

PRAKTIKUM ALGORITMA DAN STRUKTUR DATA
JOB SHEET PERTEMUAN KE-15



RIO TRI PRAYOGO

TI 1A

26

2341720236

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2024

Graph

Praktikum 1 : Implementasi Graph menggunakan Linked List

Percobaan :

Node26

```
package minggu15;

public class Node26 {
    int data;
    Node26 prev, next;
    int jarak;

    Node26(Node26 prev, int data, int jarak, Node26 next) {
        this.prev = prev;
        this.data = data;
        this.jarak = jarak;
        this.next = next;
    }
}
```

DoubleLinkedLists26

```
package minggu15;

public class DoubleLinkedLists26 {
    Node26 head;
    int size;

    public DoubleLinkedLists26() {
        head = null;
        size = 0;
    }

    public boolean isEmpty() {
        return head == null;
    }

    public int size() {
```

```

        return size;
    }

    public void clear() {
        head = null;
        size = 0;
    }

    public void addFirst(int item, int jarak) {
        if (isEmpty()) {
            head = new Node26(null, item, jarak, null);
        } else {
            Node26 newNode = new Node26(null, item, jarak, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    public void remove(int index) {
        Node26 current = head;
        while (current != null) {
            if (current.data == index) {
                if (current.prev != null) {
                    current.prev.next = current.next;
                } else {
                    head = current.next;
                }
                if (current.next != null) {
                    current.next.prev = current.prev;
                }
                break;
            }
            current = current.next;
        }
        size--;
    }

```

```

    }

    public int get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas.");
        }
        Node26 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.data;
    }

    public int getJarak(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas.");
        }
        Node26 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.jarak;
    }
}

```

Graph26

```

package minggu15;

public class Graph26 {
    int vertex;
    DoubleLinkedLists26 list[];

    public Graph26(int v) {
        vertex = v;
        list = new DoubleLinkedLists26[v];
        for (int i = 0; i < v; i++) {
            list[i] = new DoubleLinkedLists26();
        }
    }
}

```

```

    }
}

public void addEdge(int asal, int tujuan, int jarak) {
    list[asal].addFirst(tujuan, jarak);
}

public void degree(int asal) throws Exception {
    int k, totalIn = 0, totalOut = 0;
    for (int i = 0; i < vertex; i++) {
        for (int j = 0; j < list[i].size(); j++) {
            if (list[i].get(j) == asal) {
                ++totalIn;
            }
        }
        for (k = 0; k < list[asal].size(); k++) {
            list[asal].get(k);
        }
        totalOut = k;
    }

    System.out.println("InDegree dari Gedung " + (char) ('A' + asal) +
": " + totalIn);

    System.out.println("OutDegree dari Gedung " + (char) ('A' + asal) +
": " + totalOut);

    System.out.println("Degree dari Gedung " + (char) ('A' + asal) + ":
" + (totalIn + totalOut));
}

public void removeEdge(int asal, int tujuan) throws Exception {
    for (int i = 0; i < vertex; i++) {
        if (i == tujuan) {
            list[asal].remove(tujuan);
        }
    }
}

public void removeAllEdges() {

```

```

        for (int i = 0; i < vertex; i++) {
            list[i].clear();
        }
        System.out.println("Graf berhasil dikosongkan");
    }

    public void printGraph() throws Exception {
        for (int i = 0; i < vertex; i++) {
            if (list[i].size() > 0) {
                System.out.println("Gedung " + (char) ('A' + i) + "
terhubung dengan ");
                for (int j = 0; j < list[i].size(); j++) {
                    System.out.print((char) ('A' + list[i].get(j)) + " (" +
list[i].getJarak(j) + " m), ");
                }
                System.out.println("");
            }
        }
        System.out.println("");
    }
}

