SINGLE-SOURCE SHORTEST PATH PADA GRAF BERBOBOT MENGGUNAKAN ALGORITMA DIJKSTRA DAN BELLMAN-FORD

Imran Djafar, Faizal
STMIK Dipanegara Makassar
Jalan Perintis Kemerdekaan KM.9 Makassar, Telp.(0411)587194 – Fax (0411)588284
e-mail: just_imran77@yahoo.com, ichalabinurullah@gmail.com

Abstrak

Graf adalah sebuah teori matematika yang sangat tua, dan masih digunakan hingga hari ini. Graf ini memiliki karakteristik yang menarik sehingga dapat merepresentasikan berbagai macam masalah. Algoritma untuk perhitungan jarak terpendek antar graf dengan sisi yang memiliki bobot ada bermacam-macam. Pada makalah ini akan dibahas dua macam algoritma, yaitu algoritma Dijkstra dan Algoritma Bellman-Ford. Selain membahas karakteristik dari dua algoritma tersebut, akan dibahas juga perbedaan dari dua algoritma tersebut. Pencarian lintasan terpendek merupakan salah satu persoalan dalam teori graf. Persoalan ini bisa diselesaikan dengan algoritma Dijkstra. Aplikasi teori graf dapat kita saksikan sendiri bila mengendarai kendaraan bermotor di wilayah kota-kota besar pada umumnya. Kita punya banyak alternatif jalan untuk sampai ke tujuan.Namun ada jalan yang satu arah, ada titik-titik kemacetan, ada jam-jam sibuk. Contoh dalam kehidupan sehari-hari adalah masalah kemacetan lalu lintas yang sering terjadi selama perjalanan, sering mengganggu kegiatan kita sehari-hari.Setiap manusia ingin sampai ke tujuan dengan tepat waktu.Tetapi, sering kali kemacetan menyebabkan keinginan manusia terganggu.Dengan pemahaman tentang teori graf, para ahli transportasi mengatur agar lampu lalu lintas menyala dengan warna dan saat yang tepat. Selain itu, pengetahuan tentang teori graf juga diaplikasikan dalam pengaturan jalur penerbangan agar tidak ada pesawat yang bersimpangan, pengaturan jalur telekomunikasi ketika ada pelanggan yang melakukan panggilan telepon, pengaturan jalur pelayaran, serta segala sistem yang berkaitan dengan “titik” dan “garis”. Titik (vertex) dapat digunakan.untuk melambangkan kota, pelabuhan, stasiun, terminal atau airport. Oleh karena itu, dibutuhkan suatu cara untuk menanggulangi gangguan tersebut. Untuk mencapai suatu tempat tujuan dengan waktu yang lebih cepat, kita akan mencari lintasan terpendek dari tempat asal ke tempat tujuan. Hasil yang didapatkan bahwa kompleksitas algoritma dari algoritma Djikstra adalah O(1½ N2).Jika algoritma Djikstra dimodifikasi dengan metodeFord, kompleksitas dari algoritma ini akan meningkatsecara polinomial. Hal ini disebabkan setiap simpul pada graf maksimum akan dikunjungi sebanyak N – 1 kali.Akibat dari hal ini adalah peningkatan kompleksitas sebanyak N – 1 kali. Jadi, O(1 ½ N2) X (N – 1) = O(1 ½N3). Jadi setelah algoritma Djikstra dimodifikasi dengan menggunakan metode Ford, kompleksitasnya adalah O(1½ N3).

Kata kunci: *Graf, Algoritma Dijkstra, Bellman-Ford, Sisi negative, lintasan terpendek*

Abstract

Graf is a mathematical theory that is very old, and is still used to this day. This graph has interesting characteristics that can represent a wide variety of problems. The algorithm for the calculation of the shortest distance between the graph with the side that has a weight is diverse. This paper will discuss the two kinds of algorithms, the algorithm Dijkstra and Bellman-Ford algorithm. In addition to discussing the characteristics of these two algorithms, will also discuss the differences of these two algorithms. Search the shortest path is one of the problems in graph theory. This problem can be solved by Dijkstra's algorithm. Application of graph theory we can see yourself when driving a motor vehicle in the area of large cities in general. We have many alternative path to get to tujuan.Namun there is a one-way street, there are points of congestion, no rush hour. Examples in everyday life is the problem of traffic congestion that often occurs during the trip, often interfere with our daily activities hari.Setiap man wants to exact destination time.But, often jams cause human desire terganggu.Dengan understanding of graph theory , transportation experts set the traffic light in color and the right time. In addition, knowledge of graph theory is also applied in setting the flight path so no aircraft that intersects, setting telecommunications line when there are customers who make phone calls, setting the cruise lines, as well as all system related to the "point" and "line". Point (vertex) can digunakan.untuk symbolize the city, the port, the station, terminal or airport. Therefore, we need a way to cope with the disorder. To reach a

