Aplikasi *Undirected Graph* untuk Hubungan Pertemanan pada Facebook

Muhammad Althariq Fairuz - 13522027
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13522027@std.stei.itb.ac.id

***Abstract*—Media sosial adalah sebuah platform yang memungkinkan penggunanya untuk berinteraksi, berbagi, serta menciptakan konten dengan pengguna lainnya melalui internet. Salah satu contoh dari media sosial adalah Facebook. Facebook sendiri memiliki berbagai fitur yang menarik sehingga membuatnya menjadi salah satu media sosial yang paling populer di seluruh dunia. Salah satu aspek penting dari Facebook adalah hubungan pertemanan antar pengguna. Hubungan antar pengguna ini dapat direpresentasikan dengan graf. Dalam aplikasinya, simpul pada graf merepresentasikan suatu pengguna dan sisi dari graf merepresentasikan hubungan antara suatu pengguna dengan pengguna lainnya.**

***Keywords*—Media Sosial, Facebook, Graf, Hubungan Pertemanan, Simpul, Sisi.**

I. PENDAHULUAN

Kemajuan teknologi di era digital ini telah memberikan dampak yang signifikan terhadap berbagai aspek kehidupan manusia di era digital ini. Salah satu dampaknya adalah meningkatnya kepopuleran media sosial, terutama di kalangan remaja. Facebook merupakan salah satu media sosial yang paling banyak digunakan di seluruh dunia. Hal ini karena Facebook memberikan penggunanya kemudahan untuk menjalin hubungan pertemanan dengan pengguna lainnya.

Gambar 1.1 Tampilan halaman Facebook
(Sumber: https://blog.alquilercastilloshinchables.info)

Graf tak-berarah adalah bentuk model matematika yang ideal untuk mendefinisikan hubungan pertemanan dalam media sosial, termasuk Facebook. Setiap pengguna direpresentasikan sebagai simpul dan hubungannya dengan pengguna lain direpresentasikan dengan sisi yang terhubung antara suatu pengguna dengan pengguna lainnya.

II. TEORI DASAR

A. Graf

Graf adalah struktur data yang digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antar objek tersebut. Graf terdiri dari pasangan himpunan simpul (*vertex*) dan sisi (*edges*) yang dinotasikan sebagai berikut [1]:

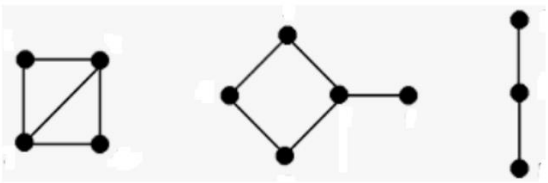
$$G = (V, E)$$

Dalam hal ini :
V = himpunan tidak-kosong dari simpul (*vertices*).
E = himpunan sisi (*edges*) yang menghubungkan sepasang simpul.

B. Jenis Graf Berdasarkan Sisi Ganda

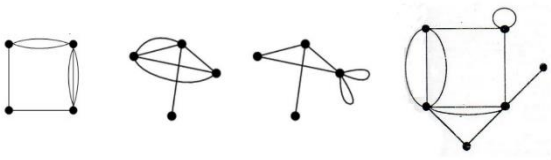
Berdasarkan ada atau tidaknya sisi ganda/gelang, graf digolongkan menjadi dua jenis, yaitu [1] :

1. Graf Sederhana (*Simple Graph*).
   Graf sederhana adalah graf yang tidak mengandung sisi ganda/gelang.



Gambar 2.1 Graf sederhana
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf)

2. Graf Tak-Sederhana (*Unsimple Graph*)
   Graf tak-sederhana adalah graf yang mengandung sisi ganda atau gelang.



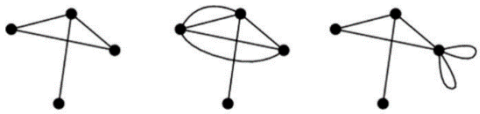
Gambar 2.2 Graf tak-sederhana
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf)

C. Jenis Graf Berdasarkan Orientasi Arah

Berdasarkan orientasi arah pada sisi, graf dibedakan menjadi dua jenis, yaitu [1]:

1. Graf Tak-Berarah (*Undirected Graph*)

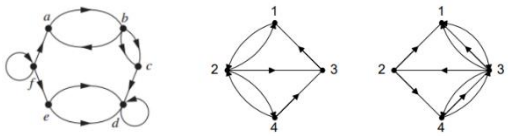
   Graf tak-berarah adalah graf yang sisinya tidak mempunyai orientasi arah.



