Aplikasi Graf Berbobot untuk Sistem Rekomendasi Pertemanan pada Media Sosial Berbasis Nearby Search

Naufal Adnan - 13522116[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522116@std.stei.itb.ac.id

***Abstract*—Media sosial telah menjadi bagian yang tak terpisahkan dari kehidupan sosial manusia modern saat ini. Sebagai makhluk sosial, manusia secara alami akan mencari koneksi dan pertemanan di lingkungannya. Salah satu fitur yang dimiliki media sosial untuk memfasilitasi penggunanya dalam interaksi sosial ini adalah fitur rekomendasi pertemanan berbasis *nearby search*. Akan tetapi, tak jarang pengguna tidak mengetahui bagaimana fitur ini dimodelkan dan diimplementasikan. Makalah ini bertujuan untuk memberikan salah satu alternatif implementasi dari sistem rekomendasi pertemanan pada media sosial berbasis nearby search. Metode yang digunakan adalah dengan menggunakan graf berbobot. Pendekatan dilakukan dengan memanfaatkan informasi geolokasi antar pengguna yang kemudian dimodelkan dalam sebuah graf yang merepresentasikan jaringan pertemanan dan bobot pada setiap sisi graf yang mencerminkan tingkat kedekatan individu berdasarkan lokasi geografis. Hasil yang didapatkan yaitu aplikasi graf berbobot dapat menjadi solusi efektif dalam meningkatkan relevansi rekomendasi pertemanan pada media sosial berdasarkan kedekatan lokasi antar pengguna.**

***Keywords*—graf berbobot, media sosial, rekomendasi pertemanan, nearby search**

I. PENDAHULUAN

Manusia merupakan makhluk sosial yang tidak bisa hidup sendiri. Kehidupan sosial membuat manusia sebagai makhluk sosial memiliki kecenderungan untuk menjalin hubungan dan berinteraksi dengan sesamanya. Manusia secara alami akan mencari koneksi dan pertemanan di lingkungannya untuk memenuhi kebutuhan sosialnya. Kebutuhan akan keterhubungan dengan manusia lainnya (konektivitas) dan pertemanan telah menjadi inti dari eksistensi manusia di masyarakat. Hal ini mendorong perkembangan interaksi sosial yang kompleks. Interaksi sosial ini telah menjadi pilar utama pembentukan masyarakat, norma, dan budaya dalam sejarah panjang peradaban manusia. [1]

Di era yang semakin modern ini, fenomena kehidupan sosial telah mengalami pergeseran akibat pesatnya perkembangan teknologi. Media sosial telah menjadi bagian yang tak terpisahkan dari kehidupan sehari-hari manusia modern. Media sosial dapat dikatakan muncul sebagai sarana yang efektif untuk memainkan peran dalam memfasilitasi konektivitas sosial di dunia maya. Tidak jarang akhir-akhir ini terdapat fenomena status sosial seseorang yang dilihat dari seberapa banyak pertemanan yang dia miliki di media sosialnya. [2]

Gambar 1 Rekomendasi pertemanan nearby search pada aplikasi telegram
(Sumber: dokumen pribadi)

Rekomendasi pertemanan pada media sosial berbasis nearby search muncul di antara berbagai fitur lainnya untuk mendukung upaya interaksi sosial ini. Fitur ini muncul untuk membantu pengguna dalam menemukan teman-teman baru dengan mempertimbangkan kedekatan geografis antar pengguna. Namun, sering kali pengguna tidak mengetahui bagaimana fitur ini dimodelkan dan diimplementasikan. Oleh karena itu, pada makalah ini akan dilakukan pemodelan dan implementasi dari sistem rekomendasi pertemanan pada media sosial berbasis nearby search dengan penerapan graf berbobot.

II. LANDASAN TEORI

A. Teori Graf

1. Definisi Graf

Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Secara matematis, graf G didefinisikan sebagai $G = (V, E)$, dengan V merupakan himpunan tidak-kosong dari simpul-simpul (*vertices*) dan E merupakan himpunan sisi (*edges*) yang menghubungkan sepasang simpul. Simpul pada graf dapat ditandai dengan huruf atau angka, sementara sisi yang menghubungkan dua simpul,

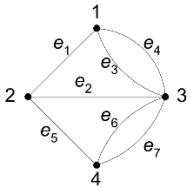
misalnya simpul 1 dengan simpul 2 dinyatakan dengan pasangan (1, 2) atau dengan lambang *e*. Jika terdapat dua buah sisi yang menghubungi dua buah simpul yang sama, maka dinamakan sisi ganda (*multiple edges*). Sementara sisi yang berawal dan berakhir pada simpul yang sama dinamakan gelang atau kalang (*loop*).

Pada Gambar 2, graf G_2 adalah graf dengan

$V = \{1, 2, 3, 4\}$

$E = \{(1, 2), (2, 3), (1, 3), (1, 3), (2, 4), (3, 4), (3, 4)\}$ atau

$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$



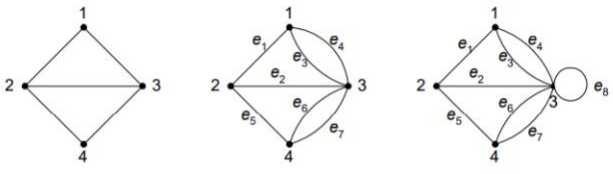
G_2

Gambar 2 Contoh graf G_2
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 08/12-2023

2. Jenis-Jenis Graf

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, maka graf dapat dikelompokkan menjadi dua jenis yaitu:

a. Graf sederhana (*simple graph*)
   Graf sederhana merupakan graf yang tidak mengandung gelang atau sisi ganda sekalipun. Sisi pada graf sederhana adalah pasangan tak terurut sehingga sisi $(a, b) = (b, a)$.
b. Graf tak-sederhana (*unsimple graph)*
   Graf tak-sederhana merupakan graf yang mengandung sisi ganda atau gelang. Ada dua macam graf tak-sederhana, yaitu graf ganda (*multigraph*) dan graf semu (*pseudograph*). Graf ganda merupakan graf yang mengandung sisi ganda. Sementara graf semu adalah graf yang mengandung gelang atau *loop*, termasuk bila memiliki sisi ganda sekalipun.

