

VERSION 1.0  
OKTOBER 4, 2021



# [STRUKTUR DATA]

## MODUL 6, GRAPH

DISUSUN OLEH: - LIDYA FANKKY O.P  
- BELLA DWI MARDIANAN

DIAUDIT OLEH: - DIDIH RIZKI CHANDRANEGARA, S.KOM., M.KOM  
- IR. GITA INDAH M ST M.KOM

PRESENTED BY: TIM LAB-IT  
UNIVERSITAS MUHAMMADIYAH MALANG

## [STRUKTUR DATA]

---

### PERSIAPAN MATERI

Mahasiswa diharapkan mempelajari materi praktikum dengan baik, sesuai dengan materi yang diberikan oleh dosen pengajar dikelas. Terutama dalam penerapan materi OOP JAVA:

1. Graph

---

### TUJUAN

Mahasiswa mampu menguasai & menjelaskan konsep dari graph.

---

### TARGET MODUL

Mahasiswa mampu memahami :

1. Graph

---

### PERSIAPAN SOFTWARE/APLIKASI

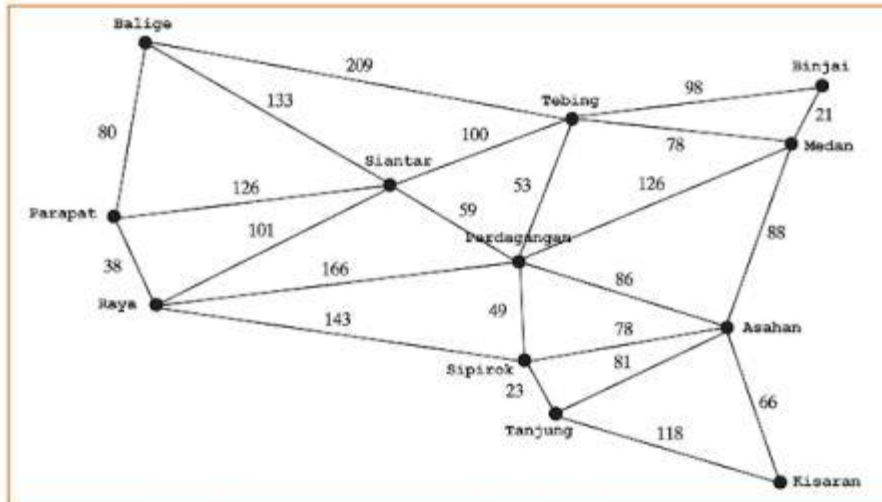
1. Java Development Kit
2. Java Runtime Environment
3. IDE (Intellij IDE, Eclipse, Netbeans, dll)

---

### MATERI POKOK

#### A. Graph

Graph adalah struktur data dengan relasi many to many, yaitu tiap element dapat memiliki 0 atau lebih dari 1 cabang. Graph terdiri dari 2 komponen yaitu Node (digunakan untuk menyimpan data) dan Edge (cabang, untuk menghubungkan node satu dengan node lain). Representasi visual dari graph adalah dengan menyatakan objek sebagai noktah, bulatan atau titik (Vertex), sedangkan hubungan antara objek dinyatakan dengan garis (Edge). Lebih Jelasnya dapat dilihat pada gambar dibawah:



1.1 Graph yang menunjukkan jarak antar kota

$$G = \{V, E\}$$

Dimana :

