan potensi penyerang terhadap keamanan jaringan dengan memanfaatkan *location-aware reachability queries* pada data geososial. Tugas akhir ini akan mengangkat permasalahan *reachability queries* pada data geososial sehingga hasil tugas akhir ini diharapkan

dapat menentukan implementasi algoritma dan struktur data yang tepat untuk memecahkan permasalahan di atas secara optimal dan diharapkan dapat memberikan kontribusi pada ilmu pengetahuan dan teknologi informasi.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana cara mendapatkan seluruh potensi penyerang atau sejumlah k potensi penyerang terhadap keamanan jaringan dengan memanfaatkan *location-aware reachability queries* pada data geososial?
2. Bagaimana menentukan dan mengimplementasikan algoritma dan struktur data yang dibangun untuk mendapatkan seluruh potensi penyerang atau sejumlah k potensi penyerang terhadap keamanan jaringan dengan memanfaatkan *location-aware reachability queries* pada data geososial?
3. Bagaimana hasil dari kinerja algoritma dan struktur data yang dibangun untuk mendapatkan seluruh potensi penyerang atau sejumlah k potensi penyerang terhadap keamanan jaringan dengan memanfaatkan *location-aware reachability queries* pada data geososial?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki batasan antara lain:

1. Algoritma ini hanya memproses nilai atribut bertipe numerik.
2. Implementasi dilakukan dengan bahasa pemrograman *Python*.

3. Dataset yang digunakan adalah data sintetis.

1.4 Tujuan

Tujuan pembuatan tugas akhir ini antara lain:

1. Menentukan algoritma dan struktur data yang tepat untuk menyelesaikan permasalahan pemetaan potensi penyerang terhadap keamanan jaringan dengan memanfaatkan *location-aware reachability queries* pada data geososial.
2. Melakukan implementasi algoritma dan struktur data yang dibangun pada permasalahan pemetaan potensi penyerang terhadap keamanan jaringan dengan memanfaatkan *location-aware reachability queries* pada data geososial.
3. Mengevaluasi hasil dan kinerja dari algoritma dan struktur data yang dibangun pada permasalahan pemetaan potensi penyerang terhadap keamanan jaringan dengan memanfaatkan *location-aware reachability queries* pada data geososial.

1.5 Manfaat

Manfaat yang diharapkan dari tugas akhir ini adalah dapat mendesain dan mengimplementasikan struktur data dan algoritma yang tepat untuk diterapkan pada pemanfaatan *location-aware reachability queries* pada data geososial untuk pemetaan potensi penyerang terhadap keamanan jaringan serta diharapkan dapat memberikan kontribusi pada perkembangan ilmu pengetahuan dan teknologi informasi.

1.6 Metodologi

Tahap-tahap yang dilakukan dalam pengerjaan tugas akhir ini adalah:

1. **Penyusunan Proposal Tugas Akhir**

Proposal tugas akhir ini berisi mengenai deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Subbab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

2. **Studi Literatur**

Pada tahap ini dilakukan pencarian informasi dan studi literatur yang diperlukan untuk penyelesaian persoalan yang akan dikerjakan. Informasi didapatkan dari materi-materi yang berhubungan dengan algoritma yang digunakan dalam pengerjaan tugas akhir ini. Materi-materi tersebut didapatkan dari *paper*, interner, maupun buku acuan.

3. **Desain Perangkat Lunak**

Pada tahap ini penulis akan mendesain beberapa algoritma pengolahan untuk menyelesaikan permasalahan pemetaan potensi penyerang terhadap keamanan jaringan dengan memanfaatkan *location-aware reachability queries* pada data geososial.

4. **Implementasi Perangkat Lunak**

Pada tahap ini dilakukan implementasi dari desain

algoritma pengolahan untuk menyelesaikan permasalahan pemetaan potensi penyerang terhadap keamanan jaringan dengan memanfaatkan *location-aware reachability queries* pada data geososial. Implementasi dilakukan dengan menggunakan bahasa pemrograman *Python*.

5. Uji Coba dan Evaluasi

Pada tahap ini dilakukan uji coba dengan menggunakan beberapa dataset yang sudah disiapkan. Pengujian dilakukan dengan beberapa cara:

(a) Pengujian Waktu Eksekusi

Pengujian ini akan berfokus pada seberapa lama waktu yang dibutuhkan untuk mengeksekusi algoritma dalam memetakan pengguna yang berpotensi melakukan penyerangan dengan memanfaatkan *location-aware reachability query* pada *geosocial graph*.

(b) Pengujian Penggunaan Memori

Pengujian ini akan berfokus pada seberapa besar memori yang digunakan saat algoritma utama dijalankan.

6. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi perangkat lunak yang sudah dibuat. Sistematika penulisan buku tugas akhir adalah sebagai berikut:

(a) Pendahuluan

- i. Latar Belakang
- ii. Rumusan Masalah
- iii. Batasan Masalah

- iv. Tujuan
 - v. Manfaat
 - vi. Metodologi
 - vii. Sistematika Penulisan
- (b) Tinjauan Pustaka
 - (c) Analisis dan Perancangan
 - (d) Implementasi
 - (e) Pengujian dan Evaluasi
 - (f) Kesimpulan dan Saran
 - (g) Daftar Pustaka

1.7 Sistematika Penulisan

Penulisan buku tugas akhir ini memiliki tujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir. Secara garis besar, buku tugas akhir ini terdiri dari beberapa bagian seperti berikut:

1. **Bab I Pendahuluan**

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

2. **Bab II Tinjauan Pustaka**

Bab ini berisi tentang penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

3. **Bab III Desain dan Perancangan**

Bab ini membahas desain dan perancangan sistem yang akan dibangun untuk menyelesaikan permasalahan

pemetaan potensi penyerang terhadap keamanan jaringan dengan memanfaatkan *location-aware reachability queries* pada data geososial.

4. **Bab IV Implementasi**

Bab ini membahas implementasi dari desain dan perancangan sistem yang telah dibuat pada bab sebelumnya.

5. **Bab V Pengujian dan Evaluasi**

Bab ini membahas tahap-tahap uji coba yang dilakukan pada sistem yang dibuat. Kemudian kinerja sistem akan dievaluasi berdasarkan hasil uji coba yang didapatkan.

6. **Bab VI Penutup**

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan serta saran untuk pengembangan aplikasi di kemudian hari.

(Halaman ini sengaja dikosongkan)

BAB II

TINJAUAN PUSTAKA

Bab ini menjelaskan teori-teori yang berkaitan dengan Pemanfaatan *Location-Aware Reachability Queries* pada Data Geososial untuk Pemetaan Potensi Penyerang Terhadap Keamanan Jaringan yang diajukan untuk tugas akhir ini. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap perangkat lunak yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

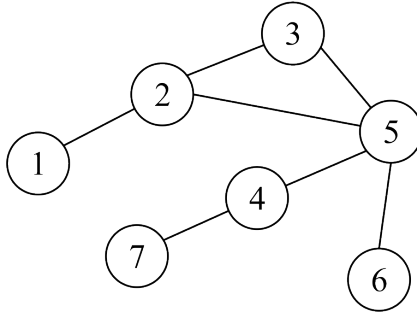
Penelitian ini menggunakan beberapa istilah penting untuk menyederhanakan penulisan. Daftar istilah yang digunakan pada penelitian ini dapat dilihat pada Tabel 2.1.

Tabel 2.1: Daftar Simbol dan Keterangan Bab II

Simbol	Keterangan
G	Graf
V	Himpunan <i>node</i>
E	Himpunan <i>edge</i>
u, v, w	<i>Node</i> yang ada dalam graf
u_i	<i>Node</i> pengguna i dalam graf, di mana $i = 1, 2, \dots, n$
p_i	<i>Node</i> lokasi i dalam graf
la_i, lo_i	Koordinat spasial lokasi i
U_{p_i}	Himpunan pengguna yang pernah melakukan <i>check-in</i> ke lokasi i
$r(u, v)$	<i>Reachability</i> dari <i>node</i> u ke <i>node</i> v

2.1 Teori Graf

Teori graf adalah ilmu yang mempelajari sifat-sifat graf. Graf terdiri dari *node* yang dihubungkan oleh sisi (*edge*). Sebuah graf dengan sisi yang berarah merupakan salah satu pengembangan dari persoalan graf. Graf seperti itu disebut sebagai graf berarah atau *directed graph* (digraph).

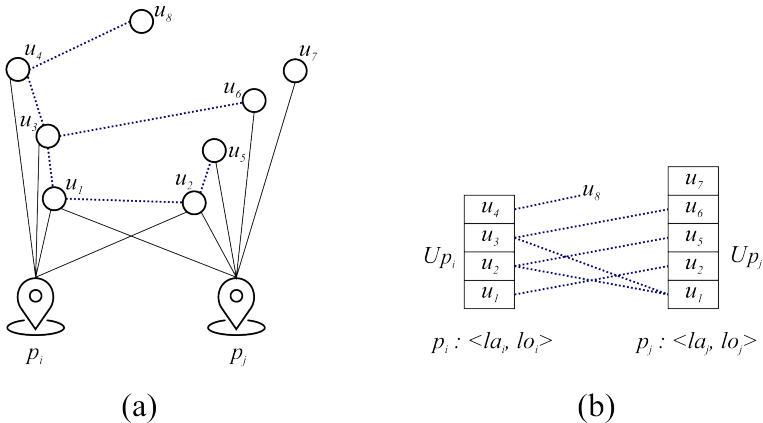


Gambar 2.1: Representasi Teori Graf

Graf dinotasikan sebagai $G = (V, E)$, di mana V merupakan himpunan *node* dan E merupakan himpunan *edge*. Sebagai contoh, graf pada Gambar 2.1 memiliki himpunan *node* V : $\{1, 2, 3, 4, 5, 6, 7\}$ dan himpunan sisi E : $\{1, 2\}, \{2, 3\}, \{2, 5\}, \{3, 5\}, \{4, 5\}, \{4, 7\}, \{5, 6\}$.

2.2 Geosocial Graph

Geosocial Graph atau biasa disebut dengan *Geosocial Networks* (GeoSN) adalah graf yang menggabungkan fungsionalitas dari media sosial dengan layanan berbasis lokasi. Pada graf GeoSN *node* merepresentasikan orang dan sisi merepresentasikan hubungan pertemanan. [1]



Gambar 2.2: Struktur Sederhana *Geosocial Graph*

Pada Gambar 2.2 (a) terdapat 8 pengguna ($u_1 - u_8$ dan 2 lokasi (p_i dan p_j)). Garis putus-putus pada gambar merepresentasikan hubungan pertemanan antar pengguna dan garis utuh merepresentasikan *check-in* dari tiap pengguna di kedua lokasi tersebut. Pada Gambar 2.2 (b) ditunjukkan bahwa tiap lokasi dimodelkan berdasarkan koordinat spasialnya. Sebagai contoh (la_i, lo_i) adalah koordinat spasial dari lokasi p_i .