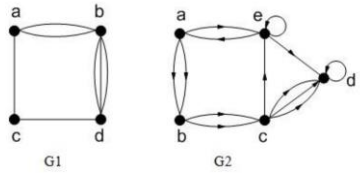


G_1 G_2 G_3

Gambar 3 Contoh graf sederhana (G_1), graf ganda (G_2), dan graf semu (G_3)
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 08/12-2023

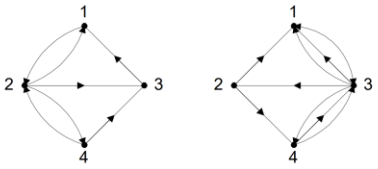
Berdasarkan orientasi arah pada sisi, maka graf dapat dikelompokkan menjadi dua jenis yaitu:

a. Graf tak-berarah (*undirected graph*)
   Graf tak-berarah merupakan graf yang tidak mempunyai orientasi arah. Urutan pasangan simpul yang dihubungkan sisi tidak diperhatikan pada graf ini sehingga sisi $(a, b) = (b, a)$.
b. Graf berarah (*digraph* atau *directed graph*)
   Graf berarah merupakan graf yang setiap sisinya diberikan orientasi arah. Urutan pasangan simpul yang dihubungkan sisi diperhatikan pada graf ini sehingga sisi $(a, b) \neq (b, a)$.



Gambar 4 Contoh graf tak-berarah (G_1), graf berarah (G_2)
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 08/12-2023

Definisi graf dapat diperluas sehingga mencakup graf-ganda berarah (*directed multigraph*) di mana sisi ganda diperbolehkan pada graf-ganda berarah ini.



(a) G_4 (b) G_5

Gambar 5 (a) graf berarah, (b) graf-ganda berarah
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 08/12-2023

Tabel 1 Jenis-jenis graf

Jenis	Sisi	Sisi ganda diperbolehkan?	Sisi gelang diperbolehkan?
Graf sederhana	Tak-berarah	Tidak	Tidak
Graf ganda	Tak-berarah	Ya	Tidak
Graf semu	Tak-berarah	Ya	Ya
Graf berarah	Berarah	Tidak	Ya
Graf-ganda berarah	Berarah	Ya	Ya

3. Terminologi di dalam Graf

a. Ketetanggaan (*adjacent*)
   Dua buah simpul dikatakan bertetangga jika keduanya terhubung langsung sehingga *a* bertetangga dengan *b* jika (a, b) adalah sebuah sisi pada graf *G*.
b. Bersisian (*incidency*)
   Untuk sembarang sisi $e = (a, b)$ dikatakan *e* bersisian dengan simpul *a*, atau *e* bersisian dengan simpul *b*.
c. Simpul terpencil (*isolated vertex*)
   Simpul terpencil merupakan simpul yang tidak mempunyai sisi yang bersisian dengannya. Simpul terpencil juga dapat disebut sebagai simpul yang tidak memiliki tetangga.
d. Graf kosong (*null graph* atau *empty graph*)

Graf kosong merupakan graf yang sisinya himpunan kosong. Ditulis sebagai N_n, dengan n merupakan jumlah simpul.

e. Derajat (*degree*)
Pada graf tak-berarah, derajat suatu simpul adalah jumlah sisi yang bersisian dengan simpul tersebut. Notasi $d(v)$ menyatakan derajat simpul *v*. Simpul terpencil adalah simpul dengan $d(v) = 0$ karena tidak ada satu pun sisi yang bersisian dengan simpul tersebut. Sisi gelang dihitung sebagai sisi dengan derajat dua. Sementara pada graf berarah, derajat simpul dibedakan lagi menjadi derajat masuk (*in-degree*) dan derajat keluar (*out-degree*). Derajat simpul *v* dapat dinyatakan dengan $d_{in}(v)$ dan $d_{out}(v)$. Simbol $d_{in}(v)$ adalah derajat masuk (jumlah busur yang masuk ke simpul *v*) dan simbol $d_{out}(v)$ adalah derajat keluar (jumlah busur yang keluar dari simpul *v*). Sisi gelang pada graf berarah menyumbangkan satu untuk derajat masuk dan satu untuk derajat keluar. Total derajat dihitung dengan menjumlahkan seluruh derajat masuk dan derajat keluar ($d_{in}(v) + d_{out}(v) = d(v)$). Menurut lemma jabat tangan, jumlah derajat semua simpul pada suatu graf adalah genap yaitu dua kali jumlah sisi pada graf tersebut. Dari lemma jabat tangan diperoleh teorema: untuk sembarang graf *G*, banyaknya simpul yang berderajat ganjil selalu genap.

