Struktur Data
Pendidikan Teknik Informatika
Universitas Negeri Padang

JOBSHEET (JS-09)

Graph

Fakultas	Proram Studi	Kode MK	Waktu
Teknik	Pendidikan Teknik Informatika	TIK1.61.2317	4 x 50 Menit

TUJUAN PRAKTIKUM

1. Mahasiswa mampu mengaplikasikan konsep graph dalam menyelesaikan kasus.

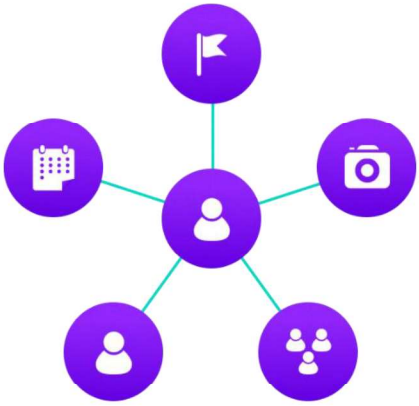
HARDWARE & SOFTWARE

1. Personal Computer
2. IDE: Dev C++

TEORI SINGKAT

Graph

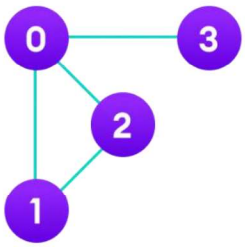
Struktur data graf adalah kumpulan node yang memiliki data dan terhubung dengan node lain. Mari kita coba memahami ini melalui sebuah contoh. Di facebook, semuanya adalah simpul. Itu termasuk Pengguna, Foto, Album, Acara, Grup, Halaman, Komentar, Cerita, Video, Tautan, Catatan, apa pun yang memiliki data adalah simpul. Setiap relasi merupakan edge dari satu node ke node lainnya. Baik Anda memposting foto, bergabung dengan grup, menyukai halaman, dll.



Gambar. Contoh struktur data graf

Semua facebook kemudian merupakan kumpulan dari node dan edge ini. Hal ini dikarenakan facebook menggunakan struktur data grafik untuk menyimpan datanya. Lebih tepatnya, graf adalah struktur data (V, E) yang terdiri dari :

1. Kumpulan simpul V vertex)
2. Kumpulan sisi E (edges), direpresentasikan sebagai pasangan terurut dari simpul (u,v)



Gambar. Simpul dan Sisi

Dalam grafik,

```
V = {0, 1, 2, 3}
E = {(0,1), (0,2), (0,3), (1,2)}
G = {V, E}
```

Terminologi Grafik

1. **Adjacency** : Suatu simpul dikatakan bertetangga dengan simpul yang lain jika ada sisi yang menghubungkannya. Simpul 2 dan 3 tidak bertetangga karena tidak ada rusuk di antara keduanya.
2. **Path** : Urutan edge yang memungkinkan Anda untuk berpindah dari vertex A ke vertex B disebut path. 0-1, 1-2 dan 0-2 adalah lintasan dari simpul 0 ke simpul 2.
3. **Graf Berarah** : Graf yang memiliki sisi (u,v) belum tentu juga memiliki sisi (v,u). Tepi dalam grafik seperti itu diwakili oleh panah untuk menunjukkan arah tepi.

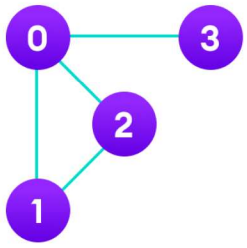
Representasi Grafik

Grafik biasanya direpresentasikan dalam dua cara:

1. Adjacency Matrix

Adjacency Matrix adalah larik 2D dari simpul V x V. Setiap baris dan kolom mewakili sebuah simpul. Jika nilai setiap elemen a[i][j]adalah 1, itu menyatakan bahwa ada tepi yang menghubungkan simpul i dan simpul j.

Adjacency matrix untuk graf yang kita buat di atas adalah :



	0	1	2	3
0	0	1	1	1
1	1	0	1	0
2	1	1	0	0
3	1	0	0	0

Gambar. Graph Adjacency Matrix

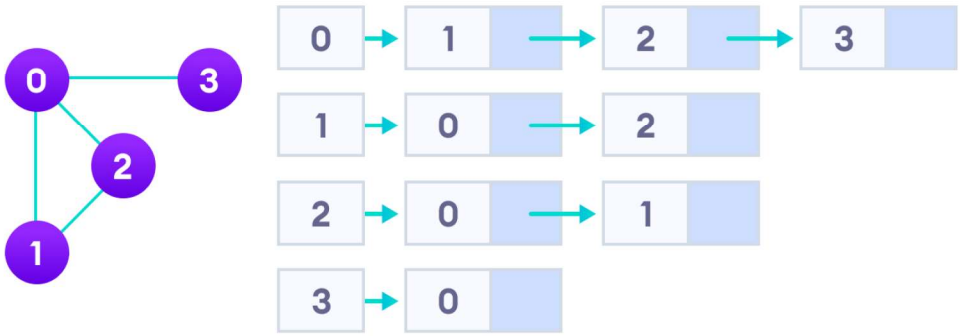
Karena merupakan graf tak berarah, untuk sisi (0,2), kita juga perlu menandai sisi (2,0); membuat adjacency matrix simetris terhadap diagonal.

Pencarian tepi (memeriksa apakah ada tepi antara simpul A dan simpul B) sangat cepat dalam representasi adjacency matrix tetapi kita harus menyediakan ruang untuk setiap kemungkinan hubungan antara semua simpul (V x V), sehingga memerlukan lebih banyak ruang.

2. Adjacency List

Adjacency list mewakili grafik sebagai larik daftar tertaut. Indeks array mewakili sebuah simpul dan setiap elemen dalam daftar tertautnya mewakili simpul lain yang membentuk tepi dengan simpul tersebut.

