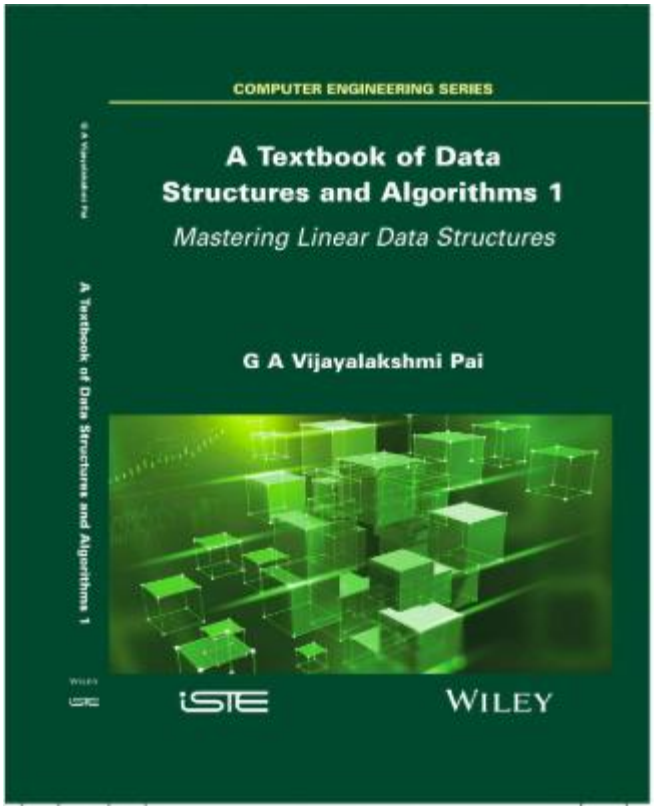
TIB29 – Struktur Data dan Algoritma

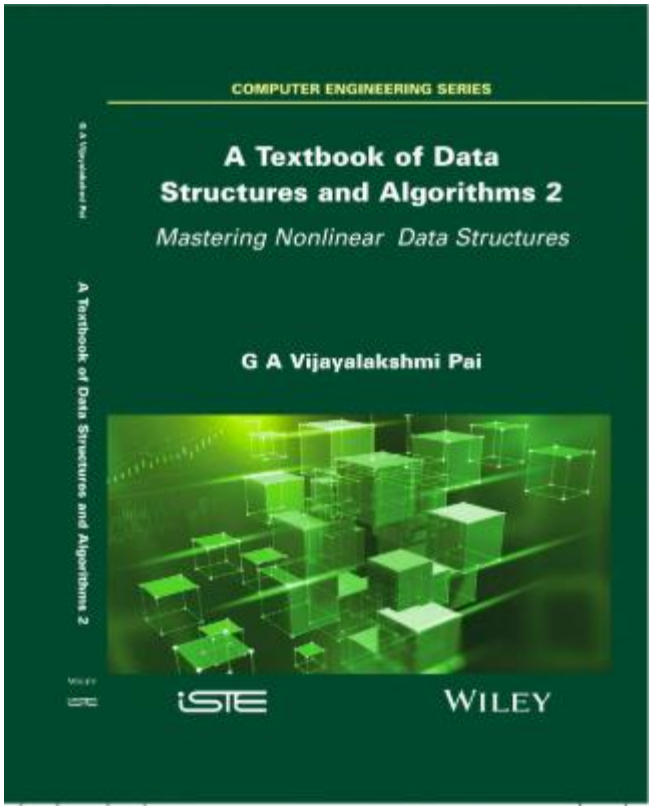
PERINGATAN HAK CIPTA

PERINGATAN HAK CIPTA

Implementasi Struktur Data Pada Graph

Diadopsi Dari Sumber:





Sub-CPMK

- Mahasiswa mampu menerapkan representasi graph dalam bentuk linked list dan array, serta melakukan penelusuran graph. (C3, A3)