Gambar 2.3 Graf tak-berarah
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf)

2. Graf Berarah (*Directed Graph*)

   Graf berarah adalah graf yang setiap sisinya memiliki orientasi arah.



Gambar 2.4 Contoh Graf berarah
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf)

Berikut adalah berbagai terminologi dalam graf [1]:

1. Dua buah simpul dari sautu graf dikatakan bertetangga (*Adjacency*) jika keduanya terhubung secara langsung.
2. Sebuah sisi dapat dikatakan bersisian (*Incidency*) dengan salah satu dari dua simpul yang dihubungkan olehnya.
3. Jika sebuah simpul tidak mempunyai sisi yang bersisian dengannya, maka simpul tersebut disebut dengan simpul terpencil.
4. Graf kosong adalah graf yang sisinya merupakan himpunan kosong.
5. Derajat (*Degree*) adalah jumlah sisi yang bersisian dengan simpul tersebut.
6. Lintasan (*Path*) adalah jumlah sisi yang harus dilalui dari simpul asal ke simpul tujuan.
7. Siklus (*Cycle*) adalah lintasan yang berawal dan berakhir di simpul yang sama.
8. Keterhubungan (*Connected*) suatu simpul dikatakan ad ajika terdapat lintasan dari simpul awal ke simpul tujuan.
9. Upagraf (*Subgraph*) adalah bagian dari suatu graf. Misalkan G = (V, E) adalah sebuah graf. G1 = (V1, E1) adalah upagraf (subgraph) dari G jika V1 $\subseteq$ V dan E1 $\subseteq$ E. Sedangkan komplemen dari upagraf G1 terhadap graf G adalah graf G2 = (V2, E2) sedemikian sehingga E2 = E - E1 dan V2 adalah himpunan simpul yang anggota-anggota E2 bersisian dengannya.
10. Upagraf Merentang (*Spanning Subgraph*) Upagraf G1 = (V1, E1) dari G = (V, E) dikatakan upagraf rentang jika V1 =V (yaitu G1 mengandung semua simpul dari G).
11. *Cut-set* dari graf terhubung G adalah himpunan sisi yang bila dibuang dari G menyebabkan G tidak terhubung.
12. Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga(bobot).

D. Facebook

Facebook adalah sebuah media sosial yang memungkinkan penggunanya untuk berkomentar, menyukai, dan berinteraksi dengan pengguna lainnya. Facebook awalnya diluncurkan pada 4 Februari 2004 dan kini telah menjadi salah satu media sosial yang paling populer di seluruh dunia.

Gambar 2.5 Logo Facebook
(Sumber: https://www.vexels.com/png-svg/preview/223136/facebook-icon-social-media)

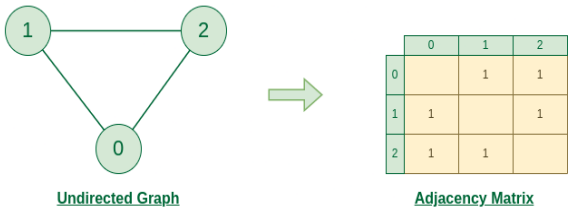
Facebook menggunakan *undirected graph* dalam merepresentasikan hubungan pertemanan antar penggunanaya. Sebagai ilustrasi, misalkan ada dua pengguna Facebook, Alice dan Bob. Jika Alice berteman dengan Bob di Facebook, Bob pasti juga berteman dengan Alice. Oleh karena itu, setiap permintaan pertemanan di Facebook harus diterima oleh kedua belah pihak sehingga membuat kedua pengguna tersebut saling terhubung melalui persahabatan dalam graf yang mendasarinya. Namun tiap sisi graf ini tidak memiliki orientasi atau arah [2].

