GRAPH

TIM AJAR

ALGORITMA DAN STRUKTUR DATA

2023/2024

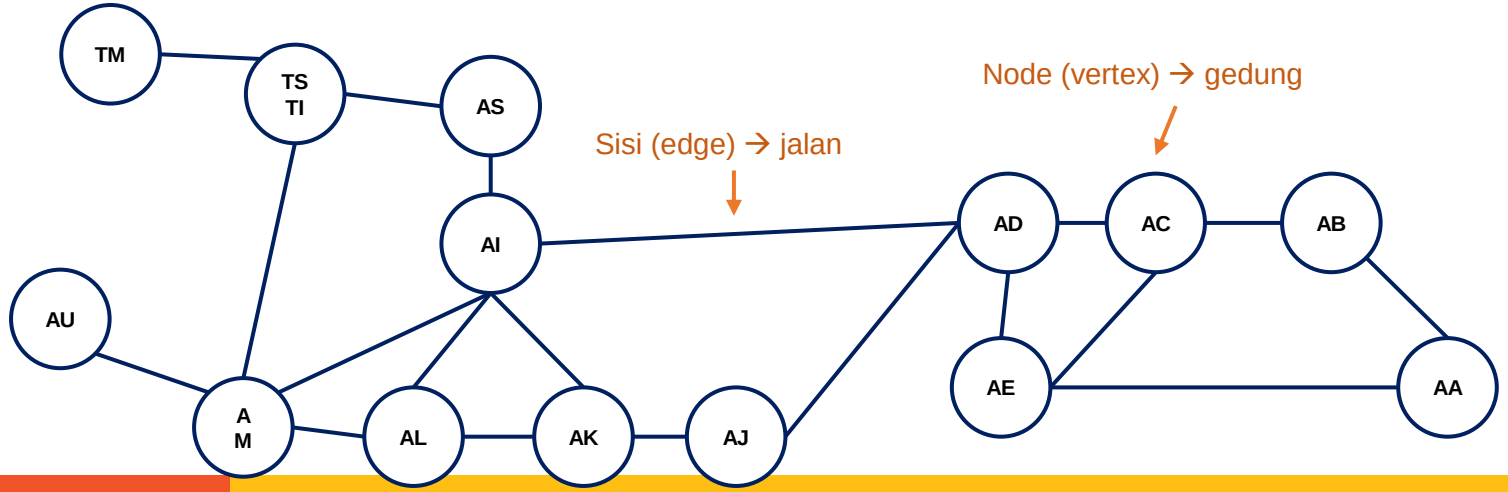
Capaian Pembelajaran

Setelah mempelajari materi Graf, mahasiswa diharapkan mampu

- Memahami definisi Graf dan terminologinya
- Memahami memodelkan permasalahan di dunia nyata menggunakan Graf
- Memahami merepresentasikan struktur data Graf

Definisi Graf

- Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut
- Contoh graf dalam mengilustrasikan sebagian gedung di Polinema:



Definisi Graf (2)

- Graf G = (V, E) adalah himpunan berhingga tak kosong V(G) dan himpunan E(G) mungkin kosong, yang elemen-elemennya merupakan himpunan pasangan tak berurut 2 elemen-elemen berbeda dari V(G)
- Komponen graf G terdiri dari:
  - V yaitu himpunan tidak kosong dari titik-titik (vertices)
    V = {a, b, …., vn}
  - E yaitu himpunan garis (edges) yang menghubungkan titik-titik
    E = {e_1, e_2, …, e_n} atau {(a,b), {a,c), (n,n)}