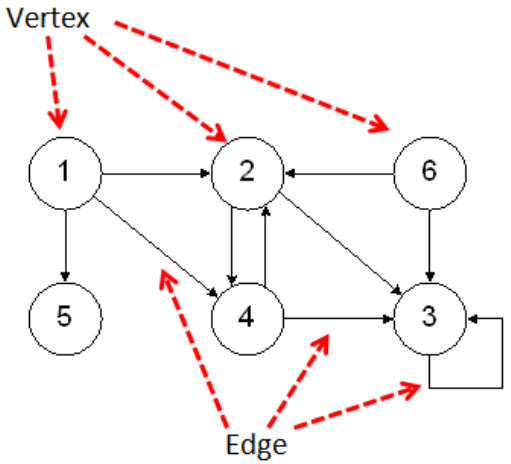
Materi

1. Pengertian Graph
2. Adjacency Matrix
3. Adjacency List
4. Penelusuran Graph

1. Pengertian Graph

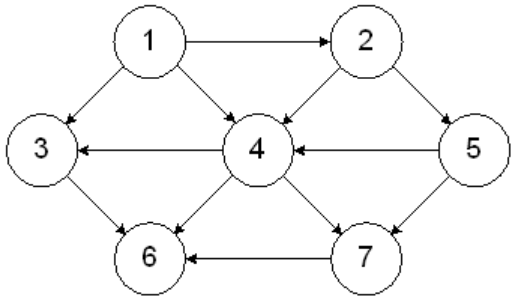
1.1 Pengertian Graph

- Graph terdiri dari satu set verteks dan satu set edge.
- Verteks adalah simpul yang membuat graph
- Edge adalah garis yang menghubungkan verteks-verteks

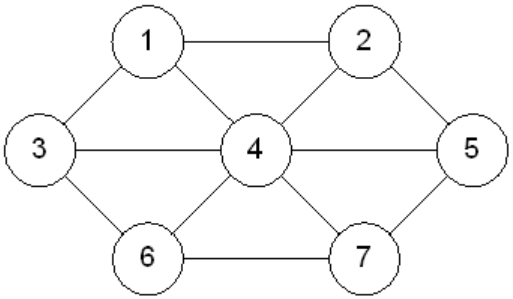


1.2 Dua Macam Graph

- Graph Berarah (Directed Graph/digraph)
- Graph tak berarah (Undirected Graph)



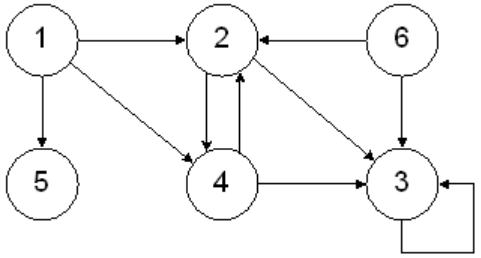
Graph Berarah



Graph Tak Berarah

1.3 Graph Berarah

- Sebuah graph berarah atau **directed graph** atau **digraph**, adalah graph yang setiap verteks nya terhubung dengan arah yang spesifik



- Contoh disamping adalah sebuah digraph dengan enam verteks yang terdiri dari verteks {1, 2, 3, 4, 5, 6}
- Dari enam verteks tersebut memiliki sepuluh edges berarah yang terdiri dari {1 ke 2, 1 ke 4, 1 ke 5, 2 ke 3, 2 ke 4, 3 dirinya sendiri, 4 ke 2, 4 ke 3, 6 ke 2, 6 ke 3}

1.3.1 Tree Pada Digraph

- Tree adalah kasus khusus dari sebuah digraph berdasarkan karakteristik:
- salah satu dari verteks yaitu root tidak memiliki edge yang masuk
- setiap verteks selain root dapat dicapai dengan kriteria hanya boleh dilalui satu kali secara berurutan

1.3.2 Perbedaan Digraph Dengan Tree

- Tidak perlu memiliki root node
- Mungkin ada beberapa jalur (atau tidak) dari satu titik ke titik yang lain
- Diproses secara berbeda.

