Perbandingan Kompleksitas Algoritma Prim, Algoritma Kruskal, Dan Algoritma Sollin Untuk Menyelesaikan Masalah *Minimum Spanning Tree*

[1]Wamiliana, [2]Didik Kurniawan, [3]Cut Shavitri N.F.

[1]Jurusan Matematika FMIPA Unila
[2]Jurusan Ilmu Komputer FMIPA Unila
[3]Jurusan Ilmu Komputer FMIPA Unila

Abstract

This study discusses about the complexity comparison among 3 algorithms, those are: Prim Algorithm, Kruskal Algorithm and Sollin Algorithm. The graphs that used for implementation are complete graphs of order n, where n is 10 to 100 with increment of 10 and the data used are generated randomly with weight ranging from 1 to 1000. C++ programming language is used to develop source code for the data implementation and from the result we found that each algorithm has complexity $O(n^2)$.

Keywords: *Complexity, Kruskal's Algorithm, Prim's Algorithm, Sollin's Algorithm.*

1 Pendahuluan

Algoritma merupakan suatu metode langkah demi langkah dari pemecahan suatu masalah. Kata "algoritma" berasal dari nama matematikawan Arab abad kesembilan Al- Khowarizmi. Algoritma didasarkan pada prinsip-prinsip matematika yang memainkan peranan penting dalam matematika dan ilmu komputer [1].

Pengembangan algoritma dan berbagai bidang kajian matematika telah banyak dilakukan guna membantu penyelesaian masalah dalam kehidupan sehari-hari. Pengembangan algoritma tersebut salah satunya adalah yang bertujuan untuk memecahkan masalah optimalisasi sumber daya. Misalnya, pada pelaksanaan pembangunan infrastruktur suatu daerah yang meliputi: pembangunan jaringan listrik, jaringan telepon, jaringan air bersih, jaringan transportasi, dan lain sebagainya. Pembangunan ini tentunya sangat membutuhkan faktor pendukung yaitu antara lain biaya, waktu, dan tenaga. Hal tersebut sebenarnya dapat diatasi dengan menggunakan pemodelan *tree* dalam graf.

Pemodelan masalah dengan menggunakan *tree* mempertimbangkan berbagai faktor, misalnya faktor biaya, jarak, waktu maupun tenaga yang diperlukan. Untuk penyelesaian pemodelan masalah *tree* tersebut biasanya menggunakan konsep pohon merentang minimum (*minimum spanning tree)* atau MST. Suatu algoritma tidak saja harus menghasilkan keluaran yang benar, tetapi juga harus efisien. Kebenaran suatu algoritma harus diuji dengan jumlah masukan tertentu untuk melihat kinerja algoritma berupa waktu yang diperlukan untuk menjalankan algoritmanya dan ruang memori yang diperlukan untuk struktur datanya [2]. Saat ini telah ada beberapa algoritma untuk menyelesaikan masalah *minimum spanning tree*, diantaranya yaitu Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin. Algoritma Sollin sejauh ini sangat jarang didiskusikan oleh para peneliti. Untuk penentuan penyelesaian *minimum spanning tree* umumnya para peneliti menggunakan Algoritma Prim dan Kruskal. Oleh karena itu pada penelitian ini penulis tertarik untuk mendiskusikan Algoritma Sollin dan membandingkannya dengan Algoritma Prim dan Kruskal berdasarkan atas kompleksitas waktu dari algoritma tersebut.

2 Metode Penelitian

Pada penelitian ini ada beberapa tahapan yang diterapkan. Tahapan-tahapan tersebut adalah:

2.1 Analisis kebutuhan

Tahapan pada penelitian ini adalah studi literatur, yakni mengumpulkan informasi sebanyak mungkin mengenai Algoritma penyelesaian MST. Informasi tersebut didapatkan dari berbagai sumber baik dari buku, jurnal dan karya tulis ilmiah, serta dari halaman-halaman *website*.

2.2 Desain

Perancangan desain ini dibuat berdasarkan hasil dari analisis kebutuhan yang telah diperoleh.

2.3 Pemrograman

Tahap ini adalah pembuatan program penerapan algoritma dengan menggunakan bahasa pemrograman C++ serta membuat visualisasi hasil yang telah didapat dengan menggunakan pemrograman Visual Basic.

