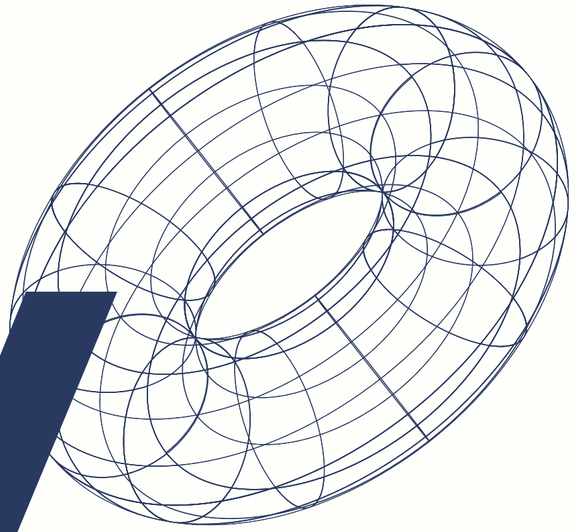
Algorithm & Data Structure Final Project

year 2025

Buddy Map



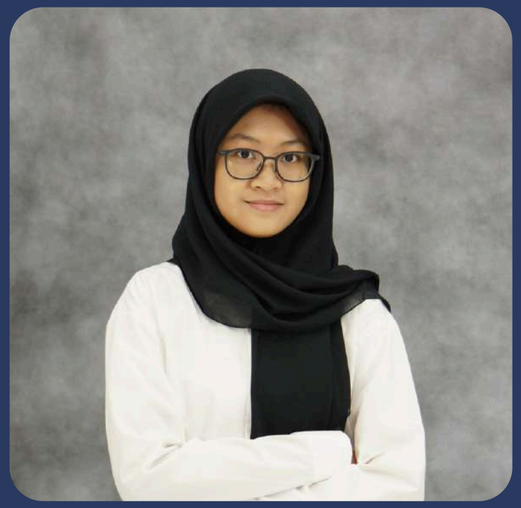
by: Group D9

/page 01

meet the group.

#groupproject

Maria Elvina P.D.
5026241012



Tavasya Alia Anjani
5026241067

Rofifah Zain Nur A.
5026241131

/page 02

#groupproject

studi kasus

TOPIK FINAL PROJECT:

Sistem Rekomendasi Teman Berdasarkan Mutual Friends

TEMA:

Implementasi Struktur Data Graph, BFS, dan Algoritma Sorting untuk Fitur Rekomendasi di Media Sosial

algoritma & struktur data yang digunakan

1. Struktur Data: Graph (Adjacency List)

- Node = pengguna, Edge = pertemanan
- Implementasi: HashMap<String, List<String>>

2. Algoritma BFS

- Mengekstrak teman level 2 (teman dari teman)
- Mempermudah perhitungan mutual friends

3. Algoritma Mutual Friends

- Hitung jumlah teman yang sama dengan user

4. Algoritma Sorting (Bubble Sort)

- Mengurutkan daftar rekomendasi alfabetical

GRAPH

```java
public class GraphLogic {  3 usages  RofifahZain +2 *

    private Map<String, List<String>> graph;  20 usages

    public GraphLogic() { graph = new HashMap<>(); }

    // menambahkan orang
    public void addPerson(String name) {  2 usages  Tavasya Alia A
        if (!graph.containsKey(name)) {
            graph.put(name, new ArrayList<String>());
        }
    }
```

Ini adalah deklarasi utama graph sebagai HashMap (key = nama pengguna, value = List teman-temannya). Metode addPerson memastikan setiap pengguna terinisialisasi dalam peta.

BFS()

```
public void displayPeopleYouMayKnow(String person) {  1 usage  RofifahZain
    Set<String> level2 = new HashSet<>();

    Set<String> visited = new HashSet<>();
    visited.add(person);
    visited.addAll(temanLangsung);

    Queue<String> queue = new LinkedList<>();

    Map<String, Integer> level = new HashMap<>();
    for (String t : temanLangsung) {
        queue.add(t);
        level.put(t, 1);
    }

    while (!queue.isEmpty()) {
        String current = queue.poll();
        int currentLevel = level.get(current);

        if (currentLevel == 2) continue;
```

Inisialisasi BFS : Digunakan Queue untuk penelusuran lapis demi lapis dan Map Level untuk melacak jarak. Penelusuran dimulai dari Teman Langsung (Level 1) yang dimasukkan ke dalam antrian.