  Penyisipan misalnya, untuk menambahkan simpul ke pohon, bidang tautan harus ditambahkan ke simpul induk. Pada digraph simpul dapat dimasukkan di mana saja, namun tidak perlu ada busur dari atau ke sana; atau edge bisa disisipkan di antara dua node yang ada.

1.4 Beberapa Definisi Yang Diasosiasikan Dengan Graph

- Dua verteks dikatakan berdampingan (adjacent) jika ada edge yang menghubungkan
- Lintasan dari sebuah graph adalah urutan dari verteks dan edge
- Panjang dari sebuah lintasan adalah jumlah edge pada lintasan, setara dengan jumlah verteks - 1
- Loop adalah lintasan yang dibuat dari edge yang mengarah ke verteks itu sendiri

1.4 Beberapa Definisi Yang Diasosiasikan Dengan Graph (Lanj.)

- Lintasan sederhana adalah lintasan dimana verteks yang dilaluinya berbeda kecuali yang pertama dan terakhir
- Siklus adalah jalur yang paling sedikit panjang 1 dimana simpul pertama dan terakhir sama dengan tepi yang berbeda untuk grafik yang tidak berarah.
- Sebuah digraph adalah sebuah non siklus graph
- Sebuah digraph dikatakan terhubung secara kuat (strongly connected) jika ada sebuah path dari setiap verteks ke setiap verteks lainnya
- Complete Graph adalah graph yang memiliki sebuah edge diantara setiap pasangan pasangan verteks

1.5 Representasi Graph Pada Struktur Data

Dapat direpresentasikan dengan

- Adjacency Matrix (ada yang menyebutnya adjacency Table)
  - Matriks biasanya diterapkan menggunakan Array 2 Dimensi
  - Ruang memory yang diperlukan adalah V^2, dimana V adalah jumlah verteks
- Adjacency List
  - Diterapkan dengan linked list atau paduan Array 1 dimensi dengan linked list
  - Ruang memory yang dibutuhkan adalah E+V, E adalah jumlah edge dan V jumlah verteks.

2. Adjacency Matrix

2.1 Adjacency Matrix

Cara membentuk Adjancency matrix

- Jumlah verteks dari digraph 1, 2, ..., n
- Bentuk sebuah matriks n x n
- Untuk setiap entry baris i dan kolom j, sisipkan sebuah angka 1 jika ada edge dari verteks i ke verteks j; jika tidak sisipkan 0

  Catatan:

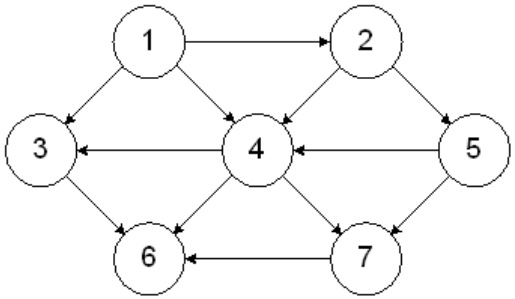
  – *untuk edge dengan nilai masukkan nilai edge*

  – *Pada beberapa penerapan, verteks yang tidak terhubung sering kali diisi dengan nilai ~ (tak berhingga) atau nilai sangat tinggi yang melebihi range nilai yang ada*

2.2 Adjacency Matriks Pada Directed Graph

- Hanya diisi nilai yang memiliki edge dari verteks i ke verteks j
- Antara baris dengan kolom tidak simetris, karena adanya arah pada edge nya

