STRUKDAT DATA
GRAPH ADJACENCY MATRIKS
C++

Konsep Graph

Konsep struktur data yang terdiri dari node (vertex atau vertices) dan garis penghubung (arc atau edge). Vertex disimbolkan dengan "V" dan edge disimbolkan dengan "E". keterhubungan graph dapat membentuk relasi One-to-One, One-to-Many, Many-to-One, dan Many-to-Many. Contoh : Informasi topologi jaringan, keterhubungan antar kota-kota, dll.

Contoh Penggunaan Graph

1. Graph banyak digunakan untuk :
    - Menggambarkan jaringan dan peta jalan
    - Lintasan pesawat
    - System perpipaan
    - Saluran telepon
    - Koneksi elektrik
    - Ketergantungan diantara task pada sistem manufaktur
    - Terdapat banyak hasil dan struktur penting yang didapatkan dari perhitungan graph

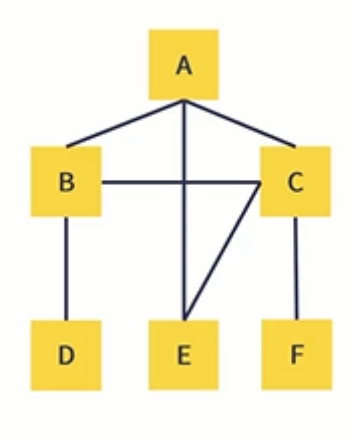
Masalah-masalah Graph

1. Shortest Path Problem (Masalah Patch Minimum)
   mencari rute dengan jarak terpendek dalam suatu jaringan transportasi.
2. Maximum Flow Problem (Masalah Aliran Maksimum)
   menghitung volume aliran BBM dari suatu reservoir ke suatu titik tujuan melalui jaringan pipa.
3. Graph Searching Problem (Masalah Pencarian dalam Graph)
   Mencari Langkah-langkah terbaik dalam program permainan catur komputer
4. Topological Ordering Problem (Masalah Pengurutan Topologi)
   Menentukan urutan pengambilan mata-mata kuliah yang saling berkaitan dalam hubungan prasyarat.
5. Task Network Problem (Masalah Jaringan Tugas)
   Penjadwalan pengerjaan suatu proyek yang memungkinkan waktu penyelesaian tersingkat.
6. Minimum Spanning Tree Problem (Masalah Pencarian Pohon Rentang Minimum)
   Mencari rentangan kabel listrik yang totalnya adalah minimal untuk menghubungkan sejumlah kota.
7. Travelling Salesperson Problem
   Tukang pos mencari lintasan terpendek melalui semua alamat penerima pos tanpa harus mendatangi suatu tempat lebih dari satu kali.
8. Four-Color Problem
   Dalam menggambar peta, memberikan warna yang berbeda pada setiap provinsi yang

saling bersebelahan.

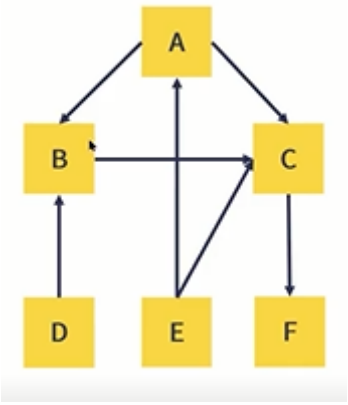
Kategori Graph

- **Undirected Graph**
  - Edge tidak ada penugasan arah dari satu arah satu vertex ke vertex yang lain.
  - Digunakan untuk merepresentasikan relasi one-to-one
  - Jika A dan B adalah vertex, maka edge dapat merepresentasikan sebagai (A,B) dan (B,A).



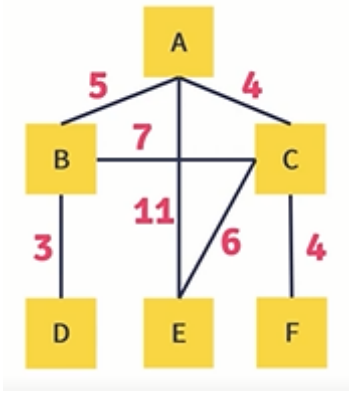
Contoh pengerjaan Undirected Graph :
Vertex = A, B, C, D, E, F
Edges = (A,B), (A,C), (A,E), (B,C), (B,D),(C,A),(C,B),(C,E),(C,F),(D,B)(E,A),(E,C),(F,C)

- **Directed Graph**
  - Edge ada penugasan arah dari satu vertex ke vertex yang lain
  - Edges antar vertexs diwakilkan dengan kurung sudut
  - Jika vertex A sebagai sumber dan menuju ke vertex B sebagai tujuan, maka edge dapat merepresentasikan sebagai <A,B>



Contoh pengerjaan Directed Graph :
Vertex = A, B, C, D, E, F
Edges = <A,B>, <A,C>,<B,C>,<C,F>,<D,B>,<E,A>,<E,C>