III. APLIKASI *GRAPH* UNTUK HUBUNGAN PERTEMANAN DI FACEBOOK

A. Aplikasi *Undirected Graph* pada List Pertemanan dari Suatu Pengguna

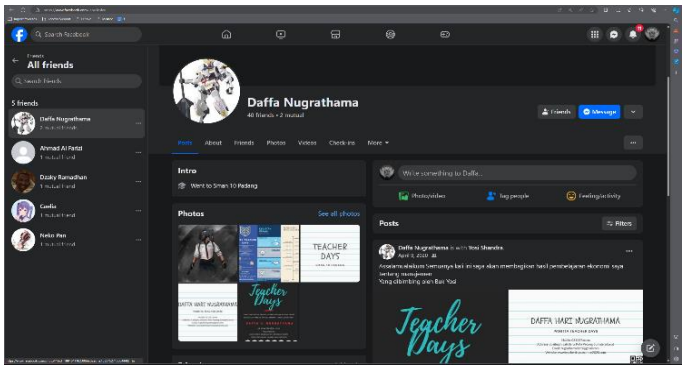
Pada Facebook, untuk menyatakan hubungan pertemanan, diperlukan implementasi graf tak-berarah. Salah satu cara mengimplementasikannya adalah dengan menggunakan matriks ketetanggaan (*adjacency matrix*).

Adjacency matrix adalah salah satu cara untuk merepresentasikan graf dalam matriks dengan *boolean* 0 dan 1. Misalkan ada N buah simpul, maka akan diinisialisasi matriks nol berukuran NxN dengan barisnya menyatakan indeks dari pengguna dan kolomnya menyatakan hubungan pengguna tersebut dengan pengguna lainnya [3].



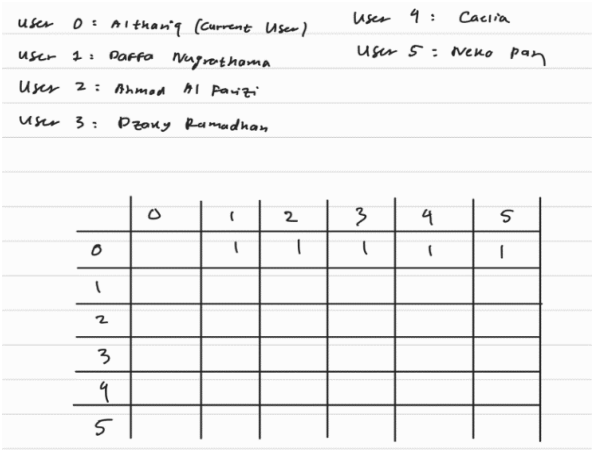
Gambar 3.1 Contoh *adjacency matrix*
(Sumber: https://www.geeksforgeeks.org/graph-and-its-representations)

Jika suatu pengguna memiliki hubungan atau berteman dengan pengguna lainnya, elemen matriks[i][j] dan elemen matriks[j][i] akan diisi dengan 1 dengan i adalah pengguna ke-i dan j adalah pengguna ke-j. Sebaliknya, jika pengguna i tidak ada berhubungan dengan pengguna j, elemen matriks[i][j] dan elemen matriks[j][i] akan diisi dengan 0. Pada contoh diatas, simpul 1 berhubungan dengan 0 dan 2, maka pada *adjacency matrix*-nya, akan diisi 1 pada baris 0 kolom 1, baris 2 kolom 1, baris 1 kolom 0, dan baris 1 kolom 2. Begitu juga untuk simpul 0 dan 2. Elemen dari indeks suatu matriks tidak perlu ditambah lagi jika elemen pada indeks yang bersangkutan sudah terisi.



Gambar 3.2 Tampilan list pertemanan di Facebook
(Sumber: Dokumentasi Pribadi)

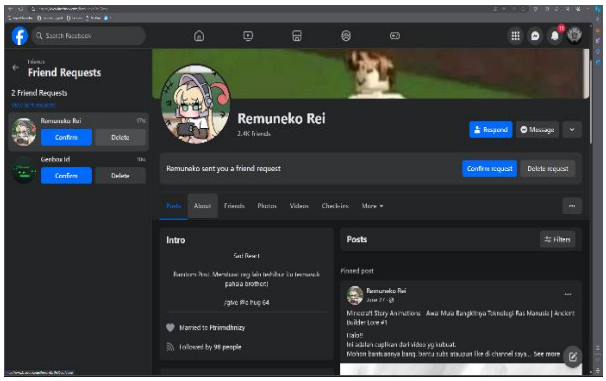
Gambar 3.2 menggambarkan tampilan list teman dari suatu pengguna Facebook. Sedangkan Gambar 3.3 merupakan interpretasi dari Gambar 3.2 dalam bentuk *adjacency matrix.*



Gambar 3.3 Representasi list pertemanan dalam *adjacency matrix*
(Sumber: Dokumen Pribadi)

B. Aplikasi *Undirected Graph* untuk permintaan pertemanan pada Facebook

Untuk mengirimkan permintaan pertemanan ke suatu pengguna, karena hubungannya masih belum bersifat timbal balik, elemen pada *adjacency matrix* masih diisi 0 baik pada elemen ke-ij maupun elemen ke-ji. Jika pengguna j menerima permintaan tersebut, barulah ditambahkan 1 pada indeks ke-ij dan indeks ke-ji sekaligus menyatakan kalau hubungan mereka sudah timbal balik.