destination with a faster time, we will seek the shortest path from point of origin to point of destination. The results showed that the complexity of the algorithm Djikstra algorithm is O (N2 1½) .If modified with metodeFord Djikstra algorithm, the complexity of this algorithm will meningkatsecara polynomials. This is due to every node on the graph the maximum will be visited as much as N - 1 kali.Akibat of this is the increase in complexity as much as N - 1 times. So, O (1 ½ N2) X (N - 1) = O (1 ½N3). So after Djikstra algorithm modified by using the method of Ford, its complexity is O (1½ N3).

***Keywords**: Graf, Dijkstra's algorithm, the Bellman-Ford, the negative side, the shortest path*

I. PENDAHULUAN

Graf merupakan salah satu cabang dari ilmu dalam ilmu matematika yang mengulas masalah titik dan garis.Dalam hal ini, teori graf dapat dimanfaatkan untuk merancang sistem prasarana transportasi, seperti membuat lintasan jalan raya yang efektif.Masalah yang paling sering ingin dipecahkan adalah bagaimana mencari lintasan terpendek yang menghubungkan beberapa wilayah yang memiliki jarak yang berbeda-beda.Kegunaan dari graf adalah untuk merepresentasikan suatu struktur.Adapun tujuanyang diinginkan dalam masalah perhubungan adalah bagaimana menghubungkan suatu lokasi dengan lokasi lainnya se-efektif mungkin. Jumlah penduduk yang semakin bertambah serta penataan kota yang kurang baik menyebabkan masalah perhubungan semakin kompleks dan dibutuhkan suatu metode yang tepat untuk menghindari kesalahan dalam menentukan keputusan yang disebabkan oleh kompleksitas dari masalah yang ada.

Aplikasi teori graf dapat kita saksikan sendiri bila mengendarai kendaraan bermotor di wilayah kota-kota besar pada umumnya.Kita punya banyak alternatif jalan untuk sampai ke tujuan.Namun ada jalan yang satu arah, ada titik-titik kemacetan, ada jam-jam sibuk.Dengan pemahaman tentang teori graf, para ahli transportasi mengatur agar lampu lalu lintas menyala dengan warna dan saat yang tepat. Selain itu, pengetahuan tentang teori graf juga diaplikasikan dalam pengaturan jalur penerbangan agar tidak ada pesawat yang bersimpangan, pengaturan jalur telekomunikasi ketika ada pelanggan yang melakukan panggilan telepon, pengaturan jalur pelayaran, serta segala sistem yang berkaitan dengan “titik” dan “garis”. Titik (vertex) dapat digunakan.untuk melambangkan kota, pelabuhan, stasiun, terminal atau airport. [2]

Mobilitas manusia saat ini meningkat seiring meningkatnya kebutuhan manusia itu sendiri.Hal ini tentu cukup menyita waktu dan biaya.Oleh karena itu, diperlukan sebuah metode jalur terpendek mencapai tujuan yang diinginkan sehingga dapat lebih efisien.

Menurut teori Graf, persoalan lintasan terpendek adalah merupakan suatu persoalan untuk mencari lintasan antara dua buah simpul pada graf berbobot yang memiliki gabungan nilai jumlah bobot pada sisi graf yang dilalui dengan jumlah yang paling minimum.Persoalan lintasan terpendek ini pun banyak sekali dijumpai di kehidupan sehari-hari. Aplikasi yang paling sering ditemui adalah pada bidang transportasi dan komunikasi, seperti pada pencarian rute terbaik untuk menempuh dua kota atau untuk mengetahui dan menelusuri proses pengiriman paket data komunikasi dalam suatu jaringan komunikasi agar dihasilkan suatu proses yang paling cepat.

Solusi untuk persoalan lintasan terpendek *(shortest path)* ini sering juga disebut sebagai *pathing algoritm.*Banyak sekali algoritma yang dapat digunakan untuk memecahkan pesoalan lintasan terpendek, salah satunya adalah algoritma Djikstra.

Algoritma *Dijkstra* merupakan salah satu algoritma yang digunakan untuk memecahkan permasalahan lintasan terpendek yang terdapat pada suatu graf . Algoritma ini digunakan pada graf berbobot dengan syarat bobot dari masing-masing sisi haruslah bernilai positif (>=0). Algoritma djikstra ini lebih sering digunakan bila dibandingkan dengan solusi untuk persoalan lintasan pendek lainnya, hal ini dikarenakan dengan menggunakan algoritma djikstra dibutuhkan waktu yang lebih sedikit di dalam penyelesaian suatu masalah rute terpendek dan lebih efisien.

