STUDI DAN IMPLEMENTASI ALGORITMA *DIJKSTRA, BELLMAN-FORD* DAN *FLOYD-WARSHALL* DALAM MENANGANI MASALAH LINTASAN TERPENDEK DALAM GRAF

Apri Kamayudi – NIM : 13505009

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15009@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang studi dan implementasi algorotma Dijkstra, Bellman-Ford, dan Floyd marshall dalam menangani masalah lintasan terpendek pada suatu graf. Lintasan terpendek merupakan salah satu dari masalah yang dapat diselesaikan dengan graf. Jika diberikan sebuah graf berbobot, masalah lintasan terpendek adalah bagaimana kita mencari sebuah jalur pada graf yang meminimalkan jumlah bobot sisi pembentuk jalur tersebut.

Masing-masing algoritma memiliki spesifikasi penyelesaian masalah, kompleksitas waktu algoritma, serta jenis masalah yang berbeda.

***Kata kunci**: Algoritma Dijkstra, algoritma Bellman-Ford, algoritma Floyd-Warshall,bobot(weight), jalur terpendek(shortest path),kompleksitas waktu algoritma (running time),simpul(vertex/node),sisi(edge).*

1. Pendahuluan

Graf merupakan suatu cabang ilmu yang memiliki banyak terapan. Banyak sekali struktur yang bisa direpresentasikan dengan graf, dan banyak masalah yang bisa diselesaikan dengan bantuan graf. Seringkali graf digunakan untuk merepresentasikan suaru jaringan. Misalkan jaringan jalan raya dimodelkan graf dengan kota sebagai simpul (*vertex/node)* dan jalan yang menghubungkan setiap kotanya sebagai sisi (*edge*) yang bobotnya (*weight*) adalah panjang dari jalan tersebut. Dalam beberapa model persoalan dimungkinkan bahwa bobot dari suatu sisi bernilai negatif. Misalkan simpul merepesentasikan bandara, sisi mereprentasikan penerbangan yang memungkinkan, dan bobot dari setiap sisi adalah biaya yang dikeluarkan dalam penerbangan tersebut. Untuk suatu kasus dimana seseorang akan dibayar untuk menempuh rute tertentu oleh suatu biro penerbangan, maka bobotnya akan bernilai negatif.

Lintasan terpendek merupakan salah satu dari masalah yang dapat diselesaikan dengan graf. Jika diberikan sebuah graf berbobot, masalah lintasan terpendek adalah bagaimana kita mencari sebuah jalur pada graf yang meminimalkan jumlah bobot sisi pembentuk jalur tersebut.
Terdapat beberapa macam persoalan lintasan terpendek antara lain:
a. Lintasan terpendek antara dua buah simpul tertentu (*a pair shortets path*).
b. Lintasan terpendek antara semua pasangan simpul
(*all pairs shortest path*).
c. Lintasan terpendek dari simpul tertentu ke semua
simpul yang lain (*single-source shoertest path*).
d. Lintasan terpendek antara dua buah simpul yang
melalui beberapa simpul tertentu (*intermediate shortest path*).

Beberapa algoritma yang digunakan untuk menyelesaikan persoalan ini adalah algoritma Dijkstra, algoritma Bellman-Ford, dan algoritma Floyd-Warshall.

Setiap algoritma penyelesaian persoalan lintasan terpendek memiliki kriteria masing-masing. Kompleksitas waktu asimptotik algoritma,

kekurangan, serta kelebihan masing-masing algoritma akan menjadi pokok bahasan studi ini.

2. Algoritma (*single source problem*)

Algoritma Dijkstra, dinamai menurut penemunya, Edsger Dijkstra, adalah algoritma dengan prinsip greedy yang memecahkan masalah lintasan terpendek untuk sebuah graf berarah dengan bobot sisi yang tidak negatif.

Misalnya, bila simpul dari sebuah graph melambangkan kota-kota dan bobot tiap simpul melambangkan jarak antara kota-kota tersebut, algoritma Dijkstra dapat digunakan untuk menemukan jarak terpendek antara dua kota.