Gambar 3.4 List dari permintaan pertemanan pada Facebook
(Sumber: Dokumentasi Pribadi)

IV. HASIL DAN PEMBAHASAN

Untuk implementasi sederhananya, pada makalah ini, akan digunakan Bahasa C untuk merepresentasikan ADT *undirected graphnya*. Pada ADT ini, jumlah pengguna akan dibatasi maksimal dua puluh pengguna saja untuk mengurangi kompleksitas program. Selain itu, akan digunakan ADT tambahan, yaitu *Priority Queue*, untuk memudahkan menyortir dan menerima daftar permintaan pertemanan. Berikut adalah gambaran dari programnya:

graph.c

```c
#define MAX_SIMPUL 20

/* Selektor */
#define SIMPUL(p) (p).simpul
#define ELMT_GRAPH(p, i, j) (p).adjMatrix[i][j] // Baris = user asal, kolom = user tujuan
#define NAME_USER(p) (p).name
#define INDEX_USER(p) (p).userIndex
#define FOLLOWER(p) (p).follower
#define REQUEST(p, i) (p).request[i]
#define PENDING_REQUEST(p, i) (p).pendingRequest[i]

/* Define structur untuk graph */
typedef struct graph
{
    int simpul;                         // simpul = jumlah user
    int adjMatrix[MAX_SIMPUL][MAX_SIMPUL]; // adjacency matrix yg berisi pointer to integer untuk menyimpan hubungan pertemanan
} Graph;

/* Define structur untuk user */
typedef struct user
{
    char name[20];
    int userIndex;
    int follower;
    int request[20];
    int pendingRequest[20];
} User;

/* Define array of user index ( 0 berarti masih belum kenal ) */
char nameOfUser[MAX_SIMPUL][20]; // Baris = user asal, kolom = panjang kata max
int indexOfUser[MAX_SIMPUL] = {0};
User listOfUser[MAX_SIMPUL];

/* Fungsi untuk membuat graph baru */
void createGraph(Graph *graph, int jumlahUser)
{
    int i, j;
    SIMPUL(*graph) = jumlahUser;

    for (int i = 0; i < jumlahUser; i++)
    {
        for (int j = 0; j < jumlahUser; j++)
        {
            ELMT_GRAPH(*graph, i, j) = 0;
        }
    }
}
```

Snipped

Gambar 4.1 ADT sederhana graf
(Sumber: Dokumentasi Pribadi)

graph.c

```c
/* Fungsi untuk mencek apakah ada hubungan pertemanan antara dua user */
boolean isTeman(Graph graph, int index_user_asal, int index_user_tujuan)
{
    if (ELMT_GRAPH(graph, index_user_asal, index_user_tujuan) == 1)
    {
        return true;
    }
    else
    {
        return false;
    }
}
/* Fungsi untuk menghitunng jumlah teman yang dimiliki oleh suatu user */
int jumlahTeman(Graph graph, int user_index)
{
    int j, jumlah = 0;
    for (j = 0; j < MAX_SIMPUL; j++)
    {
        if (ELMT_GRAPH(graph, user_index, j) == 1)
        {
            jumlah++;
        }
    }
    return jumlah;
}

/* Fungsi untuk menambahkan edge/sisi (hubungan pertemanan) pada graph */
void addTeman(Graph *graph, int index_user_asal, int index_user_tujuan)
{
    ELMT_GRAPH(*graph, index_user_asal, index_user_tujuan) = 1;
    ELMT_GRAPH(*graph, index_user_tujuan, index_user_asal) = 1;
}

/* Fungsi untuk menghapus teman */
void hapusTeman(Graph *graph, int index_user_asal, int index_user_tujuan)
{
    ELMT_GRAPH(*graph, index_user_asal, index_user_tujuan) = 0;
    ELMT_GRAPH(*graph, index_user_tujuan, index_user_asal) = 0;
}
/* Fungsi untuk mencetak daftar teman */
void printTeman(Graph graph, User user, int user_index)
{
    printf("Current User: %s\n", NAME_USER(user));
    int j = 0;
    int temanUser = jumlahTeman(graph, user_index);
    if (temanUser == 0)
    {
        printf("%s tidak memiliki teman\n", NAME_USER(user));
    }
    else
    {
        printf("%s memiliki %d teman\n", NAME_USER(user), temanUser);
        printf("Daftar teman %s: \n", NAME_USER(user));

        for (j = 0; j < MAX_SIMPUL; j++)
        {
            if (ELMT_GRAPH(graph, user_index, j) == 1)
            {
                printf("| %s \n", nameOfUser[j]);
            }
        }
    }
}
```

