STRUKTUR DATA

Struktur Data Graf

@Informatika Undip

GRAPH

- Graph adalah kumpulan dari simpul dan busur yang secara matematis dinyatakan sebagai :

  G = (V, E)

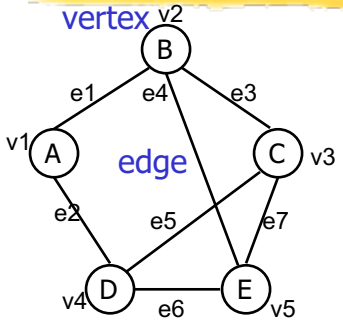
  Dimana

  G = Graph

  V = Simpul atau Vertex, atau Node, atau Titik

  E = Busur atau Edge, atau arc

Contoh graph :

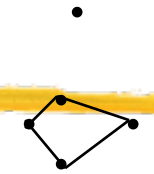


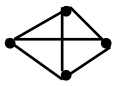
V terdiri dari v1, v2, …, v5

E terdiri dari e1, e2, … , e7

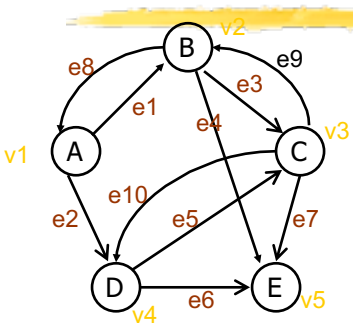
Undirected graph

- Sebuah graph mungkin hanya terdiri dari satu simpul
- Sebuah graph belum tentu semua simpulnya terhubung dengan busur
- Sebuah graph mungkin mempunyai simpul yang tak terhubung dengan simpul yang lain
- Sebuah graph mungkin semua simpulnya saling berhubungan

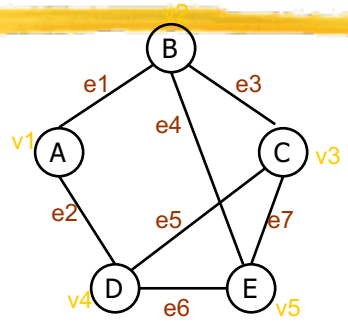




Graph Berarah dan Graph Tak Berarah :



Directed graph



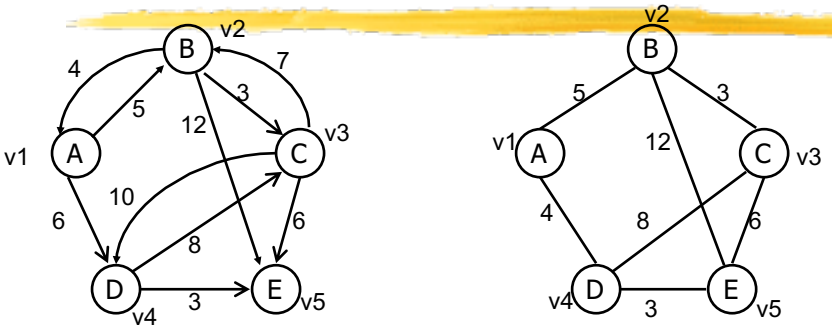
Undirected graph

Dapat dilihat dari bentuk busur yang artinya urutan penyebutan pasangan 2 simpul.

- Graph tak berarah (undirected graph atau non-directed graph) :
  - Urutan simpul dalam sebuah busur tidak dipentingkan. Mis busur e1 dapat disebut busur AB atau BA
- Graph berarah (directed graph) :
  - Urutan simpul mempunyai arti. Mis busur AB adalah e1 sedangkan busur BA adalah e8.

- Graph Berbobot (Weighted Graph)
  - Jika setiap busur mempunyai nilai yang menyatakan hubungan antara 2 buah simpul, maka busur tersebut dinyatakan memiliki bobot.
  - Bobot sebuah busur dapat menyatakan panjang sebuah jalan dari 2 buah titik, jumlah rata-rata kendaraan perhari yang melalui sebuah jalan, dll.

Graph Berbobot :



Directed graph Undirected graph

Panjang busur (atau bobot) mungkin tidak digambarkan secara panjang yang proposional dengan bobotnya. Misal bobot 5 digambarkan lebih panjang dari 7.

Istilah pada graph

Incident

Jika e merupakan busur dengan simpul-simpulnya adalah v dan w yang ditulis e=(v,w), maka v dan w disebut "terletak" pada e, dan e disebut incident dengan v dan w.

Degree (derajat), indegree dan outdegree

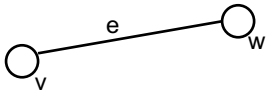
Degree sebuah simpul adalah jumlah busur yang incident dengan simpul tersebut.

Indegree sebuah simpul pada graph berarah adalah jumlah busur yang kepalanya incident dengan simpul tersebut, atau jumlah busur yang "masuk" atau menuju simpul tersebut.