Hubungan antara objek yang ada di sekitar kita merupakan hal yang sering kita amati, seperti hubungan antara kota-kota dalam atlas, struktur organisasi pada suatu lembaga, hubungan antara komputer-komputer pada suatu jaringan (LAN&W AN) dan sebagainya.

Fenomena tersebut secara abstrak dapat digambarkan dengan suatu graf *(graph)*, dimana kota-kota, unit dalam organisasi, komputer pada suatu jaringan atau yang sejenisnya digambarkan sebagai simpul *(vertex).*Sedangkan jalan yang menghubungkan antar kota, kabel/media yang menghubungkan antar komputer atau yang sejenisnya digambarkan sebagai sisi *(edges).*

Masalah lintasan terpendek untuk semua pasangan simpul *(the all-pairs shortest path)* adalah masalah menentukan lintasan terpendek atau jarak terpendek untuk setiap pasangan simpul yang ada, sehingga tercapai optimalitas fungsi tujuan (objektif) tertentu. Untuk menyelesaikan masalah ini, ada beberapa algoritma yang dapat digunakan seperti algoritrna Dijkstra, aigoritma Bellman-Ford, algoritma Johnson, algoritma Floyd dan lain sebagainya. Tapi untuk kali ini kami mencoba menggunakan algoritma Dijkstra sebagai solusi untuk menentukan jarak terpendek.

Pada prakteknya tidak semua algoritrna itu efektif untuk diterapkan atau diimplementasikan pada suatu masalah tertentu.Hal ini disebabkan karena setiap algoritrna mempunyai penampilan yang berbeda dalam menyelesaikan suatu masalah.Kelebihan suatu algoritma dibandingkan dengan algoritrna lainnya, antara lain ditentukan oleh struktur dan efisiensi dari algoritrna tersebut.

II. TINJAUAN PUSTAKA

Teori graf *(graph theory)* merupakan salah satu topik dalam matematika yang penerapanmya dapat menyelesaikan berbagai masalah nyata di berbagai bidang seperti ekonomi, industri, transportasi, sains dan lain sebagainya. Berikut akan dijelaskan teori-teori dasar graf yang mendasari masalah lintasan terpendek

A. Pengertian Graf

Graf adalah kumpulan simpul (vertices atau nodes) yang dihubungkan satu sama lain melalui sisi atau busur (edges).Secara matematis, graf didefinisikan sebagai berikut:

Graf G didefinisikan sebagai pasangan himpunan(*V*,*E*) *yang dalam hal ini:*

$V=\{v_1,v_2,\ldots,v_n\}$ *adalah himpunan tidak-kosong dari simpul-simpul (vertices atau node), dan*

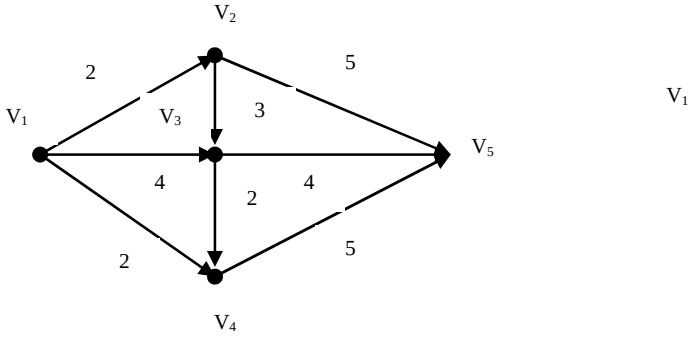
$E=\{e_1,e_2,\ldots,e_n\}$ *adalah himpunan sisi (edges atau arcs) yang menghubungkan sepasang simpul.*

Atau dapat ditulis singkat notasi G=(*V*,*E*)*, dengan V tidak boleh kosong, sedangkan E boleh kosong*

Jadi, sebuah graf dimungkinkan tidak mempunyai sisi satu buah pun, tetapi simpulnya harus ada, minimal satu.

B. Graf Berbobot

Graf berbobot (weighted graph) adalah suatu graf tanpa *arc* parallel dimana setiap *arc*-nya berhubungan dengan suatu bilangan riil tak negative yang menyatakan bobot *arc* tersebut (Jong Jek Siang, 2002, hal:262).