G = Graph

V = Simpul atau Vertex, atau Node, atau Titik

E = Busur atau Edge, atau arc

1.2 Representasi Graph

## B. Jenis-jenis Graph

### a) Undirected Graph

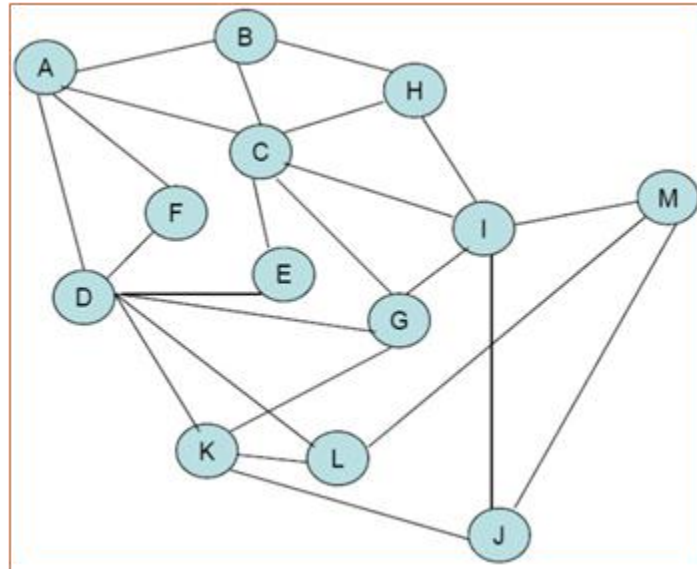
Undirected Graph merupakan graph yang tidak memiliki arah, sehingga dalam penulisannya dapat berlaku 2 arah atau bolak balik. Singkatnya perhatikanlah gambar berikut ini untuk dapat lebih memahami :



DARI GRAPH DIATAS ARAHNYA DAPAT DITULISKAN MANJADI  $\{A, B\}$  ATAU  $\{B, A\}$

Contoh lain sebagai berikut :





Notasi gambar diatas adalah :

$$G = \{V, E\}$$

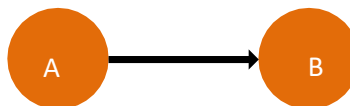
$$V = \{A, B, C, D, E, F, G, H, I, J, K, L, M\}$$

$$E = \{ (A,B), (A,C), (A,D), (A,F), (B,C), (B,H), (C,E), (C,G), (C,H), (C,I), (D,E), (D,F), (D,G), (D,K), (D,L), (E,F), (G,I), (G,K), (H,I), (I,J), (I,M), (J,K), (J,M), (L,K), (L,M) \}.$$

#### b) Directed Graph

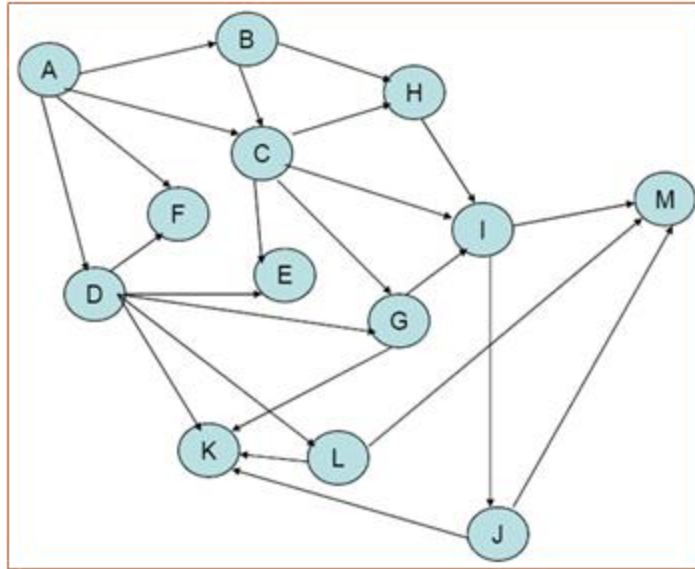
Dari graph dibawah arahnya dapat dituliskan menjadi  $\{A,B\}$  dan bukan  $\{B,A\}$ . Berbeda dengan Undirected graph yang tidak memiliki arah, directed graph memiliki arah yang tertuju pada node/vertex tertentu yang biasa dinotasikan dengan gambar anak panah.

Untuk lebih jelasnya perhatikan contoh dibawah ini :



**DARI GRAPH DIATAS ARAHNYA DAPAT DITULISKAN MANJADI  $\{A,B\}$  BUKAN  $\{B,A\}$**

Adapun untuk contoh lainnya seperti gambar dibawah ini :



Notasi dari gambar diatas:

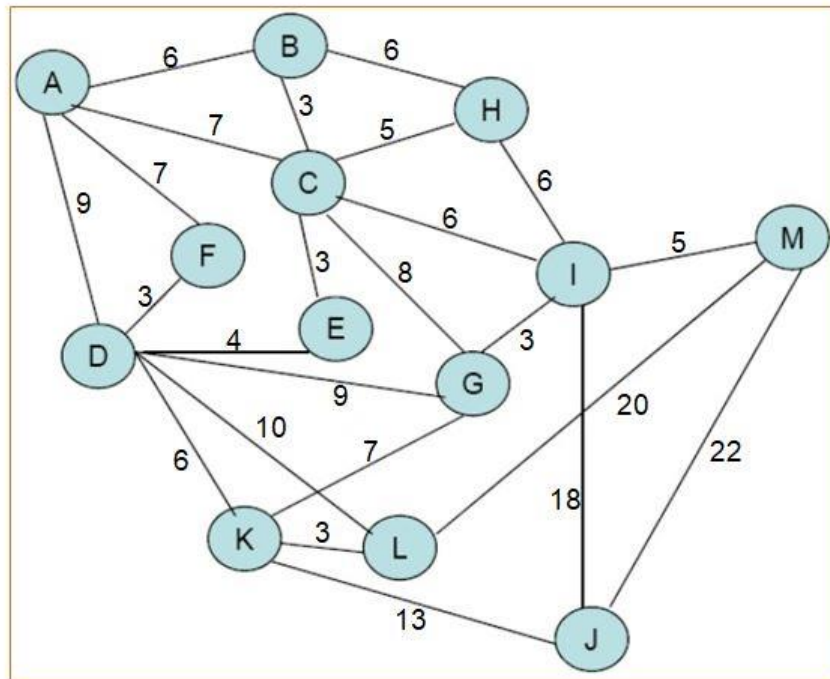
$G = \{V, E\}$

$V = \{A, B, C, D, E, F, G, H, I, J, K, L, M\}$

$E = \{(A,B), (A,C), (A,D), (A,F), (B,C), (B,H), (C,E), (C,G), (C,H), (C,I), (D,E), (D,F), (D,G), (D,K), (D,L), (E,F), (G,I), (G,K), (H,I), (I,J), (I,M), (J,K), (J,M), (L,K), (L,M)\}$ .

**c) Weight Graph**

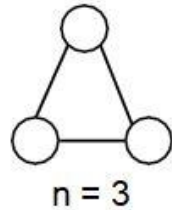
Sesuai namanya, Weight graph merupakan yang memiliki bobot atau nilai pada tiap - tiap nodenya. Untuk Lebih Jelasnya dapat memperhatikan gambar berikut:



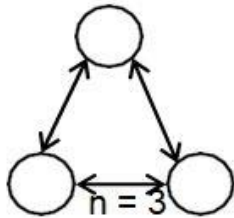
### C. Istilah – istilah pada Graph

#### a) Jumlah Edge

Jumlah pasangan edge yang mungkin (banyak maksimal edge) dapat dilihat dari jumlah node ( $n$ ). Dapat dihitung menggunakan formula Undirected Graph :  $n(n-1)/2$  dan untuk Directed Graph :  $n(n-1)$



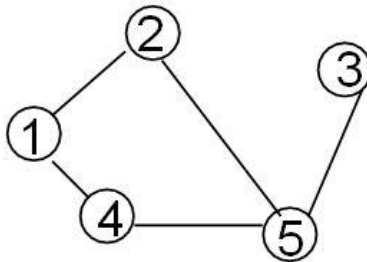
Undirected graph  
jumlah maks edge :  
3



Directed graph  
jumlah maks edge :  
6

#### b) Degree

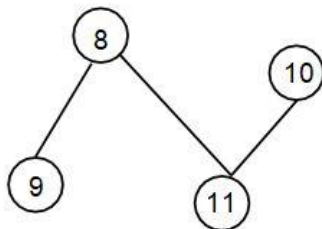
Jumlah cabang atau jumlah edge yang dimiliki node.