BFS (2)

```
        for (String friend : graph.get(current)) {
            if (!visited.contains(friend)) {

                int nextLevel = currentLevel + 1;

                if (nextLevel == 2) {
                    level2.add(friend);
                }

                visited.add(friend);
                queue.add(friend);
                level.put(friend, nextLevel);
            }
        }
    }
```

Logika inti BFS : Untuk setiap tetangga yang belum dikunjungi, dihitung level pertemanan. Jika levelnya mencapai Level 2, orang tersebut adalah Teman dari Teman dan ditambahkan sebagai calon rekomendasi.

MUTUAL

```
// Hitung mutuals
Map<String, List<String>> mutualMap = new HashMap<>();

for (String calon : level2) {
    mutualMap.put(calon, new ArrayList<>());

    for (String t : temanLangsung) {
        if (graph.get(t).contains(calon)) {
            mutualMap.get(calon).add(t);
        }
    }
}
```

Kode ini mengiterasi setiap calon rekomendasi (calon) dan membandingkannya dengan daftar temanLangsung pengguna. Jika seorang teman langsung berteman dengan calon, maka mereka adalah teman bersama (mutual friend).

SORTING

```java
private void bubbleSort(List<String> list) {  1 usage  RofifahZain
    int n = list.size();
    boolean swapped;

    for (int i = 0; i < n - 1; i++) {
        swapped = false;

        for (int j = 0; j < n - i - 1; j++) {
            if (list.get(j).compareToIgnoreCase(list.get(j + 1)) > 0) {
                // swap
                String temp = list.get(j);
                list.set(j, list.get(j + 1));
                list.set(j + 1, temp);
                swapped = true;
            }
        }

        if (!swapped) break;
    }
}
```

Algoritma Bubble Sort digunakan untuk mengurutkan daftar rekomendasi secara alfabetis. Inti logikanya adalah membandingkan elemen yang berdekatan dan menukarnya (swap) jika urutannya salah, mengulangi proses hingga seluruh daftar terurut.

fitur - fitur aplikasi

1. **Add Connection**
   → menambahkan hubungan pertemanan antara 2 orang

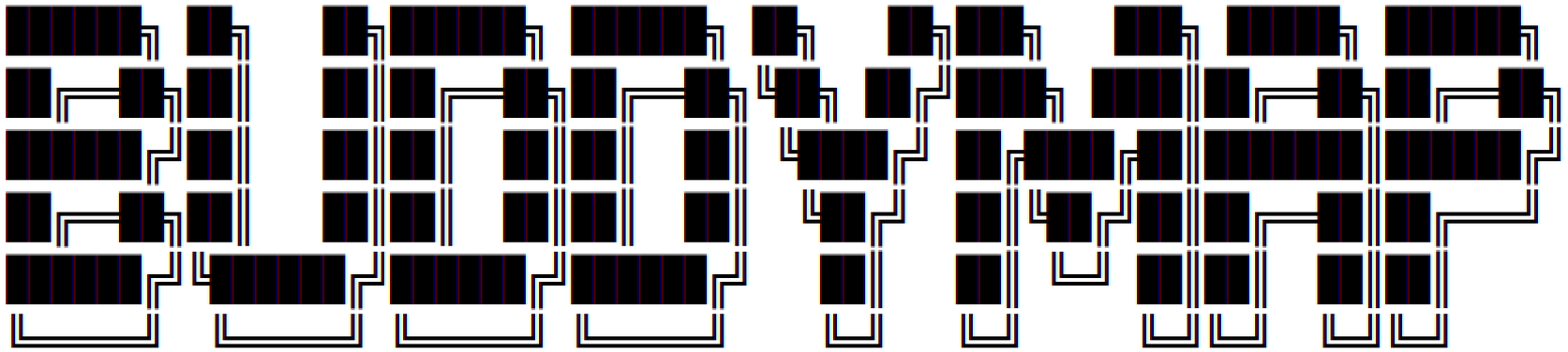
2. **People You May Know**
   → menampilkan rekomendasi teman (level 2) dari teman(level 1)

3. **Find Connection Path**
   → menunjukkan jalur pertemanan antara user pertama dan user kedua

4. **Display All Connection**
   → menampilkan list teman/koneksi dari seorang user

5. **Remove Connection**
   → menghapus koneksi pertemanan antara 2 user

Main Menu



(•‿•) ☀ BuddyMap — find your path, buddy!