Istilah pada Graf

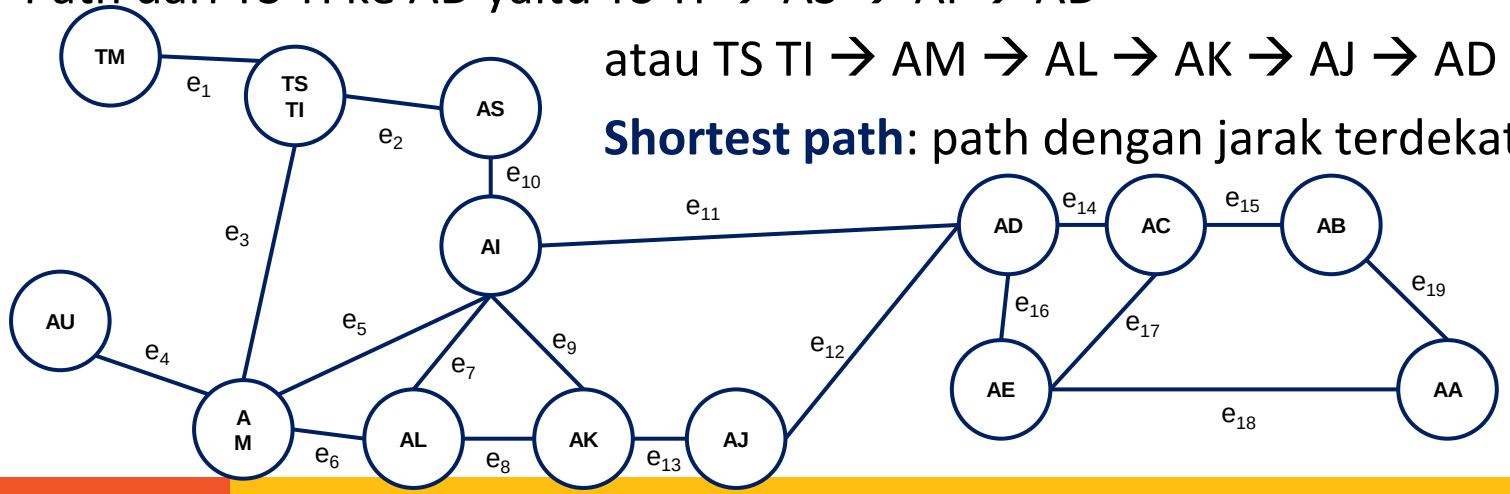
- ***Vertex* (Titik atau simpul)**
  Titik dalam *graph* disebut dengan *vertex*. Biasanya disimbolkan dengan bentuk lingkaran.
- ***Edge* (Garis atau sisi atau tepi)**
  Garis-garis penghubung antar titik dalam *graph* disebut dengan garis (*edge*)
- ***Adjacency* (Bertetangga)**
  Dua titik (*vertex*) dinamakan bertetangga (*adjacent*) jika saling terhubung melalui satu garis (*edge*).
- ***Path* (Lintasan)**
  Path atau intasan adalah representasi sebuah jalan dari satu titik ke titik lainnya.

Contoh Graf

- TS TI bertetangga dengan TM, AS, dan AM
- AI tidak bertetangga dengan TM, TS TI, AU, AJ, AC, AB, AE, dan AA
- Path dari TS TI ke AD yaitu TS TI → AS → AI → AD

atau TS TI → AM → AL → AK → AJ → AD

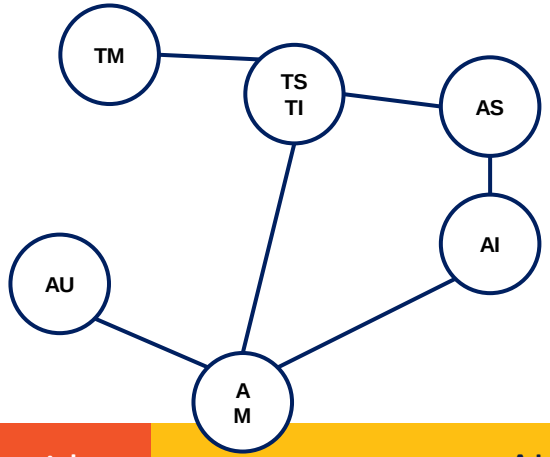
Shortest path: path dengan jarak terdekat



