LAPORAN PRAKTIKUM
STRUKTUR DATA NON LINIER
MODUL 6

"GRAPH"

Dosen Pengampu

JB. Budi Darmawan S.T., M.Sc.

DISUSUN OLEH

Yohanis Celvin D. P. U. Pati

245314033

PROGRAM STUDI INFORMATIKA

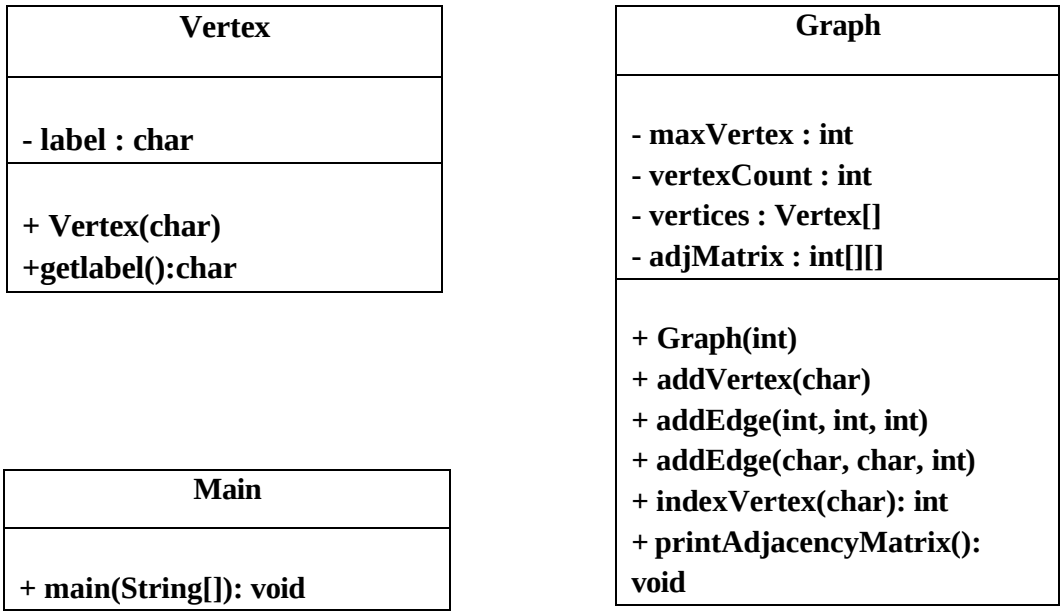
FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS SANATA DHARMA

YOGYAKARTA

2025

A. DIAGRAM UML



B. SOURCE CODE

- **Vertex**

```
6
7   /**
8    *
9    * @author Celvin Pati
10   */
11  public class Vertex {
        private char label;
13
14      public Vertex(char label) {
15          this.label = label;
16      }
17
18      public char getLabel() {
19          return label;
20      }
21  }
22
```

- **Graph**

Source History

```java
 * @author Celvin Pati
 */
public class Graph {
    private int maxVertex;
    private int vertexCount;
    private Vertex[] vertices;
    private int[][] adjMatrix;

    public Graph(int maxV) {
        this.maxVertex = maxV;
        this.vertexCount = 0;
        vertices = new Vertex[maxV];
        adjMatrix = new int[maxV][maxV];

        for (int i = 0; i < maxV; i++) {
            for (int j = 0; j < maxV; j++) {
                adjMatrix[i][j] = -1;
            }
        }
    }

    public void addVertex(char node) {
        vertices[vertexCount++] = new Vertex(node);
    }

    public int indexVertex(char label) {
        for (int i = 0; i < vertexCount; i++) {
            if (vertices[i].getLabel() == label) {
                return i;
            }
        }
        return -1;
    }

    public void addEdge(int a, int b, int c) {
        adjMatrix[a][b] = c;
        adjMatrix[b][a] = c; // Undirected
    }

    public void addEdge(char a, char b, int c) {
        int i = indexVertex(a);
        int j = indexVertex(b);

        if (i != -1 && j != -1) {
            adjMatrix[i][j] = c;
            adjMatrix[j][i] = c;
        }
    }

    public void printAdjacencyMatrix() {
        System.out.print("\t");
        for (int i = 0; i < vertexCount; i++) {
            System.out.print(vertices[i].getLabel() + "\t");
        }
        System.out.println();

        for (int i = 0; i < vertexCount; i++) {
            System.out.print(vertices[i].getLabel() + "\t");
            for (int j = 0; j < vertexCount; j++) {
                System.out.print(adjMatrix[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

- **Main**