```

GraphMain26

```

package minggu15;

public class GraphMain26 {
    public static void main(String[] args) throws Exception {
        Graph26 gedung = new Graph26(6);
        gedung.addEdge(0, 1, 50);
        gedung.addEdge(0, 2, 100);
        gedung.addEdge(1, 3, 70);
        gedung.addEdge(2, 3, 40);
        gedung.addEdge(3, 4, 60);
        gedung.addEdge(4, 5, 80);
        gedung.degree(0);
        gedung.printGraph();
        gedung.removeEdge(1, 3);
        gedung.printGraph();
    }
}

```

```

    }
}

```

Output :

Sebelum removeEdge()	Setelah removeEdge()
<pre> InDegree dari Gedung A: 0 OutDegree dari Gedung A: 2 Degree dari Gedung A: 2 Gedung A terhubung dengan C (100 m), B (50 m), Gedung B terhubung dengan D (70 m), Gedung C terhubung dengan D (40 m), Gedung D terhubung dengan E (60 m), Gedung E terhubung dengan F (80 m), </pre>	<pre> Gedung A terhubung dengan C (100 m), B (50 m), Gedung C terhubung dengan D (40 m), Gedung D terhubung dengan E (60 m), Gedung E terhubung dengan F (80 m), </pre>

Pertanyaan :

- Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!
 - Kode dalam *jobsheet* memiliki error yaitu **Edge** yang telah dihapus dengan `removeEdge()` akan tetap muncul pada saat ditampilkan menggunakan `printGraph()`. Untuk memperbaiki error tersebut dengan menambahkan `size--;` dalam method `remove()` seperti pada laporan diatas.
- Pada class `Graph`, terdapat atribut `list[]` bertipe `DoubleLinkedList`. Sebutkan tujuan pembuatan variabel tersebut!
 - Atribut `list[]` berfungsi untuk menyimpan setiap **Graph** dari class **DoubleLinkedLists** dimana setiap elemen dari array mempresentasikan satu buah **Graph**.
- Jelaskan alur kerja dari method **removeEdge**!
 - Method `removeEdge()` berfungsi untuk menghapus garis penghubung antar `Graph` dan memiliki alur dimana pada awalnya melakukan perulangan pada seluruh **Graph(vertex)** dan jika dalam perulangan tersebut terdapat **Graph** yang sesuai dengan parameter tujuan maka **Edge** yang menghubungkan parameter asal dan tujuan akan dihapus menggunakan method `remove()` dari class **DoubleLinkedLists**.
- Apakah alasan pemanggilan method **addFirst()** untuk menambahkan data, bukan method `add` jenis lain saat digunakan pada method **addEdge** pada class `Graph`?
 - Penggunaan method `addFirst()` dalam method `addEdge()` memiliki fungsi supaya **Graph** dimasukkan dalam keadaan terurut. **Graph** dimasukkan didepan supaya lebih efisien karena jika dimasukkan dibelakang maka harus melakukan penelusuran seluruh **Graph** untuk menemukan data paling belakang sehingga tidak efisien.
- Modifikasi kode program sehingga dapat dilakukan pengecekan apakah terdapat jalur antara suatu node dengan node lainnya, seperti contoh berikut (Anda dapat memanfaatkan Scanner).

Masukkan gedung asal: 2

Masukkan gedung tujuan: 3

Gedung C dan D bertetangga

Masukkan gedung asal: 2

Masukkan gedung tujuan: 5

Gedung C dan F tidak bertetangga

 - Untuk mengecek jalur antar node tersebut bisa menambahkan:
 - Kode**

```

...
public void checkEdge(int asal, int tujuan) throws Exception {

```

```

        for (int i = 0; i < list[asal].size(); i++) {
            if (list[asal].get(i) == tujuan) {
                System.out.println(
                    "Gedung " + (char) ('A' + asal) + " dan
Gedung " + (char) ('A' + tujuan) + " bertetangga");
            } else {
                System.out.println("Gedung " + (char) ('A' + asal)
+ " dan Gedung " + (char) ('A' + tujuan)
                    + " tidak bertetangga");
            }
        }
    }
}
...