Snipped

Gambar 4.2 Fungsi untuk menerima pertemanan
(Sumber: Dokumentasi Pribadi)

makalah.c

```c
/* Fungsi buat ngirim request pertemanan*/
void sendRequest(Graph *graph, User *user_asal, User *user_tujuan)
{

    /* Jika ngirim freind request ke diri sendiri */
    if (INDEX_USER(*user_asal) == INDEX_USER(*user_tujuan))
    {
        printf("Kamu tidak bisa mengirim request ke diri sendiri.\n");
        return;
    }
    else
    {
        /* Case 1 : Jika sudah berteman dengan user tertentu*/
        if (isTeman(*graph, INDEX_USER(*user_asal), INDEX_USER(*user_tujuan)))
        {
            printf("Anda sudah berteman dengan %s\n.", nameOfUser[INDEX_USER(*user_tujuan)]);
            return;
        }
        /* Case 2 : Jika sudah ngirim request ke user tertentu */
        else if (REQUEST(*user_asal, INDEX_USER(*user_tujuan)) == 1)
        {
            printf("Anda sudah mengirimkan permintaan pertemanan kepada %s. Silakan tunggu hingga permintaan Anda disetujui\n.", nameOfUser[INDEX_USER(*user_tujuan)]);
            return;
        }
        /* Case 3 : Jika belum ngirim request sama sekali ke user tertentu */
        else
        {
            REQUEST(*user_asal, INDEX_USER(*user_tujuan)) = 1;
            PENDING_REQUEST(*user_tujuan, INDEX_USER(*user_asal)) = 1;
            printf("Permintaan pertemanan kepada %s telah dikirim. Tunggu beberapa saat hingga permintaan Anda disetujui.\n", nameOfUser[INDEX_USER(*user_tujuan)]);
            return;
        }
    }
}

/* Fungsi buat nge cancel request pertemanan*/
void cancelRequest(User *user_asal, User *user_tujuan)
{
    /* Case 1 : Jika belum ada ngirim pertemanan ke user tertentu */
    if (REQUEST(*user_asal, INDEX_USER(*user_tujuan)) == 0)
    {
        printf("Anda belum mengirimkan permintaan pertemanan kepada %s.\n", nameOfUser[INDEX_USER(*user_tujuan)]);
    }
    /* Case 2: Jika benar ada udah ngirim request ke user tertentu */
    else
    {
        REQUEST(*user_asal, INDEX_USER(*user_tujuan)) = 0;
        PENDING_REQUEST(*user_tujuan, INDEX_USER(*user_asal)) = 0;
        printf("Permintaan pertemanan kepada %s telah dibatalkan.\n", nameOfUser[INDEX_USER(*user_tujuan)]);
    }
}
```

Snipped

Gambar 4.3 Fungsi untuk mengirim dan membatalkan permintaan pertemanan
(Sumber: Dokumentasi Pribadi)