Istilah pada Graf

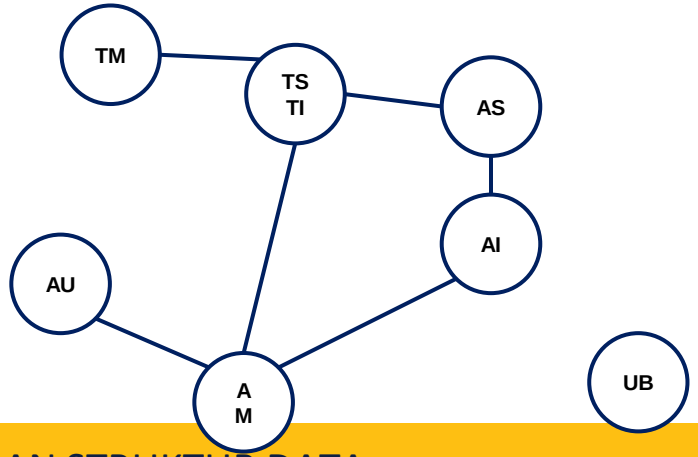
Connected Graph (Terhubung)

Ada setidaknya satu garis (*edge*) antara satu node (*vertex*) ke node lainnya



Unconnected Graph (Tidak Terhubung)

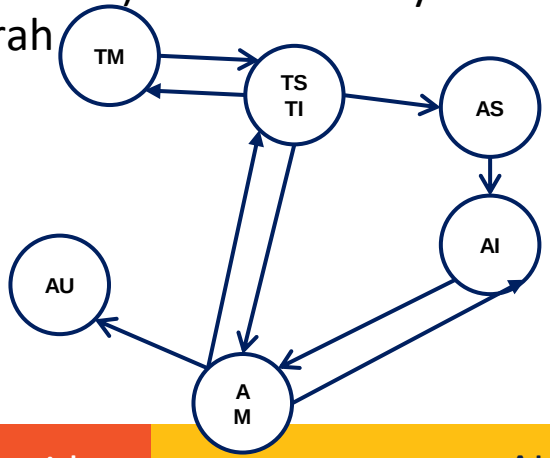
Satu atau lebih nodenya tidak terhubung ke node lainnya



Istilah pada Graf (2)

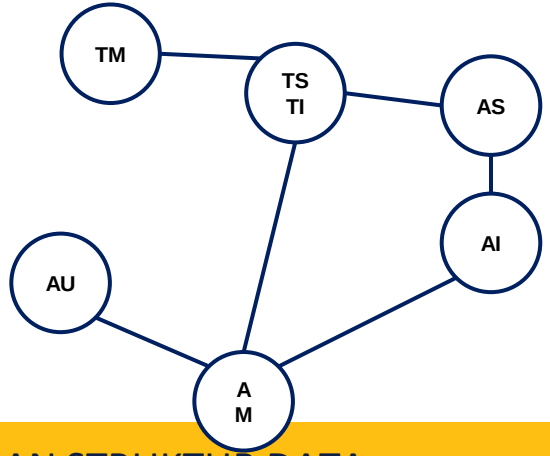
Directed Graph (Berarah)

Setiap sisinya (edge) memiliki arah. Sisi menunjukkan hubungan dari satu node (vertex) ke node lainnya secara satu arah



Undirected Graph (Tidak Berarah)

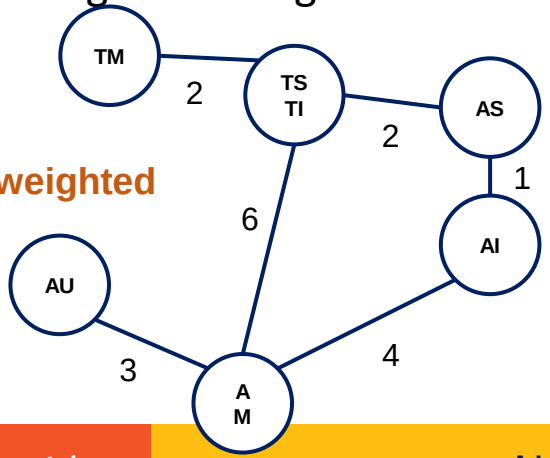
Setiap sisinya (edge) tidak memiliki arah