2.4 Pengujian

Tahap pengujian ini mencari kesalahan-kesalahan yang telah terlewati dari tahap sebelumnya.

2.5 Implementasi

Implementasi merupakan tahap dimana program yang telah dibuat bisa dipergunakan oleh pihak-pihak yang membutuhkan.

3 Pembahasan

3.1 Analisis Kebutuhan Sistem

Perbandingan Kompleksitas Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin dalam menyelesaikan masalah *Minimum Spanning Tree* ini dibangun dengan menggunakan bahasa pemrograman C++ dan juga dibuat visual dari hasil penyelesaian masalah MST tersebut dengan menggunakan pemrograman Visual Basic. Dalam penelitian ini, algoritma yang telah dibangun akan dibandingkan dan dianalisis guna melihat algoritma mana yang paling efisien berdasarkan pada komplestitas dari masing-masing algoritma tersebut.

Data yang digunakan dalam penelitian ini merupakan graf lengkap dengan titik dimulai dari 5, 10 dan kelipatan dari 10 hingga mencapai 100 titik. Data tersebut merupakan data acak dengan nilai dari masing-masing titik dimulai dari nilai 1-1000.

3.2 Implementasi Program

Tahapan implementasi ini merupakan tahapan implementasi pembuatan kode program (*source code*) dari masing-masing algoritma yaitu Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin dengan menggunakan bahasa pemrograman C++. Pembuatan kode program ini untuk membandingkan dari masing-masing algoritma tersebut yang manakah yang merupakan algoritma yang paling efisien dengan menganalisis masing-masing algoritma.

1. Algoritma Prim

Algoritma Prim merupakan algoritma paling sederhana dalam menyelesaikam masalah *minimum spanning tree*. Dalam algoritma ini langkah awal yang dilakukan adalah memilih salah satu titik secara acak yang akan dijadikan sebagai *root*. Setelah titik *root* tersebut terpilih, maka pilih titik selanjutnya yang berhubungan dengan *root* tersebut yang memiliki jarak atau nilai *edge* terkecil. Selanjutnya pilih lagi titik yang memiliki nilai *edge* terkecil yang berhubungan dengan dua titik yang telah terpilih sebelumnya. Lakukan langkah ini hingga n-1 sampai terbentuk suatu penyelesaian *minimum spanning tree*. Pada langkah pemilihan titik selanjutnya memungkinkan terjadinya suatu sirkuit jika saat nilai *edge* yang terkecil selanjutnya merupakan titik yang telah terpilih sebelumnya, tetapi jika terjadi kondisi tersebut maka algoritma ini secara otomatis menghindari dan tidak memilih titik tersebut serta memilih titik dengan nilai *edge* terkecil selanjutnya yang belum terhubung dengan titik-titik yang telah terpilih.

Langkah-langkah penyelesaian *minimum spanning tree* dari Algoritma Prim yang telah dibuat dengan menggunakan bahasa pemrograman C++ adalah sebagai berikut [3]:

1. Menginisilisasi bahwa nilai dari graf T masih berupa graf kosong.
2. Memilih titik vertek v_1 sebagai titik awal dari Graf T. Variabel telah_dikunjungi[i] dalam algoritma ini berfungsi sebagai pencegah terjadinya suatu sirkuit. Pada awal program variabel tersebut diberi nilai 0, maka pada saat program dijalankan akan terdapat kondisi dimana nilai variabel tersebut menjadi 1 dan ini yang akan menjadi pencegah terjadinya suatu sirkuit.
3. Memilih wakil edge dari titik tersebut yang memiliki bobot paling minimum.
   - Pilih titik v_1 sebagai langkah awal. Masukkan v_1 ke dalam V(T).
   - Untuk i=1, 2, 3,.., n-1 lakukan:
     1. Pilih e $\in$ E(G) dan e $\in$ E(T) dengan syarat:
        i. e berhubungan dengan satu titik dalam T dan tidak membentuk sirkuit. Dalam menghindari sirkuit pada potongan program ini juga dengan menggunakan variabel telah_dikunjungi[i] sama seperti pada bagian langkah 2..
        ii. e mempunyai bobot terkecil dibandingkan dengan semua garis yang berhubungan dengan titik-titik dalam T.
     2. Tambahkan e ke E(T) dan w ke V(T).

