Jurnal UJMC, Volume 4, Nomor 2, Hal. 33 - 41
pISSN : 2460-3333 eISSN : 2579-907X

PERBANDINGAN ALGORITMA DJIKSTRA DAN WARSHALL DALAM PENENTUAN LINTASAN TERPENDEK KE KOTA KLATEN

Niken Retnowati[1], Rizka Safitri Lutfiyani[2]

[1]Universitas Widya Dharma Klaten, niken.retnowati@unwidha.ac.id
[2]Universitas Widya Dharma Klaten, rizka.s@unwidha.ac.id

Abstract. Search for the shortest path is a problem that is faced when we want to travel or go somewhere. Klaten City is one of the cities in Central Java. Cities around Klaten include Boyolali. When we go to Klaten city there are several alternative roads or trails that can be chosen from Boyolali. Mathematically this condition can be applied in graphical form to find the shortest path to the city of Klaten. Determination of the shortest path can be found using the Djikstra Algorithm and the Warshall Algorithm. Researchers tried to compare the two methods to get the most optimal result. From the study concluded that the Djikstra algorithm is more effective than the Warshall algorithm for the shortest path from Boyolali to Klaten.

Keywords*: *Graf, Djikstra Algorithm, and Warshall Algorithm

Abstrak. Pencarian jalur atau lintasan terpendek adalah masalah yang dihadapi saat kita hendak melakukan perjalanan atau menuju kesuatu tempat.Salah satu kota yang ada di Jawa tengah yaitu Klaten. Boyolali adalah salah satu kota yang berada disekitar Klaten, dimana ada beberapa jalan atau lintasan alternatif yang dapat dipilih dari Boyolali untuk menuju kota Klaten. Secara matematis kondisi ini dapat diterapkan dalam bentuk graf guna mencari lintasan terpendek menuju kota Klaten. Penentuan lintasan terpendek dapat dicari menggunakan Algoritma *Djikstra* dan Algoritma *Warshall*. Peneliti mencoba membandingkan kedua metode untuk memperoleh hasil yang lebih efektif dalam menentukan lintasan terpendek menuju kota Klaten. Dari penelitian dititikkan bahwa algoritma *Djikstra* lebih efektif dalam penentuan lintasan terpendek menuju kota Klaten dari Boyolali.

Kata Kunci*: *Graf, Algoritma Djikstra, dan Algoritma Warshall

1 Pendahuluan

Permasalahan yang sering timbul ketika ingin menuju ke suatu tempat tertentu adalah banyaknya opsi jalan yang bisa dilalui. Jalan yang dipilih menentukan cepat tidaknya kita sampai ketujuan. Oleh karena itu banyak ilmuwan mencoba mengembangkan algoritma pencarian jalur terpendek. Salah satu metode pencarian dalam mencari jalur terpendek adalah *Warshall* dan *Djikstra.*

Algoritma *Warshall* adalah algoritma yang mudah dalam implementasinya dan sederhana , untuk mencari lintasan terpendek. Algoritma ini ditemukan oleh Warshall [8]. Prinsip Greedy digunakan Pada

Algortima *Djikstra*, guna mengetahui lintasan minimum dari node satu ke titik yang lainnya yang berarah sama diawali mulai titik pertama hingga titik yang ditujuan. Lokasi-lokasi strategis yang dikenal secara umum merupakan cikal bakal titik-titik yang nantinya akan dihitung. [3].

Dua metode akan digunakan peneliti untuk mencari lintasan terpendek yaitu Algoritma *Warshall* dan Algoritma *Djikstra* untuk membandingkan mana yang lebih efektif jika di aplikasikan pada lintasan terpendek dari Boyolali menuju kota Klaten. Dimana disini saya memilih Unwidha Klaten sebagai titik lokasi tujuan, dan pusat Kota Boyolali sebagai titik awal.