Degree(2): 2  
Degree(3): 1  
Degree(5): 3

#### c) Jumlah Degree

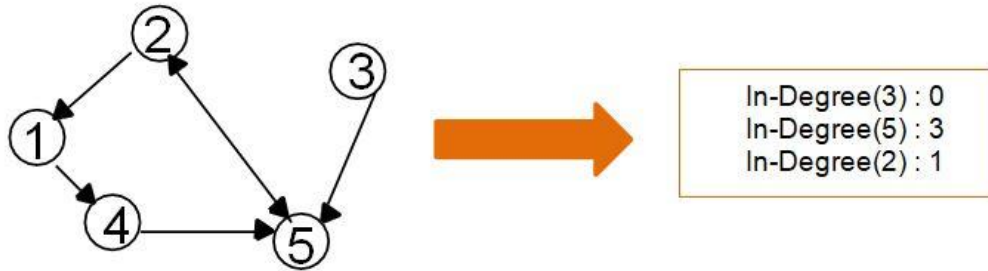
Jumlah degree adalah jumlah total cabang/degree yang ada pada graph. Dapat dirumuskan dengan notasi :  $2e$  ( $e$  merupakan jumlah edge).



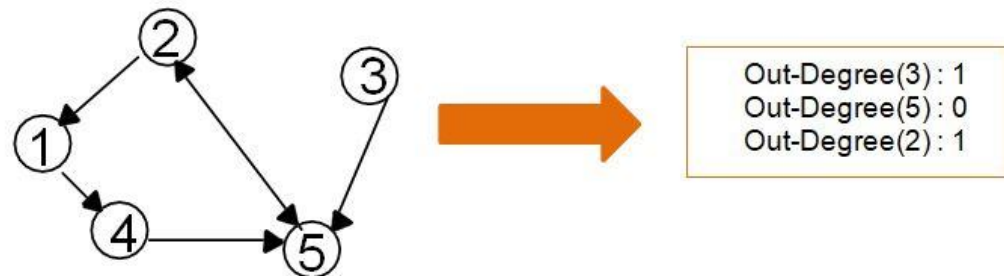
Jumlah Degree: 6

**d) In-Degree**

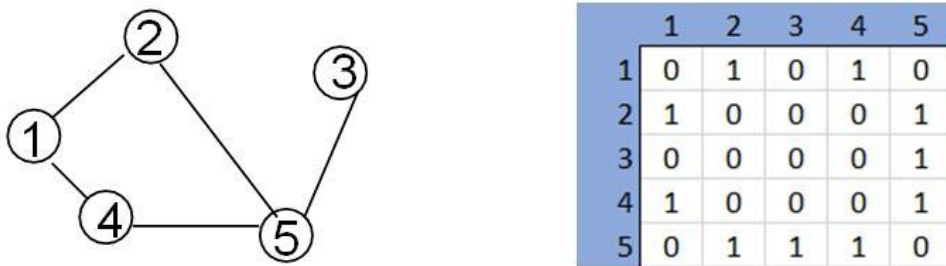
Jumlah edge yang masuk atau mengarah ke Node.

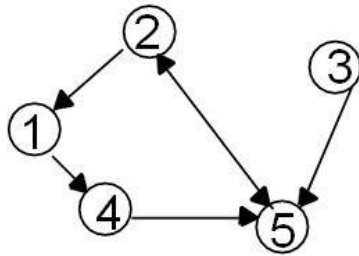
**e) Out-Degree**

Jumlah edge yang keluar dari Node.

**D. Adjacency Matriks**

Adjacency Matriks pada graph merupakan representasi graph kedalam bentuk matriks dengan ordo  $n \times n$ , dimana  $n$  adalah node. Dalam Adjacency Matriks pada graph ini, baris pada matriks berisi node asal sedangkan kolom pada matriks berisi node tujuan. Jika terdapat edge antar node maka diisi dengan nilai 1 sedangkan jika tidak ada node antar node diisi dengan nilai 0.





	1	2	3	4	5
1	0	0	0	1	0
2	1	0	0	0	1
3	0	0	0	0	0
4	0	0	0	0	1
5	0	1	1	0	0

### E. Graph Traversal

Graph Traversal merupakan metode penelusuran yang digunakan untuk mengunjungi tiap simpul/node secara sistematis. Metode penelusuran pada graph dibagi menjadi 2, yaitu:

- DFS (Depth First Search), melakukan kunjungan ke node-node dengan cara mengunjungi node terdalam/kebawah, setelah itu mencari tempat yang lainnya, dan sistemnya menggunakan stack.
- BFS (Breadth First Search), melakukan visit ke node-node dengan cara melebar kesamping, dan sistemnya menggunakan queue.