2.3 Contoh Adjacency Matriks Untuk Directed Graph

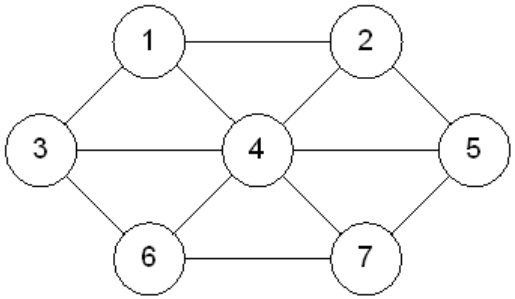


	[1]	[2]	[3]	[4]	[5]	[6]	[7]
[1]	0	1	1	1	0	0	0
[2]	0	0	0	1	1	0	0
[3]	0	0	0	0	0	1	0
[4]	0	0	1	0	0	1	1
[5]	0	0	0	1	0	0	1
[6]	0	0	0	0	0	0	0
[7]	0	0	0	0	0	1	0

2.4 Adjacency Matriks Pada Graph Tak Berarah

- Antara baris dengan kolom simetris, karena tidak adanya arah pada edge nya
- Setiap verteks i dan verteks j yang terhubung akan mempunyai nilai

2.5 Contoh Adjacency Matriks Untuk Undirected Graph



	[1]	[2]	[3]	[4]	[5]	[6]	[7]
[1]	0	1	1	1	0	0	0
[2]	1	0	0	1	1	0	0
[3]	1	0	0	1	0	1	0
[4]	1	1	1	0	1	1	1
[5]	0	1	0	1	0	0	1
[6]	0	0	1	1	0	0	1
[7]	0	0	0	1	1	1	0

2.6 Kerugian dan Keuntungan Adjacency Matriks

Kerugian

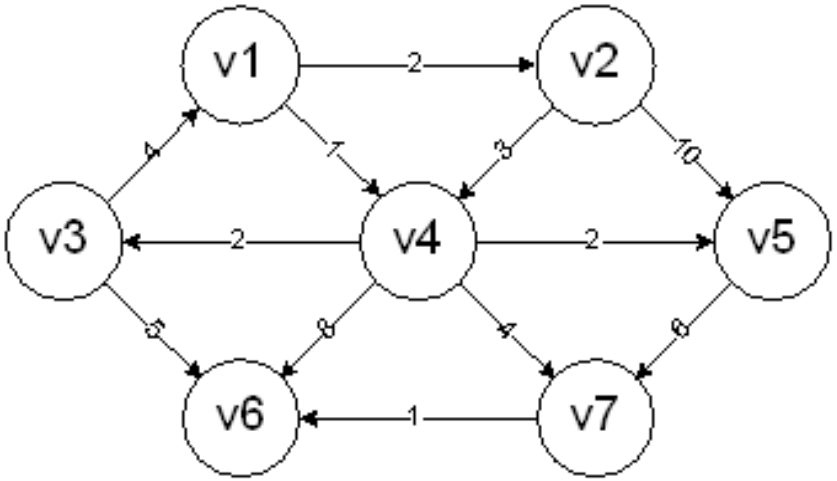
- Terpakai atau atau tidak terpakai, ruang memory harus dialokasikan. Sehingga akan banyak yang terisi dengan 0 atau mirroring dari diagonalnya pada Undirected Graph

Keuntungan

- Lebih mudah mengaksesnya, ukup menggunakan nested loop per baris untuk tiap kolomya

2.7 Graph Dengan Harga Pada Edge

- Edge yang menghubungkan verteks-verteks dapat mempunyai nilai
- Representasi pada Adjacency Matriks, sel verteks i ke verteks j dapat diisi dengan nilai pada edge tersebut