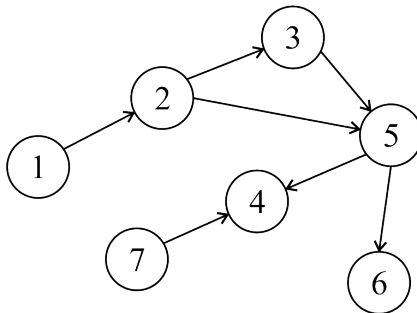
2.3 Reachability Query

Permasalahan umum yang sering terjadi dalam aplikasi berbasis graf adalah untuk memastikan apakah suatu *node* dapat dijangkau dari *node* yang lain. Sebagai contoh $G = (V, E)$ dinyatakan sebagai sebuah *directed graph* dengan V adalah sekumpulan *node* dan $E \subseteq V^2$ adalah sekumpulan sisi. *Reachability query* $r(u, v)$ memeriksa apakah $v \in V$ dapat dijangkau dari $u \in V$, sebagai contoh apakah terdapat sebuah jalur dari u ke v pada G . [2]

Definisi (Reachability Query).

$\forall (u, v) \in V^2$, v dapat dijangkau dari u dinotasikan dengan $r(u, v)$,

jika dan hanya jika $\begin{cases} u = v \\ \text{atau} \\ \exists (u, w) \in E \wedge r(w, v) \end{cases}$



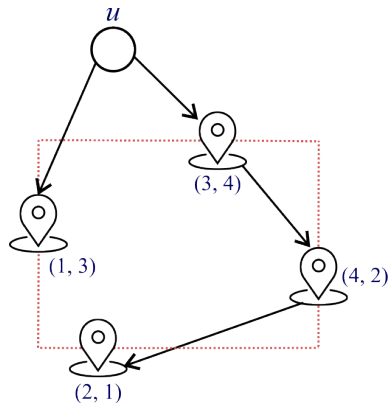
Gambar 2.3: Contoh *Reachability Query*

Dengan menggunakan definisi dari *reachability query* untuk Gambar 2.3, maka apabila dilakukan *query* untuk $r(1, 6)$ maka nilai yang akan dikembalikan bernilai *true*, sedangkan untuk $r(1, 7)$ nilai yang akan dikembalikan adalah *false*.

2.4 *Reachability Minimum Bounding Rectangle*

Minimum Bounding Rectangle adalah ekspresi dari luasan maksimum objek dua dimensi, dengan kata lain $\min(x)$, $\max(x)$, $\min(y)$, $\max(y)$. MBR adalah kotak 2 dimensi dari kotak batas minimum.

MBR sering digunakan sebagai indikasi posisi geografis atau tujuan pengindeksan spasial. Dalam tugas akhir ini MBR akan dibentuk dengan memperhatikan graf karena akan dibuat dengan melihat dari keterjangkauan *node* atau disebut *Reachability Minimum Bounding Rectangle* (RMBR).



Gambar 2.4: Contoh *Reachability Minimum Bounding Rectangle*

Dengan menggunakan Gambar 2.4, seorang pengguna u dapat menjangkau empat lokasi berdasarkan *reachability* baik dari pengguna maupun lokasi. Dari keempat lokasi tersebut nilai dari luasan maksimum akan disimpan untuk merepresentasikan RMBR pengguna.

2.5 *Graph Traversal*

Graf traversal atau pencarian graf adalah proses menelusuri (memeriksa dan/atau memperbarui) tiap *node* yang ada pada suatu graf. Algoritma pada graf traversal diklasifikasikan berdasarkan urutan ketika menelusuri *node*.

2.6 *Depth-First Search*

Depth-First Search merupakan salah satu algoritma graf traversal yang sering digunakan untuk menyelesaikan permasalahan graf. Algoritma ini akan mengunjungi *node* anak terlebih dahulu sebelum mengunjungi *node* pada tingkat yang sama; algoritma akan melintasi kedalaman jalur tertentu sebelum menjelajahi luasnya. [3]

Pada tugas akhir ini, DFS digunakan untuk memperbarui nilai RMBR lokasi karena perlu menelusuri graf hingga yang paling dalam sehingga didapatkan nilai RMBR yang benar; RMBR lokasi paling dalam diperbarui terlebih dahulu sebelum ke lokasi terluar.

2.7 *Breadth-First Search*

Breadth-First Search merupakan salah satu algoritma graf traversal yang sering digunakan untuk menyelesaikan permasalahan graf, berbeda dengan *Depth-First Search*, algoritma ini mengunjungi semua *node* pada tingkat yang sama terlebih dahulu sebelum melanjutkan ke *node* dengan tingkat yang lebih dalam. [4]

Pada tugas akhir ini, BFS digunakan untuk mencatat banyaknya hop yang dibutuhkan tiap *node* asal untuk mencapai *node* tujuan. Penggunaan BFS dipilih untuk tahapan mencatat banyaknya hop karena pada tahap ini nilai hop yang akan

disimpan adalah nilai hop yang paling kecil dari semua kemungkinan jalur yang tersedia.

2.8 Struktur Data

Dalam istilah ilmu komputer, struktur data adalah cara penyimpanan, pengorganisasian, dan pengaturan data di dalam media penyimpanan komputer sehingga data tersebut dapat digunakan secara efisien. [5]

Struktur data yang berbeda cocok digunakan untuk permasalahan yang berbeda pula. Beberapa struktur data berguna untuk masalah umum yang sederhana seperti pengambilan data yang telah disimpan dengan pengidentifikasi tertentu. Pada sisi lain, terdapat struktur data khusus yang telah dirancang untuk memecahkan masalah yang lebih kompleks. [6]

2.9 Python

Python adalah bahasa pemrograman interpretatif, interaktif dan berorientasi objek. Python menggabungkan modul, pengecualian, penulisan secara dinamis, tipe data dinamis yang sangat tinggi dan kelas. Python memiliki antarmuka ke banyak *system call* dan pustaka diberbagai sistem dan dapat diperluas ke bahasa pemrograman C atau C++. Python dapat berjalan pada berbagai sistem operasi seperti Unix, Linux, Mac Os dan Windows.

Python adalah bahasa pemrograman tingkat tinggi yang dapat diterapkan pada berbagai masalah. Bahasa ini dilengkapi pustaka yang besar untuk melakukan pemrosesan *string*, protokol internet, rekayasa perangkat lunak dan antarmuka sistem operasi. [7]

(Halaman ini sengaja dikosongkan)

BAB III

ANALISIS DAN PERANCANGAN

Bab ini menjelaskan tentang analisis dan perancangan mengenai sistem yang akan dibuat.

3.1 Analisis Sistem

Analisis sistem terbagi menjadi dua bagian, yaitu analisis permasalahan yang diangkat pada tugas akhir ini dan deskripsi umum sistem yang dibangun.

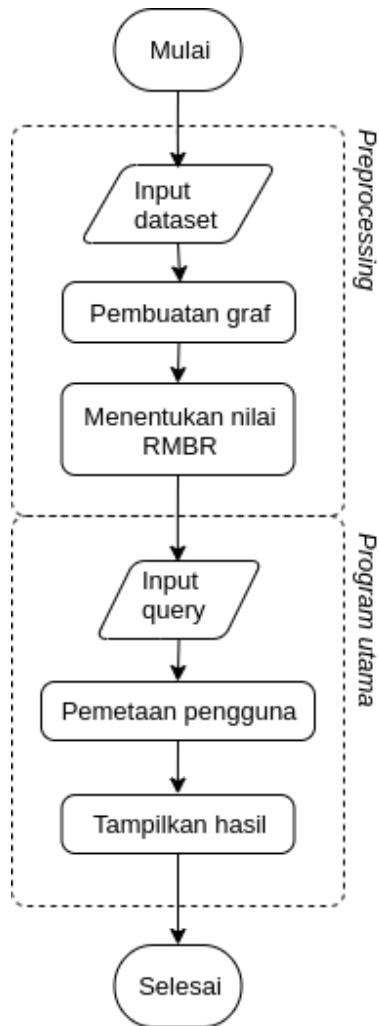
3.1.1 Analisis Permasalahan

Permasalahan yang diangkat pada tugas akhir ini adalah pemetaan orang-orang yang berpotensi dapat melakukan penyerangan berdasarkan informasi *check-in* serta daftar teman yang dimiliki, baik keseluruhan maupun sejumlah k yang didefinisikan oleh pengguna. Terdapat dua solusi yang ditawarkan untuk memetakan pengguna, solusi pertama dengan memanfaatkan RMBR dan solusi kedua dengan melihat keterjangkauan pengguna ke lokasi-lokasi yang termasuk ke dalam area kejadian.

3.1.2 Deskripsi Umum Sistem

Secara garis besar fitur yang diimplementasikan pada tugas akhir ini terdiri dari dua proses, yaitu *preprocessing* data untuk membantu meringankan perhitungan pada proses utama dan proses utama di mana pemetaan pengguna akan dilakukan.

Agar lebih mudah dipahami, alur kerja sistem pada penelitian ini dapat dilihat pada diagram alur pada Gambar 3.1.



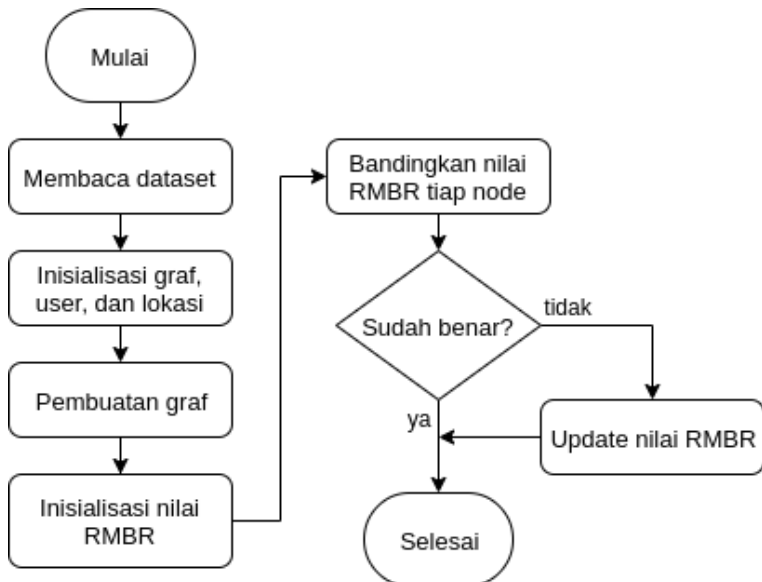
Gambar 3.1: Diagram Alur Deskripsi Umum Sistem

3.2 Perancangan Sistem

Perancangan sistem menjelaskan perancangan proses pada sistem, yaitu *preprocessing* data dan proses utama.

3.2.1 *Preprocessing* Data

Pada tahap ini dilakukan pembuatan graf serta penentuan nilai RMBR untuk tiap *node* pada graf. Tahap *preprocessing* mempunyai alur seperti terlihat pada Gambar 3.2.



Gambar 3.2: Diagram Alur *Preprocessing*

3.2.1.1 Dataset

Terdapat lima jenis data yang digunakan dalam tugas akhir ini, yaitu, data pengguna, data lokasi, data relasi antar pengguna, data relasi antar lokasi, dan data *check-in*.

Data pengguna memiliki atribut berupa id dan nama seperti yang bisa dilihat pada Tabel 3.1.

Tabel 3.1: Atribut Data Pengguna

id	nama
p_1	Morris
p_2	Ronald
p_3	Alberta
p_4	Milton
p_5	Miguel
p_6	William
p_7	Harvey

Data lokasi memiliki atribut id, nama, *latitude*, dan *longitude* seperti yang bisa dilihat pada Tabel 3.2.