2 Tinjauan Pustaka

2.1 Teori Graf

Suatu Graf *G=(V,E)* merupakan pasangan himpunan sisi dan titik dengan V(G) = Himpunan Titik {vi,v2,…vn} dan E(G) = himpunan sisi {e1,e2,….,en} . Setiap sisi bergandengan dengan satu atau dua titik. Dua buah titik dikatakan bersambungan atau bertetangga jika ada sisi yang menghubungkan keduanya. Graf dapat dikelompokkan menjadi dua yaitu : graf berarah dan graf tidak berarah. Graf berarah adalah graf yang pada tiap sisinya diberikan rute sehingga untuk dua titik v_i dan v_j maka $(v_i, v_j) \neq (v_j, v_i)$. Dilain sisi graf tidak berarah adalah graf yang sisinya tidak mempunyai arah/rute sehingga untuk dua titik v_i dan v_j maka $(v_i, v_j) = (v_j, v_i)$. Selain itu juga dikenal graf berbobot yaitu graf yang memiliki nilai[1]

2.2 Lintasan (*path*) Terpendek

Path minimum adalah jalur minumun yang dibutuhkan guna mendekati suatu lokasi dari lokasi tertentu. path minimum bisa diselesaikan dengan mengaplikasikan graf. Graf yang dipakai adalah graf bernilai, dimana setiap sisi graf terdapat suatu nilai. [2]

Masalah rute terpendek yaitu permasalahan untuk menemukan rute dua atau lebih titik graf bernilai yang merupakan perpaduan nilai sisi graf yang dilewati dengan jumlah paling sedikit, Nilai dapat menunjukkan jarak antara tempat yang satu dengan yang lain, waktu tempuh, biaya perjalanan dan lain-lain [6]

Bermacam-macam persoalan rute minmal, diantaranya :

a) rute minimal dua titik tertentu
b) rute minimal semua partner titik
c) rute minimal satu titik awal (yang ditunjuk) ke seluruh titik yang ada
d) rute minimal titik a dan titik b dimana melalui titik yang lain.

2.3 Representasi Graf

2.3.1 Matriks Ketetanggaan

Misalkan sebuah Graf G = *(V,E)* dengan jumlah titik n. Matriks ketetanggan G adalah matriks bujursangkar dengan rumus n x n atau $M = [m_{ij}]$, dengan $m_{ij} = 1$ apabila titik I dan j bersisian, sebaliknya $m_{ij} = 0$ jika titik I dan j tidak bersisian [4]

Karena Matriks bersisian hanya bernilai 0 dan 1, maka matriks tersebut disebut juga matriks nol-satu. Selain dengan 0 dan 1, elemen matriks bisa juga diterangkan dengan nilai *false* (menyatakan 0) dan *true* (menyatakan 1). Perhatikanlah bahwa matriks bersisian didasarkan pada urutan nomor titik. Di sini rumus n! merupakan cara pengurutan nomor titik, yang berarti ada n! matriks bersisian berbeda untuk graf dengan n titik. Matriks ketetanggan selalu simetri untuk graf sederhana dan tidak berarah, sedangkan untuk graf yang mempunyai arah matriks bersisian belum tentu simetri, selain itu diagonalnya selalu bernilai nol karena tidak ada sisi yang menuju kedirinya sendiri. Sayangnya matriks ketetanggan nol-satu tidak dapat dimanfaatkan untuk graf yang tidak mempunyai sisi ganda, untuk mengusahakannya, maka elemen a_{ij} pada matriks bersisian sama dengan jumlah sisi yang bersosiasi dengan (v_{ij}, v_j). Matriks ketetanggaan tentu tidak lagi matriks nol-satu. Pada graf semu, gelang titik v_i dinyatakan dengan nilai 1 pada kondisi *(i,i)* dimatriks ketetanggaan.

Jumlah elemen matris bersisian untuk graf dengan n titik adalah n^2. Apabila setiap elemen memerlukan ruang simpan sebesar p, maka ruang simpan yang diperlukan semuanya pn^2. Pada matriks berisisan untuk graf tak berarah sederhana simetri cukup dengan menyisihkan elemen segitiga atas saja, karena matriksnya simetri, oleh karena itu ruang simpan yang diperlukan dapat disingkat $\frac{pn^2}{2}$.