Input algoritma ini adalah sebuah graf berarah dan berbobot, G dan sebuah source vertex s dalam G. V adalah himpunan semua simpul dalam graph G. Setiap sisi dari graph ini adalah pasangan vertices (u,v) yang melambangkan hubungan dari vertex u ke vertex v. Himpunan semua edge disebut E. Weights dari edges dihitung dengan fungsi w: $E \rightarrow [0, \infty)$; jadi $w(u,v)$ adalah jarak non-negatif dari vertex u ke vertex v. Cost dari sebuah edge dapat dianggap sebagai jarak antara dua vertex, yaitu jumlah jarak semua edge dalam path tersebut. Untuk sepasang vertex s dan t dalam V, algoritma ini menghitung jarak terpendek dari s ke t.

2.1 Deskripsi

Algoritma Dijkstra melibatkan pemasangan label pada verteks. Kita misalkan L(v) menyatakan label dari verteks v. Pada setiap pembahasan, beberapa verteks mempunyai label sementara dan yang lain mempunyai label tetap. Kita misalkan T menyatakan himpunan verteks yang mempunyai label sementara. Dalam menggambarkan algoritma tersebut, kita akan melingkari verteks-verteks yang mempunyai label tetap. Selanjutnya akan kita tunjukkan bahwa jika L(v) adalah label tetap dari verteks v, maka L(v) merupakan panjang lintasan terpendek dari a ke v. Sebelumnya semua verteks mempunyai label sementara. Setiap iterasi dari algoritma tersebut mengubah status satu label dari sementara ke tetap; sehingga kita dapat mengakhiri algoritma tersebut jika z menerima sebuah label tetap. Pada bagian ini L(z) merupakan panjang lintasan terpendek dari a ke z.

2.2 Algoritma

Algoritma ini mencari panjang lintasan terpendek dari verteks a ke z dalam sebuah graf berbobot tersambung. Bobot dari rusuk (i,j) adalah w(i,j)>0 dan label verteks x adalah L(x). Hasilnya, L(z) merupakan panjang lintasan terpendek dari a ke z.

Masukan : Sebuah graf berbobot tersambung dengan bobot positif. Verteks a sampai z.
Keluaran : L(z), panjang lintasan terpendek dari a ke z.

```
1.  procedure dijkstra (w,a,z,L)
2.  L(a) := 0
3.  for semua verteks x ≠a do
4.  L(x) := ∞
5.  T := himpunan semua verteks
6.  // T adalah himpunan verteks yang panjang
    terpendeknya dari a belum ditemukan
7.  while z∈T do
8.  begin
9.  pilih v∈T dengan minimum L(v)
10. T:=T-{v}
11. for setiap x ∈T di samping v do
12. L(x):=min{L(x), L(v)+w(v,x)}
13. end
14. end dijkstra
```

2.3 Kompleksitas waktu algoritma (*Running time*)

Algoritma Dijkstra'menggunakan waktu sebesar O(V*log V + E)di mana V dan E adalah banyaknya sisi dan titik.

3. Algoritma Bellman-Ford (negative weighted problem)

Algoritma Bellman-Ford menghitung jarak terpendek (dari satu sumber) pada sebuah digraf berbobot. Maksudnya dari satu sumber ialah bahwa ia menghitung semua jarak terpendek yang berawal dari satu titik node. Algoritma Dijkstra dapat lebih cepat mencari hal yang sama dengan syarat tidak ada sisi (edge) yang berbobot negatif. Maka Algoritma Bellman-Ford hanya digunakan jika ada sisi berbobot negatif.

3.1 Deskripsi

Kebenaran dari algoritma Bellman-Ford dapat ditunjukkan dengan induksi sebagai berikut:

Lemma. Setelah pengulangan *i* dari siklus*for*:

- Jika Distance(u) terhingga, akan sebanding dengan panjang dari beberpapa lintasan dari *s* menuju *u*;
- Jika terdapat lintasan dari *s* menuju *u* pada kebnyakan sisi *i* , kemudian Distance(u) adalah kebnyakan panjang pada lintasan terpendek dari *s* menuju *u* dengan kebanyakan sisi *i*.

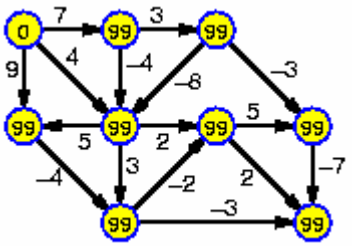
Bukti. Untuk setiap dasar induksi, perhatikan i=0 dan saat kejadian sebelum siklus for yang dieksekusi pertama kali. Kemudian, untuk setiap simpul asal, source.jarak = 0, adalah benart. Untuk setiap simpul *u*, lainnya u.jarak = tak terhingga**,** juga benar karena tidak terdapat dari simpul asal ke simpul u dengan sisi berbobot 0.