3. Adjacency List

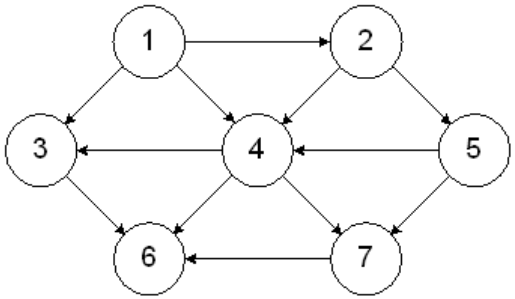
3.1 Adjacency List Dengan Linked List

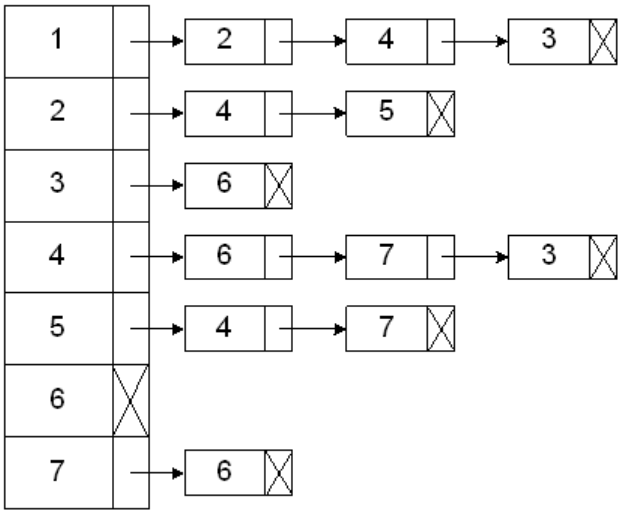
- List Vertex direpresentasikan sebagai node/simpul secara berurutan
- Pada tiap simpul list verteks terdapat dua link, link ke node urutan berikutnya dan link ke salah satu verteks tujuan
- Verteks tujuan digambarkan sebagai node/simpul dengan satu link ke node/simpul tujuan lain dan disusun secara berurutan

3.2 Adjacency List Dengan Paduan Array dan Linked List

- List Vertex direpresentasikan dengan Array satu dimensi secara berurutan
- Pada tiap simpul array terdapat sebuah link ke salah satu verteks tujuan
- Verteks tujuan digambarkan sebagai node/simpul dengan satu link ke node/simpul tujuan lain dan disusun secara berurutan

3.3 Contoh Adjacency List





4. Penelusuran Graph

4.1 Penelusuran Graph

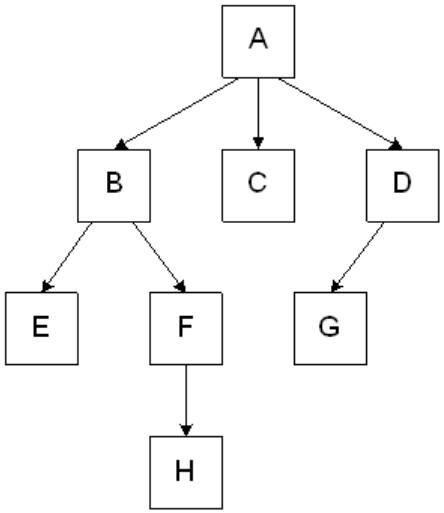
Dua macam penelusuran

- Depth First Search
- Breadth First Search

4.2 Depth-first Search

Pencarian graph terarah yang dimulai dari simpul yang dipilih dan diarahkan oleh busur sebagai "terdalam" sebanyak mungkin untuk mengunjungi simpul yang dapat dijangkau simpul yang belum pernah dikunjungi. Setelah akhir rantai tercapai, lakukan backtrack / langkah mundur ke simpul sebelumnya dan lanjutkan pencarian.