Gambar 1 : graf berbobot

C. Lintasan Terpendek (*Shortest Path*)

Lintasan terperndek adalah lintasan minimum yang diperlukan untuk mencapai suatu tempat dari tempat tertentu.Lintasan minimum yang dimaksud dapat dicari dengan menggunakan graf.Graf yang digunakan adalah graf yang berbobot, yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot.Dalam kasus ini, bobot yang dimaksud berupa jarak dan waktu kemacetan terjadi.

1.*Single-source shortest path*

Ada 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*). [4]

Dalam makalah ini, persoalan yang digunakan adalah *single-source shortest path*. Diberikan sebuah persoalan:

"Diberikan sebuah graf berbobot G(V, *E*). Tentukan lintasan terpendek dari simpul awal, s, ke setiap simpul lainnya di *G*. Asumsi bahwa bobot semua sisi bernilai positif." Algoritma *greedy* untuk mencari lintasan terpendek dapat dirumuskan sebagai berikut:

1. Perikasa semua sisi yang langsung bersisian dengan simpul *s*. Pilih sisi yang bobotnya terkecil. Sisi ini menjadi lintasan terpendek pertama, sebut saja L(1).
2. Tentukan lintasan terpendek kedua dengan cara berikut:
   a. hitung: *d*(*i*) = panjang *L*(1) + bobot sisi dari simpul akhir *L*(1) ke simpul *i* yang lain (ii) pilih *d*(*i*) yang terkecil
   b. Bandingkan *d*(*i*) dengan bobot sisi (*a*, *i*). Jika bobot sisi (*a*, *i*) lebih kecil daripada *d*(*i*), maka *L*(2) = *L*(1) U (sisi dari simpul akhir *L*(*i*) ke simpul *i*)
3. Dengan cara yang sama, ulangi langkah 2 untuk menentukan lintasan terpendek berikutnya.

D. Algoritma Dijkstra

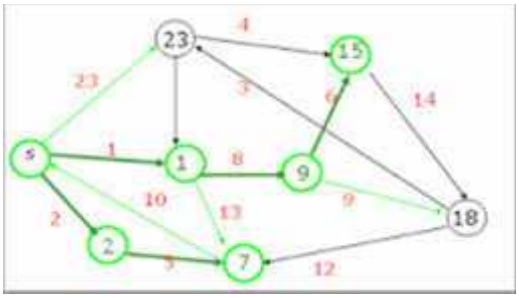
Algoritma Dijkstra ditemukan oleh Edger Wybe Dijkstra.Algoritma ini merupakan algoritma yang paling terkenal untuk mencari lintasan terpendek.Algoritma Dijkstra diterapkan pada graf berarah, tetapi selalu benar untuk graf tak-berarah.Algoritma ini menggunakan strategi *Greedy* sebagai berikut:

"Pada setiap langkah, ambil sisi yang berbobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya ke simpul-simpul yang belum terpilih."

Berikut ini ditampilkan pseducode dari algoritma Djikstra:

function Dijkstra(Graph, source):

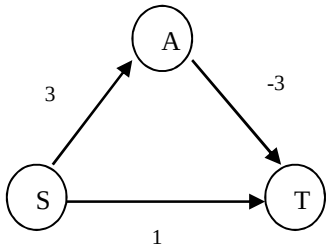
```
    for each vertex v in Graph:                 // Initializations
       dist[v] := infinity ;                    // Unknown distance function from
                                                // source to v
       previous[v] := undefined ;               // Previous node in optimal path
    end for                                     // from source

    dist[source] := 0 ;                         // Distance from source to source
    Q := the set of all nodes in Graph ;        // All nodes in the graph are
                                                // unoptimized - thus are in Q
   while Q is not empty:                        // The main loop
      u := vertex in Q with smallest distance in dist[] ;    // Start node in first case
      remove u from Q ;
      if dist[u] = infinity:
         break ;                                // all remaining vertices are
      end if                                    // inaccessible from source

      for each neighbor v of u:                 // where v has not yet been
                                                // removed from Q.
         alt := dist[u] + dist_between(u, v) ;
         if alt < dist[v]:                      // Relax (u,v,a)
            dist[v] := alt ;
            previous[v] := u ;
            decrease-key v in Q;                // Reorder v in the Queue
         end if
      end for
   end while
return dist;
```



Gambar 2. Algoritma Djikstra

E. Algoritma Bellman-Ford

Untuk kasus dimana terdapat graf yang mempunyai
bobot negatif, misalkan graf tersebut merupakan graf pada gambar 3.