2. Algoritma Kruskal

Algoritma kedua yang akan dianalisis adalah Algortima Kruskal. Algoritma ini lebih rumit bila dibandingkan dengan Algoritma Prim karena pada Algoritma Kruskal data harus diurutkan dulu berdasarkan dari data *edge* terkecil ke data *edge* terbesar, dan proses penyelesaian masalah *minimum spanning tree* pun lebih rumit karena *edge* yang dipilih bisa sembarang *edge* dengan syarat *edge* tersebut memiliki nilai bobot terkecil dan tidak terjadi sirkuit diantara *edge-edge* yang dipilih tersebut. Dalam Algoritma Kruskal pada proses pembentukannya mungkin saja terjadi *forest*, tetapi hasil akhir tetap berupa *tree*.

Langkah-langkah penyelesaian masalah *minimum spanning tree* dengan menggunakan Algoritma Kruskal adalah sebagai berikut [3]:

1. Inisialisasi bahwa graf T masih berupa graf kosong.
2. Inisialisasi data yang akan diurutkan lalu data diurutkan dari bobot terkecil ke bobot terbesar.
3. Pilih sisi atau *edge* (u, v) dengan bobot minimum yang tidak membentuk sirkuit di T dari data yang telah diurutkan. Tambahkan (u, v) ke dalam T. Lakukan langkah ini hingga n-1 dan terbentuk penyelesaian *minimum spanning tree*.

3. Algoritma Sollin

Algoritma ketiga yang akan dianalisis adalah Algoritma Sollin. Konsep dari algoritma ini adalah memilih wakil *edge* dari masing-masing titik/*vertex* dengan nilai bobot terkecil. *Edge* yang telah terpilih tersebut lalu dihubungkan untuk menemukan solusi dari *minimum spanning tree*. Tetapi ada kemungkinan *edge* yang terpilih dari masing-masing titik akan sama atau duplikat, maka akan ada proses penghapusan duplikat. Pada Algoritma Sollin juga memungkinkan terjadinya suatu *forest.*

Langkah-langkah dalam menentukan hasil dari Algoritma Sollin ini adalah:

1. Menginisialisasi bahwa graf T masih berupa graf kosong.
2. Memilih *edge* dengan nilai bobot terendah dari masing-masing titik/*vertex* yang saling berhubungan.
3. Mengeliminasi *edge* duplikat jika pada *edge* yang telah terpilih terdapat *edge* dengan titik yang sama.
4. mengelompokkan *edge* yang telah terpilih kedalam *tree*-nya masing-masing. Pada langkah ini pengecekan jika terbentuk sirkuit pada data hasil pemilihan wakil *edge* yang terpilih, dan dilakukan penghapusan jika terjadi duplikat. Jika pada data yang akan ditemukan solusi dari *minimum spanning tree* telah terbentuk pada langkah ini, maka langkah selanjutnya tidak perlu dilakukan.
5. Menghubungkan masing-masing *tree* yang telah terbentuk di dalam graf T agar menjadi suatu penyelesaian minimum spanning tree. Penghubungan ini diawali dengan sorting titik-titik yang belum terhubung diurutkan dari paling kecil ke besar. Pada langkah ini data sebelum diurutkan diberi label terlebih dahulu agar pada saat setelah data diurutkan data tersebut masih merupakan nilai *edge* dari *vertex* yang sama.

3.3 *User Interface* Program

Pada tahap ini dilakukan pengujian dari program yang telah dibuat berdasarkan dari algoritma masing-masing. Pengujian ini bertujuan untuk mencari kesalahan-kesalahan yang telah terlewati pada tahap sebelumnya. Pengujian ini dilakukan untuk melihat apakah dari program yang telah dibuat terdapat kesalahan dan untuk membuktikan bahwa tidak terjadi sirkuit didalam program penyelesaian *minimum spanning tree* dari masing-masing algoritma. Pengujian pada masing-masing algoritma dilakukan dengan menggunakan perlakuan yang sama yaitu dengan menggunakan data yang sama berupa graf lengkap dengan jumlah *node* 10 titik. Dari hasil pengujian ini akan diketahui waktu *running* yang dibutuhkan oleh masing-masing algoritma dan juga titik-titik mana yang terhubung yang berupa penyelesaian dari masalah penyelesaian *minimum spanning tree* dari graf lengkap yang digunakan.

