Studi dan Implementasi Struktur Data Graf

Fajar Dwi Anggara
Program Studi Informatika,
Sekolah Teknik Elektro dan Informatika,
Institut Teknologi Bandung,
Jalan Ganesha 10, Bandung
email: if15039@students.if.itb.ac.id

Abstract
Makalah ini membahas tentang pokok bahasan dalam matematika diskrit yaitu Teori Graf dan implementasinya berupa Struktur Data Graf. Teori graf merupakan konsep yang sudah cukup lama dipakai dan diterapkan pada banyak bidang. Makalah ini menyajikan bagaimana tataran konseptual graf, yaitu tentang gambaran umum, definisi graf, hingga sampai pada tataran implementasi, yaitu bagaimana konsep tersebut diterapkan khususnya dalam bidang ilmu komputer. Untuk melengkapi sudut pandang tersebut, makalah ini juga menyertakan implementasi struktur data graf dalam potongan kode program Java.

***Kata Kunci**: teori graf, struktur data graf, java*

1. PENDAHULUAN

Dalam bidang matematika dan ilmu komputer, teori graf mempelajari tentang graf yaitu struktur yang menggambarkan relasi antar objek dari sebuah koleksi objek.

Definisi dari suatu graf adalah himpunan benda-benda yang disebut verteks (atau node) yang terhubung oleh sisi (atau edge atau arc). Biasanya graf digambarkan sebagai kumpulan titik-titik (melambangkan verteks) yang dihubungkan oleh garis-garis (melambangkan sisi).

Definisi tersebut dituangkan dalam notasi matematika sebagai berikut :

$G = (V,E)$

,dengan

V = himpunan tidak-kosong dan berhingga dari simpul-simpul (vertices)
= {v1, v2, ... , vn}

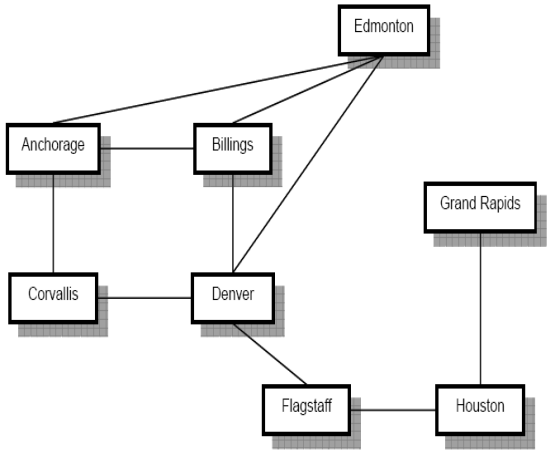
E = himpunan sisi (edges) yang menghubungkan sepasang simpul
= {e1, e2, ... , en}

Graf memiliki banyak jenis, di antaranya berdasarkan ada tidaknya gelang atau sisi ganda, kemudian ada juga yang dibedakan bedasar orientasi arah pada sisi-sisinya. Pembahasan akan dikhususkan pada jenis graf berdasar orientasi arah pada sisi, karena definisi dari golongan graf ini diperlukan dalam pembahasan selanjutnya.

Berdasarkan orientasi arah pada sisi, secara umum graf dapat dibedakan menjadi 2 jenis [1] :

1. Graf tak berarah (*undirected graph*)
Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak berarah. Pada graf tak-berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan.

Gambar 1: Graf tak berarah

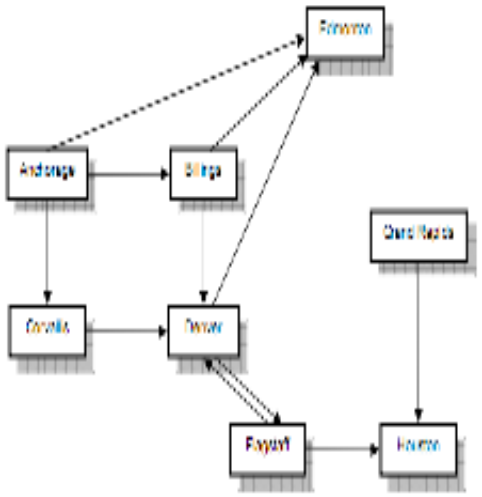


Graf di atas merupakan salah satu contoh graf tak berarah dimana sisi-sisi yang menghubungkan antar simpul dalam graf tersebut tidak memiliki orientasi arah. Sesuai apa yang telah disebutkan di atas, urutan pasangan simpul tidak diperhatikan, dalam contoh ini,

(Anchorage,Corvallis) = (Corvallis,Achorage)

, menyatakan pasangan simpul yang sama.

