LAPORAN PRAKTIKUM

Algoritma Dan Struktur Data

Jobsheet - 12 : Graph

Nama : Ghoffar Abdul Ja’far
NIM : 41720035
Kelas : 1E

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2023/2024

Praktikum 1

Hasil:

```
InDegree dari Gedung A: 0
Degree dari Gedung A: 2
OutDegree dari Gedung A: 2
Gedung A terhubung dengan
C (2m), B (1m),
Gedung B terhubung dengan
D (3m),
Gedung C terhubung dengan
D (3m),
Gedung D terhubung dengan
E (4m),
Gedung E terhubung dengan
F (5m),
```

```
Gedung A terhubung dengan
C (2m), B (1m),
Gedung C terhubung dengan
D (3m),
Gedung D terhubung dengan
E (4m),
Gedung E terhubung dengan
F (5m),
```

Pernyataan

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!

```
public void remove(int index) {
    Node11 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;
    }
}
```

Before

```
public void remove(int index) {
    Node11 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--;
}
```

After

2. Pada class Graph, terdapat atribut list[] bertipe DoubleLinkedList. Sebutkan tujuan pembuatan variabel tersebut!
   = Digunakan untuk menyambungkan graph, karena graph dengan DLL pada prev disambungkan pada tiap list pada list[]
3. Jelaskan alur kerja dari method removeEdge!
   = Loop index i dari 0 hingga kurang dari vertex, lalu pengecekan kondisi apakah index i sama dengan tujuan jika iya maka vertex asal akan dihapus tujuannya.
4. Apakah alasan pemanggilan method addFirst() untuk menambahkan data, bukan method add jenis lain saat digunakan pada method addEdge pada class Graph?
   = Karena menggunakan method addFirst() dapat memungkinkan penambahan yang efisien tanpa perlu melalui seluruh elemen list
5. 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
```

Code:

```java
public boolean isPath(int asal, int tujuan) {
    boolean[] visited = new boolean[vertex];
    Stack<Integer> stack = new Stack<>();
    stack.push(asal);

    while (!stack.isEmpty()) {
        int current = stack.pop();

        if (current == tujuan) {
            return true;
        }

        if (!visited[current]) {
            visited[current] = true;

            for (int i = 0; i < list[current].size(); i++) {
                int neighbor = list[current].get(i);
                if (!visited[neighbor]) {
                    stack.push(neighbor);
                }
            }
        }
    }

    return false;
}
```

```java
public void checkPath() {
    Scanner scanner = new Scanner(System.in);

    System.out.print(s:"Masukkan gedung asal: ");
    int asal = scanner.nextInt();

    System.out.print(s:"Masukkan gedung tujuan: ");
    int tujuan = scanner.nextInt();

    if (isPath(asal, tujuan)) {
        System.out.println("Gedung " + (char) ('A' + asal) + " dan " + (char) ('A' + tujuan) + " bertetangga");
    } else {
        System.out.println("Gedung " + (char) ('A' + asal) + " dan " + (char) ('A' + tujuan) + " tidak
        bertetangga");
    }
}
```

Hasil :

```
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 bertetangga
```

Praktium 2

Hasil:

```
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!

```
ublic void removeEdge(int asal, int tujuan)
    matriks[asal][tujuan] = -1;
```

Before

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

After

2. Apa jenis graph yang digunakan pada Percobaan 2?
   = Directed Unweighted Graph
3. Apa maksud dari dua baris kode berikut?

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

   = Digunakan untuk membuat edge dengan asal, tujuan, dan jarak.
4. Modifikasi kode program sehingga terdapat method untuk menghitung degree, termasuk inDegree dan outDegree!
   Class GraphMatriks1:

```
public void degree(int asal) {
    int t = 0, totalIn = 0, totalOut = 0;
    for (int i = 0; i < vertex; i++) {
        // inDegree
        for (int j = 0; j < matriks[i].length; j++) {
            if (matriks[i][j] == asal) {
                ++totalIn;
            }
        }
        // outDegree
        for (int k = 0; k < matriks[asal].length; k++) {
            if (matriks[asal][k] > 0) {
                t++;
            }
        }
        totalOut = t;
    }
    System.out.println("InDegree dari Gedung " + (char) ('A' + asal) + ": " + totalIn);
    System.out.println("Degree dari Gedung " + (char) ('A' + asal) + ": " + (totalIn + totalOut));
    System.out.println("OutDegree dari Gedung " + (char) ('A' + asal) + ": " + totalOut);
}
```

   Class GraphMain11:

```
gdg.printGraph();
gdg.degree(asal:0);
```

   Hasil:

```
InDegree dari Gedung A: 11
Degree dari Gedung A: 15
OutDegree dari Gedung A: 4
```

Link repo:
https://github.com/GhoffarFitassin/P_Algoritma_Struktur_Data/tree/main/jobsheet12_Graph