### F. Interface Graph

Interface Graph berisi method-method dasar dari graph.

Method	Deskripsi
<code>boolean addEdge(T v1, T v2, int w)</code>	Jika edge (v1,v2) belum terdapat pada graph, tambahkan edge dengan bobot w dan mengembalikan nilai true. Jika edge(v1,v2) sudah ada maka mengembalikan nilai false. Jika v1 dan v2 bukan merupakan vertex di graph maka throws <code>IllegalArgumentException</code> .
<code>boolean addVertex(T v)</code>	Jika vertex v tidak terdapat pada graph maka tambahkan pada graph dan mengembalikan nilai true. Jika vertex v sudah ada maka mengembalikan nilai false.
<code>void clear()</code>	Menghapus semua vertex dan edge pada graph.
<code>boolean containsEdge(T v1, T v2)</code>	Mengembalikan nilai true jika terdapat sebuah edge dari v1 dan v2 di graph dan mengembalikan nilai false jika sebaliknya. Jika v1 atau v2 bukan merupakan vertex pada graph maka throws <code>IllegalArgumentException</code> .
<code>boolean containsVertex(Object v)</code>	Mengembalikan nilai true jika v adalah vertex pada Graph dan mengembalikan nilai false jika v bukan merupakan vertex pada Graph.
<code>Set&lt;T&gt; getNeighbors(T v)</code>	Mengembalikan vertex-vertex yang terhubung dengan vertex v, dan vertex-vertex tersebut disimpan dalam object S. Jika v bukan merupakan vertex pada graph maka throws <code>IllegalArgumentException</code> .
<code>int getWeight(T v1, T v2)</code>	Mengembalikan bobot dari edge yang menghubungkan vertex v1 dan v2, jika edge (v1,v2) tidak ada maka mengembalikan nilai -1. Jika v1 atau v2 bukan merupakan vertex pada graph maka throws <code>IllegalArgumentException</code> .

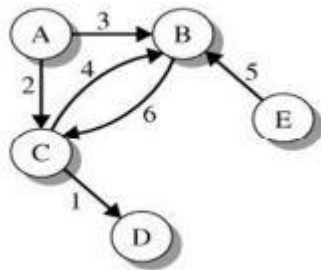


	<code>IllegalArgumentException</code> .
<code>boolean isEmpty()</code>	Mengembalikan nilai true jika graph sama sekali tidak mempunyai vertex dan mengembalikan nilai false jika memiliki minimal 1 buah vertex.
<code>int numberOfEdges()</code>	Mengembalikan jumlah edge pada graph.
<code>int numberOfVertices()</code>	Mengembalikan jumlah vertex pada graph.
<code>boolean removeEdge()</code>	Jika (v1,v2) merupakan edge, menghapus edge tersebut dan mengembalikan nilai true dan mengembalikan nilai false jika sebaliknya. Jika v1 atau v2 bukan merupakan vertex pada graph maka throws <code>IllegalArgumentException</code> .
<code>boolean removeVertex(Object v)</code>	Jika v adalah vertex pada graph, menghapus vertex v dari graph dan mengembalikan nilai true, dan mengembalikan nilai false jika sebaliknya.
<code>int setWeight(T v1, T v2, int w)</code>	Jika edge (v1, v2) terdapat pada graph, ubah bobot edge dan mengembalikan bobot sebelumnya jika tidak mengembalikan nilai false. Jika v1 atau v2 bukan vertex di graph, maka throws <code>IllegalArgumentException</code> .
<code>Set&lt;T&gt; vertexSet()</code>	Mengembalikan vertex-vertex yang terdapat pada Graph disimpan dalam object Set.