==== MAIN MENU ====
1. Add Connection
2. People You May Know
3. Find Connection Path
4. Display All Connection
5. Remove Connection
0. Exit
Choose option:

Main Menu

```
public class Menu {  2 usages   Maria Elvina +1

    public void start() {  1 usage   Maria Elvina
        int choice;
        do {
            printHeader();
            showMenu();
            choice = getUserChoice();

            switch (choice) {
                case 1 -> addFriendshipMenu();
                case 2 -> peopleYouMayKnowMenu();
                case 3 -> findPathMenu();
                case 4 -> showGraphMenu();
                case 5 -> removeConnectionMenu();
                case 0 -> System.out.println("Exiting program...");
                default -> System.out.println("Invalid option!");
            }

        } while (choice != 0);
        sc.close();
```

ADD CONNECTION MENU

```
private void addFriendshipMenu() {  1 usage   Maria Elvina
    while (true) {
        System.out.println("\n=== ADD CONNECTION ===");
        System.out.print("Enter your name: ");
        String p1 = sc.nextLine();
        System.out.print("Enter your friend: ");
        String p2 = sc.nextLine();
        graph.addFriendship(p1, p2);
        System.out.println("Connection added!");
        waitEnter();

        System.out.println("\nSelect next action:");
        System.out.println("1. Add another");
        System.out.println("0. Back to Main Menu");
        System.out.print("Choose: ");
        String pilih = sc.nextLine();

        if (!pilih.equals("1")) return;
    }
}
```

ADD CONNECTION METHOD

```
public void addFriendship(String personA, String personB) {
    if (!graph.containsKey(personA)) {
        addPerson(personA);
    }
    if (!graph.containsKey(personB)) {
        addPerson(personB);
    }
    List<String> daftarTemanPersonA = graph.get(personA);
    List<String> daftarTemanPersonB = graph.get(personB);
    if (!daftarTemanPersonA.contains(personB)) {
        daftarTemanPersonA.add(personB);
    }
    if (!daftarTemanPersonB.contains(personA)) {
        daftarTemanPersonB.add(personA);
    }
}
```

PEOPLE YOU MAY KNOW MENU

```
private void peopleYouMayKnowMenu() {  1 usage   Maria Elvina
    while (true) {
        System.out.println("\n=== PEOPLE YOU MAY KNOW ===");
        System.out.print("Enter name: ");
        String name = sc.nextLine();

        graph.displayPeopleYouMayKnow(name);

        waitEnter();

        System.out.println("\nSelect next action: ");
        System.out.println("1. Search again");
        System.out.println("0. Back to Main Menu");
        System.out.print("Choose: ");
        String pilih = sc.nextLine();

        if (!pilih.equals("1")) return;
    }
}
```

PEOPLE YOU MAY KNOW METHOD (1)

```
public void displayPeopleYouMayKnow(String person) {  1 usage   RofifahZain

    if (!graph.containsKey(person)) {
        System.out.println("User \"" + person + "\" doesn't exist in the network.");
        return;
    }

    List<String> temanLangsung = graph.get(person);

    Set<String> level2 = new HashSet<>();

    Set<String> visited = new HashSet<>();
    visited.add(person);
    visited.addAll(temanLangsung);

    Queue<String> queue = new LinkedList<>();

    Map<String, Integer> level = new HashMap<>();
    for (String t : temanLangsung) {
        queue.add(t);
        level.put(t, 1);
    }
```

PEOPLE YOU MAY KNOW METHOD (2)

```
while (!queue.isEmpty()) {
    String current = queue.poll();
    int currentLevel = level.get(current);

    if (currentLevel == 2) continue;

    for (String friend : graph.get(current)) {
        if (!visited.contains(friend)) {

            int nextLevel = currentLevel + 1;

            if (nextLevel == 2) {
                level2.add(friend);
            }

            visited.add(friend);
            queue.add(friend);
            level.put(friend, nextLevel);
        }
    }
}
```

FIND CONNECTION PATH MENU

```
private void findPathMenu() {  1 usage   Tavasya Alia A

    while (true) {
        System.out.println("\n=== FIND CONNECTION PATH ===");

        System.out.print("From: ");
        String start = sc.nextLine();

        System.out.print("To: ");
        String target = sc.nextLine();

        graph.findPath(start, target);
        waitEnter();

        System.out.println("\nSelect next action: ");
        System.out.println("1. Search again");
        System.out.println("0. Back to Main Menu");
        System.out.print("Choose: ");
        String pilih = sc.nextLine();

        if (!pilih.equals("1")) return;
    }
}
```