Gambar 3. Graf dengan salah satu bobot sisinya negatif

Misalkan kita tentukan simpul awal adalah simpul S dan simpul akhir adalah simpul T. Secara mudah kita dapat melihat bahwa lintasan terpendek yang dapat ditemukan antara kedua simpul ini adalah (S, A), (A, T) dimana panjang jalannya adalah 2 + (-2) = 0. Jika kita terapkan algoritma Djikstra pada kasus ini akan terdapat kekeliruan, algoritma Djikstra akan mengambil sisi (S,T) yang berjarak 1 sebagai lintasan terpendek antara simpul S dan T. Dari hal ini sudah dipastikan bahwa tidak ada jaminan algoritma Djikstra akan menemukan hasil yang benar untuk masalah *shortest path* jika ada bobot sisi yang negatif [5].

Kitamemerlukan sebuah metode yang dapatmenyelesaikannya, oleh karena itu lahirlah sebuah metodeFord (1956) dimana metode ini tidak lain merupakansebuah hasil modifikasi dari algoritma Djikstra. Algoritma Ford dapat dinyatakandalam langkah berikut ini :

Langkah 1.Inisiasi awal, semua simpul dan sisi belumditandai.Isi semua nilai d(x) untuk x adalah setiapsimpul pada graf dengan aturan sebagai berikut. Untuksimpul awal S yang merupakan awal titik perjalanan*shortest path* kita beri nilai d(S) = 0, untuk simpul laind(X) = ∞, X ≠ S. Kemudian kita ambil Y = S.

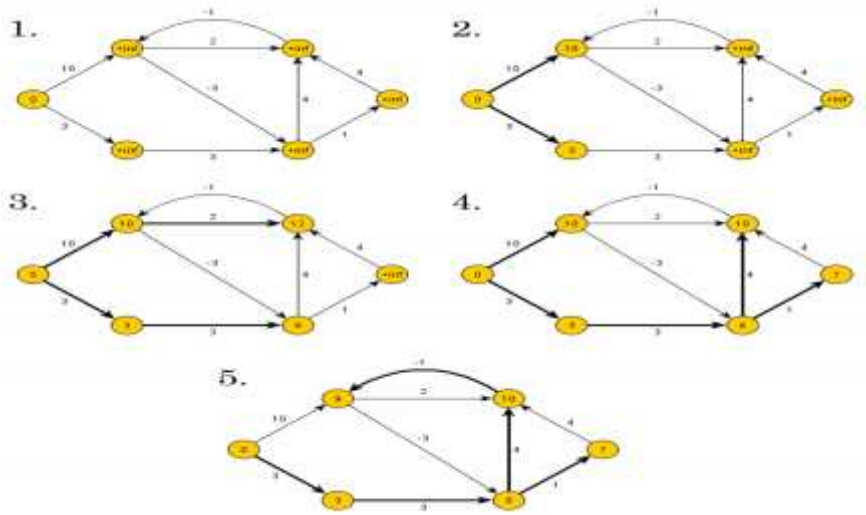
Langkah 2.Untuk setiap simpul kecuali simpul pada Y,definisikan ulang d(X) dengan rumus berikut padapersamaan 1.dimana Y adalah simpul lain yang dapatdikunjungi oleh X serta belum ditandai sebelumnya,maksud a(Y, X) adalah bobot sisi yang dibentuk olehsimpul Y ke simpul X. Jika d(X) = ∞ untuk semuasimpul yang belum ditandai, berhentilah karena tidakada jalan lagi dari simpul S ke simpul yang lainnya. Jikaada, tandai simpul yang mempunyai nilai d(X) terkecil,juga tandai simpul yang terbentuk antara simpul S kesimpul X tersebut. Kemudian kita ambil Y = X. Jikatidak ada satu pun sisi berarah yang keluar dari simpulY = X, kita ganti pilihan kita, jangan memilih Y = Xtetapi simpul yang terkecil berikutnya misal A. Dengandemikian Y = A. Pada langkah ini juga jangan lupamenerapkan butir kedua pada daftar perubahan yangdilakukan yang ada pada bagian awal subbab ini.

Langkah 3. Jika simpul akhir T sudah ditandai dan
pada langkah kedua gagal untuk mendapatkan nilai sisiyang lebih kecil lagi, berhentilah, karena lintasanterpendek *(shortest path)* dari S ke T sudah ditemukan.Jika simpul T belum ditandai, ulangi langkah 2.