```

GraphMain26

```

...

    System.out.print("Masukkan gedung asal: ");
    int asal = scan.nextInt();
    System.out.print("Masukkan gedung tujuan: ");
    int tujuan = scan.nextInt();
    gedung.checkEdge(asal, tujuan);
}
...

```

- **Output**

Masukkan gedung asal: 2	Masukkan gedung asal: 2
Masukkan gedung tujuan: 3	Masukkan gedung tujuan: 5
Gedung C dan Gedung D bertetangga	Gedung C dan Gedung F tidak bertetangga

Praktikum 2 : Implementasi Graph menggunakan Matriks

Percobaan :

GraphMatriks26

```

package minggu15;

public class GraphMatriks26 {
    int vertex;
    int[][] matriks;

    public GraphMatriks26(int v) {

```



```

        vertex = v;
        matriks = new int[v][v];
    }

    public void makeEdge(int asal, int tujuan, int jarak) {
        matriks[asal][tujuan] = jarak;
    }

    public void removeEdge(int asal, int tujuan) {
        matriks[asal][tujuan] = 0;
    }

    public void printGraph() {
        for (int i = 0; i < vertex; i++) {
            System.out.print("Gedung " + (char) ('A' + i) + ": ");
            for (int j = 0; j < vertex; j++) {
                System.out.print("Gedung " + (char) ('A' + j) + " (" +
matriks[i][j] + " m), ");
            }
            System.out.println();
        }
    }
}

```

GraphMain26

```

package minggu15;

public class GraphMain26 {
    public static void main(String[] args) {
        GraphMatriks26 gdg = new GraphMatriks26(4);
        gdg.makeEdge(0, 1, 50);
        gdg.makeEdge(1, 0, 60);
        gdg.makeEdge(1, 2, 70);
        gdg.makeEdge(2, 1, 80);
        gdg.makeEdge(2, 3, 40);
        gdg.makeEdge(3, 0, 90);
        gdg.printGraph();
    }
}

```

```

        System.out.println("Hasil setelah penghapusan edge");

        gdg.removeEdge(2, 1);

        gdg.printGraph();

    }

}

```

Output :

```

Gedung A: Gedung A (0 m), Gedung B (50 m), Gedung C (0 m), Gedung D (0 m),
Gedung B: Gedung A (60 m), Gedung B (0 m), Gedung C (70 m), Gedung D (0 m),
Gedung C: Gedung A (0 m), Gedung B (80 m), Gedung C (0 m), Gedung D (40 m),
Gedung D: Gedung A (90 m), Gedung B (0 m), Gedung C (0 m), Gedung D (0 m),
Hasil setelah penghapusan edge
Gedung A: Gedung A (0 m), Gedung B (50 m), Gedung C (0 m), Gedung D (0 m),
Gedung B: Gedung A (60 m), Gedung B (0 m), Gedung C (70 m), Gedung D (0 m),
Gedung C: Gedung A (0 m), Gedung B (0 m), Gedung C (0 m), Gedung D (40 m),
Gedung D: Gedung A (90 m), Gedung B (0 m), Gedung C (0 m), Gedung D (0 m),

```

Pertanyaan :

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!
 - Kode dalam *jobsheet* memiliki output tidak sesuai dengan percobaan dan sudah saya perbaiki pada laporan diatas. Baris yang tidak sesuai berada pada method `printGraph()` dimana setelah perulangan `for j` dilakukan seleksi `if (matriks[i][j] != -1)` sehingga output tidak memunculkan **Edge Graph** yang dihapus, dengan menghapus seleksi tersebut maka **Edge Graph** yang dihapus akan dimunculkan sesuai dengan nilai dalam method `removeEdge()` dimana supaya sesuai dengan output yang diharapkan nilai dalam method tersebut diubah menjadi '0'.
2. Apa jenis graph yang digunakan pada Percobaan 2?
 - Pada percobaan 2 jenis **Graph** yang digunakan adalah jenis **Graph Matrix** yang menggunakan *array* 2D.
3. Apa maksud dari dua baris kode berikut?


```
gdg.makeEdge(1, 2, 70);
gdg.makeEdge(2, 1, 80);
```