2. Graf Berarah (*directed graph*)
Graf yang setiap sisinya memiliki orientasi arah disebut sebagai graf berarah. Sisi berarah dalam graf ini dapat dinamakan sebagai busur (*arc*). Lain halnya dengan graf tak-berarah, urutan pasangan simpul disini sangat diperhatikan karena dapat menyatakan hal yang berbeda. Pada graf berarah, (v_j,v_k) dan (v_k,v_j) menyatakan dua buah busur yang berbeda.



Gambar 2 : Graf Berarah

Graf di atas merupakan contoh dari graf berarah yang memiliki sisi-sisi dengan orientasi arah (busur).

(Denver,Flagstaff) dan *(Flagstaff,Denver)* pada graf di atas menyatakan dua busur yang berbeda sebagaimana bisa dilihat dalam gambar, terdapat dua buah busur yang menghubungkan antara kota Denver dan Flagstaff. Untuk busur *(Denver,Flagstaff),* simpul Denver dinamakan sebagai simpul asal (*initial vertex*) dan simpul Flagstaff dinamakan simpul terminal (*terminal vertex*).

2. STRUKTUR DATA GRAF

Dalam bidang ilmu komputer, sebuah graf dapat dinyatakan sebagai sebuah struktur data, atau secara spesifik dinamakan sebagai ADT(abstract data type) yang terdiri dari kumpulan simpul dan sisi yang membangun hubungan antar simpul. Konsep ADT graf ini merupakan turunan konsep graf dari bidang kajian matematika.

Pokok bahasan sebelumnya menjelaskan bahwa graf menampilkan visualisasi data dan hubungannya. Sedangkan jika berbicara masalah implementasi struktur data graf itu sendiri, isu utama yang dihadapi adalah bagaimana informasi itu disimpan dan dapat diakses dengan baik, ini yang dapat disebut dengan representasi internal.

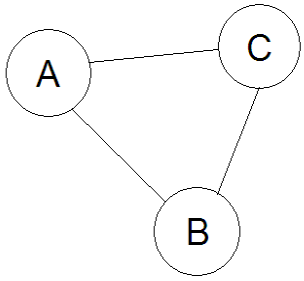
Secara umum terdapat dua macam representasi dari struktur data graf yang dapat diimplementasi. Pertama, disebut *adjacency list*, dan diimplementasi dengan menampilkan masing-masing simpul sebagai sebuah struktur data yang mengandung senarai dari semua simpul yang saling berhubungan.

Yang kedua adalah representasi berupa *adjacency matrix* dimana baris dan kolom dari matriks (jika dalam konteks implementasi berupa senarai dua dimensi) tersebut merepresentasikan simpul awal dan simpul tujuan dan sebuah entri di dalam senarai yang menyatakan apakah terdapat sisi di antara kedua simpul tersebut.

2.1. *Adjacency List*

Dalam teori graf, *adjacency list* merupakan bentuk representasi dari seluruh sisi atau busur dalam suatu graf sebagai suatu senarai. Simpul-simpul yang dihubungkan sisi atau busur tersebut dinyatakan sebagai simpul yang saling terkait.

Dalam implementasinya, hash table digunakan untuk menghubungkan sebuah simpul dengan senarai berisi simpul-simpul yang saling terkait tersebut.



Gambar 3 : *Undirected Cyclic Graph*

Graf pada gambar 3 dapat dideskripsikan sebagai senarai {a,b},{a,c},{b,c}. Dan representasi adjacency list dapat digambarkan melalui tabel di bawah ini.

Tabel 1. Representasi *Adjacency List*

Vertex	*Adjacency*	*Array of Adjacent Vertices*
a	*adjacent to*	b,c
b	*adjacent to*	a,c
c	*adjacent to*	a,b

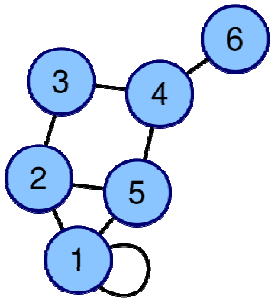
Salah satu kekurangan dari teknik representasi ini adalah tidak adanya tempat untuk menyimpan nilai yang melekat pada sisi. Contoh nilai ini antara lain berupa jarak simpul, atau beban simpul.