Tabel 3.2: Atribut Data Lokasi

id	nama	lat	long
l_1	Knights	120	171
l_2	Dunham	180	77
l_3	Bilski	257	303
l_4	Johnson	175	435
l_5	Brown	432	173

Data relasi antar pengguna memiliki atribut berupa id pengguna satu dan id pengguna dua seperti yang bisa dilihat pada Tabel 3.3.

Tabel 3.3: Data Relasi Pengguna

id1	id2
<i>p1</i>	<i>p2</i>
<i>p2</i>	<i>p7</i>
<i>p1</i>	<i>p3</i>
<i>p2</i>	<i>p4</i>
<i>p3</i>	<i>p4</i>
<i>p1</i>	<i>p4</i>
<i>p3</i>	<i>p6</i>
<i>p3</i>	<i>p3</i>
<i>p1</i>	<i>p5</i>

Data relasi antar lokasi memiliki atribut berupa id lokasi satu dan id lokasi dua seperti yang bisa dilihat pada Tabel 3.4.

Tabel 3.4: Data Relasi Lokasi

id1	id2
<i>l1</i>	<i>l2</i>
<i>l3</i>	<i>l5</i>
<i>l5</i>	<i>l2</i>
<i>l2</i>	<i>l4</i>

Data *check-in* memiliki atribut berupa id pengguna dan id lokasi seperti yang bisa dilihat pada tabel 3.5.

Tabel 3.5: Data *Check-in*

id pengguna	id lokasi
p_3	l_5
p_5	l_4
p_7	l_4
p_5	l_3
p_2	l_1

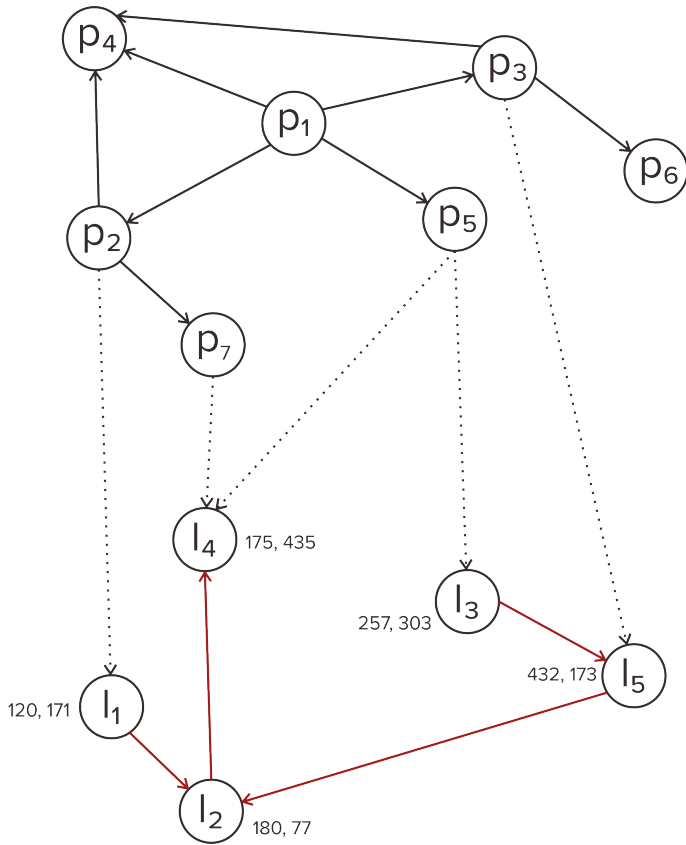
3.2.1.2 Inisialisasi

Tahap *preprocessing* diawali dengan menginisialisasi graf, user, dan lokasi. Pada user akan disimpan informasi yang berkaitan dengan pengguna, lokasi akan menyimpan informasi yang berkaitan dengan lokasi, kemudian graf akan menyimpan seluruh *node* yaitu *node* pengguna dan lokasi.

3.2.1.3 Pembuatan Graf

Proses pembuatan graf dilakukan dengan mengekstraksi data *node* dan *edge* yang tersedia. Graf yang digunakan pada tugas akhir ini adalah *directed graph* di mana tiap sisi akan diarahkan dari satu *node* ke *node* yang lain.

Dengan menggunakan data relasi yang telah dijabarkan pada Tabel 3.3 dan 3.4, graf yang dibuat pada tugas akhir ini akan bersifat *one directional* graf.



Gambar 3.3: Contoh Grafik

3.2.1.4 Inisialisasi RMBR

Tiap lokasi memiliki nilai koordinat spasial yang akan dijadikan sebagai nilai RMBR awal. Nilai RMBR akan diperbarui dengan melihat keterjangkauan lokasi satu dengan lokasi lainnya.

Gambar 3.3 adalah grafik yang dibuat berdasarkan data yang dicontohkan pada bagian 3.2.1.1, dengan menggunakan gambar tersebut maka nilai RMBR awal untuk tiap lokasi adalah sebagai berikut:

Tabel 3.6: Nilai Awal RMBR untuk Tiap Lokasi

id	min x	min y	max x	max y
l_1	120	171	120	171
l_2	180	77	180	77
l_3	257	303	257	303
l_4	175	435	175	435
l_5	432	173	432	173

Nilai RMBR pengguna akan dibiarkan kosong pada saat inisialisasi dan baru akan diperbarui pada tahapan selanjutnya.

3.2.1.5 Membandingkan Nilai RMBR Tiap *Node*

Nilai RMBR lokasi akan diperbarui dengan melihat daftar lokasi lain yang diikuti. Tiap nilai RMBR yang ada dalam daftar akan dibandingkan dengan RMBR lokasi itu sendiri untuk diperiksa apakah nilainya sudah benar. Apabila nilai RMBR belum benar maka nilainya akan diperbarui dengan nilai yang benar. Dengan melihat keterjangkauan tiap lokasi yang ada pada Gambar 3.3, maka nilai RMBR terbaru dari tiap lokasi dapat dilihat pada Tabel 3.7.

Tabel 3.7: Nilai RMBR Lokasi Setelah Diperbarui

id	min x	min y	max x	max y
l_1	120	77	180	435
l_2	175	77	180	435
l_3	175	77	432	435
l_4	175	435	175	435
l_5	175	77	432	435

Nilai RMBR pengguna akan ditentukan dari informasi *check-in* serta teman yang dimiliki. Terdapat tiga kondisi untuk menentukan nilai RMBR pengguna, yaitu:

1. Jika pengguna tidak memiliki teman tetapi pernah melakukan *check-in* ke satu atau beberapa lokasi maka RMBR akan ditentukan dari RMBR lokasi.
2. Jika pengguna tidak pernah melakukan *check-in* tetapi memiliki teman maka RMBR akan ditentukan dari RMBR teman pengguna yang tidak kosong.
3. Jika pengguna memiliki teman dan pernah melakukan *check-in* maka RMBR akan ditentukan dari RMBR lokasi serta teman pengguna.

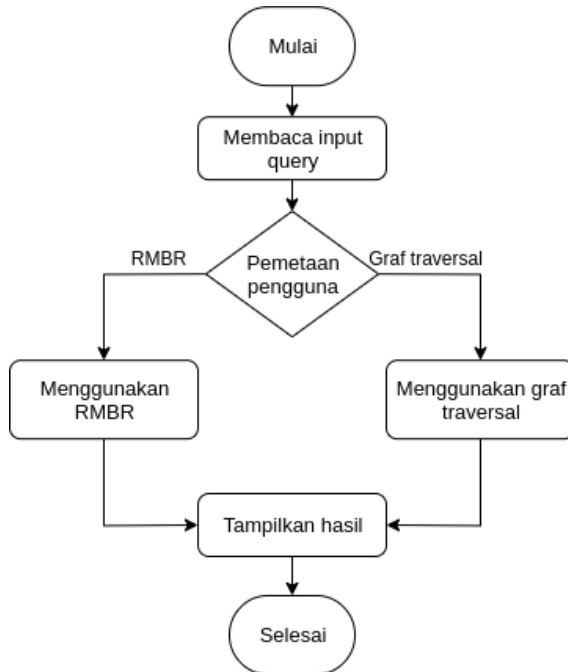
Dari setiap kondisi yang disebutkan, program akan membandingkan RMBR dari satu *node* ke *node* lainnya. Apabila nilai RMBR belum benar maka nilainya akan diperbarui dengan nilai yang benar. RMBR pengguna akan tetap kosong apabila tidak memenuhi satu dari tiga kondisi yang disebutkan. Nilai RMBR pengguna setelah dilakukan pembaruan dapat dilihat pada Tabel 3.8.

Tabel 3.8: Nilai RMBR Pengguna Setelah Diperbarui

id	min x	min y	max x	max y
p_1	120	77	432	435
p_2	120	77	180	435
p_3	175	77	432	435
p_4	-	-	-	-
p_5	175	77	432	435
p_6	-	-	-	-
p_7	175	435	175	435

3.2.2 Proses Utama

Pada tahap ini dilakukan proses utama yaitu pemetaan pengguna. Tahap ini memiliki alur utama seperti terlihat pada Gambar 3.4.



Gambar 3.4: Diagram Alur Proses Utama

3.2.2.1 *Input Query*

Input Query pada tugas akhir ini berupa koordinat yang merepresentasikan suatu lokasi maupun luasan area di mana penyerangan berlangsung.

3.2.2.2 **Pemetaan Pengguna**

Tahap ini adalah tahap di mana program akan melakukan pemetaan pengguna yang berpotensi dapat melakukan penyerangan. Pada tahap ini terdapat dua solusi yang ditawarkan untuk menyelesaikan permasalahan, yaitu solusi dengan menggunakan RMBR dan dengan menggunakan graf traversal. *User* dibebaskan memilih ingin melakukan pemetaan dengan menggunakan algoritma RMBR maupun graf traversal.

3.2.2.2.1 **RMBR-Based Solution**

Tahap penyelesaian dengan menggunakan RMBR memiliki alur seperti yang ditampilkan pada Gambar 3.5.

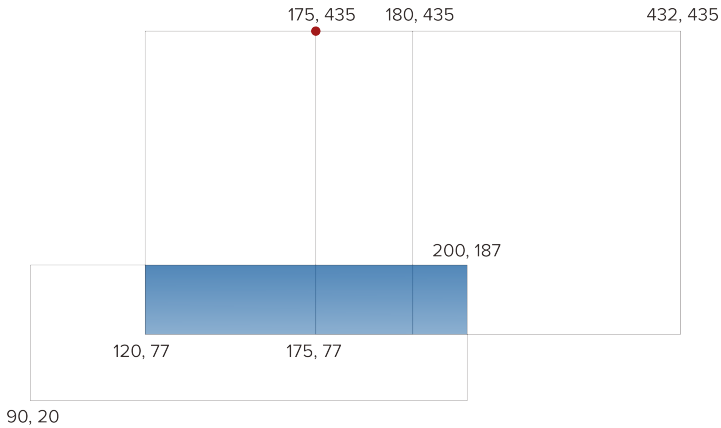
Penyelesaian dengan solusi ini memanfaatkan informasi RMBR yang dimiliki pengguna. Dari *input query* berupa koordinat spasial yang dimasukkan akan dicari RMBR pengguna yang beririsan. Daftar pengguna yang memiliki nilai RMBR beririsan akan disimpan untuk ditampilkan kepada *user*.