Contoh *user interface* :

Nama file yang di uji adalah file 10/1.txt, dan nilai dari file tersebut dapat dilihat pada Tabel 1.

Tabel 1 Data bobot graf lengkap dengan jumlah vertex n=10

NO	*EDGE*		BOBOT
	I	J	
1	1	2	904
2	1	3	785
3	1	4	10
4	1	5	263
5	1	6	51
6	1	7	693
7	1	8	507
8	1	9	633
9	1	10	661
10	2	3	587
11	2	4	695
12	2	5	890
13	2	6	530
14	2	7	206
15	2	8	355
16	2	9	605
17	2	10	905
18	3	4	61
19	3	5	822
20	3	6	590
21	3	7	454
22	3	8	544
23	3	9	180
24	3	10	705
25	4	5	152

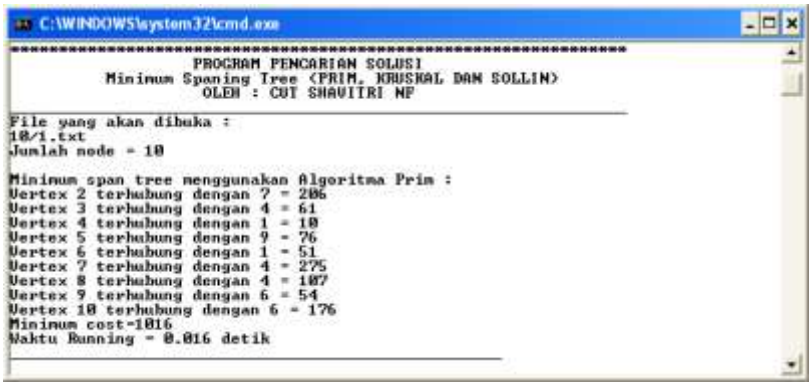
26	4	6	297
27	4	7	275
28	4	8	107
29	4	9	720
30	4	10	678
31	5	6	263
32	5	7	805

33	5	8	956
34	5	9	76
35	5	10	193
36	6	7	672
37	6	8	451
38	6	9	54
39	6	10	176

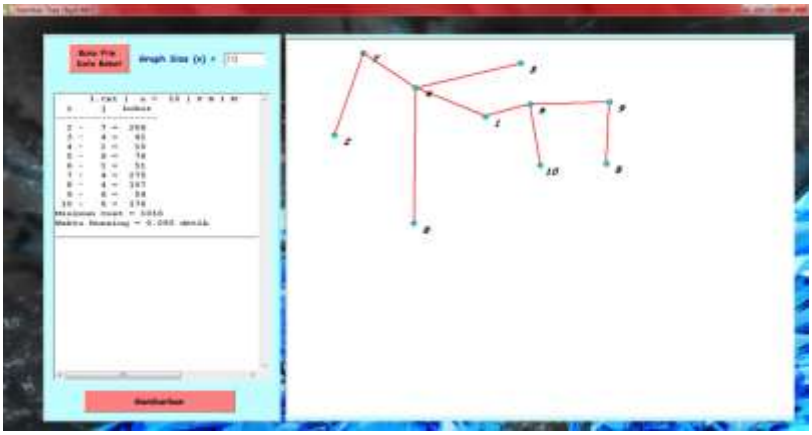
40	7	8	917
41	7	9	690
42	7	10	878
43	8	9	550
44	8	10	796
45	9	10	734

Berikut ini merupakan hasil dari pengujian program:

1. User Interface program Algoritma Prim.

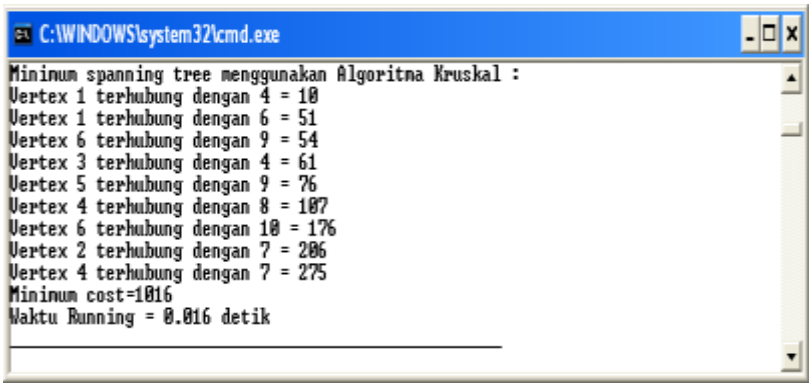


Gambar 1 *User interface* program Algoritma Prim.