 - Dua baris kode diatas bermaksud untuk menyambungkan/membuat jalur(*Edge*) untuk 2 buah **Graph**. Dimana parameter pertama merupakan "Graph asal", kedua merupakan "Graph tujuan", dan ketiga merupakan jarak dari kedua **Graph**.
4. Modifikasi kode program sehingga terdapat method untuk menghitung degree, termasuk `inDegree` dan `outDegree`!
 - Untuk memunculkan Degree, `InDegree`, dan `OutDegree` bisa menambahkan:
 - **Kode**

```

...

public void degree(int asal) {
    int totalIn = 0, totalOut = 0;
    for (int i = 0; i < vertex; i++) {
        if (matriks[i][asal] != 0) {
            totalIn++;
        }
    }
}

```

```

    }

    for (int j = 0; j < vertex; j++) {
        if (matriks[asal][j] != 0) {
            totalOut++;
        }
    }

    System.out.println("InDegree dari Gedung " + (char) ('A' +
asal) + ": " + totalIn);

    System.out.println("OutDegree dari Gedung " + (char) ('A'
+ asal) + ": " + totalOut);

    System.out.println("Degree dari Gedung " + (char) ('A' +
asal) + ": " + (totalIn + totalOut));

    }

    ...

```

GraphMain26

```

...

    gdg.degree(0);

...

```

- **Output**

```

Gedung A: Gedung A (0 m), Gedung B (50 m), Gedung C (0 m), Gedung D (0 m),
Gedung B: Gedung A (60 m), Gedung B (0 m), Gedung C (70 m), Gedung D (0 m),
Gedung C: Gedung A (0 m), Gedung B (0 m), Gedung C (0 m), Gedung D (40 m),
Gedung D: Gedung A (90 m), Gedung B (0 m), Gedung C (0 m), Gedung D (0 m),
InDegree dari Gedung A: 2
OutDegree dari Gedung A: 1
Degree dari Gedung A: 3

```

Tugas Praktikum

1. Modifikasi kode program pada class GraphMain sehingga terdapat menu program yang bersifat dinamis, setidaknya terdiri dari:

- a) Add Edge
- b) Remove Edge
- c) Degree
- d) Print Graph
- e) Cek Edge

Pengguna dapat memilih menu program melalui input Scanner

➤ **Kode:**

(Kode dalam *Node26*, *DoubleLinkedLists26*, dan *Graph26* sama seperti dalam Percobaan 1)

GraphMain26

```

package minggu15;

import java.util.Scanner;

```

```

public class GraphMain26 {
    public static void main(String[] args) throws Exception {
        Scanner scan = new Scanner(System.in);
        System.out.print("Masukkan kapasitas Graph: ");
        int kapasitas = scan.nextInt();
        Graph26 gedung = new Graph26(kapasitas);
        boolean run = true;
        do {
            System.out.println("\n=====");
            System.out.println("Selamat datang di program Graph");
            System.out.println("=====");
            System.out.println("1. Add Edge");
            System.out.println("2. Remove Edge");
            System.out.println("3. Degree");
            System.out.println("4. Print Graph");
            System.out.println("5. Cek Edge");
            System.out.println("0. Keluar");
            System.out.print("(1/2/3/4/5/0): ");
            int input = scan.nextInt();
            System.out.println("=====");
            switch (input) {
                case 1:
                    System.out.println("Add Edge");
                    System.out.println("=====");
                    System.out.print("Masukkan gedung asal: ");
                    int addAsal = scan.nextInt();
                    System.out.print("Masukkan gedung tujuan: ");
                    int addTujuan = scan.nextInt();
                    System.out.print("Masukkan jarak antar gedung: ");
                    int addJarak = scan.nextInt();
                    System.out.println("=====");
                    gedung.addEdge(addAsal, addTujuan, addJarak);
                    System.out.println("=====");
            }
        } while (run);
    }
}