Kelebihan menggunakan matriks berisisian adalah matriks bagiannya bisa disalurkan secara serta merta dengan indeks. Dilain sisi, dapat juga kita menunjuk dengan langsung apakah titik *i* dan *j* bertetangga.

nilai per-titik *i* bisa dicaari dari dengan matriks bersisian, untuk graf yang tidak mempunyai arah $d(v_i) = \sum_{j=1}^{n} a_{ij}$

Sedangkan untuk graf berarah

$$d_{in} v_j = jumlah\ nilai\ pada\ kolom\ j = \sum_{i=1}^{n} a_{ij}$$

$$d_{out} v_i = jumlah\ nilai\ pada\ baris\ i = \sum_{j=1}^{n} a_{ij}$$

Tanda ∞ menunjukkan tidak terdapat sisi titik *i* ke titik *j* atau dari titik *i* ke dirinya sendiri, oleh karena itu a_{ij} nilai tak berhingga dapat diberikan.

2.3.2 Matriks Bersisian

Jika matriks bersisian menjelaskan ketetanggan titik-titik pada graf, maka matriks bertetanggaan menerangkan kebersisian titik dengan sisi. Misalkan G = (V, E) adalah graf dengan *n* titik dan *m* buah sisi. Matriks bertetanggaan G adalah matriks berukuran *nxm*. Baris menyatakan label titik, sedangkan kolom menyatakan label sisi. Bila matriks tersebut dipanggil A = $[a_{ij}]$, maka a_{ij}=1

jika titik *i* bertetangaan dengan sisi *j*, sebaliknya $a_{ij} = 0$ bila titik *i* tidak bertetanggaan dengan sisi *j*.
Matriks bertetanggaan bisa digunakan untuk mereprentasikan graf yang memuat sisi ganda. Derajat setiap titik *i* dapat dicari dengan menjumlahkan semua elemen pada baris *i* (kecuali graf yang mempunyai gelang). Jumlah elements matriks bertetanggaan adalah *nm*. Bila setiap element memerlukan ruang simpan sebesar *p*, maka ruang simpan yang dibutuhkan semuanya adalah *pnm.*

2.4 Algoritma *Warshall*

Dalam usahanya guna menemukan rute / jalur minimum, algoritma *Warshall* iterasi diawali dari node pertama yang ditunjuk selanjutnya meneruskan *rute* melalui jalur yang sudah dipilah-pilih dengan mengeliminasi node satu ke node yang lain, sampai menemukan node yang dituju dengan totalan nilai sekecil mungkin. Contoh W_0 merupakan matriks penghubung graf yang mempunyai arah dan nilai mula-mula/awal. W^* adalah matriks hubung terkecil dengan W_{ij}^* = jalur terpendek dari titik v_i ke v_j.

Algortima Warshall untuk menemukan rute/ path minimum yaitu :

1. $W = W_0$
2. buat k=1 sampai n, kerjakan :
   buat i=1 sampai n, kerjakan :
   buat j=1 sampai n, kerjakan :
   apabila $W[i,j] > W[i,k] + W[k,j]$ sehingga
   ganti W[I,j] dengan $W[i,k] + W[k,j]$
3. $W^* = W$

Pada perulangannya guna mendapatkan rute/*path* minimum, *Warshall* menghasilkan matriks sejumlah n sama dengan perulangan-k, sehingga membuat waktu pengerjaannya lama, jika n semakin besar maka semakin lama. Walaupun waktu perulangan atau iterasinya tidaklah yang paling cepat, *Warshall* sering diperuntukan untuk mencari rute minimum karena simple.[8]

2.5 *Djikstra*

Algoritma Djikstra adalah salah satu metode guna mendapat rute minimum dari sebuah titik ke semua titik dalam graf yang hanya memilik bobot positif. Secara formal, masalah rute minimum semua pasangan titik adalah untuk mendapatkan jalur minimum diantara semua pasangan titik $v_i, v_j \in V$ sedemikian sehingga $i \neq j$ [7]
Dalam mendapatkan rute minimum dari suatu titik ke semua pasangan titik algoritma Djikstra melewati sejumlah step yang mengaplikasikan prinsip *Greed.*Prinsip *Greedy* pada algoritma *Djikstra* menerangkan bahwa pada tiap

step kita memastikan sisi yang bernilai terkecil dan mengaplikasikannya pada himpunan solusi [4]