2.2. *Adjacency Matrix*

Adjacency Matrix merupakan representasi matriks nxn yang menyatakan hubungan antar simpul dalam suatu graf. Kolom dan baris dari matriks ini merepresentasikan simpul-simpul, dan nilai entri dalam matriks ini menyatakan hubungan antar simpul, apakah terdapat sisi yang menghubungkan kedua simpul tersebut.

Pada sebuah matriks nxn, entri non-diagonal a_{ij} merepresentasikan sisi dari simpul i dan simpul j. Sedangkan entri diagonal a_{ii} menyatakan sisi

kalang(loop) pada simpul i.



Gambar 4: Graf tak berarah berlabel

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Gambar 5: Adjacency matrix

Gambar 5 merupakan adjacency matrix yang berkorelasi dengan graf tak berarah pada gambar 4. Kolom dan baris pada matriks merupakan simpul-simpul berlabel 1-6.

Kelebihan dari *adjacency matrix* ini adalah elemen matriksnya dapat diakses langsung melalui indeks, dengan begitu hubungan ketetanggan antara kedua simpul dapat ditentukan dengan langsung.

Sedangkan kekurangan pada representasi ini adalah bila graf memiliki jumlah sisi atau busur yang relatif sedikit, karena matriksnya bersifat jarang yaitu hanya mengandung elemen bukan nol yang sedikit. Kasus seperti ini merugikan, karena kebutuhan ruang memori untuk matriks menjadi boros dan tidak efisien karena komputer menyimpan elemen 0 yang tidak perlu.

3. HASIL DAN PEMBAHASAN

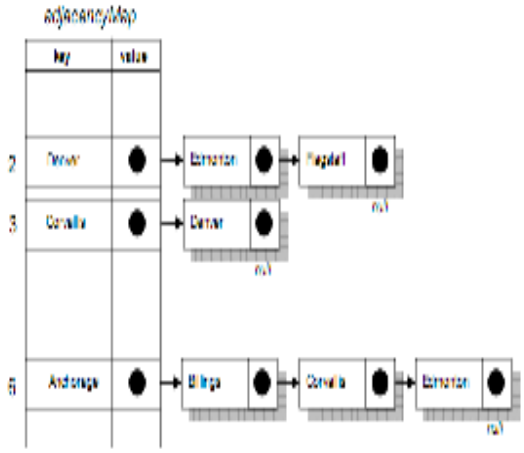
Representasi internal yang dipakai dalam struktur data graf ini merupakan implementasi dari representasi senarai ketetanggaan dengan menggunakan sebuah *hashmap*.

Simpul disimpan sebagai kunci dalam sebuah *Map structure* (struktur pemetaan) dengan tujuan agar mempermudah pencarian sebuah simpul. Struktur pemetaan ini selanjutnya disebut sebagai *adjacency map.*

Sisi yang berawal dari setiap simpul disimpan sebagai senarai simpul yang berhubungan. Senarai ini disimpan sebagai nilai yang memiliki kaitan dengan kunci yang sesuai dalam *adjacency map.*

Sebagai contoh digunakan graf pada gambar 2. Representasi dari graf di atas akan terdiri dari sebuah pemetaan dengan masukan sebagai berikut :

Key : "Anchorage"
Value : ["Billings", "Corvallis", "Edmonton"]



Gambar 6: Adjacency Map

Jika *adjacency map* tersebut adalah sebuah *HashMap* dan sisi dari masing-masing simpul disimpan dalam sebuah senarai ketetanggan maka gambar di representasi internal dari contoh graf di atas ditunjukkan melalui gambar 5.