makalah.c

```c
/* Fungsi buat print daftar teman yang ada */
void daftarPermintaanTeman(User user)
{
    List listRequest;
    CreateList(&listRequest);
    printf("Current User: %s\n", nameOfUser[INDEX_USER(user)]);
    /* Case 1 : Jika ga ada request pertemanan */
    if (isNoPendingRequest(user))
    {
        printf("Tidak ada permintaan pertemanan untuk Anda.\n");
        return;
    }
    /* Case 2 : Jika ada request */
    else
    {
        printf("Terdapat %d permintaan pertemanan untuk Anda.\n", jumlahPendingRequest(user));
        for (int i = 0; i < MAX_SIMPUL; i++)
        {
            if (PENDING_REQUEST(user, i) == 1)
            {
                enqueue(&listRequest, listOfUser[i], FOLLOWER(listOfUser[i]));
            }
        }
        printf("\n");
        while (!IsEmpty(listRequest))
        {
            printf("| %s\n", nameOfUser[INDEX_USER(DATA(FIRST_QUEUE(listRequest)))]);
            printf("| Jumlah Teman: %d \n\n", FOLLOWER(DATA(FIRST_QUEUE(listRequest))));
            dequeue(&listRequest);
        }
    }
}

/* Fungsi buat menerima request pertemanan */
void acceptRequest(Graph *graph, User *user_asal, User *user_tujuan)
{
    printf("Current User: %s\n", nameOfUser[INDEX_USER(*user_asal)]);
    if (PENDING_REQUEST(*user_asal, INDEX_USER(*user_tujuan)) == 0)
    {
        printf("Anda belum menerima permintaan pertemanan dari %s.\n", nameOfUser[INDEX_USER(*user_tujuan)]);
    }
    else
    {
        addTeman(graph, INDEX_USER(*user_asal), INDEX_USER(*user_tujuan));
        REQUEST(*user_tujuan, INDEX_USER(*user_asal)) = 0;
        PENDING_REQUEST(*user_asal, INDEX_USER(*user_tujuan)) = 0;
        printf("Permintaan pertemanan dari %s telah disetujui. Selamat! Anda telah berteman dengan %s.\n", nameOfUser[INDEX_USER(*user_tujuan)], nameOfUser[INDEX_USER(*user_tujuan)]);
    }
}

/* Fungsi buat request nolak pertemanan */
void declineRequest(User *user_asal, User *user_tujuan)
{
    printf("Current User: %s\n", nameOfUser[INDEX_USER(*user_asal)]);
    if (PENDING_REQUEST(*user_asal, INDEX_USER(*user_tujuan)) == 0)
    {
        printf("Anda belum menerima permintaan pertemanan dari %s.\n", nameOfUser[INDEX_USER(*user_tujuan)]);
    }
    else
    {
        REQUEST(*user_tujuan, INDEX_USER(*user_asal)) = 0;
        PENDING_REQUEST(*user_asal, INDEX_USER(*user_tujuan)) = 0;
        printf("Permintaan pertemanan dari %s telah ditolak.\n", nameOfUser[INDEX_USER(*user_tujuan)]);
    }
}
```

Snipped

Gambar 4.4 Fungsi untuk menampilkan daftar permintaan dengan bantuan ADT *Priority Queue*, menerima, serta menolak permintaan pertemanan
(Sumber: Dokumentasi Pribadi)

makalah.c

```c
/* Selektor */
#define DATA(p) (p)->user
#define NEXT_QUEUE(p) (p)->next
#define PRIORITY(p) (p)->priority
#define FIRST_QUEUE(l) (l) // Buat ambil node dari list l

/* Address = Pointer to Node */
typedef struct node *Address;

/* Node = Element dari Priority Queue Linked List */
typedef struct node
{
    User user;
    int priority;
    struct node *next;
} Node;

/* List = Address (Elemen pertama queue) */
typedef Address List;

/* Buat convert data user ke node */
Address newNode(User user)
{
    Address temp = (Address)malloc(sizeof(Node));
    if (temp != NULL)
    {
        DATA(temp) = user;
        PRIORITY(temp) = FOLLOWER(user);
        NEXT_QUEUE(temp) = NULL;
    }
    return temp;
}

/* Buat Priority Queue*/
void CreateList(List *l)
{
    FIRST_QUEUE(*l) = NULL;
}

/* Cek apakah queue kosong*/
boolean isEmpty(List l)
{
    return (FIRST_QUEUE(l) == NULL);
}

/* Fungsi untuk hapus elemen/node dengan prioritas tertinggi*/
void dequeue(List *l)
{
    Address temp = FIRST_QUEUE(*l);
    FIRST_QUEUE(*l) = NEXT_QUEUE(FIRST_QUEUE(*l));
    free(temp);
}

/* Fungsi untuk menambahkan node*/
void enqueue(List *l, User user, int follower)
{
    Address start = FIRST_QUEUE(*l);

    /* Convert data user ke node */
    Address temp = newNode(user);

    /* Case 1: Jika list kosong*/
    if (FIRST_QUEUE(*l) == NULL)
    {
        NEXT_QUEUE(temp) = FIRST_QUEUE(*l);
        FIRST_QUEUE(*l) = temp;
    }
    /* Case 2 : Jika elemen pertama memiliki prioritas lebih kecil dari user yang akan diinput*/
    else if (PRIORITY(FIRST_QUEUE(*l)) < follower)
    {

        // Insert New Node before head
        NEXT_QUEUE(temp) = FIRST_QUEUE(*l);
        FIRST_QUEUE(*l) = temp;
    }
    /* Case 3 : Mencari posisi yang pas untuk node baru, perlu dilakukan looping */
    else
    {

        while (NEXT_QUEUE(start) != NULL && PRIORITY(NEXT_QUEUE(start)) > follower)
        {
            start = NEXT_QUEUE(start);
        }

        /* Node akan diinput diakhir atau di posisi yang sesuai */
        NEXT_QUEUE(temp) = NEXT_QUEUE(start);
        NEXT_QUEUE(start) = temp;
    }
}

int length_queue(List l)
{
    int count = 0;
    Address temp = FIRST_QUEUE(l);
    while (temp != NULL)
    {
        count++;
        temp = NEXT_QUEUE(temp);
    }
    return count;
}

/* Fungsi baut ngecek apakah ada pending request */
boolean isNoPendingRequest(User user)
{
    boolean empty = true;
    for (int i = 0; i < MAX_SIMPUL; i++)
    {
        if (PENDING_REQUEST(user, i) == 1)
        {
            empty = false;
        }
    }
    return empty;
}
```