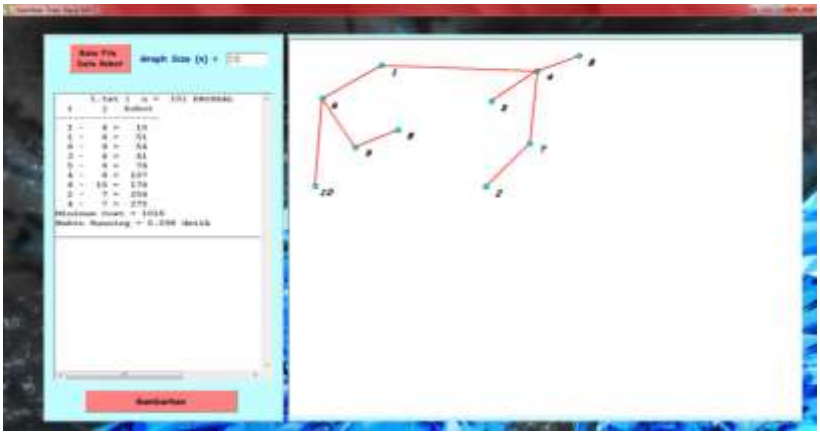


Gambar 2 Gambar *tree* yang terbentuk dari Algoritma Prim

2. User Interface program Algoritma Kruskal.

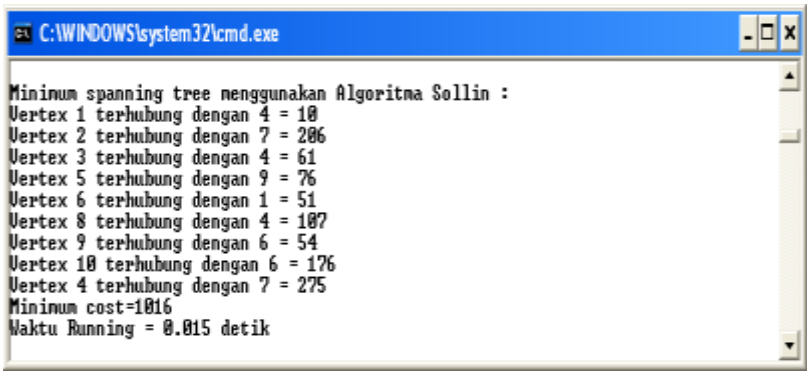


Gambar 3 *User interface* program Algoritma Kruskal.

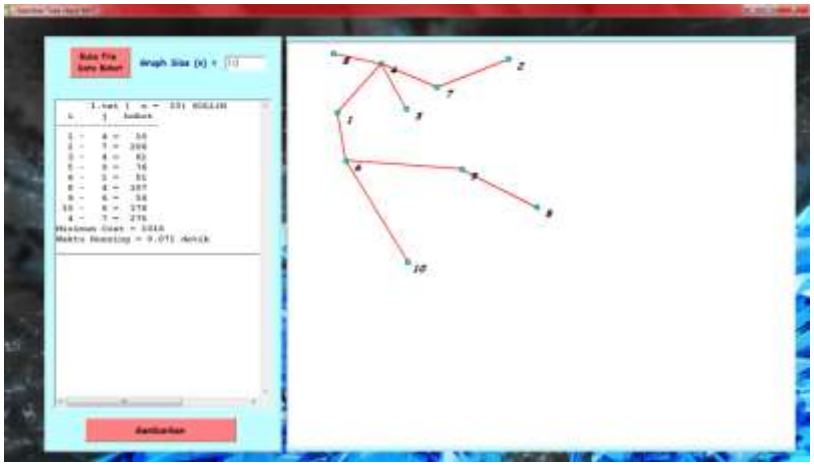


Gambar 4 Gambar *tree* yang terbentuk dari Algoritma Kruskal

3. User Interface program Algoritma Sollin.



Gambar 5 *User interface* program Algoritma Sollin.