## LATIHAN PRAKTIKUM

### LATIHAN 1

Tentukan matriks Adjency dari graph dibawah ini :

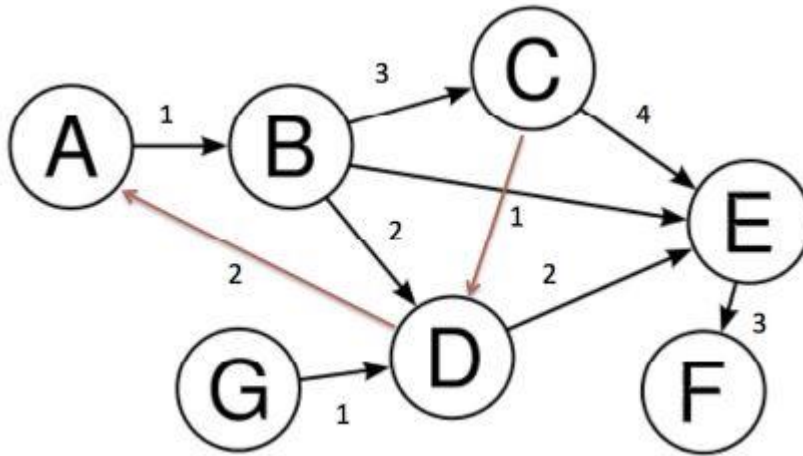


Kemudian tentukan jumlah edge, in-degree, dan out-degree.

## TUGAS

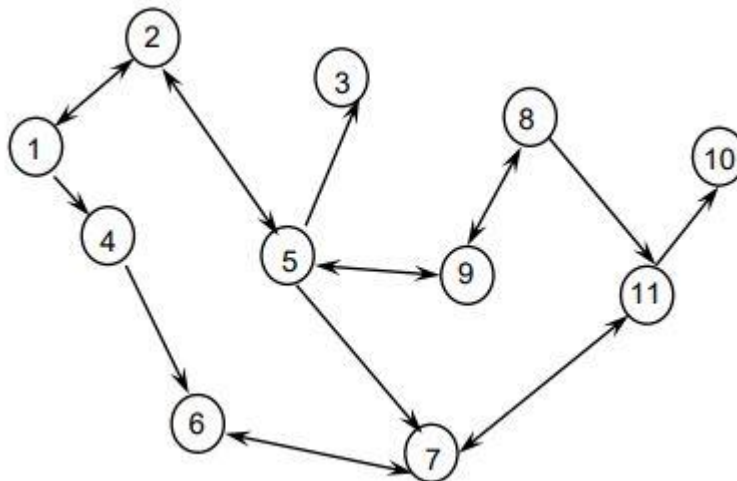
### KEGIATAN 1

1. Jelaskan kepada asisten tentang apa yang anda ketahui tentang graph dan jelaskan pula perbedaan antara Struktur data Tree dan Struktur data Graph.
2. Perhatikan gambar dibawah ini :



Jelaskan kepada Asisten :

- a. Notasi dari graph diatas (  $G=\{V,E\}$  )
  - b. Jumlah in-degree, jumlah out-degree, dan jumlah Edge
  - c. Adjacency Matriks dari gambar tersebut
  - d. Tentukan weight pada tiap-tipa node
3. Buatlah program yang dapat menentukan DFS (Depth First Search) dan BFS (Breadth First Search) dari gambar dibawah ini :



Kemudian jelaskan pula rute dari graph diatas sehingga menghasilkan output seperti pada program yang telah anda buat.

---

**CATATAN**

Diperbolehkan menggunakan library pada kegiatan 1.

**Aturan umum penulisan bahasa JAVA agar mudah di koreksi oleh asisten:**

1. Untuk nama kelas,interface,enum, dan yang lainnya biasakan menggunakan gaya CamelCase (diawali dengan huruf besar pada tiap kata untuk mengganti spasi) seperti: Kursi , JalanRaya, ParkiranGedung, dan lain seterusnya.
2. Untuk penulisan nama method, dan attribute diawali dengan huruf kecil di awal kata dan menggunakan huruf besar untuk kata setelahnya, seperti: getNamaJalan, namaJalan, harga, setNamaJalan, dan lain seterusnya.
3. Jika menggunakan IDE IntelliJ jangan lupa untuk memformat penulisan kode agar terlihat rapi menggunakan menu code -> show reformat file dialog -> centang semua field dan klik ok. Silahkan dikerjakan tanpa **copy – paste**.

---

**DETAIL PENILAIAN TUGAS**

Soal	Nilai
Kegiatan 1	80
Pemahaman Materi	20