Snipped

Gambar 4.5 ADT *Priority Queue* untuk membantu menerima dan menyoritr permintaan pertemanan berdasarkan pengikut/jumlah teman terbanyak.
(Sumber: Dokumentasi Pribadi)

Misalkan ada dua user, yaitu Alice dan Bob. Alice mengirim permintaan pertemanan ke Bob, maka Bob bisa menerima atau menolak permintaan tersebut. Jika Bob menerima, graf yang merepresentasikan Bob akan terhubung dengan Alice sehingga Alice dan Bob menjadi teman.

Berikut adalah realisasinya:

makalah.c

```c
/* Inisialisasi Graph*/
    Graph graph;
    createGraph(&graph, MAX_SIMPUL);
    /* Create user*/
    User user1;
    createUser(&user1, "Alice", 50);
    User user2;
    createUser(&user2, "Bob", 40);
    User user3;
    createUser(&user3, "Bacin", 30);
    User user4;
    createUser(&user4, "Dewolover", 150);
```

Snipped

Gambar 4.6 Inisialisasi pengguna dan *Undirected Graph* untuk semua pengguna beserta jumlah temannya
(Sumber: Dokumentasi Pribadi)

makalah.c

```c
    sendRequest(&graph, &user1, &user2);
    printf("\n");
    daftarPermintaanTeman(user2);
    printf("\n");
    acceptRequest(&graph, &user2, &user1);
    printf("\n");
    printTeman(graph, user1, INDEX_USER(user1));
    printf("\n");
    printTeman(graph, user2, INDEX_USER(user2));
    printf("\n");
```

Snipped

Gambar 4.7 User1 (Alice) mengirim permintaan pertemanan ke User2 (Bob) dan User2 (Bob) menerimanya
(Sumber: Dokumentasi Pribadi)

```
Current User: Alice
Permintaan pertemanan kepada Bob telah dikirim. Tunggu beberapa saat hingga permintaan Anda disetujui.

Current User: Bob
Terdapat 1 permintaan pertemanan untuk Anda.

| Alice
| Jumlah Teman: 50


Current User: Bob
Permintaan pertemanan dari Alice telah disetujui. Selamat! Anda telah berteman dengan Alice.

Current User: Alice
Alice memiliki 1 teman
Daftar teman Alice:
| Bob

Current User: Bob
Bob memiliki 1 teman
Daftar teman Bob:
| Alice
```

Gambar 4.8 Output dari program
(Sumber: Dokumentasi Pribadi)

makalah.c

```
281     daftarPermintaanTeman(user3);
282     printf("\n");
283     acceptRequest(&graph, &user4, &user1);
284     printf("\n");
285     printTeman(graph, user3, INDEX_USER(user3));
286     printf("\n");
287     sendRequest(&graph, &user3, &user4);
288     printf("\n");
289     cancelRequest(&user3, &user4);
290     printf("\n");
291     sendRequest(&graph, &user4, &user3);
292     printf("\n");
293     declineRequest(&user4, &user3);
294     printf("\n");
```