Kecuali matriks bersisian *M*, algoritma ini memerlukan tabel $S = [s_i]$, dengan $s_i = 1$, jika titik i merupakan rute minimum dan sebaliknya $s_i = 0$, jika titik i tidak masuk rute minimum dan juga tabel $D = [d_i]$, dengan d_i = panjang rute diawali titik pertama a ke titik i

Urutan mencari rute minimum dari graf G diawali n-buah titik dimana titik awal a memakai algoritma *Djikstra* yaitu:

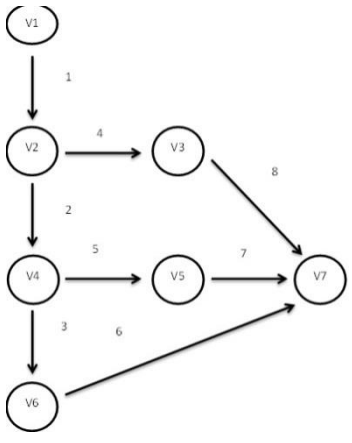
1. (pemberian nilai): $s_i = 0$ dan $d_i = m_{ai}$ untuk $i = 1, 2, \ldots, n$
2. meng*input*kan s_a dengan 1 dan isi d_a dengan ∞
3. dimana tiap $s_i = 0$ dengan $i = 1, 2, \ldots, n$, pilih $d_j = \min\{d_1, d_2, \ldots, d_n\}$ kemudian isi s_j dengan 1 dan perbarui d_i, dengan: d_i (baru) = $\min\{d_i$ (lama), $d_j + m_{ji}\}$. Pada rute, penambahan titik j sebagai titik terpilih untuk rute berikutnya.
4. memperbarui langkah 2 sampai $s_j = 1$, untuk $j = 1, 2, \ldots, n$
5. menciptakan kumpulan titik berdasarkan susunan yang didapat yang merupakan rute minimum dengan nilai d_i [5]

3 Metodologi Penelitian

3.1 Jenis dan Bahan Penelitian

Pada Penelitian ini digunakan data jarak kota Boyolali menuju kota Klaten melalui beberapa lintasan yang mungkin, dengan satuan km. Dalam hal ini akan dibentuk suatu graf dimana kota/ tempat sebagai titik yang akan diberi nama V1,…., Vn dan garis mewakili jarak yang ditempuh.

3.2 Deskripsi Data



Gambar 1. Graf dari Boyolali ke Klaten

Gambar 1 adalah graf menuju kota Klaten dari Boyolali. Dimana ada beberapa asumsi yang diambil peneliti antara lain graf dibuat dengan mengabaikan arah mata angin, hanya arah panah sebagai penanda titik satu

ke titik yang lain. Selain itu jarak juga tidak berdasarkan panjang pendeknya line, serta tidak semua jalan yang mungkin dipaparkan dalam graf ini. Berikut keterangan dari Gambar 1

Tabel 1 Keterangan Graf

Titik	Kota/tempat yang diwakili	no	Jarak	Keterangan
V1	Boyolali	1	6.6	Jarak dari Kota Boyolali menuju Tugu Logerit
V2	Tugu Logerit	2	10.2	Jarak dari Tugu logerit ke Toko Pojok Jatinom
V3	Indomaret Sudimoro	3	8.0	Jarak dari Toko Pojok Jatinom ke Puskesmas Jambu Kulon
V4	Toko Pojok Jatinom	4	1.6	Jarak dari Tugu Logerit ke Indomaret Sudimoro
V5	Apotik Sehati	5	8.0	Jarak dari Toko Pojok Jatinom ke Apotik Sehati
V6	Puskesmas Jambu Kulon	6	6.2	Jarak dari Puskesmas Jambu Kulon Ke Unwidha
V7	Kota Klaten (khususnya Unwidha)	7	1.6	Jarak dari Apotik Sehati ke Unwidha
		8	19.8	Jarak dari Indomaret Sudimoro ke Unwidha

4 Pembahasan

Dari data yang sudah diperoleh maka sebelum mengaplikasikannya ke dalam algoritma *Warshall* maupun *Djikstra* maka langkah awal adalah membuat matrik keterhubungan.