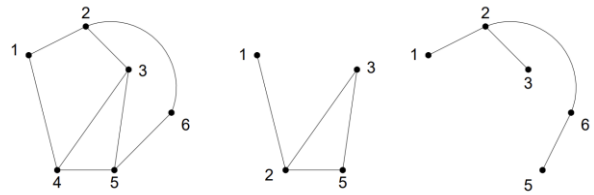
f. Lintasan (*path*)
Lintasan yang panjangnya *n* dari simpul awal v_0 ke simpul tujuan v_n di dalam graf *G* ialah barisan berselang-seling simpul-simpul dan sisi-sisinya yang berbentuk v_0, e_1, v_1, e_2, v_2, …, e_{n-1}, v_{n-1}, e_n, v_n sedemikian sehingga $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2)$, …, $e_n = (v_{n-1}, v_n)$ adalah sisi-sisi pada graf *G*. Panjang lintasan adalah jumlah sisi dalam lintasan tersebut. Sebuah lintasan dikatakan lintasan sederhana (*simple path*) jika semua simpulnya berbeda atau setiap sisi hanya dilalui satu kali. Lintasan yang berawal dan berakhir pada simpul yang sama dinamakan lintasan tertutup (*closed path*), sementara lintasan yang tidak berawal dan berakhir pada simpul yang sama disebut lintasan terbuka (*open path*).

g. Siklus (*cycle*) atau sirkuit (*circuit*)
Lintasan yang berawal dan berakhir pada simpul yang sama disebut sirkuit atau siklus. Panjang sirkuit adalah jumlah sisi di dalam sirkuit tersebut.

h. Keterhubungan (*connected*)
Dua buah simpul v_1 dan simpul v_2 disebut terhubung jika terdapat lintasan dari v_1 ke v_2. Pada graf tak-berarah *G* dikatakan graf terhubung (*connected graph*) jika untuk setiap pasang simpul v_i dan v_j dalam himpunan V terdapat lintasan dari v_i ke v_j. Jika tidak, maka *G* disebut graf tak-terhubung (*disconnected graph*). Sementara pada graf berarah, graf berarah *G* dikatakan terhubung jika graf tak-berarahnya terhubung. Graf tak-berarah dari *G* diperoleh dengan menghilangkan arahnya. Graf berarah *G* disebut terhubung kuat (*strongly connected graph*) jika untuk setiap pasang simpul sembarang v_i dan v_j di *G* terhubung kuat. Jika tidak, maka graf *G* disebut graf terhubung lemah.

i. Upagraf (*subgraph*) dan komplemen upagraf
Misalkan $G = (V, E)$ adalah sebuah graf. $G_1 = (V_1, E_1)$ adalah upagraf (*subgraph*) dari *G* jika $V_1 \subseteq V$ dan $E_1 \subseteq E$. Komplemen dari upagraf G_1 terhadap graf *G* adalah graf $G_2 = (V_2, E_2)$ sedemikian sehingga $E_2 = E - E_1$ dan V_2 adalah himpunan simpul yang anggota-anggota E_2 beririsan dengannya. Jika graf tidak terhubung maka graf tersebut terdiri atas beberapa komponen terhubung (*connected component*). Komponen terhubung adalah upagraf terhubung dari graf *G* yang tidak terdapat di dalam upagraf terhubung *G* yang lebih besar sehingga setiap komponen terhubung di dalam graf *G* saling lepas (*disjoint*). Pada graf berarah, komponen terhubung kuat (*strongly connected component*) adalah upagraf terhubung-kuat dari graf *G* yang tidak terdapat di dalam upagraf terhubung-kuat dari *G* yang lebih besar.

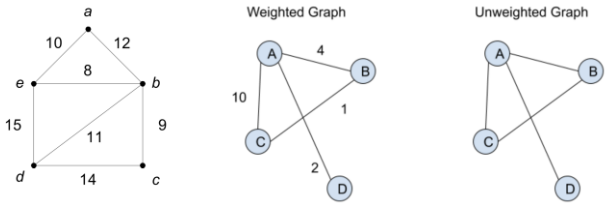
j. Upagraf merentang (*spanning subgraph*)
Upagraf $G1 = (V1, E1)$ dari $G = (V, E)$ dikatakan upagraf merentang jika $V1 = V$ yaitu $G1$ mengandung semua simpul dari *G*.



Gambar 6 (a) Graf G1 (b) Sebuah upagraf (c) komplemen dari upagraf (b)
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 08/12-2023

k. Cut-Set
Cut-set dari sebuah graf terhubung *G* adalah himpunan sisi yang bila dibuang dari *G* menyebabkan *G* tidak terhubung. Jadi, cut-set selalu menghasilkan dua buah komponen terhubung.

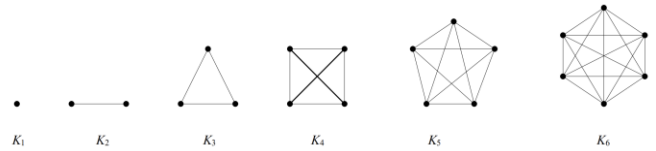
l. Graf berbobot (*weighted graph*)
Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot).



Gambar 7 Contoh graf berbobot dan graf tidak berbobot
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 08/12-2023

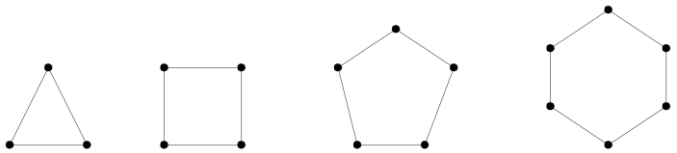
4. Graf Khusus

a. Graf lengkap (*complete graph*)
Graf lengkap adalah graf sederhana yang setiap simpulnya mempunyai sisi ke semua simpul yang lainnya. Graf lengkap dengan *n* buah simpul dilambangkan dengan K_n. Jumlah sisi pada graf lengkap yang terdiri dari n buah simpul adalah $n(n - 1)/2$.