Untuk kasus induktif, pertama kali kita membuktikan bagian awal. Bayangkan saat jarak setiap simpul diperbarui sebagai berikut v.jarak := u.jarak + uv.bobot. Dengan menggunakan asumsi induktif, u.jarak adalah panjang dari beberapa lintasan yang menghubungkan simpul awal dengan *u*. Kemudian u.jarak + uv.bobot adalah panjang lintasan yang berasal dari simpul awal menuju *v* yang mengikuti lintasan yang berasal dari simpul awal menuju *u* dan kemudian menuju ke *v*.

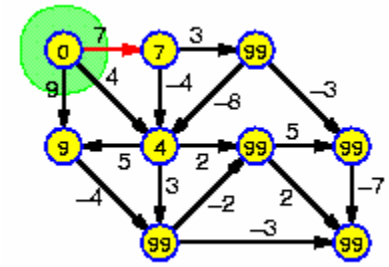
Untuk bagian kedua,perhatikan bahwa lintasan terpendek dari simpul asal menuju *u* denagn kebanyakan terdapat pada *i* sisi. Jadikan *v* sebagai simpul terakhir sebelum mencapai *u* pada lintasan tersebut. Kemudian, bagian suatu lintasan dari simpul awal menuju *v* adalah lintasan terpendek dari simpul asal menuju *v* pada kebanyakan sisi-sisi *i-1*. Dengan asumsi induktif ini, v.jarak setelah siklus *i-1* kebanyakan panjang dari lintasan ini. Dengan demikian, uv.bobot + v.jarak berada padsa kebanyakn panjang lintasan dari *s* menuju *u*. Pada siklus ke- *i*, u.jarak akan dibandingkan dengan uv.weight + v.jarak, dan himpunan sebanding dengannya jika uv.bobot + v.jarak lebih kecil. Kemudian, setelah siklus *i*, u.jarak pada kebanyakan panjang lintasan terpendek dari simpul asal menuju *u* yang melewati kebanyakan sisi *i*.

Ketika *i* sebanding dengan banyaknya simpul pada graf, setiap lintasan akan dijadikan sebagai shortest path overall, kecuali jika terdapat bobot-siklus yang negatif. Jika ada bobot-siklus negative dan dapat diakses dari simpul asal, kemudian diberikan langkah manapun, akan terdapat sebuah lintasan yang lebih pendek one, sehingga tidak terdapat langkah terpendek. Di lain pihak, langkah terpendek tidak akan mengikutsertakan siklus manapun (karena dengan berputar pada siklus tersebut akan membuat langkahnya menjadi semakin pendek), jadi setiap lintasan terpendek akan mengunjungi setiap simpul paling tidak 1 kali, dan banyaknya sisi lebih sedikit dari banyaknya simpul di dalam graf.

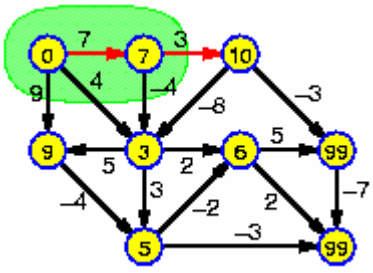
Skema Pencarian Lintasan Terpendek dengan algoritma Bellman Ford



Gambar 1 Contoh graf berbobot negatif

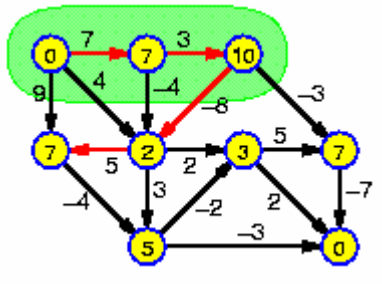


Gambar 2 Tahap pertama Algoritma Bellman-Ford untuk penyelesaian contoh graf pada gambar 1