$$W = W_0 = \begin{bmatrix} \infty & 6.6 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 1.6 & 10.2 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 19.8 \\ \infty & \infty & \infty & \infty & 8.0 & 8.0 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 1.6 \\ \infty & \infty & \infty & \infty & \infty & \infty & 6.2 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Gambar 2. Matriks Keterhubungan

4.1 Algoritma Warshall

Setelah membentuk matriks keterhubungan pada Gambar 2 maka proses pencarian jalur terpendek dengan algoritma warshall dimulai dengan iterasi-k (sesuai dengan jumlah titik) dengan kata lain ada 7 iterasi.

a. Iterasi k=1
   Untuk setiap sel matrik W di cek apakah W[i,j] > W[I,1]+W[1,j]. Jika ya, maka W[i,j] diganti dengan W[I,1]+W[1,j]. Dari iterasi k=1 didapat matriks
   Jadi jika W[1,2] = 6.6 maka W [1,2] ⊁ W[1,1] + W[1,2] = ∞ sehingga tetap ditulis 6.6

$$\begin{bmatrix} \infty & 6.6 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 1.6 & 10.2 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 19.8 \\ \infty & \infty & \infty & \infty & 8.0 & 8.0 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 1.6 \\ \infty & \infty & \infty & \infty & \infty & \infty & 6.2 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Gambar 3. Matriks Iterasi 1 Algoritma Warshall

b. Iterasi ke = 2
   misal jika W[1,3] = ∞ maka W [1,3] > W[1,2] + W[2,3] = 6.6 + 1,6 =8.2 sehingga tetap ditulis 8.2. Dari semua perhitungan di dapat matrik

$$\begin{bmatrix} \infty & 6.6 & 8.2 & 16.8 & \infty & \infty & \infty \\ \infty & \infty & 1.6 & 10.2 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 19.8 \\ \infty & \infty & \infty & \infty & 8.0 & 8.0 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 1.6 \\ \infty & \infty & \infty & \infty & \infty & \infty & 6.2 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Gambar 4. Matriks Iterasi 2 Algoritma Warshall

Lakukan langkah yang sama hingga iterasi ke 7 (atau sejumlah titik). Dari perhitungan yang sudah dilakukan di dapat matriks akhir pada iterasi ke 7 adalah sebgai berikut

c. Iterasi ke = 7

$$\begin{bmatrix} \infty & 6.6 & 8.2 & 16.8 & 24.8 & 24.8 & 28 \\ \infty & \infty & 1.6 & 10.2 & 18.2 & 18.2 & 21.4 \\ \infty & \infty & \infty & \infty & \infty & \infty & 19.8 \\ \infty & \infty & \infty & \infty & 8.0 & 8.0 & 9.6 \\ \infty & \infty & \infty & \infty & \infty & \infty & 1.6 \\ \infty & \infty & \infty & \infty & \infty & \infty & 6.2 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Gambar 4. Matriks Iterasi 7 Algoritma Warshall

Dari Gambar 4 diatas terlihat bahwa dari V1 menuju V7 diperoleh jarak terendah yaitu 28 km dengan melalui V1 -V2 – V4- V5-V7.

4.2 Algoritma Djikstra

Dari matriks keterhubungan pada Gambar 2 , langkah selanjutnya adalah :

1. Tentukan L={}
   Dan V={v2, v3, …v7}
2. D(2) = 6.6, D(3,4,5,…7) = ∞
3. V-L= {v2,v3,…,v7}-{}={v1,v2,…,v7}
4. Vn=V7 ∉ L sehingga langkah selanjutnya adalah ambil D(k) terkecil yaitu D(2)=6.6
   Sehingga vk=v2 dimana $L = L \cup \{Vk\} = L \cup \{V2\} = \{V2\}$
   V-L ={v2,v3,…,v7}-{v2}={v3,v4,…,v7}
   K=2 selidiki setiap titik dalam V-L
5. Untuk j=3
   D(j)=D(3)= ∞ ; D(k) + W(k,j) = D(2) + W(2,3)= 6.6 + 1.6 = 8.2
   Oleh karena D(3) > dari D(2) + W(2,3) maka D(3) diubah jadi 8.2
6. Untuk j=4
   D(j)=D(4)= ∞ ; D(k) + W(k,j) = D(2) + W(2,4)= 6.6 + 10.2 = 16.8
   Oleh karena D(4) > dari D(2) + W(2,4) maka D(4) diubah jadi 16.8
7. Untuk j=5
   D(j)=D(5)= ∞ ; D(k) + W(k,j) = D(2) + W(2,5)= 6.6 + ∞ = ∞
   Oleh karena D(5) ≯ dari D(2) + W(2,5) maka D(5) tetap ∞
8. Untuk j=6
   D(j)=D(6)= ∞ ; D(k) + W(k,j) = D(2) + W(2,6)= 6.6 + ∞ = ∞
   Oleh karena D(6) ≯ dari D(2) + W(2,6) maka D(6) tetap ∞
9. Untuk j=7
   D(j)=D(7)= ∞ ; D(k) + W(k,j) = D(2) + W(2,7)= 6.6 + ∞ = ∞
   Oleh karena D(7) ≯ dari D(2) + W(2,7) maka D(7) tetap ∞