Daftar adjacency untuk grafik yang kita buat pada contoh pertama adalah sebagai berikut:



Gambar. Representasi Adjacency List

Adjacency list efisien dalam hal penyimpanan karena kita hanya perlu menyimpan nilai untuk edge. Untuk graf dengan jutaan simpul, ini bisa berarti banyak ruang yang dihemat.

Operasi Grafik

Operasi grafik yang paling umum adalah:

1. Periksa apakah elemen ada dalam grafik
2. Grafik Traversal
3. Tambahkan elemen (titik (vertex), sisi (edges)) ke grafik
4. Menemukan jalur dari satu simpul ke simpul lainnya

LATIHAN

1. Adjacency Matrix

```
/* Nama File   : adjacency matrix
Pembuat        : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/
```

```
/* Adjacency Matrix representation of an undirected graph
in C */

#include <stdio.h>
#define V 4 /* number of vertices in the graph */

/* function to initialize the matrix to zero */
void init(int arr[][V]) {
  int i, j;
  for (i = 0; i < V; i++)
    for (j = 0; j < V; j++)
      arr[i][j] = 0;
}

/* function to add edges to the graph */
void insertEdge(int arr[][V], int i, int j) {
  arr[i][j] = 1;
  arr[j][i] = 1;
}

/* function to print the matrix elements */
void printAdjMatrix(int arr[][V]) {
  int i, j;
  for (i = 0; i < V; i++) {
    printf("%d: ", i);
    for (j = 0; j < V; j++) {
      printf("%d ", arr[i][j]);
    }
    printf("\n");
  }
}

int main() {
  int adjMatrix[V][V];

  init(adjMatrix);
```

```
  insertEdge(adjMatrix, 0, 1);
  insertEdge(adjMatrix, 0, 2);
  insertEdge(adjMatrix, 1, 2);
  insertEdge(adjMatrix, 2, 0);
  insertEdge(adjMatrix, 2, 3);

  printAdjMatrix(adjMatrix);

  return 0;
}
```

2. Adjacency List

```
/* Nama File  : adjacency list
Pembuat       : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/

/* Adjacency list representation of a graph in C */
#include <stdio.h>
#include <stdlib.h>

/* structure to represent a node of adjacency list */
struct AdjNode {
    int dest;
    struct AdjNode* next;
};

/* structure to represent an adjacency list */
struct AdjList {
    struct AdjNode* head;
};

/* structure to represent the graph */
struct Graph {
    int V; /*number of vertices in the graph*/
    struct AdjList* array;
};
```

```
struct AdjNode* newAdjNode(int dest)
{
    struct AdjNode* newNode = (struct
AdjNode*)malloc(sizeof(struct AdjNode));
    newNode->dest = dest;
    newNode->next = NULL;
    return newNode;
}

struct Graph* createGraph(int V)
{
    struct Graph* graph = (struct
Graph*)malloc(sizeof(struct Graph));
    graph->V = V;
    graph->array = (struct AdjList*)malloc(V *
sizeof(struct AdjList));

    /* Initialize each adjacency list as empty by making
head as NULL */
    int i;
    for (i = 0; i < V; ++i)
        graph->array[i].head = NULL;
    return graph;
}

/* function to add an edge to an undirected graph */
void addEdge(struct Graph* graph, int src, int dest)
{
    /* Add an edge from src to dest. The node is added at
the beginning */
    struct AdjNode* check = NULL;
    struct AdjNode* newNode = newAdjNode(dest);

    if (graph->array[src].head == NULL) {
```

```
        newNode->next = graph->array[src].head;
        graph->array[src].head = newNode;
    }
    else {

        check = graph->array[src].head;
        while (check->next != NULL) {
            check = check->next;
        }
        // graph->array[src].head = newNode;
        check->next = newNode;
    }

    /* Since graph is undirected, add an edge from dest to
src also */
    newNode = newAdjNode(src);
    if (graph->array[dest].head == NULL) {
        newNode->next = graph->array[dest].head;
        graph->array[dest].head = newNode;
    }
    else {
        check = graph->array[dest].head;
        while (check->next != NULL) {
            check = check->next;
        }
        check->next = newNode;
    }
}
/* function to print the adjacency list representation of
graph*/
void print(struct Graph* graph)
{
    int v;
    for (v = 0; v <= graph->V; ++v) {
        struct AdjNode* pCrawl = graph->array[v].head;
        printf("\n The Adjacency list of vertex %d is: \n
```

```
head ", v);
        while (pCrawl) {
            printf("-> %d", pCrawl->dest);
            pCrawl = pCrawl->next;
        }
        printf("\n");
    }
}

int main()
{
    int V = 5;
    struct Graph* g = createGraph(V);
    addEdge(g, 0, 1);
    addEdge(g, 0, 3);
    addEdge(g, 1, 2);
    addEdge(g, 1, 3);
    addEdge(g, 2, 4);
    addEdge(g, 2, 3);
    addEdge(g, 3, 4);
    print(g);
    return 0;
}
```

DAFTAR PUSTAKA

Kelley, Al and Pohl, Ira. 2003. *C by Dissection: The Essentials of C Programming*. Addison-Weasley.

Khannedy, Eko Kurniawan. 2007. *Diktat Pemrograman C*.Bandung : Unikom.

Liem, Inggriani. 2003. *Catatan Singkat Bahasa C*. Jurusan Teknik Informatika. Institut Teknologi Bandung.

Reema Thareja. 2014. *Data Structures Using C*. India : Oxford University Press.

Solichin, Achmad. 2003. *Pemrograman Bahasa C dengan Turbo C*. Kuliah Berseri ilmukomputer.com