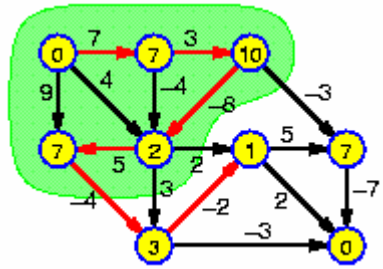


Gambar 3 Tahap kedua Algoritma

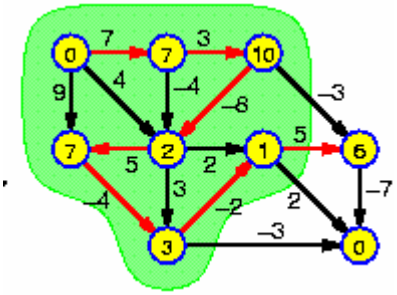
Bellman-Ford untuk penyelesaian contoh graf pada gambar 1



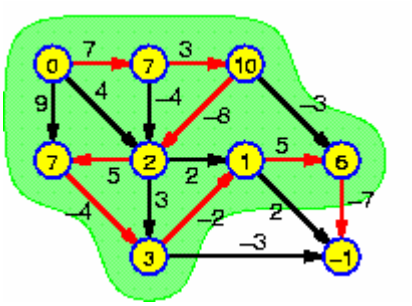
Gambar 4 Tahap ketiga Algoritma Bellman-Ford untuk penyelesaian contoh graf pada gambar 1



Gambar 5 Tahap keempat Algoritma Bellman-Ford untuk penyelesaian contoh graf pada gambar 1



Gambar 6 Tahap Kelima Algoritma Bellman-Ford untuk penyelesaian contoh graf pada gambar 1



Gambar 7 Lintasan terpendek untuk penyelesaian contoh graf pada gambar 1 sebesar -1

3.2 Algoritma

```
// Definisi tipe data dalam graf
record titik {
        list sisi2
        real jarak
        titik sebelum
}
record sisi {
        titik dari
        titik ke
        real bobot
}
 function BellmanFord(list semuatitik, list semuasisi, titik dari)
        // Argumennya ialah graf, dengan bentuk daftar titik
        // and sisi. Algoritma ini mengubah titik-titik dalam
        // semuatitik sehingga atribut jarak dan sebelum
        // menyimpan jarak terpendek.
        // Persiapan
        for each titik v in semuatitik:
                if v is dari then v.jarak = 0
                else v.jarak := tak-hingga
                v.sebelum := null
        // Perulangan relaksasi sisi
        for i from 1 to size(semuatitik):
                for each sisi uv in semuasisi:
                        u := uv.dari
                        v := uv.ke          // uv adalah sisi dari u ke v
                        if v.jarak > u.jarak + uv.bobot
                                v.jarak := u.jarak + uv.bobot
                                v.sebelum := u
        // Cari sirkuit berbobot(jarak) negatif
```

```
        for each sisi uv in semuasisi:
                u := uv.dari
                v := uv.ke
                if v.jarak > u.jarak + uv.bobot
                        error "Graph
mengandung siklus berbobot total negatif"
```

3.3 Kompleksitas waktu algoritma (*Running time*)

Algoritma Bellman-Ford menggunakan waktu sebesar O(V.E), di mana V dan E adalah banyaknya sisi dan titik.

Inisialisasi: O(v)

Loop utama: O(v.e)

Pengecekan loop negative: O(e)

Total: O(v.e)

4. Algoritma Floyd-Warshall (all pairs source problem).

Algoritma Floyd-Warshall memiliki input graf berarah dan berbobot (*V*,*E*), yang berupa daftar titik (node/vertex *V*) dan daftar sisi (edge *E*). Jumlah bobot sisi-sisi pada sebuah jalur adalah bobot jalur tersebut. Sisi pada *E* diperbolehkan memiliki bobot negatif, akan tetapi tidak diperbolehkan bagi graf ini untuk memiliki siklus dengan bobot negatif. Algoritma ini menghitung bobot terkecil dari semua jalur yang menghubungkan sebuah pasangan titik, dan melakukannya sekaligus untuk semua pasangan titik.

4.1 Deskripsi

Dasar algoritma ini adalah sebagai berikut:

- Asumsikan semua simpul graf berarah *G* adalah $V = \{1, 2, 3, 4, ..., n\}$, perhatikan subset $\{1, 2, 3, ..., k\}$.
- Untuk setiap pasangan simpul *i*, *j* pada *V*, perhatiakm semua lintasan dari *i* ke *j* dimana semua simpul pertengahan diambil dari $\{1, 2, ..., k\}$, dan *p* adalah lintasan berbobot minimum diantara semuanya.
- Algoritma ini mengeksploitasi relasi antara lintasan *p* dan lintasan terpendek dari i ke j dengan semua simpul pertengahan berada pada himpunan $\{1, 2, ..., k-1\}$.
- Relasi tersebut bergantung pada apakah *k* adalah simpul pertengahan pada lintasan *p*.