Langkah selanjutnya adalah mengambil D(k) terkecil dan mengulangi kembali proses ke 4 dan seterusnya. Hasil perhitungan Algoritma *Djikstra* dapat dilihat pada Tabel 2 .

Tabel 2. Hasil Algoritma *Djikstra*

Indeks k shg D{k} minimum	L	V-L	D{2}	D{3}	D{4}	D{5}	D{6}	D{7}
	∅	{v2,v3,……, v7}	6.6	∞	∞	∞	∞	∞
2	{V2}	{v3,v4,……, v7}	6.6	8.2	16.8	∞	∞	∞
3	{V2,V3}	{v4,……, v7}	6.6	8.2	16.8	∞	∞	28
4	{V2,V3,V4}	{v5,……, v7}	6.6	8.2	16.8	24.8	24.8	28
5	{V2,V3,V4,V5}	{v6,……, v7}	6.6	8.2	16.8	24.8	24.8	26.1
6	{V2,V3,V4,V5,V6}	{v7}	6.6	8.2	16.8	24.8	24.8	31
7	{V2,V3,V4,V5,V6,V7}							

Dari Hasil diatas diperoleh bahwa jarak terdekat dari Boyolali ke Klaten yaitu 26.1 km dengan rute V1-V2-V4-V5-V7

5 Kesimpulan

Dari pembahasan diatas dapat dititikkan bahwa, secara algoritma *Warshall* lebih sederhana dan lebih mudah diimplementasikan, namun *Warshall* memiliki waktu yang lebih lama untuk memproses dari pada *Djikstra*. Selain itu dilihat dari hasil dimana *Warshall* dan *Djikstra* sama-sama menghasilkan jalur terpendek V1-V2-V4-V5-V7 tetapi *Djikstra* lebih mendekati jarak sebenarnya yaitu 25,8 km sedangkan *Djikstra* menghasilkan 26.1 km dan *Warshall* 28 km

6 Daftar Pustaka

[1] Ahuja dkk. 1993. *Network flow : theory algorithms and applications*. Pretince Hall, New Jersey

[2] Defindal, Irvan Prama dkk. 2005. *Algoritma Greedy untuk Menentukan Lintasan Terpendek*. Bandung : ITB

[3] Ferdiansyah & Ahmad Rizal. 2013. *Penerapan Algoritma Djikstra untuk Menentukan Rute Terpendek Pembacaan Water Meter Induk PDAM Tirta Kerta Raharja Kabupaten Tangerang*. Jurnal TICOM vol 2. No 1 September 2013

[4] Munir. 2005. *Diktat kuliah IF2251 strategi algorithm*. Bandung

[5] Munir. 2008. Matematika Diskrit. Bandung : Penerbit Informatika

[6] Pradhana, B. A. 2009. *Studi dan Implementasi Persoalan Lintasan Terpendek suatu Graf dengan algoritma Djikstra dan Algoritma Bellman-ford*.

[7] Sarwoko, E.A. 2003. *Perancangan Arsitektuur Pemaralelan untuk Mencari Shortest path dengan Algoritma Djikstra. Jurnal Matematika dan Komputer*. 6 : 137-143

[8] Siang, Jong Jek. 2006. *Matematika Diskrit dan Aplikasinya pada Ilmu Komputer*. Yogyakarta : ANDI