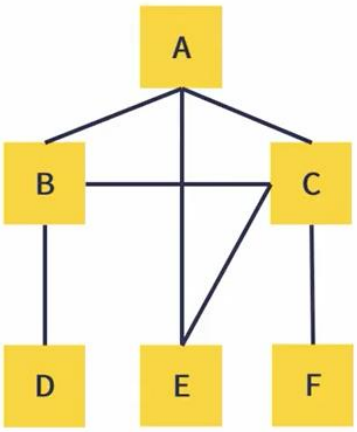
- **Weighted Graph**
  - Edge diberikan suatu nilai sebagai weight dari suatu vertex ke vertex yang lain
  - Dapat direpresentasikan oleh Undirected atau Directed Graph
  - Weight dapat direpresentasikan sebagai jarak, lama waktu, dan lain sebagainya



Contoh pengerjaan Undirected Weight Graph :
Vertex = A, B, C, D, E, F
Edges = (A,B) = 5, (A,C) = 4, (A,E) = 11, (B,A) = 5, (B,C) = 7, (B,D) = 3, (C,A) = 4, (C,B) = 7, (C,E) = 6, (C,F) = 4, (D,B) = 3, (E,A) = 11, (E,C) = 6, (F,C) = 4

Adjacency Matriks

- Representasinya dengan array 2 dimensi atau matriks
- Pada undirect dan direct graph 0 berarti tidak ada relasi edge, sedangkan nilai 1 ada relasi edge.
- Pada weighted graph nilainya diisi nilai beban edgenya.

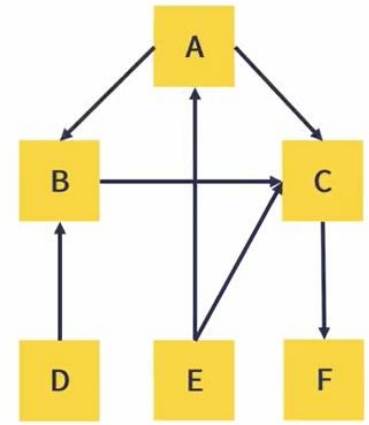


Tujuan

Sumber	A	B	C	D	E	F
A	0	1	1	0	1	0
B	1	0	1	1	0	0
C	1	1	0	0	1	1
D	0	1	0	0	0	0
E	1	0	1	0	0	0
F	0	0	1	0	0	0

Undirect Graph

Contoh Direct Graph

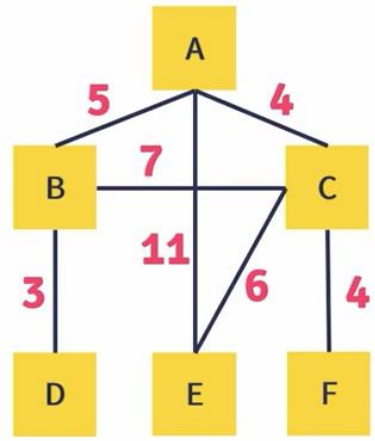


Tujuan

Sumber	A	B	C	D	E	F
A	0	1	1	0	0	0
B	0	0	1	0	0	0
C	0	0	0	0	0	1
D	0	1	0	0	0	0
E	1	0	1	0	0	0
F	0	0	0	0	0	0

Direct Graph

Contoh Weight Graph

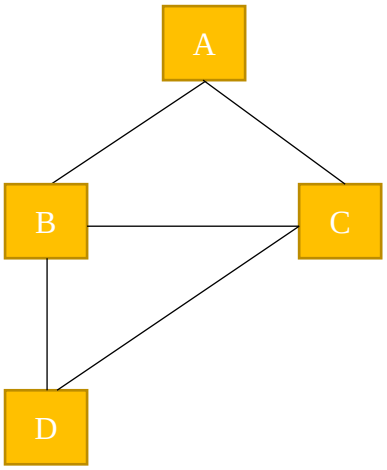


Tujuan

Sumber	A	B	C	D	E	F
A	0	5	4	0	11	0
B	5	0	7	3	0	0
C	4	7	0	0	6	4
D	0	3	0	0	0	0
E	11	0	6	0	0	0
F	0	0	4	0	0	0

Undirect Weighted Graph

Contoh Program Adjacency Matriks



0	**1**	**1**	**0**
1	**0**	**1**	**1**
1	**0**	**0**	**1**
0	**1**	**1**	**0**

Permisalan :
0 = A
1 = B
2 = C
3 = D

```
#include <iostream>
#include <vector>

using namespace std;

void printAdjacencyMatrix(vector<vector<int>>& adjMatrix) {
    int n = adjMatrix.size();
    for (int i = 0; i < n; i++) {
        cout << '[';
        for (int j = 0; j < n; j++) {
            cout << adjMatrix[i][j] << " ";
        }
        cout << ']';
        cout << endl;
    }
}

int main() {
    int n = 4; // Jumlah simpul
    vector<vector<int>> adjMatrix(n, vector<int>(n, 0));

    // Mengatur sisi-sisi pada adjacency matrix
    adjMatrix[0][1] = 1;
    adjMatrix[0][2] = 1;
    adjMatrix[1][0] = 1;
    adjMatrix[1][3] = 1;
    adjMatrix[1][2] = 1;
    adjMatrix[2][0] = 1;
    adjMatrix[2][3] = 1;
    adjMatrix[3][1] = 1;
    adjMatrix[3][2] = 1;
    adjMatrix[3][2] = 1;


    cout << "Adjacency matrix:" << endl;
    printAdjacencyMatrix(adjMatrix);

    return 0;
}
```

Tugas Praktikum

Buatlah program dari gambar graph di bawah ini!