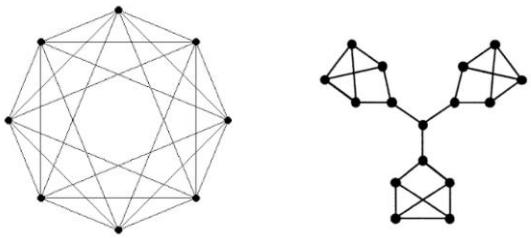
Gambar 8 Contoh graf lengkap
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 08/12-2023

b. Graf lingkaran
Graf lingkaran adalah graf sederhana yang setiap simpulnya berderajat dua. Graf lingkaran dengan *n* simpul dilambangkan dengan C_n.



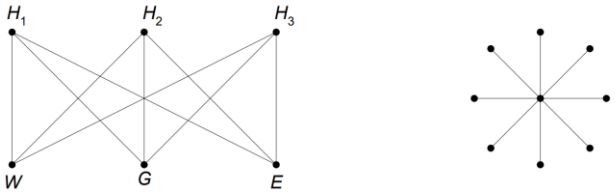
Gambar 9 Contoh graf lingkaran
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 08/12-2023

c. Graf teratur (*regular graph*)
Graf teratur adalah graf yang setiap simpulnya mempunyai derajat yang sama. Apabila derajat setiap simpul adalah *r*, maka graf tersebut disebut sebagai graf teratur derajat *r*. Jumlah sisi pada graf teratur adalah *nr*/2.



Gambar 10 Contoh graf teratur
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 08/12-2023

d. Graf bipartite (*bipartite graph*)
Graf bipartit merupakan graf *G* yang himpunan simpulnya dapat dikelompokkan menjadi dua himpunan bagian V_1 dan V_2 sedemikian sehingga setiap sisi di dalam *G* menghubungkan sebuah simpul di V_1 ke sebuah simpul di V_2 dan dapat dinyatakan sebagai $G(V_1, V_2)$. [3]



Gambar 11 Contoh graf bipartite
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 08/12-2023

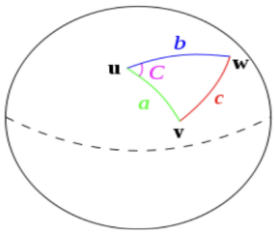
B. Nearby Search

1. Pengukuran Jarak

Informasi pengguna mengenai geolokasi adalah elemen utama dalam *nearby search*. Informasi tentang lokasi geografis pengguna digunakan untuk menentukan objek atau pengguna lain yang berdekatan. Dengan memanfaatkan koordinat geografis, sistem rekomendasi dapat memetakan bobot untuk graf interaksi sosial berdasarkan kedekatan fisik. Perhitungan jarak dilakukan dengan mengambil informasi posisi *latitude* dan *longitude* pengguna [4]. Dari data *latitude* dan *longitude* masing-masing pengguna, maka dapat dihitung jarak antar dua pengguna dengan *Haversine Formula*. Jika diberikan dua titik dengan koordinat geografis, misalkan (lat_1, lon_1) dan (lat_2, lon_2), maka

$R = 6371$
$\Delta lat = lat_2 - lat_1$
$\Delta lon = lon_2 - lont_1$
$a = sin^2(\frac{\Delta lat}{2}) + cos(lat_1).\ cos(lat_2).\ sin^2(\frac{\Delta lon}{2})$
$c = 2.atan2(\sqrt{(a)}, \sqrt{(1-a)})$
$d = R.c$

keterangan:
R: jari-jari bumi sebesar 6371 *km*
Δlat: besaran perubahan latitude
Δlon: besaran perubahan longitude
a: parameter sementara dalam perhitungan
c: kalkulasi perpotongan sumbu
d: jarak antara dua titik lokasi



Gambar 12 *Haversine Triangle*
Sumber:https://www.lppm.unmer.ac.id/webmin/assets/uploads/lj/LJ20210109161016316836 6.pdf
diakses pada 08/12-2023

Formula ini digunakan untuk mengukur jarak garis lurus antara dua titik pada permukaan bola dan sering digunakan dalam perhitungan jarak geografis pada peta. [5]

III. METODOLOGI

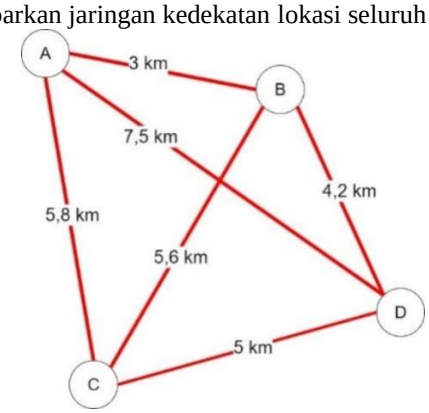
A. Pemodelan Graf

Persoalan keterhubungan antar pengguna dapat diimplementasikan dalam bentuk graf berbobot (*weighted graph*). Graf yang merepresentasikan jaringan kedekatan geolokasi ini memiliki total pengguna sebanyak simpul yang ada pada graf. Bobot pada setiap sisi graf mencerminkan tingkat kedekatan individu berdasarkan lokasi geografis. Jika terdapat pengguna sebanyak *n* pengguna (termasuk dirinya sendiri), maka akan ada sebanyak *n*-1 data jarak terhadap pengguna lainnya yang dimiliki oleh salah satu pengguna. Hal ini juga berarti bahwa akan ada *n* buah simpul dan *n*-1 buah sisi pada graf. Pemodelan ini dilakukan untuk menunjukkan keterhubungan yang dimiliki oleh salah satu pengguna saja.