Gambar 3.5: Diagram Alur Penyelesaian dengan Menggunakan RMBR

Sebagai contoh *user* memasukkan (90, 20, 200, 187) sebagai *input query*. Dari area ini akan dicari RMBR pengguna mana saja yang beririsan dengan area tersebut. Gambar 3.6 dengan area berwarna biru menunjukkan RMBR mana saja yang beririsan dengan masukan pengguna. Nilai RMBR pada Gambar 3.6 didasarkan pada nilai RMBR yang telah terdapat pada Tabel 3.8.

Dari sini didapatkan daftar pengguna yang berpotensi melakukan penyerangan dengan menggunakan algoritma RMBR, yaitu pengguna dengan id sebagai berikut; p_1 , p_2 , p_3 , dan p_5 .

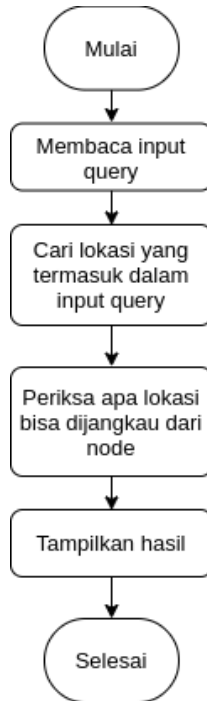


Gambar 3.6: Ilustrasi RMBR Pengguna yang Beririsan

Hasil dengan menggunakan algoritma ini hanya akan menampilkan daftar pengguna yang memiliki potensi melakukan penyerangan. Algoritma ini tidak akan menunjukkan seberapa dekat pengguna dari lokasi kejadian atau seberapa besar kemungkinan pengguna dapat melakukan penyerangan karena dalam algoritma ini program hanya akan memetakan pengguna berdasarkan nilai RMBR masing-masing.

3.2.2.2 Graph Traversal-Based Solution

Tahap penyelesaian dengan menggunakan graf traversal memiliki alur seperti yang ditampilkan pada Gambar 3.7.



Gambar 3.7: Diagram Alur Penyelesaian dengan Menggunakan Graf Traversal

Solusi ini akan memanfaatkan graf yang sudah dibuat pada tahap *preprocessing*. Langkah pertama yang dilakukan adalah dengan mencari daftar lokasi yang termasuk ke dalam *input query*. Daftar lokasi ini didapatkan dengan mencari lokasi mana saja yang memiliki nilai RMBR yang beririsan dengan *input query*. Setelah didapatkan daftar lokasi, graf akan ditelusuri dari lokasi-lokasi yang termasuk dalam daftar dan *node* yang bisa menjangkau lokasi akan disimpan beserta skor berupa banyaknya *hop* yang dibutuhkan untuk sampai ke lokasi.

Dengan menggunakan data RMBR lokasi yang ada pada Tabel 3.7, didapatkan empat lokasi yang memiliki nilai RMBR beririsan dengan *input query*, yaitu, l_1 , l_2 , l_3 , dan l_5 . Empat lokasi ini akan dijadikan titik awal di mana penelusuran graf akan dilakukan.

Setelah graf ditelusuri sebanyak empat kali maka akan didapatkan empat pengguna yang berpotensi melakukan penyerangan dengan nilai *hop* yang berbeda-beda dari satu pengguna ke tiap lokasi. Tabel 3.9 menunjukkan hasil pemetaan dengan menggunakan algoritma graf traversal.

Tabel 3.9: Hasil Pemetaan dengan Menggunakan Algoritma Graf Traversal

id lokasi	id pengguna	hop
l_1	p_2	1
l_1	p_1	2
l_2	p_2	2
l_2	p_3	2
l_2	p_1	3
l_2	p_5	3
l_3	p_5	1
l_3	p_1	2
l_5	p_3	1
l_5	p_1	2
l_5	p_5	2

Dengan menggunakan algoritma graf traversal, pengguna akan dipetakan dengan skor mereka atau disebut dengan hop. Nilai hop ini menggambarkan seberapa dekat pengguna dengan lokasi kejadian. Dari hasil pemetaan dengan menggunakan algoritma ini, seberapa besar kemungkinan pengguna dapat melakukan penyerangan dapat disimpulkan. Semakin kecil nilai hop, maka semakin besar kemungkinan pengguna dapat melakukan penyerangan.

(Halaman ini sengaja dikosongkan)

BAB IV

IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan sebelumnya. Pada bagian implementasi ini juga akan dijelaskan mengenai fungsi-fungsi yang digunakan dalam program tugas akhir ini dan disertai dengan pseudocode masing-masing fungsi utama. Implementasi dilakukan dalam bahasa pemrograman Python.

4.1 Lingkungan Implementasi Perangkat Lunak

Lingkungan implementasi adalah lingkungan di mana fitur temu kembali informasi dibangun. Lingkungan implementasi dibagi menjadi dua yaitu perangkat keras dan perangkat lunak.

4.1.1 Perangkat Keras

Spesifikasi perangkat keras yang digunakan dalam tugas akhir ini dapat dilihat pada Tabel 4.1.

Tabel 4.1: Tabel Perangkat Keras Implementasi Sistem

Spesifikasi	Deskripsi
Tipe	Asus A455L
Prosesor	Intel® Core™ i7-5500U CPU @ 2.40GHz x 4
Memori (RAM)	12 GB

4.1.2 Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam tugas akhir ini dapat dilihat pada Tabel 4.2.

Tabel 4.2: Tabel Perangkat Lunak Implementasi Sistem

Spesifikasi	Deskripsi
Sistem Operasi	Ubuntu 18.04.03 LTS
Bahasa Pemrograman	Python 3.6.9
Text Editor	Visual Studio Code 1.40.2

4.2 Implementasi Program *Preprocessing* Data

Subbab ini akan menjelaskan mengenai implementasi algoritma pembuatan grafik dan penentuan RMBR untuk tiap pengguna dan lokasi.

4.2.1 Pembuatan Graf

Algorithm 1: Algoritma Pembuatan Graf
<pre> for l <i>in</i> $location$ do for i <i>in</i> $follow_list$ do $loc_graph \leftarrow i$; $full_graph \leftarrow i$; end end for p <i>in</i> $user$ do for i <i>in</i> $friend_list$ do $full_graph \leftarrow i$; end for j <i>in</i> $checkin_list$ do $full_graph \leftarrow j$; end end </pre>

Pseudocode 4.1: Algoritma Pembuatan Graf

Dari dua graf yang dibuat, *loc_graph* akan digunakan untuk memperbarui nilai RMBR lokasi dan *full_graph* akan digunakan untuk penyelesaian solusi dengan graf traversal.

4.2.2 Update RMBR

Algorithm 2: Algoritma Traversal Graf Lokasi

```

let  $Q$  be list ;
while  $Q$  do
     $v = Q.pop()$ ;
    if  $v$  not in path then
        path[ $v$ ] = path + [ $v$ ];
         $Q = graph[v] + Q$ ;
    end
end
return path

```

Pseudocode 4.2: Algoritma Traversal Graf Lokasi

Pseudocode di atas menggunakan algoritma *depth-first search* untuk mendapatkan jalur terdalam untuk tiap lokasi. Algoritma ini akan mengunjungi *node* anak terlebih dahulu sebelum mengunjungi *node* pada tingkat yang sama. Algoritma ini digunakan untuk memperbarui nilai RMBR lokasi karena untuk mendapatkan nilai yang benar nilai RMBR lokasi dengan kedalaman paling jauh harus diperbarui terlebih dahulu sebelum lokasi yang paling luar.

Algorithm 3: *Algoritma Update RMBR*

```

for  $l$  in location do
  for  $i$  in follow_list do
    if  $l.rmbbr[min] > i.rmbbr[min]$  then
       $l.rmbbr[min] = i.rmbbr[min]$ ;
    end
    if  $l.rmbbr[max] < i.rmbbr[max]$  then
       $l.rmbbr[max] = i.rmbbr[max]$ ;
    end
  end
end
for  $p$  in user do
  if  $p.checkin > 0$  and  $p.friends = 0$  then
    for  $i$  in checkin_list do
       $p.rmbbr \leftarrow i.rmbbr$ ;
    end
  end
  if  $p.friends > 0$  and  $p.checkin = 0$  then
    for  $j$  in friend_list do
       $p.rmbbr \leftarrow j.rmbbr$ ;
    end
  end
  if  $p.rmbbr \neq \emptyset$  and  $p.friends > 0$  then
    for  $k$  in friend_list do
      if  $p.rmbbr[min] > k.rmbbr[min]$  then
         $p.rmbbr[min] = k.rmbbr[min]$ ;
      end
      if  $p.rmbbr[max] < k.rmbbr[max]$  then
         $p.rmbbr[max] = k.rmbbr[max]$ ;
      end
    end
  end
end

```

Pseudocode 4.3: *Algoritma Update RMBR*

Tiap lokasi pada awalnya memiliki nilai RMBR berdasarkan koordinat spasial masing-masing, nilai RMBR akan diperbarui dengan membandingkan nilai RMBR dari tiap lokasi yang diikuti suatu lokasi.

RMBR pengguna ditentukan dengan melihat daftar teman serta *check-in* dari masing-masing pengguna. Terdapat tiga kondisi untuk menentukan nilai RMBR pengguna, yaitu:

1. Jika pengguna tidak memiliki teman tetapi pernah melakukan *check-in* ke satu atau beberapa lokasi maka RMBR akan ditentukan dari RMBR lokasi.
2. Jika pengguna tidak pernah melakukan *check-in* tetapi memiliki teman maka RMBR akan ditentukan dari RMBR teman pengguna yang tidak kosong.
3. Jika pengguna memiliki teman dan pernah melakukan *check-in* maka RMBR akan ditentukan dari RMBR lokasi serta teman pengguna.

Nilai RMBR pengguna akan kosong apabila tidak memenuhi satu dari tiga kondisi yang sudah disebutkan.

4.3 Implementasi Program Utama

Algoritma pada proses utama diimplementasikan sesuai dengan rancangan sistem yang sudah dibahas pada bab desain dan perancangan. Tujuan dari proses utama adalah memetakan pengguna yang berpotensi melakukan penyerangan. Terdapat dua solusi yang ditawarkan untuk menyelesaikan permasalahan, yaitu solusi dengan menggunakan RMBR dan solusi dengan menggunakan graf traversal.

4.3.1 Pemetaan Pengguna dengan Menggunakan RMBR

Algorithm 4: Algoritma Pemetaan Pengguna dengan Menggunakan RMBR

```

for  $p$  in user do
  if  $p.rnbr \neq \emptyset$  then
    if  $p.rnbr[\textit{min}] > \textit{input}[\textit{max}]$  or  $p.rnbr[\textit{max}] < \textit{input}[\textit{min}]$ 
      then
        | continue;
      end
    suspect  $\leftarrow p$ ;
  end
end
return suspect

```

Pseudocode 4.4: Algoritma Pemetaan Pengguna dengan Menggunakan RMBR

Dari tahapan sebelumnya, setiap pengguna akan memiliki nilai RMBR berdasarkan daftar teman serta lokasi yang pernah dikunjungi. Dengan menggunakan solusi ini tiap RMBR pengguna akan diperiksa apakah beririsan dengan area masukan atau tidak. Jika RMBR pengguna beririsan maka pengguna akan dimasukkan ke daftar pengguna yang berpotensi melakukan penyerangan.