Istilah pada Graf (3)

Weighted Graph (Berbobot)

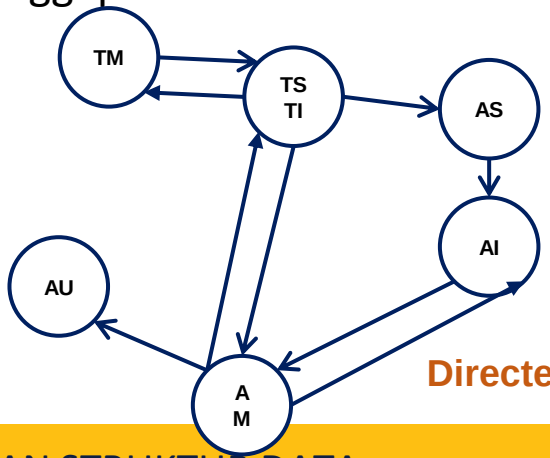
Setiap sisi (edge) memiliki bobot yang menunjukkan biaya atau jarak yang terkait dengan hubungan antara dua node



Undirected weighted

Unweighted Graph (Tidak Berbobot)

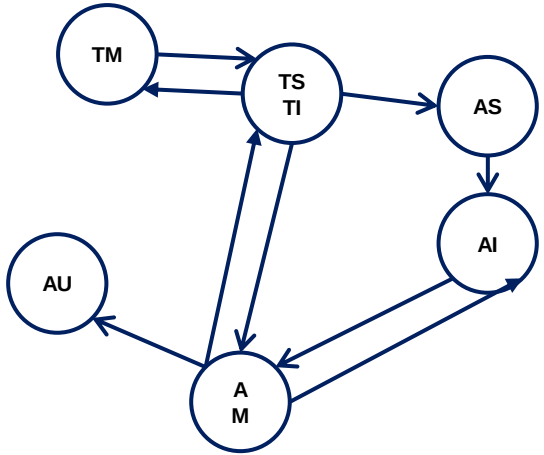
Setiap sisi (edge) tidak memiliki bobot. Semua hubungan antara node (vertex) dianggap setara



Directed unweighted

Istilah pada Graf (3)

- **Degree** (derajat) sebuah node adalah jumlah sisi yang bersebelahan dengan node tersebut atau jumlah garis yang keluar dari node
- **In-degree** sebuah node pada graph berarah adalah jumlah sisi yang “masuk” atau menuju node tersebut
- **Out-degree** sebuah node pada graph berarah adalah jumlah sisi yang “keluar” atau berasal dari node tersebut