Berikut ini ditampilkan pseducode dari algoritma Bellman-Ford:

```
function BellmanFord(list vertices, list edges, vertex source)
  ::distance[],predecessor[]
  // This implementation takes in a graph, represented as
  // lists of vertices and edges, and fills two arrays
  // (distance and predecessor) with shortest-path
  // (less cost/distance/metric) information
```

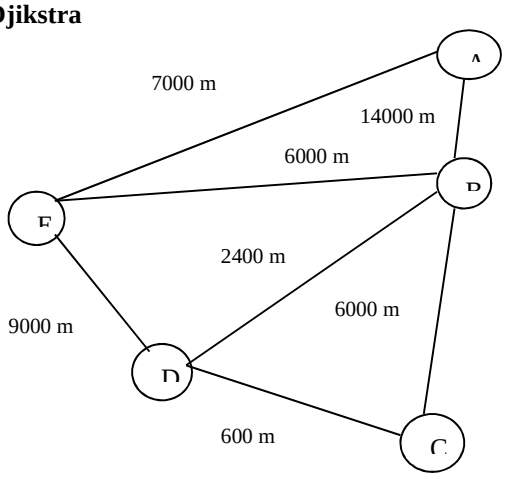
```
  // Step 1: initialize graph
  for each vertex v in vertices:
     if v is source then distance[v] := 0
     else distance[v] := inf
     predecessor[v] := null

  // Step 2: relax edges repeatedly
  for i from 1 to size(vertices)-1:
     for each edge (u, v) in Graph with weight w in edges:
        if distance[u] + w < distance[v]:
           distance[v] := distance[u] + w
           predecessor[v] := u

  // Step 3: check for negative-weight cycles
  for each edge (u, v) in Graph with weight w in edges:
     if distance[u] + w < distance[v]:
        error "Graph contains a negative-weight cycle"
  return distance[],
predecessor[]
```



Gambar 4. Algortima Bellman-Ford

III. METODE PENELITIAN

1. Kasus untuk Algoritma Djikstra



Gambar 2. Penggambaran graf berbobot

Keterangan:

A : MTC Karebosi

B : Masjid Raya

C : Kantor Gubernur Sulsel

D :STMIK Dipanegara

E :Kantor Catatan Sipil

Tabel 1.Panjang lintasan dari satu tempat ke tempat yang lainnya.

	A	B	C	D	E
A	0	1400	∞	∞	7000
B	1400	0	600	2400	6000
C	∞	1700	0	600	∞
D	∞	2400	1200	0	9000
E	7000	6000	∞	9000	0

Persoalan: Carilah jalur terpendek dari STMIK Dipanegara menuju Kantor Catatan Sipil!
Jadi, lintasan terpendek dari STMIK Dipanegara ke Kantor Catatan Sipil adalah yaitu sepanjang :
D-B-E: 2400 m + 6000 m = 8400 m
Tabel 2. Waktu rawan macet

	A	B	C	D	E
A	0	Siang	∞	∞	siang
B	Siang	0	Siang sore	Siang sore	Siang sore
C	∞	Siang sore	0	siang	Siang
D	∞	Siang sore	siang	0	Siang
E	Siang	Siang sore	siang	siang	0

Jarak yang digunakan dari satu tempat ke tempat yang lainnya sama dengan jarak yang digunakan pada tabel 1.

Contoh persoalan: carilah lintasan terpendek dari STMIK Dipanegara menujuKantor Catatan Sipil pada siang hari!
Jadi, lintasan terpendek dari STMIK Dipanegara menuju Kantor Catatan Sipil adalah: D-E, yaitu sepanjang: 9000 m

2. Kasus untuk Bellman-Ford

Kita akan lebih mendalami pemahaman mengenaimetode ini dengan mencoba menerapkannya pada sebuahcontoh kasus. Kembali lagi kita akan menggunakangambar 1 sebagai contoh penerapan metode Ford dalampenyelesaian masalah lintasan terpendek (*shortest path*)pada graf tersebut. Tujuannya adalah untuk memberikangambaran sederhana mengenai metode ini.Kita mulai dengan mendefinisikan sebuah titik awalyaitu simpul S dan titik akhir yaitu simpul T. Kita akanmenerapkan metode Ford dalam mencari lintasanterpendek antara simpul S dan T pada graf di gambar 1.
Adapun langkah – langkah yang digunakan denganmenggunakan metode Ford adalah sebagai berikut :

Langkah 1. Inisial awal, hanya simpul S yang ditandaid(S) = 0, d(A) = ∞, d(T) = ∞.

Langkah 2. Y = S
d(A) = Min { d(A), d(S) + a(S, A) } = Min{∞,0+2} = 2
d(T) = Min { d(T), d(S) + a(S, T) } = Min{∞,0+1} = 1
karena d(T) = Min{d(A), d(T)}, jadi simpul T ditandai
dan sisi (S, T) juga ditandai. Lintasan terpendeksementara kali ini adalah (S, T).

Langkah 3.karena belum semua simpul dan sisiditandai. Maka iterasi kedua dijalankan yaitu kembalike langkah 2.