4.3.2 Pemetaan Pengguna dengan Menggunakan Graf Traversal

Sebelum memetakan pengguna yang berpotensi melakukan penyerangan, program akan mencari lokasi mana saja yang termasuk ke dalam area masukan. Lokasi akan masuk ke daftar lokasi kejadian apabila nilai RMBR lokasi beririsan dengan area masukan.

Algorithm 5: Algoritma untuk Mendapatkan Daftar Lokasi Kejadian

```

for  $l$  in location do
  | if  $l.rnbr[min] > input[max]$  or  $l.rnbr[max] < input[min]$  then
  |   | continue;
  | end
  |  $sites \leftarrow l$ ;
end

```

Pseudocode 4.5: Algoritma untuk Mendapatkan Daftar Lokasi Kejadian

Setiap lokasi yang termasuk ke dalam daftar lokasi kejadian akan dijadikan *start point* untuk fungsi pemetaan ini. Fungsi ini akan menelusuri graf dari *start point* dan menyimpan nilai hop untuk setiap *node* yang dapat dijangkau.

Algorithm 6: Algoritma Pemetaan Pengguna dengan Menggunakan Graf Traversal

```

let path and Q be dict ;
Q[start] = 0;
while Q do
    v, hop = Q.popitem();
    if not v in path then
        path[v] = hop;
        for neighbor in graph[v] do
            if neighbor not in path, q then
                q[neighbor] = hop + 1;
            end
        end
    end
end
end

```

Pseudocode 4.6: Algoritma Pemetaan Pengguna dengan Menggunakan Graf Traversal

(Halaman ini sengaja dikosongkan)

BAB V

PENGUJIAN DAN EVALUASI

Bab ini akan membahas tahap pengujian dan evaluasi sistem yang sudah dibangun berdasarkan desain dan perancangan pada bab sebelumnya. Hasil yang didapatkan dari tahap pengujian akan dievaluasi sehingga dapat didapatkan kesimpulan untuk bab selanjutnya.

5.1 Lingkungan Pengujian

Pada proses pengujian perangkat lunak, dibutuhkan suatu lingkungan pengujian yang sesuai dengan standar kebutuhan. Lingkungan pengujian dalam tugas akhir ini adalah sama untuk setiap kasus. Spesifikasi dari lingkungan pengujian dapat dilihat pada Tabel 5.1.

Tabel 5.1: Lingkungan Pengujian

Spesifikasi	Deskripsi
Prosesor	Intel® Core™ i7-5500U CPU @ 2.40GHz x 4
Arsitektur Prosesor	64 bit
Sistem Operasi	Ubuntu 18.04.03 LTS
Memori (RAM)	12 GB

5.2 Jenis Data Pengujian

Pada pengujian ini dataset yang digunakan adalah data buatan. Koordinat spasial untuk *node* lokasi dan jumlah *edge* akan diacak dalam rentang yang telah ditentukan. Selain itu data acak juga akan digunakan sebagai input pengguna.

5.3 Skenario Pengujian

Skenario uji coba akan dibedakan menjadi dua jenis, yaitu uji coba fungsionalitas dan uji coba performa. Uji coba fungsionalitas bertujuan untuk menguji apakah sistem yang dibuat sudah berhasil menjawab kebutuhan fungsional. Sedangkan uji coba performa digunakan untuk mengetahui perbandingan dua solusi yang ditawarkan untuk menyelesaikan permasalahan pada tugas akhir ini.

5.3.1 Uji Coba Fungsionalitas

Uji coba fungsionalitas adalah tahap pengujian untuk memeriksa apakah aplikasi dapat digunakan sesuai dengan kebutuhan fungsional. Kebutuhan fungsional dapat dilihat pada Tabel 5.2.

Tabel 5.2: Kebutuhan Fungsional

Kode	Deskripsi Kebutuhan
F01	Menerima masukan <i>dataset</i>
F02	Menerima masukan titik kueri
F03	Menghasilkan keluaran berupa file yang berisi daftar pengguna yang berpotensi melakukan penyerangan

5.3.2 Uji Coba Performa

Skenario uji coba performa dilakukan dengan cara menguji masing-masing algoritma dan mencatat waktu beserta memori yang digunakan. Pengujian dilakukan sebanyak 10 kali untuk tiap skenario.

Tabel 5.3: Graf Pengujian

Jumlah Node	Jumlah Edge
1000	5449
2000	10371
5000	26108
10000	52115

5.3.2.1 Parameter Pengujian

Pada uji coba performa akan dilakukan beberapa skenario pengujian dengan parameter yang berbeda-beda seperti yang dapat dilihat pada daftar berikut:

1. Skenario Variasi Jumlah *Node*

Tabel 5.4: Skenario Variasi Jumlah *Node*

Kode	Jumlah Node
A01	1000
A02	2000
A03	5000
A04	10000

2. Skenario Rasio Jumlah Pengguna dan Lokasi

Dari total *node*, rasio jumlah *node* pengguna dan lokasi akan dibuat seperti yang dapat dilihat pada Tabel 5.5.

Tabel 5.5: Skenario Rasio Jumlah Pengguna dan Lokasi

Kode	Jumlah <i>Node</i>	Pengguna	Lokasi
B01	5000	0.25	0.75
B02	5000	0.50	0.50
B01	5000	0.75	0.25

3. Skenario Jumlah *Edge* Pengguna

Tiap pengguna memiliki jumlah *edge* yang acak dalam rentang jumlah minimal hingga maksimal. Skenario ini akan membatasi jumlah maksimal *edge* tiap pengguna seperti yang dapat dilihat pada Tabel 5.6.

Tabel 5.6: Skenario Jumlah *Edge* Pengguna

Kode	Jumlah <i>Node</i>	Jumlah <i>Edge</i>
C01	5000	5
C02	5000	10
C03	5000	20
C04	5000	30
C05	5000	40

5.4 Analisis Hasil Uji Coba

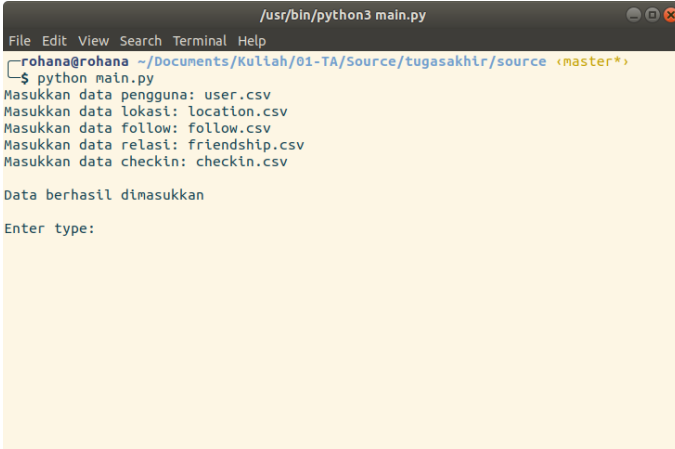
Bagian ini akan menjelaskan hasil uji coba dan analisis terhadap hasil tersebut. Terdapat dua hasil uji coba yang akan ditampilkan, yaitu hasil uji coba fungsionalitas dan hasil uji coba performa.

5.4.1 Hasil Uji Coba Fungsionalitas

Skenario uji coba fungsionalitas dilakukan berdasarkan skenario pada Tabel 5.2. Hasil uji coba dapat dilihat pada Tabel 5.7.

Tabel 5.7: Hasil Uji Coba Kebutuhan Fungsional

Kode	Keterangan
F01	Berhasil
F02	Berhasil
F03	Berhasil



```
~/usr/bin/python3 main.py
File Edit View Search Terminal Help
rohana@rohana ~/Documents/Kuliah/01-TA/Source/tugasakhir/source (master*)
$ python main.py
Masukkan data pengguna: user.csv
Masukkan data lokasi: location.csv
Masukkan data follow: follow.csv
Masukkan data relasi: friendship.csv
Masukkan data checkin: checkin.csv

Data berhasil dimasukkan

Enter type:
```

Gambar 5.1: Hasil Uji Coba F01

```
~/usr/bin/python3 main.py
File Edit View Search Terminal Help

Enter type: rnbr
----- Masukkan Area -----
Enter min_x: 200
Enter min_y: 172
Enter max_x: 200
Enter max_y: 172

Hasil dapat dilihat di hasil_rnbr.csv
-----

Enter type: graf
----- Masukkan Area -----
Enter min_x: 200
Enter min_y: 172
Enter max_x: 200
Enter max_y: 172

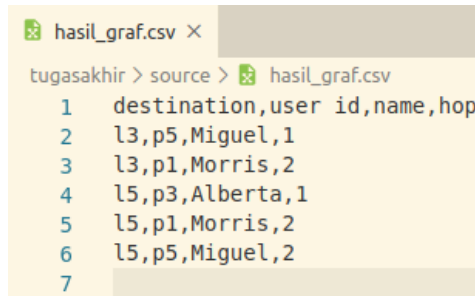
Hasil dapat dilihat di hasil_graf.csv
-----

Enter type:
```

Gambar 5.2: Hasil Uji Coba F02

```
hasil_rnbr.csv ×
tugasakhir > source > hasil_rnbr.csv
1 user id,name
2 | p1,Morris
3 | p3,Alberta
4 | p5,Miguel
5 |
```

Gambar 5.3: Hasil Uji Coba F03



```

hasil_graf.csv ×
tugasakhir > source > hasil_graf.csv
1 destination,user id,name,hop
2 l3,p5,Miguel,1
3 l3,p1,Morris,2
4 l5,p3,Alberta,1
5 l5,p1,Morris,2
6 l5,p5,Miguel,2
7

```

Gambar 5.4: Hasil Uji Coba F03

5.4.2 Hasil Uji Coba Performa

Skenario uji coba performa dilakukan berdasarkan Tabel 5.4, Tabel 5.5, dan Tabel 5.6. Performa yang dilihat adalah waktu yang dibutuhkan untuk menjalankan program dan besar memori yang digunakan.

5.4.2.1 *Preprocessing Data*

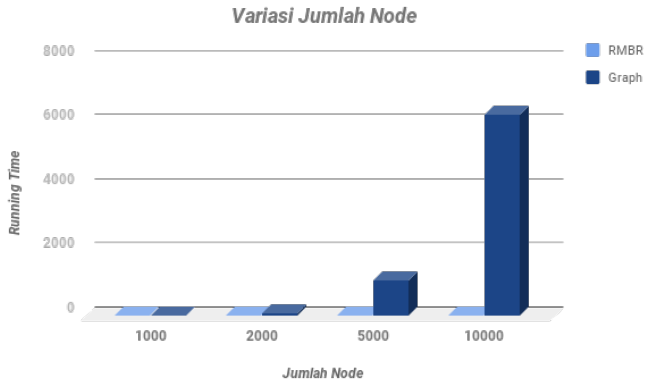
Tabel 5.8 menampilkan rata-rata waktu *preprocessing* untuk variasi jumlah *node*. Dari tabel tersebut dapat disimpulkan bahwa semakin banyak jumlah *node*, maka akan semakin lama pula waktu *preprocessing* yang dibutuhkan.