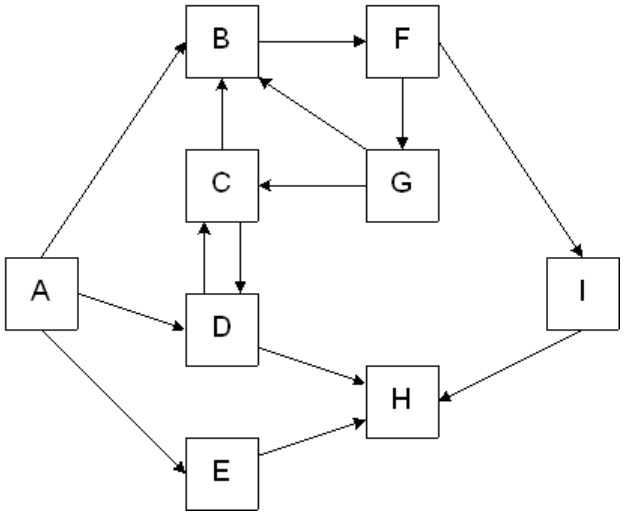
4.3 Contoh Pada Non Siklus Graph



- Hasil Penelusuran:

  A, B, E, F, H, C, D, G

4.4 Contoh Pada Graph Dengan Siklus



Hasil Penelusuran

- Jika dimulai dari A

  A, B, F, I, H, G, C, D, E

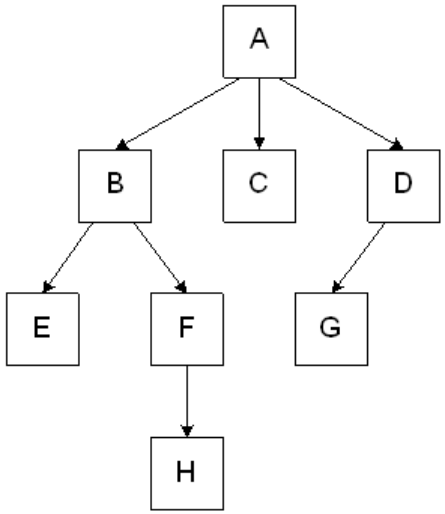
- Jika dimulai dari B

  B, F, I, H, G, C, D

4.5 Breadth-first Search

- Pencarian graph berarah secara “melebar”, yang dimulai dari simpul yang dipilih dan kemudian semua simpul yang berdekatan dikunjungi. Setelah selesai, simpul pertama yang berdekatan menjadi titik awal yang baru, dan semua simpulnya yang berdekatan dikunjungi selama belum pernah dikunjungi sebelumnya.

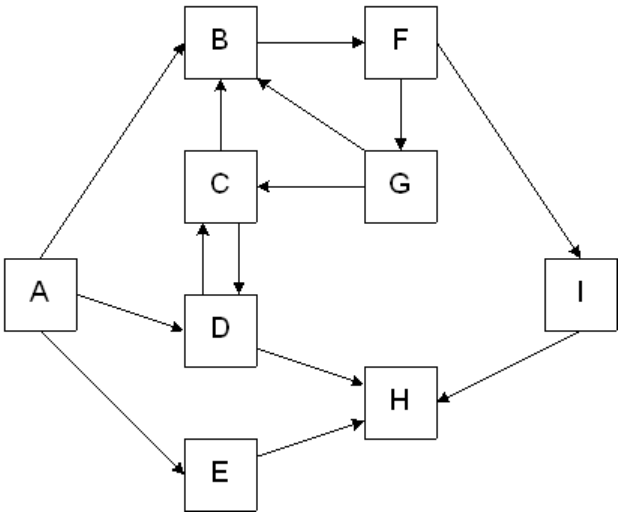
4.6 Contoh Pada Non Siklus Graph



Hasil:

- A, B, C, D, E, F, G, H

4.7 Contoh Pada Siklus Graph



Hasil:

- Jika dimulai dari A

  A, B, D, E, F, C, H, G, I

- Jika dimulai dari B

  B, F, G, I, H, C, D

4.8 Pencarian Lintasan Terpendek

- Kasus yang sering muncul pada Graph adalah kasus pencarian lintasan terpendek berdasarkan harga pada edge-edge nya
- Salah satu algoritma pencarian lintasan terpendek adalah algoritma Djikstra, berikut ini algoritma pencarian lintasan terpendek yang cukup terkenal, semua algoritma ini sudah dibahas pada mata kuliah matematika diskrit dan teori graph
  1. Unweighted shortest path
  2. Dijkstra's algorithm
  3. Graphs with negative edge costs
  4. Acyclic graphs