Langkah 2. Y = T

Karena tidak ada sisi yang meninggalkan simpul T, jadisimpul T diganti dengan simpul A. Simpul A kemudianditandai dan sisi (S, A) juga ditandai.Jadi, lintasanterpendek sementara sekarang adalah (S, T), (S, A).

Langkah 3.Kembali ke langkah 2 untuk mendapatkankemungkinan nilai sisi yang lebih kecil lagi.

Langkah 2. Y = A
d(T) = Min{d(T), d(A) + a(A, T)}=Min{1,2-2}=0
d(S) = Min{d(S), d(A) + a(A, S)}=Min{0,2+∞}=0
Karena d(T) mengecil nilainya dari 1 ke 0, jadi simpul Tdan sisi (S,T) dicabut lagi label tertandanya menjadibelum ditandai. Jadi lintasan pendek sementarasekarang adalah (S,A). Simpul T adalah satu-satusimpul yang belum ditandai akibat dicabut pada proses
sebelumnya. Akibatnya simpul T dan sisi (A,T) harusditandai kembali. Jadi lintasan terpendek sekarangadalah (S, A), (A, T).

Langkah 3. Kembali ke langkah 2 untuk Y = T

Langkah 2. Y = T

Karena tidak ada sisi yang keluar dari T dan tidak adasimpul yang bisa dikecilkan nilainya, serta tidak adapula sisi yang belum ditandai.Langsung ke langkah 3.

Langkah 3.karena semua simpul dan sisi sudahditandai dan tidak ada lagi sisi yang nilainya bisadikurangi, dengan begitu berhantilah algoritma Ford ini.Jadi, Lintasan terpendek terakhir adalah solusinya yaitu(S,A), (A,T). Dimana nilainya adalah 2 + (-2) = 0.

IV. HASIL DAN PEMBAHASAN

Untuk menangani kasus pencarian jalan terpendek pada suatu graf yang mempunyai bobot sisi yang negatif,metode Ford memang merupakan metode yang cukuphebat untuk penyelesaian kasus ini.Hal ini disebabkanmetode ini hanya mengubah beberapa bagian padaalgoritma Djikstra yang sudah banyak dikenal kalanganorang.Akan tetapi metode masih mempunyai kelemahan yangcukup berarti. Metode mempunyai kelemahan jika pada
suatu graf mempunyai sirkuit yang total semua bobot sisiyang membentuknya berjumlah nilai negatif atau kurangdari nol [3]. Misalkan pada suatu graf terdapat sirkuit Sdimana S terdiri dari kumpulan sisi (x1, x2), (x2, x3), (x3,x4), ..., (xn, x1). Jika kita jumlahkan bobot semua sisiyang ada pada sirkuit tersebut adalah kurang dari nol.Kemudian jika kita terapkan metode Ford, kita akan
menemukan bahwa perulangan pengubahan nilai d(X)dimana X adalah simpul – simpul pembentuk sirkuit akandilakukan terus menerus tanpa henti. Jadi dalam halseperti ini bisa dikatakan kalau metode Ford gagalmenyelesaikan masalah pencarian jalan terpendek.Jika kita tidak yakin pada suatu graf tertentu terdapatsirkuit dengan jumlah total bobot sisi pembentuknyaadalah negatif, kita dapat menerapkan langsung metodeFord pada graf yang kita tentukan tersebut. Dalam proses,kita hitung berapa banyak jumlah salah satu simpul padagraf dikunjungi, jika ada salah satu simpul yangdikunjungi lebih besar sama dengan N kali, dimana Nadalah jumlah sisi pada graf, hentikan algoritma tersebut,karena pada graf tersebut terdapat sirkuit yang sedang kitabahas. Jika kita teruskan, kita akan terus mengunjungisimpul tersebut sampai tidak terbatas kali.Hal lain yang umum dibahas jika kita membicarakanmengenai algoritma adalah mengenai kompleksitasnya.Seperti kita ketahui sebelumnya bahwa metode Ford inimerupakan sebuah metode yang diterapkan padaalgoritma Djikstra.Tetapi jika algoritma Djikstra sudahdimodifikasi dengan metode ini maka kompleksitasnyabukan menurun melainkan naik secara polinomial.Pada algoritma Djikstra yang murni, pada iterasi yangpertama, sebanyak N – 1 sisi yang belum ditandai harusdikunjungi.Dari persamaan (1), proses ini membutuhkanN – 1 Penambahan, N – 1 proses minimisasi, danmenyeleksi sebanyak N – 1 angka. Jadi pada iterasi yangpertama akan ada 3(N – 1) proses. Proses yang samadilakukan pada iterasi selanjutnya, karena jumlah sisiyang dikunjunngi berukuran satu jadi total proses yangdilakukan adalah 3(N – 2) proses. Sehingga proses totalyang dilakukan adalah sebagai berikut :

$$\sum_{i=1}^{i=N} 3\,N - i = 3N(N-1)/2 = 1\tfrac{1}{2}\,N(N-1) \quad \text{..... (2)}$$