Tabel 5.8: Rata-rata Waktu *Preprocessing*

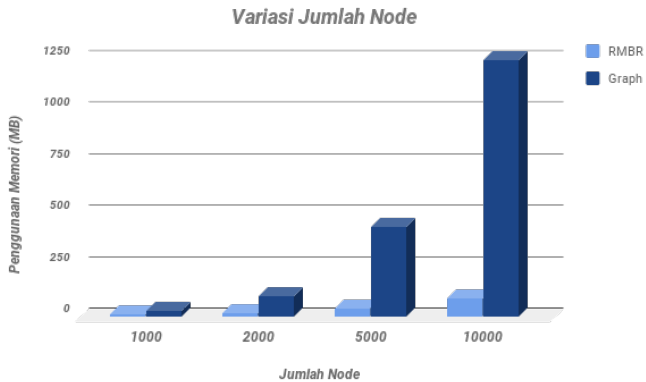
Jumlah Node	Waktu <i>Preprocessing</i> (detik)
1000	6.32
2000	26.28
5000	285.71
10000	1660.18

5.4.2.2 Program Utama

1. Skenario Variasi Jumlah *Node*



Gambar 5.5: Grafik Perbandingan Waktu Eksekusi dengan Jumlah *Node*



Gambar 5.6: Grafik Perbandingan Penggunaan Memori dengan Jumlah *Node*

Tabel 5.9: Perbandingan Waktu Eksekusi dengan Jumlah *Node*

Kode	RMBR	Graph
A01	0.000939 s	8.655362 s
A02	0.002842 s	79.90775 s
A03	0.005395 s	1113.099 s
A04	0.008528 s	6291.224 s

Tabel 5.10: Perbandingan Penggunaan Memori dengan Jumlah *Node*

Kode	RMBR	Graph
A01	15.11 MB	32.563 MB
A02	18.90 MB	103.69 MB
A03	40.27 MB	439.01 MB
A04	90.09 MB	1244.8 MB

Tabel 5.11: Perbandingan Jumlah Pengguna pada Hasil Tiap Skenario

A01	A02	A03	A04
476	967	1099	1085
482	965	1287	2055
476	961	2379	4848
482	972	2429	4706
482	971	2424	4709
476	968	2379	4815
481	957	2412	4826
485	957	2414	4706
482	966	2416	4716
474	973	2418	4781

Tabel 5.12: Perbandingan Jumlah Lokasi pada Hasil Tiap Skenario dengan Menggunakan Algoritma Graf Traversal

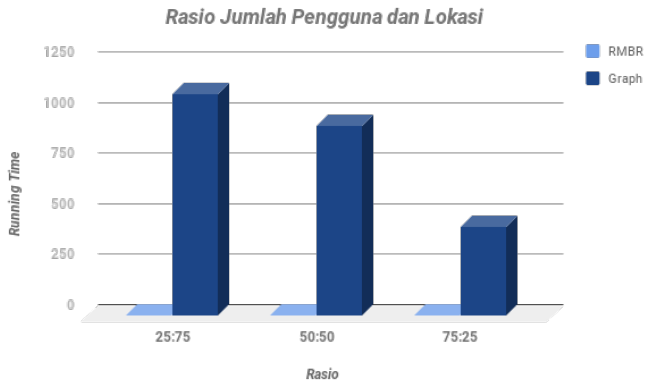
A01	A02	A03	A04
287	569	1168	1980
230	494	1237	2232
250	520	1204	1988
243	525	1370	2326
240	529	1498	2628
250	579	1329	2489
255	533	1535	3361
274	481	1411	2256
286	609	1481	3199
273	587	1496	2610

Berdasarkan Tabel 5.9 dapat dilihat bahwa waktu eksekusi dengan menggunakan RMBR memerlukan waktu rata-rata kurang dari satu detik untuk tiap skenario. Pada uji coba dengan menggunakan graf traversal, hasil menunjukkan bahwa lamanya waktu eksekusi bervariasi tergantung dengan jumlah *node*. Semakin banyak jumlah *node* maka semakin besar pula waktu yang dibutuhkan.

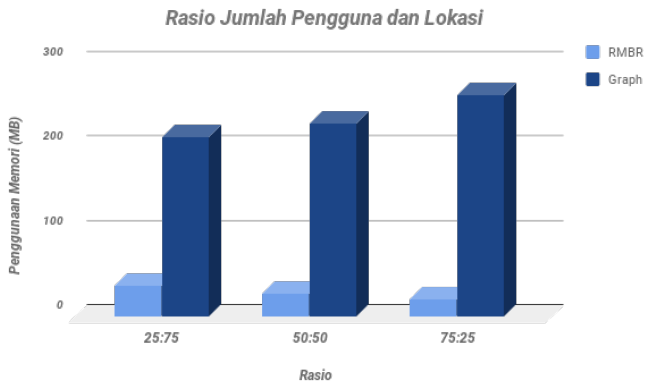
Pada Tabel 5.10 dapat dilihat bahwa penggunaan memori kedua algoritma sama-sama dipengaruhi oleh banyaknya jumlah *node*. Semakin banyak jumlah *node*, maka semakin besar pula memori yang digunakan.

Dari hasil uji coba skenario ini dapat disimpulkan bahwa jumlah *node* mempengaruhi waktu eksekusi serta penggunaan memori terutama pada algoritma dengan menggunakan graf. Hal ini dikarenakan kemungkinan jumlah lokasi yang termasuk ke daftar tempat kejadian akan semakin banyak sehingga program harus menelusuri graf sebanyak tempat kejadian pula.

2. Skenario Rasio Jumlah Pengguna dan Lokasi



Gambar 5.7: Grafik Perbandingan Waktu Eksekusi dengan Rasio Jumlah Pengguna dan Lokasi



Gambar 5.8: Grafik Penggunaan Memori dengan Rasio Jumlah Pengguna dan Lokasi

Tabel 5.13: Perbandingan Waktu Eksekusi dengan Rasio Jumlah Pengguna dan Lokasi

Kode	RMBR	Graph
B01	0.0017 s	1094.31 s
B02	0.0039 s	940.669 s
B03	0.0090 s	441.749 s

Tabel 5.14: Perbandingan Penggunaan Memori dengan Rasio Jumlah Pengguna dan Lokasi

Kode	RMBR	Graph
B01	36.59 MB	212.50 MB
B02	26.87 MB	229.90 MB
B03	20.86 MB	263.04 MB

Tabel 5.15: Perbandingan Jumlah Pengguna pada Hasil Tiap Skenario

B01	B02	B03
859	822	1109
781	973	1165
1190	777	1475
1180	1169	3649
1197	2372	3648
1167	2400	3570
1192	2424	3612
1190	2412	3658
1208	2415	3629
1173	2405	3573

Tabel 5.16: Perbandingan Jumlah Lokasi pada Hasil Tiap Skenario dengan Menggunakan Algoritma Graf Traversal

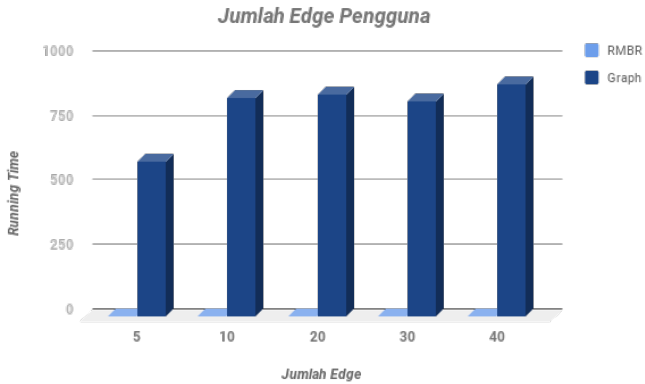
B01	B02	B03
1518	941	509
1547	992	514
1475	1063	542
1606	1153	816
1803	1389	529
1899	1195	549
1519	1054	508
1513	1019	676
1515	1050	640
1629	1027	576

Berdasarkan Tabel 5.13 dan Tabel 5.14 dapat dilihat bahwa waktu eksekusi serta pengguna memori dengan menggunakan RMBR dipengaruhi dengan banyaknya jumlah *node* pengguna.

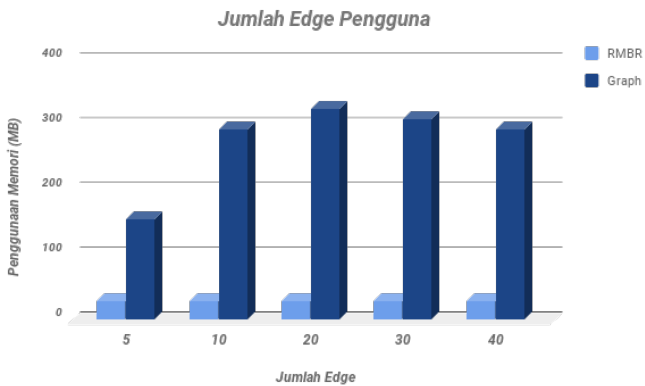
Pada uji coba dengan menggunakan graf traversal, Tabel 5.13 dan Tabel 5.14 menunjukkan bahwa lamanya waktu eksekusi serta penggunaan memori dipengaruhi dengan banyaknya jumlah *node* lokasi.

Dari hasil uji coba skenario ini dapat disimpulkan bahwa performa dengan menggunakan RMBR dipengaruhi oleh jumlah *node* pengguna sedangkan performa dengan menggunakan graf traversal dipengaruhi oleh jumlah *node* lokasi. Hal ini dikarenakan pada penyelesaian dengan menggunakan RMBR, iterasi dilakukan sebanyak jumlah pengguna. Sedangkan pada graf traversal penelusuran graf dilakukan sebanyak jumlah lokasi yang ada pada daftar tempat kejadian.

3. Skenario Jumlah *Edge* Pengguna



Gambar 5.9: Grafik Perbandingan Waktu Eksekusi dengan Jumlah *Edge* Pengguna



Gambar 5.10: Grafik Penggunaan Memori dengan Jumlah *Edge* Pengguna

Tabel 5.17: Perbandingan Waktu Eksekusi dengan Jumlah *Edge* Pengguna

Kode	RMBR	Graph
C01	0.0033 s	602.17 s
C02	0.0033 s	848.79 s
C03	0.0057 s	859.19 s
C04	0.0058 s	833.18 s
C05	0.0071 s	901.84 s

Tabel 5.18: Perbandingan Penggunaan Memori dengan Jumlah *Edge* Pengguna

Kode	RMBR	Graph
C01	28.75 MB	155.53 MB
C02	28.39 MB	294.58 MB
C03	29.39 MB	325.41 MB
C04	28.02 MB	310.07 MB
C05	28.79 MB	293.88 MB

Tabel 5.19: Perbandingan Jumlah Pengguna pada Hasil Tiap Skenario

C01	C02	C03	C04	C05
1035	2163	667	1463	2453
1121	2187	1027	1337	2454
1098	2162	1099	1824	2476
1111	2188	1287	2446	2466
1249	2216	2379	2419	2455
1143	2175	2411	2424	2475
1177	2181	2414	2450	2478
1135	2246	2381	2451	2461
1239	2219	2380	2459	2472
1221	2195	2419	2460	2474

Tabel 5.20: Perbandingan Jumlah Lokasi pada Hasil Tiap Skenario dengan Menggunakan Algoritma Graf Traversal

C01	C02	C03	C04	C05
1167	1158	1169	1090	1134
1138	1134	1196	1036	1138
1151	1122	1189	1035	1144
1155	1074	1204	1452	1157
1167	1514	1325	1455	1216
1512	1264	1228	1159	1510
1250	1539	1209	1343	1492
1354	1432	1284	1294	1230
1394	1098	1326	1414	1241
1433	1487	1171	1171	1367

Pada Tabel 5.17 dapat dilihat bahwa waktu eksekusi kedua algoritma ini tidak terlalu dipengaruhi oleh jumlah *edge* pengguna. Perubahan waktu eksekusi yang paling signifikan terjadi pada algoritma dengan menggunakan graf traversal dari skenario C01 ke C02.