Graf ini mencerminkan graf bipartite (*bipartite graph*) karena himpunan simpulnya dapat dikelompokkan menjadi dua himpunan bagian V_1 dan V_2 sedemikian sehingga setiap sisi di dalam G menghubungkan sebuah simpul di V_1 ke sebuah simpul di V_2 dan dapat dinyatakan sebagai G (V_1, V_2).

Gambar 13 Graf jarak kedekatan salah satu pengguna
(Sumber: dokumen pribadi)

Sementara itu, untuk mendapatkan visualisasi antar pengguna secara menyeluruh dapat dilakukan dengan pemodelan graf lengkap (*complete graph*). Hal ini karena setiap pengguna (simpul) memiliki keterhubungan (sisi) ke pengguna (simpul) lainnya. Jika terdapat pengguna sebanyak *n*, maka akan ada *n* buah simpul dan *n* (*n* - 1)/2 buah sisi pada graf yang menggambarkan jaringan kedekatan lokasi seluruh pengguna.



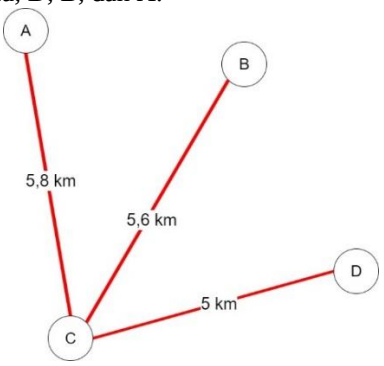
Gambar 14 Graf jarak kedekatan seluruh pengguna
(Sumber: dokumen pribadi)

B. Pengisian Bobot

Pengisian bobot dilakukan dengan menghitung jarak berdasarkan lintang dan bujur antara kedua lokasi pengguna. Hasil dari tahap ini akan diperoleh data jarak antara dua pengguna. Setiap pengguna akan memiliki data jarak dengan seluruh pengguna lainnya, kecuali dengan dirinya sendiri. Perhitungan jarak dilakukan dengan mengambil informasi posisi *latitude* dan *longitude* pengguna. Dari data *latitude* dan *longitude* masing-masing pengguna, maka dapat dihitung jarak antar dua pengguna dengan *Haversine Formula*. Bobot pada sisi (*a*, *b*) adalah hasil dari perhitungan jarak antara simpul *a* dan *b*.

C. Rekomendasi Pertemanan

Untuk setiap pengguna, dilakukan pencarian simpul yang bertetangga berdasarkan batas jarak tertentu. Rekomendasi pertemanan diberikan berdasarkan jarak, di mana pengguna-pengguna yang berjarak kurang dari atau sama dengan batas jarak dianggap sebagai rekomendasi pertemanan. Rekomendasi untuk salah satu pengguna ditampilkan dengan menelusuri pemodelan graf bipartite dari salah satu pengguna yang akan dicari rekomendasi pertemanannya tersebut. Setelah itu, rekomendasi akan diurutkan berdasarkan jarak yang terdekat. Pada Gambar 12, untuk pengguna C memiliki graf bipartite seperti pada Gambar 13. Kemudian ditinjau dari graf bipartite tersebut, pengguna C memiliki rekomendasi teman secara berurutan yaitu, D, B, dan A.



Gambar 15 Graf bipartite jarak kedekatan pengguna C
(Sumber: dokumen pribadi)

IV. EKSPERIMEN

A. Data Pengguna

Eksperimen dilakukan dengan menggunakan total 52 data pengguna dan disimpan dalam file bernama data.py. Data yang digunakan dalam eksperimen ini adalah daftar koordinat pengguna di Kota Bandung. Setiap entri data pengguna mencakup informasi sebagai berikut.

1. Nama Pengguna (*user*): nama identifikasi unik untuk setiap pengguna.
2. Lintang (latitude): koordinat lintang lokasi pengguna.
3. Bujur (longitude): koordinat bujur lokasi pengguna.

```
matdis > data.py > ...
# Daftar koordinat pengguna di Kota Bandung
user_coordinate = [
    ("Adit", -6.9175, 107.6191), ("Budi", -6.9271, 107.6028), ("Cito", -6.9204, 107.6494),
    ("Dian", -6.8932, 107.6166), ("Evan", -6.9128, 107.6069), ("Fian", -6.9345, 107.6232),
    ("Gani", -6.9529, 107.6902), ("Heri", -6.9307, 107.5764), ("Isna", -6.8922, 107.6022),
    ("Jeno", -6.8851, 107.6178), ("Kila", -6.9069, 107.6353), ("Lana", -6.9397, 107.5909),
    ("Musa", -6.8893, 107.6229), ("Nano", -6.9189, 107.6241), ("Opah", -6.8785, 107.5902),
    ("Pika", -6.9509, 107.6887), ("Quin", -6.9273, 107.6538), ("Rere", -6.9083, 107.6760),
    ("Siti", -6.9121, 107.6077), ("tino", -6.8869, 107.6168), ("Umar", -6.9286, 107.6462),
    ("Vina", -6.9156, 107.6491), ("Wira", -6.9476, 107.6536), ("Xena", -6.8993, 107.5879),
    ("Yoga", -6.9294, 107.6335), ("Zara", -6.8974, 107.6210), ("Acha", -6.9056, 107.6154),
    ("Bima", -6.9520, 107.6487), ("Cici", -6.9354, 107.6175), ("Doni", -6.9227, 107.5865),
    ("Eris", -6.9371, 107.6285), ("Fita", -6.8990, 107.6315), ("Gito", -6.9098, 107.6467),
    ("Hana", -6.9368, 107.6039), ("Ivan", -6.9165, 107.6025), ("Jula", -6.9481, 107.6264),
    ("Kian", -6.8988, 107.6620), ("Lulu", -6.8898, 107.5975), ("Mila", -6.9449, 107.6721),
    ("Nana", -6.9152, 107.6629), ("Omar", -6.9203, 107.6286), ("Pina", -6.9085, 107.6423),
    ("Qori", -6.9190, 107.6693), ("Rian", -6.8952, 107.6575), ("Sara", -6.8995, 107.6753),
    ("Toni", -6.9091, 107.6818), ("Ulan", -6.9302, 107.6778), ("Vian", -6.9181, 107.5999),
    ("Widi", -6.9390, 107.6653), ("Xian", -6.9355, 107.5918), ("Yuli", -6.9289, 107.6686),
    ("Zian", -6.9334, 107.6355)
]
```