Dari persamaan (2) diatas dapat kita simpulkan bahwa kompleksitas algoritma dari algoritma Djikstra adalah O(1½ N2).Jika algoritma Djikstra dimodifikasi dengan metodeFord, kompleksitas dari algoritma ini akan meningkatsecara polinomial. Hal ini disebabkan setiap simpul pada
graf maksimum akan dikunjungi sebanyak N – 1 kali.Akibat dari hal ini adalah peningkatan kompleksitassebanyak N – 1 kali. Jadi, O(1 ½ N2) X (N – 1) = O(1 ½N3). Jadi setelah algoritma Djikstra dimodifikasi denganmenggunakan metode Ford, kompleksitasnya adalah O(1½ N3).

V. KESIMPULAN

Dari dua algoritma yang telah dijelaskan penulis dapat disimpulkan bahwa, kedua algoritma memiliki kelebihan dan kekurangan masing-masing.Algoritma Dijkstra dengan kompleksitas waktu yang lebih kecil dan algoritmaBellman-Ford yang handal dalam menangani kasus- kasus graf bersisi negatif.

Dalam penerapannya, pemilihan salah satu algoritma akan sangat penting jika terdapat kasus-kasus sisi berbobot negatif ataupun banyaknya jumlah simpul dan sisi. Namun pada kenyataannya jarang ditemui graf yang mempunyai bobot negatif jika merepresentasikan dunia nyata.Penulis menaruh perhatian pada algoritma Dijkstra karena algoritma ini dapat dengan mudah dimodifikasi sehingga dapat menampilkan jalur atau *path* dari suatu simpul ke simpul lainnya dengan jarak terpendek. Hal ini sangat penting jika graf merepresentasikan sebuah jaringan yang besar dimana kecepatan transfer data anatarkomputer menjadi suatu nilai penting untuk diperhatikan. Pencarian lintasan terpendek sangat berguna untuk menentukan jalan tersingkat untuk menuju suatu tempat.Sehingga, kita dapat sampai tepat waktu menuju tempat tujuan.Hasil analisis berdasarkan bobot-bobot yang berbeda, menunjukkan bahwa semakin banyak bobot yang diberikan, maka semakin akurat pula data yang dihasilkan.Sehingga menghasilkan waktu yang efisien. Ini ditunjukkan dari bahwa kompleksitas algoritma dari algoritma Djikstra adalah O(1½ N2).Jika algoritma Djikstra dimodifikasi dengan metodeFord, kompleksitas dari algoritma ini akan meningkatsecara polinomial. Hal ini disebabkan setiap simpul pada graf maksimum akan dikunjungi sebanyak N – 1 kali.Akibat dari hal ini adalah peningkatan kompleksitassebanyak N – 1 kali. Jadi, O(1 ½ N2) X (N – 1) = O(1 ½N3). Jadi setelah algoritma Djikstra dimodifikasi denganmenggunakan metode Ford, kompleksitasnya adalah O(1½ N3).

DAFTAR PUSTAKA

[1] Arif Y, Fazmah ,Diktat Kuliah CS3024 Desain dan Analisis Strategi , Bandung, 2006.

[2] Aristhia Gita Mentari, dkk. Penentuan Rute Terpendek Jadwal Pertandingan Paris-Saint-Germain Pada Kompetisi Liga Prancis Dengan Menggunakan Algoritma Djikstra. 2010

[3] Edward Minieka, “Optimization Algorithms for Networks and Graphs”, Marcel Dekker, Inc, 1978.

[4] Liu, C.L, “Element of Discrete Mathematics”, McGraw-Hill, Inc, International, 1985.

[5] Prama, Irvan, dkk. Makalah Algoritma Greedy untuk Mencari Lintasan Terpendek, Departemen Teknik Informatika ITB, 2005.

[6] Rinaldi Munir, “Strategi Algoritmik”, ITB, 2007.

[7] Rinaldi Munir, “Strategi Algoritmik”, ITB, 2007.

[8] S., Mehran, April 2005, Exhaustive Recursion and Backtracking.

[9] Strategi Greedy. http://kur2003.if.itb.ac.id/strategialgoritmik. Diakses tanggal 18 Maret 2006.

[10] Thomas H.Cormen, dkk, Introduction to Algorithms Second Edition, MIT Press, London, 2001.