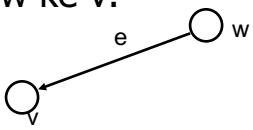
Outdegree sebuah simpul pada graph berarah adalah jumlah busur yang ekornya incident dengan simpul tersebut, atau jumlah busur yang "keluar" atau berasal dari simpul tersebut.

3. Adjacent

Pada graph tidah berarah, 2 buah simpul disebut adjacent bila ada busur yang menghubungkan kedua simpul tersebut. Simpul v dan w disebut adjacent.



Pada graph berarah, simpul v disebut adjacent dengan simpul w bila ada busur dari w ke v.

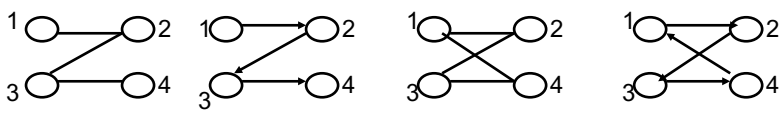


4. Successor dan Predecessor

Pada graph berarah, bila simpul v adjacent dengan simpul w, maka simpul v adalah successor simpul w, dan simpul w adalah predecessor dari simpul v.

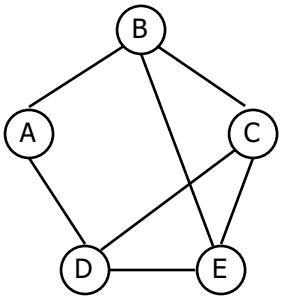
5. Path

Sebuah path adalah serangkaian simpul-simpul yang berbeda, yang adjacent secara berturut-turut dari simpul satu ke simpul berikutnya.



Representasi Graph dalam bentuk matrix

- Adjacency Matrix Graph tak berarah



Graph

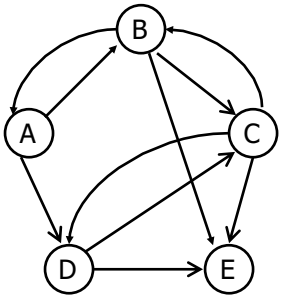
Urut abjad →

	A 0	B 1	C 2	D 3	E 4
A 0	0	1	0	1	0
B 1	1	0	1	0	1
C 2	0	1	0	1	1
D 3	1	0	1	0	1
E 4	0	1	1	1	0

Degree simpul : 3

Representasi Graph dalam bentuk matrix

- Adjacency Matrix Graph berarah



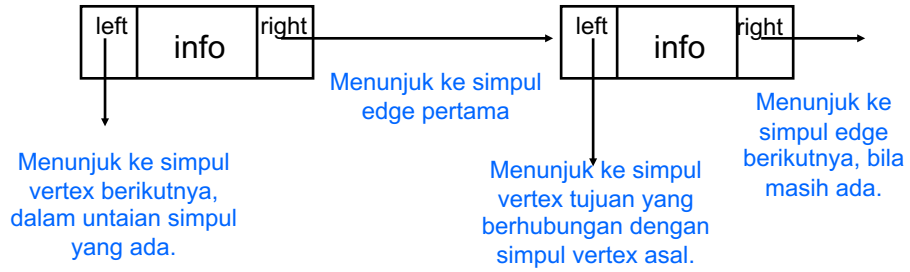
Graph

ke →
dari ↓

	A 0	B 1	C 2	D 3	E 4	
A 0	0	1	0	1	0	
B 1	1	0	1	0	1	out →
C 2	0	1	0	1	1	
D 3	0	0	1	0	1	
E 4	0	0	0	0	0	
						in ↓

Representasi Graph dalam bentuk Linked List

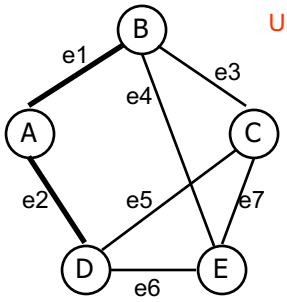
- Adjency List graph tak berarah
- Digambarkan sebagai sebuah simpul yang memiliki 2 pointer.
- Simpul vertex : Simpul edge :



- Define struct untuk sebuah simpul yang dapat digunakan sebagai vertex maupun edge.

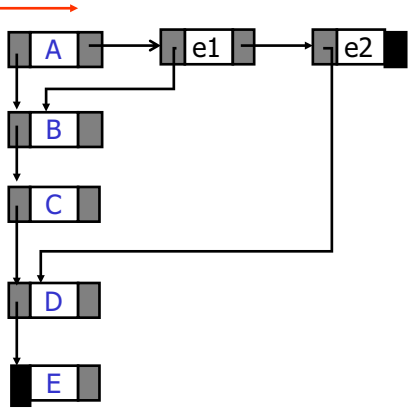
```
typedef struct tipeS {
        tipeS *Left;
        int INFO;
        tipeS *Right;
        };
tipeS *FIRST, *PVertex, *PEdge;
```

Contoh : untuk vertex A, memiliki 2 edge yang terhubung yaitu e1 dan e2.