FIND CONNECTION PATH METHOD (1)

```java
// Cari jalur path BFS
public void findPath(String start, String target) {  1 usage  Maria Elvina
    if (!graph.containsKey(start) || !graph.containsKey(target)) {
        System.out.println("One Name Not Found in Graph.");
        return;
    }

    Queue<String> queue = new LinkedList<>();
    Map<String, String> parent = new HashMap<>();
    Set<String> visited = new HashSet<>();

    queue.add(start);
    visited.add(start);
    parent.put(start, null);
```

FIND CONNECTION PATH METHOD (2)

```
boolean found = false;

while (!queue.isEmpty()) {
    String current = queue.poll();

    if (current.equals(target)) {
        found = true;
        break;
    }

    for (String neighbor : graph.get(current)) {
        if (!visited.contains(neighbor)) {
            visited.add(neighbor);
            parent.put(neighbor, current);
            queue.add(neighbor);
        }
    }
}
```

FIND CONNECTION PATH METHOD (3)

```
        if (!found) {
            System.out.println("\nNo Path exists from " + start + " to " + target);
            return;
        }

        List<String> path = new ArrayList<>();
        String step = target;
        while (step != null) {
            path.add(step);
            step = parent.get(step);
        }
        Collections.reverse(path);

        System.out.println("\nPath from " + start + " to " + target + ":");
        for (int i = 0; i < path.size(); i++) {
            System.out.print(path.get(i));
            if (i < path.size() - 1) System.out.print(" -> ");
        }
        System.out.println();
    }
```

DISPLAY ALL CONNECTION MENU

```
private void showGraphMenu() {  1 usage   Tavasya Alia A
    System.out.println("\n=== ALL CONNECTION ===");
    graph.printGraph();
    waitEnter();
}
```

DISPLAY ALL CONNECTION METHOD

```
    public void printGraph() {  1 usage   Maria Elvina
        System.out.println("\n===== Graph =====");
        for (String namaOrang : graph.keySet()) {
            System.out.println(namaOrang + " -> " + graph.get(namaOrang)); }
    }
}
```

REMOVE CONNECTION MENU (1)

```
private void removeConnectionMenu() {  1 usage   Tavasya Alia A

    while (true) {
        System.out.println("\n=== REMOVE CONNECTION ===");

        System.out.print("Enter your name: ");
        String p1 = sc.nextLine().trim();

        System.out.print("Enter name of friend to remove: ");
        String p2 = sc.nextLine().trim();

        if (!graph.contains(p1) || !graph.contains(p2)) {
            System.out.println("\nOne or both names do not exist in the network!");
            waitEnter();
        } else if (!graph.areConnected(p1, p2)) {
            System.out.println("\nThey are not connected!");
            waitEnter();
        } else {
            graph.removeConnection(p1, p2);
            System.out.println("\nConnection removed!");
            waitEnter();
        }
```

REMOVE CONNECTION MENU (2)

```
                System.out.println("\nSelect next action:");
                System.out.println("1. Remove another");
                System.out.println("0. Back to Main Menu");
                System.out.print("Choose: ");

                String pilih = sc.nextLine();

                if (!pilih.equals("1")) return;
            }
        }
}
```

REMOVE CONNECTION METHOD

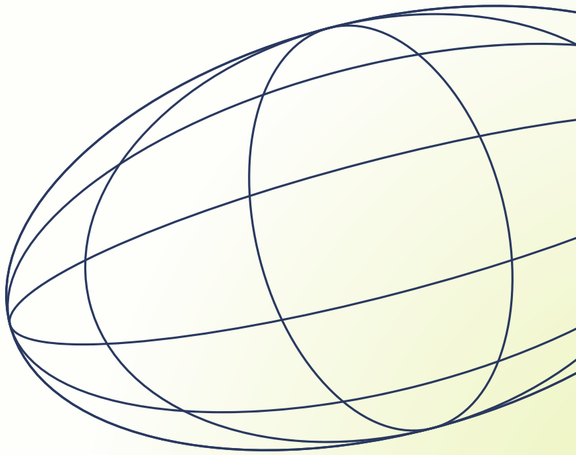
```
public boolean contains(String name) {  3 usages  Tavasya Alia A
    return graph.containsKey(name);
}

public boolean areConnected(String a, String b) {  1 usage  Tavasya Alia A
    return contains(a) && graph.get(a).contains(b);
}

public void removeConnection(String a, String b) {  1 usage  Tavasya Alia A
    graph.get(a).remove(b);
    graph.get(b).remove(a);
}
```

LINK GITHUB

https://github.com/mariaelvina/FinalProjectD9ASD

Algorithm & Data Structure

thank you.



by: Group 9