```

```

        break;
    case 2:
        System.out.println("Hapus Edge");
        System.out.println("=====
==");

        System.out.print("Masukkan gedung asal: ");
        int removeAsal = scan.nextInt();
        System.out.print("Masukkan gedung tujuan: ");
        int removeTujuan = scan.nextInt();
        System.out.println("=====
==");

        gedung.removeEdge(removeAsal, removeTujuan);
        System.out.println("=====
==");

        break;
    case 3:
        System.out.println("Graph Degree");
        System.out.println("=====
==");

        System.out.print("Masukkan gedung: ");
        int cekDegree = scan.nextInt();
        System.out.println("=====
==");

        gedung.degree(cekDegree);
        System.out.println("=====
==");

        break;
    case 4:
        System.out.println("Print Graph");
        System.out.println("=====
==");

        gedung.printGraph();
        System.out.println("=====
==");

        break;
    case 5:
        System.out.println("Cek Edge");
        System.out.println("=====
==");

        System.out.print("Masukkan gedung asal: ");

```

```

        int cekAsal = scan.nextInt();

        System.out.print("Masukkan gedung tujuan: ");

        int cekTujuan = scan.nextInt();

        System.out.println("=====
==");

        gedung.checkEdge(cekAsal, cekTujuan);

        System.out.println("=====
==");

        break;

    case 0:

        System.out.println("Keluar
program\nTerimakasih!");

        System.out.println("=====
==");

        run = false;

        break;

    default:

        System.out.println("Pilihan tidak valid!");

        System.out.println("=====
==");

        break;

    }

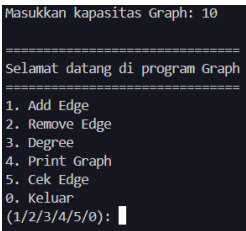
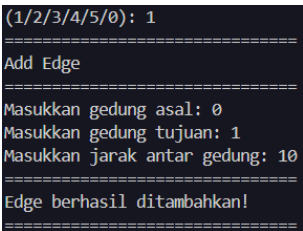
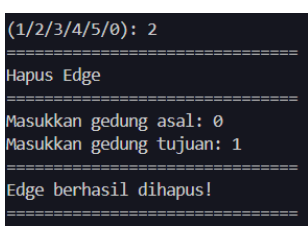
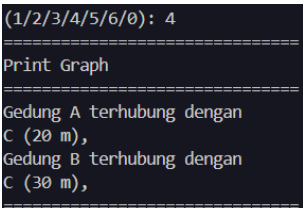
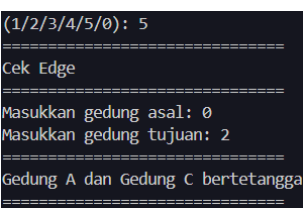
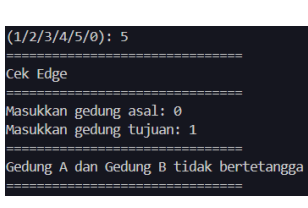
    } while (run);

}

}

```

➤ Output

Menu Utama	Add Edge	Remove Edge
		
Print Graph	Cek Edge (Bertetangga)	Cek Edge (Tidak Bertetangga)
		

2. Tambahkan method `updateJarak` pada Percobaan 1 yang digunakan untuk mengubah jarak antara dua node asal dan tujuan!

➤ **Kode**

(Kode merupakan tambahan dari Nomor 1)