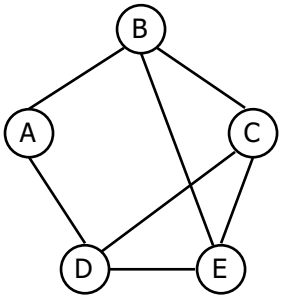


Graph

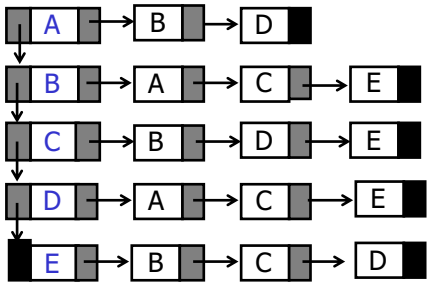
Urut abjad



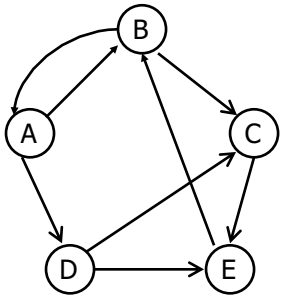
Gambar di atas dapat disusun dengan lebih sederhana, sbb :

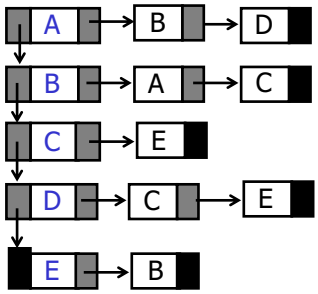


Graph

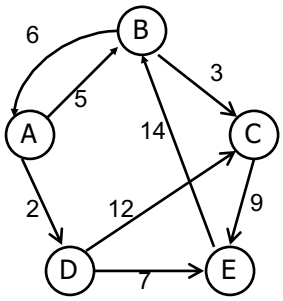


Adjency List graph berarah





Graph berarah dan berbobot



		A 0	B 1	C 2	D 3	E 4
A	0	0	5	0	2	0
B	1	6	0	3	0	0
C	2	0	0	0	0	9
D	3	0	0	12	0	7
E	4	0	14	0	0	0

Perhatikan pemilihan nilai 0.

Penyelesaian kasus Graph halaman sebelumnya :

- Define simpul untuk vertex dan edge
- Mengidentifikasi Simpul pertama sebagai vertex yang pertama
- Tambahkan vertex sisanya
- Tambahkan edge pada masing-masing vertex yang telah terbentuk
- Tampilkan representasi graph berikut bobotnya

```
#include<stdio.h>
typedef struct tipeS {
   struct tipeS *Left;
   int INFO;
   struct tipeS *Right;
   };
typedef struct tipeS simpul;
simpul *P,*FIRST,*LAST,*PVertex,*PEdge,*Q,*R,*S;
simpul *PointS[5];
void main() {
   int A[5][5] = {0,5,0,2,0,  6,0,3,0,0,
                  0,0,0,0,9,  0,0,12,0,7,  0,14,0,0,0};
   char NmS[5] = "ABCDE";
   int I,J;

//Simpul Vertex yang pertama
I=0;J=0;
P=new simpul;
P->INFO = NmS[0];
FIRST = P;
LAST = P;
P->Left = NULL;
P->Right = NULL;
PointS[0] = P;
printf("%c", P->INFO);
printf(" alamat %d  ", PointS[0]);
```

```
//Simpul Vertex yang berikutnya
for (I=1;I<=4;I++)
    {P = new simpul;
    P->INFO = NmS[I];
    LAST->Left = P;
    LAST = LAST->Left;
    P->Left = NULL;
    P->Right = NULL;
    PointS[I] = P;
    printf("\n%c ", P->INFO);
    printf("alamat %d  ", PointS[I]);
    }
```

```
//Simpul Edge untuk semua Vertex
Q = FIRST;
for (I=0; I<=4; I++)
   {R=Q;
   printf("Vertex %c .... ", Q->INFO);
   for (J=0; J<=4; J++)
      {if(A[I][J]!=0)
            {
            P = new simpul;
            P->INFO = A[I][J];
            R->Right = P;
            P->Left = PointS[J];
            printf("berhubungan dengan %c: ", P->Left->INFO);
            printf("bobot %d;", P->INFO);
            P->Right = NULL;
            R = P;
            }
      }
   printf("\n");
   Q = Q->Left;
   }
}
```

Hasil :

```
A alamat 2126
B alamat 238
C alamat 222
D alamat 206
E alamat 190
Vertex A .... berhubungan dengan B: bobot 5;berhubungan dengan D: bobot 2;
Vertex B .... berhubungan dengan A: bobot 6;berhubungan dengan C: bobot 3;
Vertex C .... berhubungan dengan E: bobot 9;
Vertex D .... berhubungan dengan C: bobot 12;berhubungan dengan E: bobot 7;
Vertex E .... berhubungan dengan B: bobot 14;
```