Implementasi struktur data graf dalam bahasa pemograman java adalah berupa kelas bernama graph sebagai berikut :

```
/**
 * class Graph
 *
 * @author Jim
 * @version 1.0
 */

public class Graph
{
   protected HashMap adjacencyMap;

  /**
   *  insialisasi Graf, membuat adjacency map
   */
   public Graph()
   {
      adjacencyMap = new HashMap();
   }
```

```
  /**
   *  Memeriksa apakah Graf mengandung simpul
   *
   *@return true – jika tidak mengandung simpul
    kosong.
   */
  public boolean isEmpty()
  {
        return adjacencyMap.isEmpty();
  }

  /**
   *
   *
   *  Mengembalikan jumlah simpul induk dalam
graf
   */
  public int size()
  {
        return adjacencyMap.size();
  }
  /**
   *
   *
   *  Mengembalikan jumlah sisi dalam graf
   */
  public int getEdgeCount()
  {
     int count = 0;
     for                                    (int
i=0;i<adjacencyMap.CAPACITY;i++){
        if (adjacencyMap.keys[i] != null){
          LinkedList edges = (LinkedList)
          adjacencyMap.get(adjacencyMap.keys[i]);
           count += edges.size();
        }
     }
     return count;
  }

  /**
   *  Menambahkan sebuah objek sebagai simpul
   *
   *  @param vertex - the specified object
   *  @return true – jika berhasil ditambahkan
   */
  public boolean addVertex (Object vertex)
  {
        if (adjacencyMap.containsKey(vertex))
              return false;
        adjacencyMap.put          (vertex,          new
LinkedList());
        return true;
  }

  /**
    *  Menambahkan sisi, dan simpul jika belum ada
dalam graf
  *
    *  @param v1 – simpul awal mula sisi
    *  @param v2 – simpul akhir dari sisi
    *  @return true - jika sisi berhasil ditambahkan
ke dalam graf
    */
   public boolean addEdge (Object v1, Object v2)
   {
        addVertex (v1); addVertex (v2);
        LinkedList l = (LinkedList)
        adjacencyMap.get(v1);
        l.add(v2);
        return true;
   }

}
```

Untuk memberikan gambaran singkat tentang struktur data graf ini, akan ditunjukan bagaimana membentuk graf seperti gambar 2 dengan menggunakan fungsi dalam struktur data graf di atas.

```
/*
*Contoh implementasi ADT graf
* new instance, bentuk objek graph1 sebagai graph
*/
Graph graph1 = new Graph();
/*
* gunakan fungsi addEdge untuk menambah
simpul-simpul ke dalam graf sesuai dengan
hubungan ketetanggaan antar simpul
*/
graph1.addEdge("Achorage","Billings");
graph1.addEdge("Achorage","Corvalis");
graph1.addEdge("Achorage","Edmonton");
graph1.addEdge("Billings","Denver");
graph1.addEdge("Billings","Edmonton");
graph1.addEdge("Corvalis","Denver");
graph1.addEdge("Denver","Edmonton");
graph1.addEdge("Denver","Flagstaff");
graph1.addEdge("Flagstaff","Denver");
graph1.addEdge("Flagstaff","Houston");
graph1.addEdge("Grand Rapids","Houston");
```

Kelas Graph tersebut memerlukan struktur data yang lain agar bisa diimplementasi, yaitu kelas HashMap dan LinkedList. Namun makalah ini tidak akan membahas kedua struktur data tersebut karena di luar bahasan masalah.

4. KESIMPULAN

Makalah ini telah menunjukan bagaiaman konsep graf diimplementasi dalam bahasa pemrograman Java. Setelah melalui pembahasan, dapat diambil kesimpulan sebagai berikut :

1. Teori Graf merupakan salah satu cabang dari bidang Matematika Diskrit yang mempunyai banyak terapan di berbagai bidang.

2. Struktur data graf merupakan bentuk implementasi dari teori graf yang mencakup definisi, dan hukum-hukum yang menyertainya.

3. Struktur data graf menggunakan representasi internal senarai ketetanggaan dengan alasan efisiensi penggunaan untuk komputasi, karena penggunaan matriks ketetanggan kurang efisisen dan cenderung boros untuk kasus jumlah sisi sedikit sedangkan matriks ketetanggaan yang dibentuk berupa matriks jarang (*sparse*)

DAFTAR REFERENSI

[1] Munir, Rinaldi. 2003. Diktat Kuliah IF2153 Matematika Diskrit *Edisi Keempat.* Agustus 2003. Bandung : Penerbit ITB.

[2] Joe Celko. 2004. *Trees and Hierarchies in SQL for Smarties*. Morgan Kaufmann.

[3] Java Software Development 2. Scotland : Bell College.
http://hamilton.bell.ac.uk/swdev2/notes/
Tanggal Akses : 3 Januari 2009 pukul 09.00

[4] http://en.wikipedia.org
Tanggal Akses : 2 Januari 2009 pukul 14.00