Snipped

Gambar 4.9 Program mencetak daftar permintaan pertemanan dari User3 (Bacin). Kemudian, program menerima permintaan pertemanan dari User1 (Alice) ke User4 (DewoLover)
(Sumber: Dokumentasi Pribadi)

```
Current User: Bacin
Tidak ada permintaan pertemanan untuk Anda.

Current User: Dewolover
Anda belum menerima permintaan pertemanan dari Alice.

Current User: Bacin
Bacin tidak memiliki teman

Current User: Bacin
Permintaan pertemanan kepada Dewolover telah dikirim. Tunggu beberapa saat hingga permintaan Anda disetujui.

Current User: Bacin
Permintaan pertemanan kepada Dewolover telah dibatalkan.

Current User: Dewolover
Permintaan pertemanan kepada Bacin telah dikirim. Tunggu beberapa saat hingga permintaan Anda disetujui.

Current User: Bacin
Permintaan pertemanan dari Dewolover telah ditolak.
```

Gambar 4.10 Output dari program
(Sumber: Dokumentasi Pribadi)

Karena User1 (Alice) belum mengirim permintaan pertemanan ke User4 (DewoLover), program mengirimkan *error message* ke terminal. Kemudian, User3 (Bacin) mengirimkan permintaan pertemanan ke User4 (DewoLover), tetapi ia akhirnya membatalkan permintaan pertemanannya. Lalu, User4 (DewoLover) mengirimkan pertemanan ke User3 (Bacin), tetapi User3 (Bacin) menolaknya.

makalah.c

```
295     printTeman(graph, user3, INDEX_USER(user3));
296     printf("\n");
297     printTeman(graph, user4, INDEX_USER(user4));
298     printf("\n");
```

Snipped

Gambar 4.11 Program menampilkan daftar teman dari User3 (Bacin) dan User4 (DewoLover)

(Sumber: Dokumentasi Pribadi)

```
Current User: Bacin
Bacin tidak memiliki teman

Current User: Dewolover
Dewolover tidak memiliki teman
```

Gambar 4.12 Output dari program

(Sumber: Dokumentasi Pribadi)

V. KESIMPULAN

Dari penerapan *undirected graph* untuk hubungan pertemanan di Facebook, kesimpulan yang diambil mencakup beberapa aspek:

1. Representasi Hubungan
   Graf tak-berarah (undirected graph) digunakan untuk merepresentasikan hubungan pertemanan antar pengguna. Setiap simpul dalam graf merepresentasikan sautu pengguna, sedangkan sisi merepresentasikan adanya hubungan pertemanan antara pengguna.

2. Ketertautan dalam Jaringan Sosial
   Dengan adanya graf, dapat dilihat bagaimana pengguna saling terhubung satu sama lain. Hal ini menciptakan jaringan sosial yang kompleks.

3. Analisis Koneksi
   Dengan menganalisis graf, dapat diidentifikasi pengguna yang berperan sebagai penghubung dalam jaringan pertemanan. Pengguna ini mungkin memiliki banyak koneksi dan memegang peranan penting dalam menyatukan berbagai bagian dari komunitas.

VI. UCAPAN TERIMA KASIH

Pertama-tama, puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena berkat rahmat-Nya penulis dapat menyelesaikan makalah ini dengan baik. Penulis juga berterimakasih kepada orang tua, serta teman-teman yang selalu memberikan semangat dan dukungan kepada penulis sehingga makalah ini dapat terselesaikan. Penulis juga tak lupa berterimakasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T., Ibu Fariska Zakhralativa Ruskanda, S.T.,M.T., dan Ibu Dr. Nur Ulfa Maulidevi, selaku dosen mata kuliah matematika diskrit yang telah memberikan banyak ilmu dan motivasi dalam kegiatan perkuliahan. Terakhir, penulis memohon maaf apabila dalam penulisan makalah ini terdapat Makalah IF2120 Matematika Diskrit – Sem. I Tahun 2023/2024 kesalahan baik disengaja maupun tidak disengaja. Penulis berharap makalah ini dapat bermanfaat bagi banyak orang.

REFRENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf . Diakses pada 11 November 2023.
[2] https://sitn.hms.harvard.edu/flash/2021/graph-theory-101/. Diakses pada 11 November 2023.
[3] https://www.geeksforgeeks.org/graph-and-its-representations. Diakses pada 11 November 2023.

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, 3 Desember 2023

Muhammad Althariq Fairuz 13522027