```
 */
public class Main {
    public static void main(String[] args) {
        Graph graph = new Graph(5);

        graph.addVertex('A');
        graph.addVertex('H');
        graph.addVertex('W');
        graph.addVertex('C');
        graph.addVertex('D');

        graph.addEdge('A', 'H', 100);
        graph.addEdge('A', 'W', 800);
        graph.addEdge('H', 'W', 750);
        graph.addEdge('W', 'C', 400);
        graph.addEdge('C', 'D', 370);

        graph.printAdjacencyMatrix();
    }
}
```

C. OUTPUT

```
cd C:\Users\Windows\OneDrive\Documents\NetBeansProjects\Modul6SDNL; "JAVA_HOME=C:\\Prog
Scanning for projects...

----------------------< com.mycompany:Modul6SDNL >----------------------
Building Modul6SDNL 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ Modul6SDNL ---
skip non existing resourceDirectory C:\Users\Windows\OneDrive\Documents\NetBeansProject

--- compiler:3.13.0:compile (default-compile) @ Modul6SDNL ---
Recompiling the module because of changed source code.
Compiling 3 source files with javac [debug release 23] to target\classes

--- exec:3.1.0:exec (default-cli) @ Modul6SDNL ---
        A       H       W       C       D
A       -1      100     800     -1      -1
H       100     -1      750     -1      -1
W       800     750     -1      400     -1
C       -1      -1      400     -1      370
D       -1      -1      -1      370     -1
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  3.582 s
Finished at: 2025-11-02T23:43:28+07:00
------------------------------------------------------------------------
```

D. ANALISA

1. **Vertex**
   - **Vertex(char label)** Konstruktor yang membuat node/simpul baru dengan label tertentu.
   - Kaitan dengan konsep Tree Sama seperti Tree Node yang menyimpan data/label pada sebuah simpul.
2. **Graph**
   - **Graph(int maxV)** Membuat struktur graph dengan jumlah vertex maksimum. Inisialisasi adjacency matrix nilai -1 (tidak ada koneksi).
   - **addVertex(char node)** Menambahkan simpul baru ke dalam graph dan menyimpannya dalam array vertex.
   - **indexVertex(char label)** Mencari index simpul berdasarkan label diperlukan agar dapat pemetaan ke adjacency matrix.
   - **addEdge(int a, int b, int c)** Menambah sisi (edge) antara node ke-a dan node ke-b dengan bobot c pada adjacency matrix.
   - **addEdge(char a, char b, int c)** Versi lebih praktis menggunakan label huruf, bukan index.