Implementasi algoritma ini dalam pseudocode:
(Graf direpresentasikan sebagai matrix keterhubungan, yang isinya ialah bobot/jarak sisi yang menghubungkan tiap pasangan titik, dilambangkan dengan indeks baris dan kolom)
(Ketiadaan sisi yang menghubungkan sebuah pasangan dilambangkan dengan Tak-hingga)

4.2 Algoritma

```
function fw(int[1..n,1..n] graph) {
   // Inisialisasi
   var int[1..n,1..n] jarak := graph
   var int[1..n,1..n] sebelum
   for i from 1 to n
      for j from 1 to n
         if jarak[i,j] < Tak-hingga
            sebelum[i,j] := i
   // Perulangan utama pada algoritma
   for k from 1 to n
      for i from 1 to n
         for j from 1 to n
            if jarak[i,j] > jarak[i,k] + jarak[k,j]
               jarak[i,j] = jarak[i,k] + jarak[k,j]
               sebelum[i,j] = sebelum[k,j]
   return jarak
}
```

4.3 Kompleksitas waktu algoritma (*Running time*)

Algoritma ini berjalan dengan waktu $O(V^3)$.

5. Kesimpulan

Kesimpulan yang dapat dimbil dari studi ini adalah:

1. Urutan algoritma penyelesaian persoalan lintasan terpendek berdasarkan kompleksitas algoritmanya adalah

$O(V.E) < O((E+V)\log V) < O(V^3)$=Bellman-Ford<Dijkstra<Floyd-Warshall

2. Algoritma Bellman-Ford dapat menangani masalah lintasan terpendek dengan kasus graf berbobot negatif yang tidak dapat diselesaikan oleh algoritma Dijkstra.

3. Berdasarkan masalah yang dapat diselesaikan.

* Algoritma Dijkstra untuk masalah Single-source Shortest Path

* Algoritma Bellman-Ford untuk masalah Single-source Shortest Path.

* Algoritma Floyd-Warshall untuk masalah All-pairs Shortest Path

7. Referensi

[1] Cormen,Thomas H., Charles E. Leiserson, Ronald L. Rivest, & Clifford Stein. (2001).Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill.

[2] Dobson, Simon. (2005). Weighted graphs and shortest paths.UCD School of Computer Science and Informatics. Dublin

[3] Drexel University.(1996). http://mathforum.org/. Tanggal akses: 30 Desember 2006 pukul 10:00.

[4] English Wikipedia. (2006). http://en.wikipedia.org/. Tanggal akses: 28 Desember 2006 pukul 13:00.

[5] Floyd, Robert W. (1962). "Algorithm 97: Shortest Path". Communications of the ACM 5. Templat:Doi.

[6] Kur2003. http://kur2003.if.itb.ac.id/strategialgoritmik. Tanggal akses: 30 Desember 2006 pukul 10:00.

[7] Munir,Rinaldi. (2006). Diktat Kuliah IF2251 Strategi Algoritmik. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

[8] Munir, Rinaldi. (2006). Diktat Kuliah IF2153 Matematika Diskrit Edisi Keempat. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

[9] NIST. (2004). National Institute of Standards and Technology. http://www.nist.gov. Tanggal akses: 30 Desember 2006 pukul 10:00.

[10] Ranau. (2005). http://ranau.cs.ui.ac.id/sda/archive/1998/handout/handout20.html. Tanggal akses: 30 Desember 2006 pukul 10:00.

[11] Rosen, H. Kenneth. (1999). Discrete Mathematics and its Applications. McGraw-Hill. USA.

[12] Topcoder. (2006). http://www.topcoder.com/tc?module=Static&d1=tutorials&d2=graphsDataStrucs3. Tanggal akses: 30 Desember 2006 pukul 10:00.

[13] Wikipedia Indonesia. (2006). http://id.wikipedia.org/. Tanggal akses: 28 Desember 2006 pukul 13:00.