Gambar 16 Data pengguna (nama, *latitude*, *longitude*)
(Sumber: dokumen pribadi)

B. Mekanisme

Algoritma yang digunakan adalah penerapan struktur data graf berbobot dengan primitif-primitif yang membantu dalam memudahkan operasi yang akan dilakukan. Implementasi dilakukan menggunakan bahasa python melalui langkah-langkah sebagai berikut.

1. Inisialisasi graf

Setiap pengguna direpresentasikan sebagai simpul (*node*) dalam graf. Class Graph terdiri dari method __init__(self) yang merupakan konstruktor yang dipanggil ketika objek graf dibuat, digunakan sebagai inisialisasi atribut-atribut dari objek graf. Atribut self.nodes merupakan himpunan (set) yang menyimpan simpul-simpul (*nodes*) dalam graf. Sementara atribut self.edges akan menyimpan informasi sisi-sisi (*edges*) dalam graf. Setiap simpul memiliki daftar sisi yang terhubung ke simpul lain, beserta bobotnya. Method add_node(self, value) digunakan untuk menambahkan simpul baru ke dalam graf dengan cara menambahkan nilai simpul ke dalam himpunan *nodes*, lalu membuat entri baru pada *edges* untuk simpul yang ditambahkan, dan menginisialisasi daftar sisi (*edges*) menjadi list kosong. Sementara method add_edge(self, from_node, to_node, weight) digunakan untuk menambahkan sisi baru ke dalam graf.

```python
class Graph:
    def __init__(self):
        self.nodes = set()
        self.edges = {}

    def add_node(self, value):
        self.nodes.add(value)
        self.edges[value] = []

    def add_edge(self, from_node, to_node, weight):
        self.edges[from_node].append((to_node, weight))
        self.edges[to_node].append((from_node, weight))
```

Gambar 17 Class Graph
(Sumber: dokumen pribadi)

2. Perhitungan jarak

Jarak antar pengguna dihitung menggunakan formula *Haversine*, yang memperhitungkan lintang dan bujur dari kedua lokasi. Formula *Haversine* menghasilkan bobot jarak dalam kilometer. Bobot pada sisi (*a, b*) adalah hasil dari perhitungan jarak antara simpul *a* dan *b*.

```python
# Fungsi Haversine
def haversine(lat1, lon1, lat2, lon2):
    lat1, lon1, lat2, lon2 = map(radians, [lat1, lon1, lat2, lon2])
    dlat = lat2 - lat1
    dlon = lon2 - lon1
    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    distance = 6371 * c  # Radius bumi (km): 6371
    return distance
```

Gambar 18 Implementasi fungsi *Haversine*
(Sumber: dokumen pribadi)

3. Pemodelan graf sosial

Setiap pengguna direpresentasikan sebagai simpul (*node*) dalam graf. Hubungan pertemanan antar pengguna direpresentasikan sebagai sisi (*edge*) graf dengan bobotnya adalah jarak berdasarkan koordinat lintang dan bujur. Dalam pemodelan graf sosial ini didapatkan hubungan antar pengguna secara menyeluruh yang jika divisualisasikan akan membentuk graf lengkap (*complete graph*). Hal ini karena setiap pengguna (simpul) memiliki keterhubungan (sisi) ke pengguna (simpul) lainnya.

Pemodelan dilakukan dengan menambahkan simpul setiap iterasi jika terdapat simpul yang belum terbentuk. Penambahan simpul dilakukan hingga seluruh pengguna telah memiliki simpul yang merepresentasikan keberadaannya. Penambahan sisi (edge) dilakukan ke dalam graf sosial yang menghubungkan pengguna pertama dan pengguna kedua, dengan bobot (*weight*) jarak yang telah dihitung menggunakan formula *Haversine* sebelumnya.

```python
def create_social_graph():
    # Inisialisasi graf
    social_graph = Graph()
    for i in range(len(var.user_coordinate) - 1):
        user1, lat1, lon1 = var.user_coordinate[i]
        if user1 not in social_graph.nodes:
        # Tambahkan simpul (user1) ke dalam graf
            social_graph.add_node(user1)
        j = i + 1
        while j < len(var.user_coordinate):
            user2, lat2, lon2 = var.user_coordinate[j]
            if user2 not in social_graph.nodes:
                # Tambahkan simpul (user2) ke dalam graf
                social_graph.add_node(user2)
            # Tambahkan sisi (keterhubungan) dan bobot (jarak) antar simpul
            distance = haversine(lat1, lon1, lat2, lon2)
            social_graph.add_edge(user1, user2, distance)
            j += 1

    return social_graph
```

Gambar 19 Implementasi pemodelan graf sosial
(Sumber: dokumen pribadi)