D_{in} (TS TI) = 2

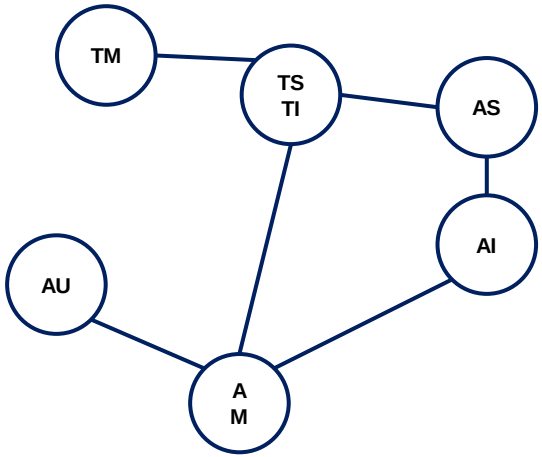
D_{out}(TS TI) = 3

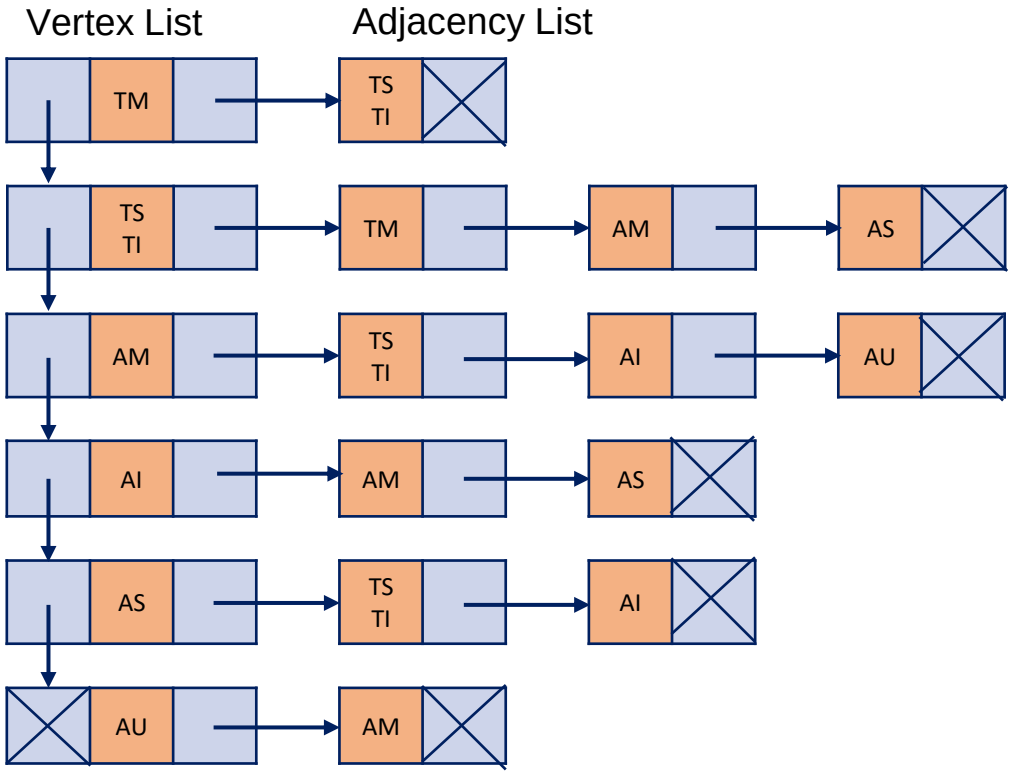
Jenis Representasi Graf

- **Adjacency List**
  Menggunakan suatu array pada linked list. Array tersebut digunakan untuk menyimpan jumlah node. Nilai pada linked list dapat digunakan untuk menyimpan bobot graf
- **Adjacency Matrix**
  Merupakan array 2D dengan size V x V dimana V adalah jumlah node pada graph. Jika adj[i][j] = 1 dapat diartikan terdapat suatu garis (edge) pada titik i ke titik j