Contoh Djikstra dengan C++

```cpp
#include <iostream.h>
#include <cmath.h>
int indeks;
int lintasan[10]={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};
char v[5] =       {'A','B','C','D','E'};
int dat[5][5] = {{999, 1,  3,999, 19},
                 {999,999, 12,  2,  7},
                 {999,999,999,  6,  8},
                 {999,999,999,999,  1},
                 {999,999,999,999,999}};
```

```cpp
void djikstra(int asal, int tujuan)
{
  if (asal != tujuan)
  {
    indeks++;
    lintasan[indeks] = asal;
    int terkecil = asal+1;
    for(int i = asal+1;i<=tujuan;i++)
      if (dat[asal][i] <= dat[asal][terkecil])
        terkecil = i;
    lintasan[indeks] = terkecil;
    djikstra(terkecil,tujuan);
  }
}

int main()
{
  indeks = 0;
  lintasan[indeks]=0;
  djikstra(0,4);
  for(int i=0;i<=indeks;i++)
    cout<<v[lintasan[i]]<<" ";
  return 0;
}
```

Contoh Djikstra dengan Python

```
matrik = []
jumpath=0
total = 0
vertek = 0
def djikstra(asal, tujuan):
    print(asal)
    if (asal != tujuan): #Recursive Rule
      terkecil = 0
      for i in range(0,vertek):
        if asal != i:
          if matrik[asal][i]<matrik[asal][terkecil]:
            terkecil = i
      return djikstra(terkecil, tujuan) + matrik[asal][terkecil]
    else:
      print("Lintasan berhasil")
      return 0
```

Contoh Djikstra dengan Python (Lanj.)

```
18  ##################
19  #Input Lintasan##
20  ##################
21  vertek = int(input("jumlah Vertek: "))
22  for j in range(0,vertek):
23    baris = []
24    for i in range(0,vertek):
25      baris.append(999)
26    matrik.append(baris)
27
28  jumpath = int(input("jumlah Lintasan: "))
29  for i in range(0,jumpath):
30    dari = vertek
31    while dari >= vertek:
32      dari = int(input("Dari: "))
33      if dari>=vertek:
34        print("ULANGI!!!")
35    ke = vertek
36    while ke >= vertek:
37      ke = int(input("menuju ke: "))
38      if ke>=vertek:
39        print("ULANGI!!!")
40    jarak = int(input("Jarak: "))
41    matrik[dari][ke] = jarak
42  for i in range(0,vertek):
43    print(matrik[i])
```

```
44
45  ######################
46  #Pencarian Lintasan
47  ######################
48  asal = 0
49  while (asal>=0) & (asal<vertek):
50    asal = int(input("Asal: "))
51    if (asal>=0) & (asal<vertek):
52      tujuan = int(input("Tujuan: "))
53      print("Jarak ditempuh: ",djikstra(asal,tujuan))
54
```

Ringkasan

- Graph terdiri dari satu set verteks dan satu set edge.
- Verteks adalah simpul yang membuat graph
- Edge adalah garis yang menghubungkan verteks-verteks
- Graph berarah atau directed graph atau digraph, adalah graph yang setiap verteks nya terhubung dengan arah yang spesifik
- Graph Pada Struktur Data terdiri dari **Adjacency Matrix** (ada yang menyebutnya adjacency Table) biasanya diterapkan menggunakan Array 2 Dimensi dan **Adjacency List** Diterapkan dengan linked list atau paduan Array 1 dimensi dengan linked list

PERINGATAN HAK CIPTA

PERINGATAN HAK CIPTA

Terimakasih

TUHAN Memberkati Anda

Teady Matius Surya Mulyana (tmulyana@bundamulia.ac.id)