4. Rekomendasi teman

Fungsi recommend_friends berperan dalam memberikan rekomendasi pertemanan untuk seorang pengguna berdasarkan jarak maksimum yang diinginkan. Pencarian simpul tetangga yang memenuhi kriteria jarak dilakukan dengan iterasi melalui daftar sisi (*edges*) yang terhubung dengan simpul (*user*) tertentu. Setiap simpul tetangga (*neighbor*) akan diperiksa bobotnya. Jika jaraknya kurang dari atau sama dengan preferensi yang diinput oleh pengguna, simpul tetangga tersebut dianggap memenuhi kriteria dan ditambahkan ke dalam list rekomendasi. Rekomendasi pertemanan lalu diurutkan berdasarkan jarak mulai dari kecil ke besar. Fungsi sort menggunakan fungsi lambda untuk menentukan kunci pengurutan, di mana x{1} merujuk pada elemen kedua dari setiap tupel (jarak atau bobot graf).

```python
def recommend_friends(graph, user, max_distance):
    recommended_friends = []
    # Cari tetangga yang berjarak kurang dari atau sama dengan max_distance
    for neighbor, distance in graph.edges[user]:
        if distance <= max_distance:
            recommended_friends.append((neighbor, distance))
    # Urutkan rekomendasi berdasarkan jarak
    recommended_friends.sort(key=lambda x: x[1])

    return recommended_friends
```

Gambar 20 Implementasi rekomendasi teman
(Sumber: dokumen pribadi)

5. Simulasi

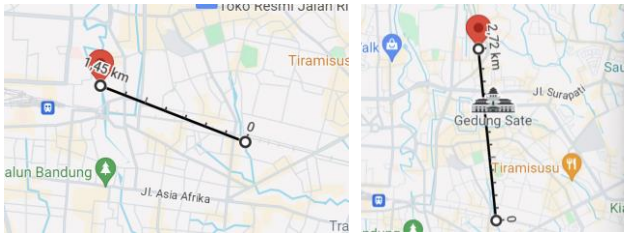
Simulasi untuk sistem rekomendasi pertemanan dilakukan melibatkan interaksi dengan pengguna. Masukan nama pengguna diperlukan untuk digunakan sebagai nama pengguna yang akan dicari rekomendasi pertemanannya. Sementara masukan batas jarak maksimum digunakan sebagai parameter preferensi pengguna untuk menerima rekomendasi pertemanan hingga radius berapa km. Melalui simulasi ini, pengguna dapat dengan mudah melihat rekomendasi pertemanan berdasarkan kriteria jarak maksimum yang diinginkan. Hasil tersebut dapat membantu pengguna dalam menemukan teman-teman baru yang berada dalam jarak tertentu dari lokasi mereka.

```python
# __Main Program__
social_graph = create_social_graph()
user_to_recommend = input("user: ")
max_distance_for_recommendation = float(input("batas jarak: "))
# Rekomendasi pertemanan untuk user_to_recommend
recommendations = recommend_friends(social_graph, user_to_recommend,
                                    max_distance_for_recommendation)
print(f"Rekomendasi pertemanan untuk {user_to_recommend}")
for friend, distance in recommendations:
    print(f"{friend} (Jarak: {distance:.2f})")
```

Gambar 21 Implementasi program utama
(Sumber: dokumen pribadi)

C. Hasil Eksperimen

Pada perhitungan jarak, pengukuran yang dilakukan oleh *Google Maps* tidak memiliki perbedaan yang signifikan dengan penerapan menggunakan formula *Haversine*. Pada Gambar 2 dapat dilihat hasil perhitungan jarak menggunakan *Google Maps* dan formula *Haversine*. Uji banding dilakukan pada pengukuran jarak (lintang, bujur): -6.9175, 107.6191 dengan -6.9128, 107.6069 (gambar kiri) dan -6.9175, 107.6191 dengan -6.8932, 107.6166 (gambar kanan). Hasil ini menunjukkan bahwa algoritma *Haversine* sangat cocok untuk pengukuran menggunakan lintang dan bujur sehingga pemodelan untuk pemberian bobot sistem rekomendasi pertemanan ini dapat dikatakan akurat.




```
Jarak (haversin): 1.44 km
```
```
Jarak (haversin): 2.72 km
```

Gambar 22 Uji banding pengukuran jarak
(Sumber: dokumen pribadi)

Pada class Graph, penambahan sisi melalui method add_edge(self, from_node, to_node, weight) dilakukan secara dua arah. Pertama, dilakukan penambahan tuple (to_node, weight) ke daftar sisi yang terhubung dengan simpul from_node sehingga menciptakan sisi yang mengarah ke simpul to_node dengan bobot weight. Kemudian sisi ditambahkan juga ke daftar sisi simpul to_node. Hal ini dilakukan untuk memastikan hubungan dua arah antara dua simpul. Dengan begitu, pada sisi diperoleh jumlah pasangan (Node, Neighbor, Distance) sebanyak n (n-1), dengan n = 52 maka ada sebanyak 2.652 pasang.