Gambar 6 Gambar *tree* yang terbentuk dari Algoritma Sollin

3.4 Perbandingan Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin.

Dari hasil analisis dan pengujian di atas dapat dikatakan bahwa penggunaan Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin mempunyai perbedaan yang mendasar, yaitu:

1. Pada Algoritma Prim langkah pertama yang dilakukan adalah memilih titik awal yang akan dimasukkan kedalam graf T, kemudian dari titik yang telah dipilih tersebut dipilih sisi/*edge* dengan bobot minimum. Sisi yang dimasukkan ke dalam *T* selain berbobot minimum juga harus

bersisian dengan sebuah titik di *T*. Di awal penentuan titik yang dipilih, perlu diketahui bahwa titik manapun yang dipilih, banyak bobot dan bentuk pohon merentang minimum tetaplah sama.

2. Pada Algoritma Kruskal pencarian *minimum spanning tree* didasarkan pada pemilihan sisi yang mempunyai bobot minimum dan sisi tersebut tidak membentuk sirkuit dengan sisi-sisi yang telah dipilih sebagai *tree*.

3. Pada Algoritma Sollin pencarian pohon merentang minimum didasarkan pada pemilihan wakil sisi dari masing-masing titik yang mempunyai bobot minimum, lalu sisi tersebut dilakukan penghapusan jika terdapat duplikat atau sisi yang memiliki titik yang sama didalamnya. Setelah penghapusan sisi duplikat maka sisi yang telah dipilih dihubungkan dengan syarat tidak terjadi sirkuit. Jika saat semua titik telah dihubungkan dan terjadi *forest* maka dilakukan pemilihan titik yang belum terhubung dengan bobot minimum dan tidak terbentuk sirkuit.

Berdasarkan langkah-langkah yang digunakan setiap algoritma dalam menyelesaikan permasalahan *minimum spanning tree* dapat dilihat perbandingan antara Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin, yaitu:
1. Jumlah bobot minimum yang dihasilkan sebagai penyelesaian *minimum spanning tree* untuk setiap graf lengkap pada Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin adalah sama.
2. Banyaknya sisi yang terbentuk setelah diperoleh pohon merentang untuk setiap graf pada algoritma Prim, algoritma Kruskal, dan algoritma Sollin adalah sama.
3. Banyaknya langkah yang ditempuh oleh setiap graf pada algoritma Prim, algoritma Kruskal, dan algoritma Sollin hingga terbentuk pohon merentang minimum adalah berbeda-beda.
4. Waktu yang dibutuhkan oleh masing-masing algoritma untuk menyelesaikan masalah *minimum spanning tree* berbeda-beda.
5. Kompleksitas dari ketiga algoritma tersebut adalah sama yaitu $O(n^2)$.

4 Kesimpulan

Berdasarkan hasil penelitian yang dilakukan maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Algoritma Prim, Algortima Kruskal, dan Algoritma Sollin merupakan algoritma yang dapat digunakan dalam membantu penyelesaian masalah *minimum spanning tree*.
2. Algoritma Prim, Algoritma kruskal, dan Algoritma Sollin memiliki langkah yang berbeda-beda meskipun hasil akhir berupa nilai bobot yang diperoleh dalam mencapai penyelesaian *minimum spanning tree* adalah sama.
3. *Running time* yang dibutuhkan Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin dalam menyelesaikan masalah minimum spanning tree berbeda-beda dan bergantung pada jumlah n yang digunakan.
4. Kompleksitas dari masing-masing algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin sama yaitu $O(n^2)$.
5. Bahasa Pemrograman C++ merupakan bahasa pemrograman yang handal yang dapat digunakan dalam menyelesaikan masalah *minimum spanning tree.*

5 Refferences

[1] Johnsonbaugh, Richard. 1997. *Matematika Diskrit.* Diterjemahkan oleh Didik Djunaedi.Yogyakarta : PT Aditya Media.

[2] Nugraha, Deni Wiria. Penerapan Kompleksitas Waktu Algoritma Prim Untuk Menghitung Kemampuan Komputer Dalam Melaksanakan Perintah. *Jurnal Ilmiah Foristek* Vol. 2, No. 2, September 2012 hal.195-207

[3] Wibisono, Samuel. 2004. *Matematika Diskrit*. Yogyakarta : Penerbit Graha Ilmu.
[4] Munir, R. 2005. *Matematika Diskrit*. Bandung: Informatika.