Pada penggunaan memori, Tabel 5.18 menunjukkan bahwa kedua algoritma ini juga tidak terlalu dipengaruhi oleh jumlah *edge* pengguna. Perbedaan penggunaan memori yang paling signifikan terjadi pada algoritma dengan menggunakan graf traversal dari skenario C01 ke C02.

Dari hasil uji coba ini dapat disimpulkan bahwa jumlah *edge* tidak berpengaruh secara signifikan pada kedua algoritma penyelesaian. Penyelesaian dengan menggunakan RMBR tidak dipengaruhi karena program tidak perlu memerhatikan *edge* pengguna saat tahap pemetaan.

Dari hasil yang tersedia, jumlah *node* adalah parameter yang paling mempengaruhi performa kedua algoritma. Jumlah *node* pengguna berpengaruh pada algoritma dengan menggunakan RMBR. Hal ini dikarenakan program perlu melakukan iterasi sebanyak jumlah pengguna untuk memeriksa nilai RMBR masing-masing pengguna. Jumlah *node* lokasi berpengaruh pada algoritma dengan menggunakan graf traversal. Hal ini dikarenakan kemungkinan jumlah lokasi yang termasuk ke daftar tempat kejadian semakin banyak sehingga program harus menelusuri graf sebanyak tempat kejadian pula.

(Halaman ini sengaja dikosongkan)

BAB VI

KESIMPULAN DAN SARAN

Bab ini akan memaparkan kesimpulan yang bisa diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1 Kesimpulan

1. Terdapat dua algoritma yang diajukan untuk menyelesaikan permasalahan, yaitu algoritma dengan menggunakan RMBR dan algoritma dengan menggunakan graf traversal.
2. Proses pemetaan pengguna pada algoritma dengan menggunakan RMBR dilakukan dengan mencari RMBR pengguna yang beririsan dengan area yang dimasukkan. Untuk algoritma dengan menggunakan graf traversal pertama program akan mencari lokasi yang masuk ke daftar tempat kejadian kemudian dari daftar tersebut graf akan ditelusuri sebanyak jumlah lokasi yang ada pada daftar.
3. Hasil performa algoritma dengan menggunakan RMBR jauh lebih baik dibandingkan dengan graf traversal, namun algoritma ini hanya akan menampilkan daftar pengguna yang berpotensi melakukan penyerangan tanpa mengetahui seberapa besar kemungkinan pengguna tersebut bisa melakukan penyerangan. Hasil performa algoritma dengan menggunakan graf traversal termasuk buruk, namun dari algoritma ini kedekatan pengguna dan lokasi dapat diketahui sehingga seberapa besar kemungkinan pengguna bisa melakukan penyerangan dapat disimpulkan.

6.2 Saran

1. Menerapkan metode threading agar proses yang dilakukan dapat berjalan secara paralel sehingga dapat meningkatkan performa dalam segi waktu eksekusi program.
2. Pembaruan nilai RMBR lokasi pada tahap *preprocessing* dilakukan dengan menggunakan graf traversal. Hal ini tentu saja tidak efisien. Diperlukan cara yang lebih efisien untuk meningkatkan performa pada tahap *preprocessing*.
3. Menentukan parameter lain untuk pemetaan pengguna supaya hasil lebih akurat.

DAFTAR PUSTAKA

- [1] N. Armenatzoglou, S. Papadopoulos, dan D. Papadias, “A General Framework for Geo-Social Query Processing,” *PVLDB*, 2013.
- [2] R. R. Veloso, L. Cerf, W. M. Jr., dan M. J. Zaki, “Reachability Queries in Very Large Graphs: A Fast Refined Online Search Approach,” in *EDBT*, 2014.
- [3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, dan C. Stein, *Introduction to Algorithm*. MIT Press, 2009, bab. 22.3.
- [4] —, *Introduction to Algorithm*. MIT Press, 2009, bab. 22.2.
- [5] “data structure,” 27 November 2019. [Daring]. Tersedia pada: <https://xlinux.nist.gov/dads/HTML/datastructur.html/>. [Diakses: 27 November 2019].
- [6] “Data Structure,” 27 November 2019. [Daring]. Tersedia pada: <https://www.britannica.com/technology/data-structure>. [Diakses: 27 November 2019].
- [7] “General Python FAQ,” 23 November 2019. [Daring]. Tersedia pada: <https://docs.python.org/3/faq/general.html#what-is-python>. [Diakses: 23 November 2019].
- [8] M. Sarwat dan Y. Sun, “Answering Location-Aware Graph Reachability Queries on Geosocial Data,” *IEEE*, 2017.
- [9] J. Shi, N. Mamoulis, D. Wu, dan D. W. Cheung, “Density-based Place Clustering in Geo-Social Networks,” *IEEE*, 2018.

(Halaman ini sengaja dikosongkan)

LAMPIRAN A

KODE SUMBER

```
1 import random
2 import names
3 import sys
4 import csv
5
6 try:
7     num_of_rows = int(sys.argv[1])
8 except IndexError:
9     num_of_rows = 100
10
11 header = ["src", "dst"]
12
13 def generate_user():
14     header = ["id", "name"]
15     data = []
16     data.append(header)
17     for i in range(num_of_rows):
18         tmp = ["p"+str(i+1), names.get_first_name()]
19         data.append(tmp)
20
21     csvfile = "../dataset/user.csv"
22     with open(csvfile, "w") as output:
23         writer = csv.writer(output, lineterminator='\n')
24         writer.writerows(data)
25
26 def generate_location():
27     header = ["id", "name", "latitude", "longitude"]
28     data = []
29     data.append(header)
30     for i in range(num_of_rows):
31         rand_a = random.randint(0, num_of_rows)
32         rand_b = random.randint(0, num_of_rows)
33         tmp = ["l"+str(i+1), names.get_last_name(), rand_a,
34             rand_b]
35         data.append(tmp)
36
37     csvfile = "../dataset/location.csv"
38     with open(csvfile, "w") as output:
39         writer = csv.writer(output, lineterminator='\n')
```

```

39         writer.writerow(data)
40
41 def generate_p_to_p():
42     users = []
43     data = []
44     data.append(header)
45     for i in range(num_of_rows):
46         u = "p"+str(i+1)
47         users.append(u)
48     for p in users:
49         n_rel = random.randint(0, 20)
50         p_to_p = int(0.7 * n_rel)
51         for i in range(p_to_p):
52             tmp = [p, "p"+str(random.randint(1, num_of_rows)
53 )]
54             data.append(tmp)
55     csvfile = "../dataset/friendship.csv"
56     with open(csvfile, "w") as output:
57         writer = csv.writer(output, lineterminator='\n')
58         writer.writerow(data)
59
60 def generate_p_to_l():
61     users = []
62     data = []
63     data.append(header)
64     for i in range(num_of_rows):
65         u = "p"+str(i+1)
66         users.append(u)
67     for p in users:
68         n_rel = random.randint(0, 20)
69         p_to_l = int(0.3 * n_rel)
70         for i in range(p_to_l):
71             tmp = [p, "l"+str(random.randint(1, num_of_rows)
72 )]
73             data.append(tmp)
74     csvfile = "../dataset/checkin.csv"
75     with open(csvfile, "w") as output:
76         writer = csv.writer(output, lineterminator='\n')
77         writer.writerow(data)
78
79 def generate_l_to_l():

```

```

78     places = []
79     data = []
80     data.append(header)
81     for i in range(num_of_rows):
82         u = "1"+str(i+1)
83         places.append(u)
84     for l in places:
85         n_rel = random.randint(0, 5)
86         l_to_l = int(0.7 * n_rel)
87         for i in range(l_to_l):
88             tmp = [l, "1"+str(random.randint(1, num_of_rows)
89 )]
90             data.append(tmp)
91     csvfile = "../dataset/follow.csv"
92     with open(csvfile, "w") as output:
93         writer = csv.writer(output, lineterminator='\n')
94         writer.writerows(data)
95     def generate_query():
96         data = []
97         for i in range(20):
98             rand_x1 = random.randint(0, num_of_rows)
99             rand_y1 = random.randint(0, num_of_rows)
100            rand_x2 = random.randint(0, num_of_rows)
101            rand_y2 = random.randint(0, num_of_rows)
102            tmp = [rand_x1, rand_y1, rand_x2, rand_y2]
103            data.append(tmp)
104            csvfile = "../dataset/query.csv"
105            with open(csvfile, "w") as output:
106                writer = csv.writer(output, lineterminator='\n')
107                writer.writerows(data)
108
109 if __name__ == '__main__':
110     generate_user()
111     generate_location()
112     generate_p_to_p()
113     generate_p_to_l()
114     generate_l_to_l()
115     generate_query()

```

Kode Sumber 1.1: Kode Sumber Generate Data

```
1 import csv
2 import math
3 import time
4 import os
5 import sys
6 import psutil
7 from collections import defaultdict
8 from collections import OrderedDict
9
10 class User:
11     def __init__(self, id, name):
12         self.id = id
13         self.name = name
14         self.friends = list()
15         self.checkin = list()
16         self.rmbr = dict()
17
18     def add_friends(self, user_id):
19         self.friends.append(user_id)
20
21     def add_checkin(self, location_id):
22         self.checkin.append(location_id)
23
24 class Location:
25     def __init__(self, id, name, lat, long):
26         self.id = id
27         self.name = name
28         self.lat = lat
29         self.long = long
30         self.follow = list()
31         self.score = dict()
32         self.rmbr = {
33             'min_x': lat,
34             'min_y': long,
35             'max_x': lat,
36             'max_y': long
37         }
38
39     def add_score(self, id, score):
40         self.score[id] = score
41
```

```

42     def add_follow(self, location_id):
43         self.follow.append(location_id)
44
45     class Graph:
46         def __init__(self):
47             self.user = list()
48             self.location = list()
49             self.loc_graph = defaultdict(list)
50             self.full_graph = defaultdict(list)
51
52         def insert_user(self, file):
53             users = dict()
54             with open(file, 'r') as csv_file:
55                 csv_reader = csv.DictReader(csv_file, delimiter=
56                 ",")
57                 for row in csv_reader:
58                     users[row[ csv_reader.fieldnames[0]]] = User(
59                     row[ csv_reader.fieldnames[0]], row[ csv_reader.fieldnames
60                     [1]])
61                     self.user.append(users[row[ csv_reader.
62                     fieldnames[0]])
63
64         def insert_location(self, file):
65             places = dict()
66             with open(file, 'r') as csv_file:
67                 csv_reader = csv.DictReader(csv_file, delimiter=
68                 ",")
69                 for row in csv_reader:
70                     places[row[ csv_reader.fieldnames[0]]] =
71                     Location((row[ csv_reader.fieldnames[0]], row[ csv_reader
72                     .fieldnames[1]], int(row[ csv_reader.fieldnames[2]]), int
73                     (row[ csv_reader.fieldnames[3]]))
74                     self.location.append(places[row[ csv_reader.
75                     fieldnames[0]])
76
77         def insert_friends(self, file):
78             with open(file, 'r') as csv_file:
79                 csv_reader = csv.DictReader(csv_file, delimiter=
80                 ",")
81                 for row in csv_reader:
82                     for p in self.user:

```

```

73         # if id csv = id obj
74         if row[csv_reader.fieldnames[0]] == p.id
75         :
76             # self loop
77             if row[csv_reader.fieldnames[1]] ==
78 p.id:
79                 del row[csv_reader.fieldnames
80 [1]]
81                 else:
82                     # add obj to friend list
83                     for i in self.user:
84                         if row[csv_reader.fieldnames
85 [1]] == i.id:
86                             p.add_friends(i)
87                             # remove redundant value
88                             p.friends = list(dict.fromkeys(p.friends
89 ))
90
91 def insert_checkin(self, file):
92     with open(file, 'r') as csv_file:
93         csv_reader = csv.DictReader(csv_file, delimiter=
94         ",")
95         for row in csv_reader:
96             for p in self.user:
97                 # if id csv = id obj
98                 if row[csv_reader.fieldnames[0]] == p.id
99         :
100             # add obj to checkin list
101             for i in self.location:
102                 if row[csv_reader.fieldnames[1]]
103 == i.id:
104                 p.add_checkin(i)
105                 # remove redundant value
106                 p.checkin = list(dict.fromkeys(p.checkin
107 ))
108
109 def insert_follow(self, file):
110     with open(file, 'r') as csv_file:
111         csv_reader = csv.DictReader(csv_file, delimiter=
112         ",")
113         for row in csv_reader:

```

```

104         for l in self.location:
105             # if id csv = id obj
106                 if row[csv_reader.fieldnames[0]] == l.id
:
107                 # self loop
108                 if row[csv_reader.fieldnames[1]] ==
l.id:
109                     del row[csv_reader.fieldnames
[1]]
110                 else:
111                     # add obj to follow list
112                     for i in self.location:
113                         if row[csv_reader.fieldnames
[1]] == i.id:
114                             l.add_follow(i)
115                             # remove redundant value
116                             l.follow = list(dict.fromkeys(l.follow))
117
118     def build_graph(self):
119         # location
120         for l in self.location:
121             if len(l.follow) > 0:
122                 for i in l.follow:
123                     self.loc_graph[l.id].append(i.id)
124
125         # full graph
126         for p in self.user:
127             if len(p.friends) > 0:
128                 for f in p.friends:
129                     self.full_graph[f.id].append(p.id)
130                     # self.full_graph[p.id].append(f.id)
131             if len(p.checkin) > 0:
132                 for l in p.checkin:
133                     self.full_graph[l.id].append(p.id)
134                     # self.full_graph[p.id].append(l.id)
135         for l in self.location:
136             if len(l.follow) > 0:
137                 for i in l.follow:
138                     self.full_graph[i.id].append(l.id)
139                     # self.full_graph[l.id].append(i.id)
140

```



```

197         for p in self.user:
198             if bool(p.rnbr) != False and len(p.friends)
> 0:
199                 for i in p.friends:
200                     if bool(i.rnbr) != False:
201                         if p.rnbr['min_x'] > i.rnbr['
min_x']:
202                             p.rnbr['min_x'] = i.rnbr['
min_x']
203                         if p.rnbr['min_y'] > i.rnbr['
min_y']:
204                             p.rnbr['min_y'] = i.rnbr['
min_y']
205                         if p.rnbr['max_x'] < i.rnbr['
max_x']:
206                             p.rnbr['max_x'] = i.rnbr['
max_x']
207                         if p.rnbr['max_y'] < i.rnbr['
max_y']:
208                             p.rnbr['max_y'] = i.rnbr['
max_y']
209
210     def total_user(self):
211         return len(self.user)
212
213     def total_location(self):
214         return len(self.location)
215
216     def get_path(graph, start, path=[]):
217         q = [start]
218         while q:
219             v = q.pop(0)
220             if v not in path:
221                 path = path + [v]
222                 q = graph[v] + q
223         return path
224
225     def get_score(graph, start):
226         path = dict()
227         q = OrderedDict()
228         q[start] = 0

```



```

261
262 def get_suspect_rnbr(area, graph):
263     suspect = list()
264     for p in graph.user:
265         if bool(p.rnbr) != False:
266             if area['min_x'] != area['max_x'] and area['
min_y'] != area['max_y']:
267                 # user has rnbr area
268                 if p.rnbr['min_x'] != p.rnbr['max_x'] and p.
rnbr['min_y'] != p.rnbr['max_y']:
269                     if p.rnbr['min_x'] >= area['max_x'] or p
.rnbr['max_x'] <= area['min_x']:
270                         continue
271                     if p.rnbr['min_y'] >= area['max_y'] or p
.rnbr['max_y'] <= area['min_y']:
272                         continue
273                 suspect.append(p)
274                 # user doesn't have rnbr area (point)
275                 else:
276                     if p.rnbr['min_x'] >= area['min_x'] and
p.rnbr['max_x'] <= area['max_x']:
277                         if p.rnbr['min_y'] >= area['min_y']
and p.rnbr['max_y'] <= area['max_y']:
278                             suspect.append(p)
279                 else:
280                     # if the given point is inside rnbr
281                     if p.rnbr['min_x'] <= area['min_x'] <= p.
rnbr['max_x'] and p.rnbr['min_y'] <= area['min_y'] <= p.
rnbr['max_y']:
282                         suspect.append(p)
283     return suspect
284
285 def get_suspect_graph(sites, graph):
286     data = defaultdict(list)
287     for i in sites:
288         data[i.id] = get_score(graph.full_graph, i.id)
289     for key, value in data.items():
290         for i, score in value.items():
291             if score > 0:
292                 for j in range(graph.total_location()):
293                     if graph.location[j].id == key:

```

```

294         graph.location[j].add_score(i, score
    )
295     return
296
297 if __name__ == '__main__':
298     user = input('Masukkan data pengguna: ')
299     location = input('Masukkan data lokasi: ')
300     follow = input('Masukkan data follow: ')
301     friend = input('Masukkan data relasi: ')
302     checkin = input('Masukkan data checkin: ')
303
304     ts = time.time()
305
306     graph = Graph()
307     graph.insert_user(user)
308     graph.insert_location(location)
309     graph.insert_follow(follow)
310     graph.insert_friends(friend)
311     graph.insert_checkin(checkin)
312     graph.build_graph()
313
314     print("\nData berhasil dimasukkan")
315
316     # for updating location rnbr
317     loc_dfs = defaultdict(list)
318     for i in list(graph.loc_graph):
319         loc_dfs[i] = get_path(graph.loc_graph, i)
320
321     graph.update_rnbr('l', loc_dfs)
322     graph.update_rnbr('p', loc_dfs)
323
324     precomputing = time.time() - ts
325     # print(precomputing)
326
327     opt = input('\nEnter type: ')
328     while(opt != 'stop'):
329
330         print('\n----- Masukkan Area -----')
331         area = dict()
332         area['min_x'] = int(input('Enter min_x: '))
333         area['min_y'] = int(input('Enter min_y: '))

```

```

334     area['max_x'] = int(input('Enter max_x: '))
335     area['max_y'] = int(input('Enter max_y: '))
336
337     # MBR BASED SOLUTION
338     if opt == "rnbr":
339         start = time.time()
340         suspect = get_suspect_rnbr(area, graph)
341         print('\nJumlah Pengguna: ' + str(len(suspect)))
342
343         # LOG
344         runtime = time.time() - start
345         process = psutil.Process(os.getpid())
346         mem_usage = process.memory_info().rss
347
348         csvlog = "log_rnbr.csv"
349         with open(csvlog, "a") as output:
350             writer = csv.writer(output, lineterminator='
\n', quoting = csv.QUOTE_MINIMAL)
351             writer.writerow([precomputing, runtime,
mem_usage])
352
353             # OUTPUT
354             hasil_rnbr = "hasil_rnbr.csv"
355             with open(hasil_rnbr, "w") as output:
356                 writer = csv.writer(output, lineterminator='
\n', quoting = csv.QUOTE_MINIMAL)
357                 writer.writerow(["user id", "name"])
358                 for i in suspect:
359                     writer.writerow([i.id, i.name])
360
361                 print("\nHasil dapat dilihat di " + hasil_rnbr)
362                 print('-----')
363
364             # GRAPH TRAVERSAL BASED SOLUTION
365             if opt == "graf":
366                 start = time.time()
367                 ts_graph = time.time()
368
369                 sites = get_sites(area, graph.location)
370                 # print(len(sites))
371                 get_suspect_graph(sites, graph)

```

```

372
373         # LOG
374         runtime = time.time() - start
375         process = psutil.Process(os.getpid())
376         mem_usage = process.memory_info().rss
377
378         csvlog = "log_graf.csv"
379         with open(csvlog, "a") as output:
380             writer = csv.writer(output, lineterminator='
\n', quoting = csv.QUOTE_MINIMAL)
381             writer.writerow([precomputing, runtime,
mem_usage])
382
383         hasil_graf = "hasil_graf.csv"
384         with open(hasil_graf, "w") as output:
385             writer = csv.writer(output, lineterminator='
\n', quoting = csv.QUOTE_MINIMAL)
386             writer.writerow(["destination", "user id", "
name", "hop"])
387             for i in graph.location:
388                 if len(i.score) > 0:
389                     for j, score in i.score.items():
390                         if j[0] != 'l':
391                             for k in graph.user:
392                                 if j == k.id:
393                                     writer.writerow([i.
id, k.id, k.name, score])
394
395             print("\nHasil dapat dilihat di " + hasil_graf)
396             print('-----')
397
398         opt = input('\nEnter type: ')

```

Kode Sumber 1.2: Kode Sumber Program Utama

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS



Rohana Qudus, akrab dipanggil Hana lahir di Gresik pada tanggal 7 Februari 1998. Penulis merupakan anak kedua dari 2 bersaudara. Penulis memiliki hobi antara lain mendengarkan musik dan menonton film. Selama berkuliah di Departemen Informatika ITS, penulis pernah menjadi asisten dosen dan praktikum untuk mata kuliah Sistem Operasi (2017) dan Jaringan Komputer (2018). Selama menempuh perkuliahan penulis juga aktif di kegiatan organisasi dan kepanitiaan diantaranya menjadi Staf Departemen Media Informasi HMTC ITS, Staf Departemen Informasi Media BEM FTIF ITS, Staf Ahli Departemen Media Informasi HMTC ITS, Staf Website dan Kesekretariatan Schematics 2016, dan Staf Ahli 3D Schematics 2017.