social_graph.csv

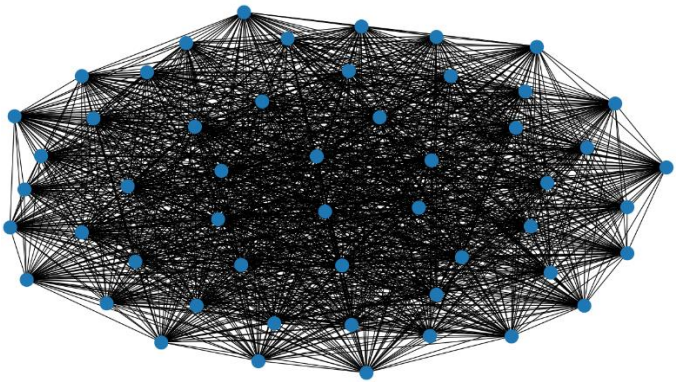
```
1  Node,Neighbor,Distance
2  Acha,Adit,1.3848200060631752
3  Acha,Budi,2.765844594678185
4  Acha,Cito,4.0980919773528415
5  Acha,Dian,1.385165784516734
6  Acha,Evan,1.2334342368921076
7  Acha,Fian,3.326878652575861
8  Acha,Gani,9.789508440501654
```

social_graph.csv

```
2645  Rere,Rian,2.5084694606867077
2646  Rere,Sara,0.981561649414695
2647  Rere,Toni,0.6463980143612984
2648  Rere,Ulan,2.4432614491268114
2649  Rere,Vian,8.4707968891149
2650  Rere,Widi,3.6122374689795325
2651  Rere,Xian,9.774096459224138
2652  Rere,Yuli,2.4319055221928423
2653  Rere,Zian,5.270268724621814
2654
```

Gambar 23 Banyaknya tuple (Node, Neighbor, Distance)
(Sumber: dokumen pribadi)

Untuk visualisasi dari graf sosial yang terbentuk dari 52 pengguna dapat dilihat pada Gambar 24 di bawah ini.



Gambar 24 Visualisasi graf yang terbentuk
(Sumber: dokumen pribadi)

Pada percobaan, salah satu masukan yang dicoba adalah dengan memasukkan pengguna bernama “Acha”. Kemudian dengan memasukkan preferensi radius teman yang akan dicari sejauh 2 km, maka diperoleh hasil rekomendasi pertemanan untuk “Acha” seperti pada Gambar 24.

```
user: Acha
batas jarak: 2
Rekomendasi pertemanan untuk Acha
Zara (Jarak: 1.10 km)
Siti (Jarak: 1.12 km)
Evan (Jarak: 1.23 km)
Adit (Jarak: 1.38 km)
Dian (Jarak: 1.39 km)
Nano (Jarak: 1.76 km)
Ivan (Jarak: 1.87 km)
Fita (Jarak: 1.92 km)
Musa (Jarak: 1.99 km)
```

Gambar 25 Rekomendasi teman pengguna Acha
(Sumber: dokumen pribadi)

V. KESIMPULAN

Graf memiliki banyak sekali kegunaan, salah satunya adalah untuk menyelesaikan persoalan sistem rekomendasi pertemanan pada media sosial berbasis *nearby search*. Dari implementasi yang telah dilakukan menunjukkan bahwa pendekatan graf berbobot dapat menjadi metode yang efektif dalam merepresentasikan persoalan keterhubungan antar pengguna dalam media sosial berdasarkan tingkat kedekatan lokasi. Penggunaan jarak dengan formula *Haversine* untuk pengisian bobot pada graf memberikan dimensi kedekatan lokasi spasial yang akurat, tidak memiliki perbedaan yang signifikan dari perhitungan menggunakan *Google Maps*. Implementasi yang dilakukan menunjukkan bahwa untuk sejumlah n pengguna akan menghasilkan graf dengan jumlah pasangan tuple (*Node*, *Neighbor*, *Distance*) sebanyak n (n-1) pasang. Daftar rekomendasi pertemanan menampilkan daftar pengguna yang direkomendasikan sebagai pertemanan untuk pengguna tertentu beserta informasi jarak antara pengguna dari pengguna yang direkomendasikan. Dengan memberikan rekomendasi pertemanan berbasis geografis, pengguna memiliki kesempatan untuk menjalin koneksi yang lebih dekat dengan individu yang berada dalam lingkup lokal atau daerah sekitar mereka sehingga dapat memberikan nuansa keakraban dan kebersamaan yang lebih kuat.

VI. Ucapan Terima Kasih

Penulis mengucapkan rasa syukur dan terima kasih yang sebesar-besarnya kepada Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya penulis dapat menyelesaikan makalah yang berjudul "Aplikasi Graf Berbobot untuk Sistem Rekomendasi Pertemanan pada Media Sosial Berbasis Nearby Search" tepat waktu. Terima kasih juga penulis ucapkan kepada kedua orang tua yang telah mendukung penulis hingga saat ini. Tidak lupa, penulis mengucapkan terima kasih kepada Ibu Dr. Fariska Zakhralativa Ruskanda, S.T., M.T. selaku dosen mata kuliah IF2120 Matematika Diskrit kelas 02 yang telah memberikan pengajaran kepada penulis dan kepada Bapak Dr. Ir. Rinaldi Munir, MT. yang telah memberikan referensi dan sumber pembelajaran mata kuliah IF2120 Matematika Diskrit.

Referensi

[1] Newcomb, T. M., Turner, R. H., & Converse, P. E. 2015. Social Psychology: The Study of Human Interaction. Psychology Press.

[2] Baumer, S. 2013. Social Media, Human Connectivity and Psychological Well-Being. The Sage Handbook of Digital Technology Research, London, 71-87.

[3] Munir, Rinaldi. 2023. Graf (Bag.1). https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 8 Desember 2023.

[4] Shaw, B., Shea, J., Sinha, S., & Hogue, A. 2013. Learning to Rank for Spatiotemporal Search. In Proceedings of the sixth ACM international conference on Web search and data mining (pp. 717-726).

[5] Prasetya, D. A., Nguyen, P. T., Faizullin, R., Iswanto, I., & Armay, E. F. 2020. Resolving The Shortest Path Problem Using The Haversine Algorithm. Journal of Critical Reviews, 7(1), 62-64.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 10 Desember 2023

Naufal Adnan
13522116