Graph26

```
...  
  
    public void updateJarak(int asal, int tujuan, int newJarak)  
    throws Exception {  
  
        int i;  
  
        for (i = 0; i < list[asal].size(); i++) {  
  
            if (list[asal].get(i) == tujuan) {  
  
                list[asal].setJarak(i, newJarak);  
  
                System.out.println("Jarak antara Gedung " + (char)  
( 'A' + asal) + " dan Gedung " + (char) ( 'A' + tujuan)  
  
                    + " berhasil diubah menjadi " + newJarak + "  
m");  
  
                break;  
  
            }  
  
        }  
  
        if (i == list[asal].size()) {  
  
            System.out.println("Gedung " + (char) ( 'A' + asal) + "  
dan Gedung " + (char) ( 'A' + tujuan)  
  
                + " tidak memiliki Edge");  
  
        }  
  
    }  
  
    ...
```

DoubleLinkedLists26

```
...  
  
    public void setJarak(int index, int jarak) throws Exception {  
  
        if (index < 0 || index >= size) {  
  
            throw new Exception("Index di luar batas");  
  
        }  
  
        Node26 tmp = head;  
  
        for (int i = 0; i < index; i++) {  
  
            tmp = tmp.next;  
  
        }  
  
        tmp.jarak = jarak;  
  
    }  
  
    ...
```

...

Main26

```
...
        System.out.println("6. Update Jarak");
        System.out.println("0. Keluar");
        System.out.print("(1/2/3/4/5/6/0): ");
...
...
        case 6:
            System.out.println("Update Jarak");
            System.out.println("=====
==");

            System.out.print("Masukkan gedung asal: ");
            int updateAsal = scan.nextInt();
            System.out.print("Masukkan gedung tujuan: ");
            int updateTujuan = scan.nextInt();
            System.out.print("Masukkan jarak baru: ");
            int updateJarak = scan.nextInt();
            System.out.println("=====
==");

            gedung.updateJarak(updateAsal, updateTujuan,
updateJarak);

            System.out.println("=====
==");

            break;
...

```

➤ Output

Jarak Awal	Update Jarak
<pre>===== Print Graph ===== Gedung A terhubung dengan C (20 m), Gedung B terhubung dengan C (30 m), =====</pre>	<pre>(1/2/3/4/5/6/0): 6 ===== Update Jarak ===== Masukkan gedung asal: 0 Masukkan gedung tujuan: 2 Masukkan jarak baru: 50 ===== Jarak antara Gedung A dan Gedung C berhasil diubah menjadi 50 m =====</pre>
Jarak Akhir	Tidak Ada Edge
<pre>===== Print Graph ===== Gedung A terhubung dengan C (50 m), Gedung B terhubung dengan C (30 m), =====</pre>	<pre>(1/2/3/4/5/6/0): 6 ===== Update Jarak ===== Masukkan gedung asal: 0 Masukkan gedung tujuan: 1 Masukkan jarak baru: 50 ===== Gedung A dan Gedung B tidak memiliki Edge =====</pre>

3. Tambahkan method `hitungEdge` untuk menghitung banyaknya edge yang terdapat di dalam graf!

➤ **Kode**

(Kode merupakan tambahan dari Nomor 2)

Graph26

```
...
    public int hitungEdge() {
        int count = 0;
        for (int i = 0; i < vertex; i++) {
            count += list[i].size();
        }
        return count;
    }
...
```

Main26

```
...
        System.out.println("7. Banyak Edge");
        System.out.println("0. Keluar");
        System.out.print("(1/2/3/4/5/6/7/0): ");
...
...
        case 7:
            System.out.println("Banyak Edge");
            System.out.println("=====
==");
            System.out.println("Banyak Edge dalam Graph: " +
gedung.hitungEdge());
            System.out.println("=====
==");
            break;
...
...
```

➤ **Output**

Graph	Jumlah Edge
<pre>===== Print Graph ===== Gedung A terhubung dengan C (50 m), Gedung B terhubung dengan C (30 m), =====</pre>	<pre>(1/2/3/4/5/6/7/0): 7 ===== Banyak Edge ===== Banyak Edge dalam Graph: 2 =====</pre>