   - **printAdjacencyMatrix()** Menampilkan hubungan antar node dalam bentuk matriks. Jika -1 berarti tidak terhubung.
   - Kaitannya dengan konsep Tree program ini untuk menampilkan sekumpulan objek (vertex/kota) dan hubungan antar objek tersebut berupa jarak (edge). Representasi yang dipakai adalah adjacency matrix, yaitu tabel yang menyimpan bobot hubungan antar vertex.
3. **Main**
   - **main(String[] args)** Program utama: membuat Graph, menambah vertex, menambah edge, lalu mencetak adjacency matrix. Ini menjalankan lifecycle Graph.
   - Hubungannya dengan konsep Tree Dalam struktur Tree, ini seperti membentuk tree, menambahkan node, dan menampilkan struktur tree hasilnya.

4. Output

Setelah program saya jalankan maka output yang ditampilkan output dalam bentuk Adjacency Matrix, yaitu tabel yang menunjukan hubungan antara simpul kota dalam Graph berbobot.

- Baris adalah kota asal.
- Kolom itu kota tujuan.
- Nilai jarak **(**km**)** itu menunjukkan kota tersebut terhubung.
- Nilai –1 menunjukkan kota tidak terhubung langsung.

a. Penjelasan data antar kota

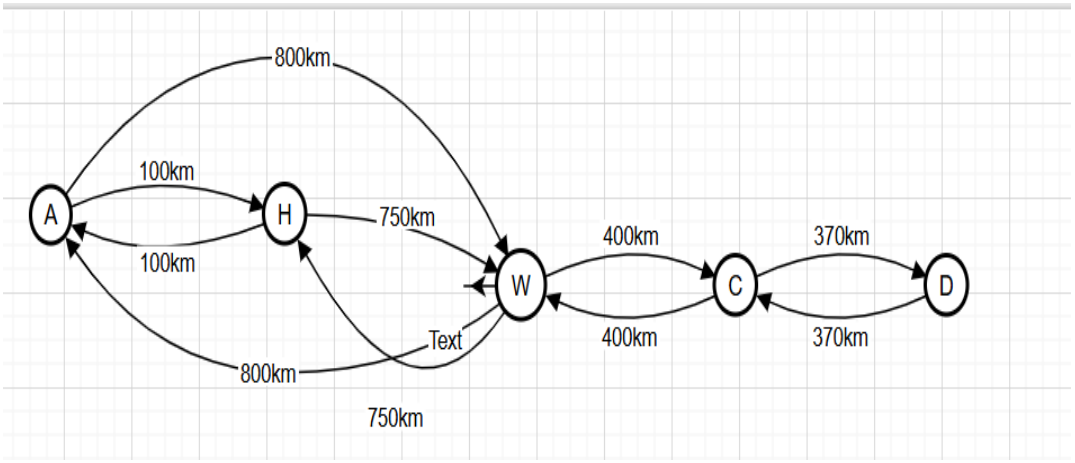
- Baris A H = 100 km Ada jalur langsung A ke H dengan jarak 100 km
- Baris A W = 800 km Jalur langsung A ke W tersedia
- Baris A - C dan A - D = -1 Tidak ada jalur langsung ke C dan D
- Karena Graph ini tidak berarah**,** hubungan bersifat timbal balik Contoh: Jika A terhubung ke H maka H juga terhubung ke A.

b. Analisis perkota

1. Kota A (Auckland)
    - Terhubung dengan H (Hamilton) jarak 100 km
    - Terhubung dengan W (Wellington) jarak 800 km
    - Tidak terhubung langsung ke C dan D
2. Kota H (Hamilton)
    - Terhubung dengan kota A jarak 100 km
    - Terhubung dengan kota W jarak 750 lm
    - kota penghubung antara A dan W
3. Kota W (Wellington)
    - Terhubung dengan kota A jarak 800 km
    - Terhubung dengan kota H jarak 750 km
    - Terhubung dengan C (Christchurch) jarak 400 km
    - Memiliki 3 hubungan

4. Kota C (Christchurch)
    - Terhubung dengan kota W jarak 400 km
    - Terhubung dengan kota D (Dunedin) jarak 370 km
    - Bagian dari jalur selatan New Zealand
5. Kota D (Dunedin)
    - Terhubung hanya ke C
    - kota paling ujung atau terminal jalur

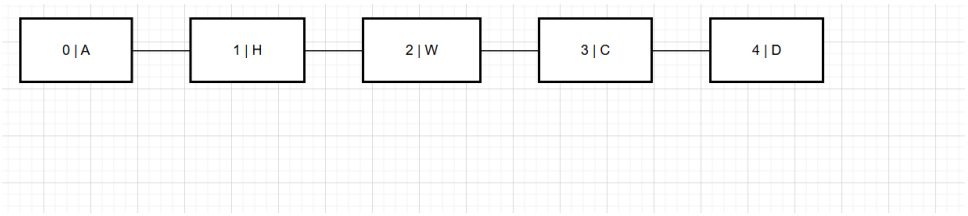
c. Ilustrasi graph (node & edge dengan bobot)



d. Ilustasi adjacency matrix nya

AdjacencyMatrix
Kota A H W C D
A -1 100 800 -1 -1
H 100 -1 750 -1 -1
W 800 750 -1 400 -1
C -1 -1 400 -1 370
D -1 -1 -1 370 -1

e. VertexList



E. REFERENSI