Contoh Adjacency List

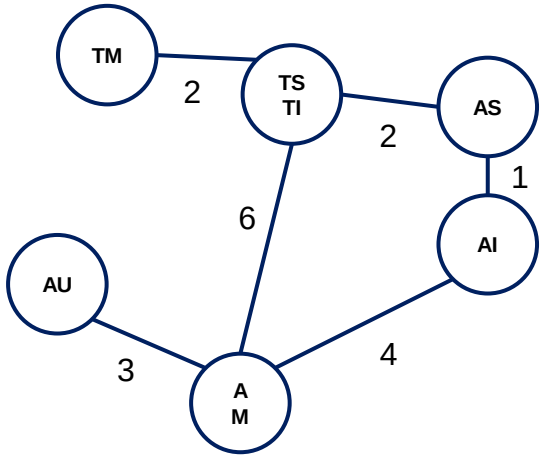
Undirected Graph





Contoh Adjacency Matrix

Undirected Graph



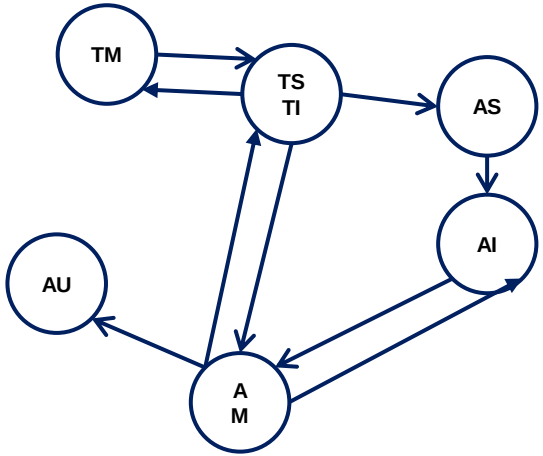
Vertex Vector
TM
TS TI
AS
AI
AM
AU

Adjacency Matrix

	TM	TS TI	AS	AI	AM	AU
TM	0	2	0	0	0	0
TS TI	2	0	2	0	6	0
AS	0	2	0	1	0	0
AI	0	0	1	0	4	0
AM	0	6	0	4	0	3
AU	0	0	0	0	3	0

Contoh Adjacency Matrix

Directed Graph



TM
TS TI
AS
AI
AM
AU

Vertex Vector

	TM	TS TI	AS	AI	AM	AU
TM	0	1	0	0	0	0
TS TI	1	0	1	0	1	0
AS	0	1	0	0	0	0
AI	0	0	0	0	1	0
AM	0	1	0	1	0	1
AU	0	0	0	0	0	0

Adjacency Matrix

Operasi Dasar pada Graf

- Menyisipkan node: memasukkan node ke dalam graf
- Menghapus vertex : menghapus sebuah node dari graf
- Membuat lintasan: menghubungkan dua node menggunakan edge
- Mencari entitas: mencari node atau edge pada graf
- Mencari lintasan: melintasi jalur di antara dua node
- Traversal: melintasi semua node dalam graf

Implementasi Graf

- Menggunakan **Linked List**

  Class Graph mempunyai atribut:

  int vertex

  LinkedList list[]

- Menggunakan **Matrix**

  Class Graph mempunyai atribut:

  int vertex

  int[][] array

Latihan 1

Ubah matrix berikut ke dalam bentuk graf!

a.

	V1	V2	V3	V4	V5	V6
V1	0	1	0	0	0	0
V2	1	1	1	0	0	0
V3	0	1	0	1	1	1
V4	0	0	1	0	0	0
V5	0	0	1	0	0	0
V6	0	0	1	0	0	0

b.

	N1	N2	N3	N4	N5	N6
N1	0	2	0	6	0	0
N2	0	0	3	0	0	0
N3	0	0	0	0	1	0
N4	0	0	0	0	4	2
N5	0	0	0	0	0	7
N6	5	0	0	0	0	0

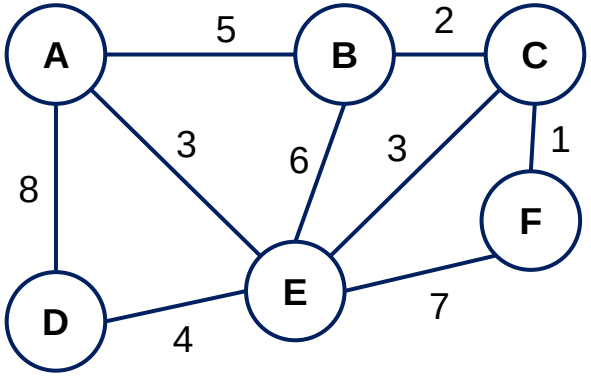
Latihan 2

Ubah matrix berikut ke dalam bentuk graf!

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
V1	1	1	0	1	1	0	0	0
V2	1	0	1	0	0	0	0	0
V3	0	1	1	0	0	1	1	0
V4	0	0	0	1	0	1	0	1
V5	0	0	0	0	0	0	0	1

Latihan 3

Perhatikan graf berikut!



a. Ubahlah graf tersebut ke dalam bentuk adjacency matrix!
b. Tentukan shortest path dari A ke F!
c. Tentukan lintasan traversal untuk menghubungkan semua node dengan jarak terpendek!