Journal Profile

Jurnal Ilmiah Matrik

eISSN : 26218089 | pISSN :

Universitas Bina Darma

S5
Sinta Score

10
H-Index

9
H5-Index

707
Citations

629
5 Year Citations

JURNAL ILMIAH MATRIK

DI TERBITKAN OLEH :
DIREKTORAT RISET DAN PENGABDIAN KEPADA MASYARAKAT
UNIVERSITAS BINA DARMA, PALEMBANG
JENDR. A.YANI NO.03
http://journal.binadarma.ac.id
email :jurnalmatrik@binadarma.ac.id | Hp. 081532791703

VOL. 21 NO. 2 AGUSTUS 2019 P-ISSN : 1411 - 1624 E-ISSN : 2621 - 8089 HAL . 84 -155

Vol. 21 No.2, Agustus 2019 **ISSN : 1411-1624**
E-ISSN: 2621-8089

Jurnal Ilmiah MATRIK

Jurnal Ilmiah MATRIK diterbitkan oleh Direktorat Riset dan Pengabdian Kepada Masyarakat (DRPM) Universitas Bina Darma Bekerja sama dengan Fakultas Ilmu Komputer dan Jurnal Ilmiah Terpadu Universitas Bina Darma (JIT-UBD) Publikasi dilakukan secara berkala setiap tahun 3 (tiga) kali (April, Agustus dan Desember). Terbit pertama kali April 1999.

Alamat Redaksi:
Kampus Utama Lantai II Universitas Bina Darma (UBD)
Jalan Ahmad Yani No.3 Palembang
Telp.0711-515679, Fax.0711-515582,
Website: http://journal.binadarma.ac.id/index.php/jurnalmatrik
Email: jurnalmatrik@binadarma.ac.id.

Dicetak di Pusat Penerbitan dan Percetakan Universitas Bina Darma Press (PPP-UBD Press).
Isi Diluar Tanggung Jawab Percetakan.

Vol. 21 No.2, Agustus 2019 **ISSN : 1411-1624**
E-ISSN: 2621-8089

Jurnal Ilmiah MATRIK

PENGANTAR REDAKSI

Puji Syukur kehadirat Allah SWT Tuhan Yang Mahas Esa , Jurnal Ilmiah Matrik untuk Edisi Bulan Agustus 2019 volume 21 Nomor 2 telah terbit sesuai dengan jadwal walaupun terdapat beberapa kendala.

Jurnal Ilmiah Matrik untuk edisi ini telah menerima kiriman artikel dengan jumlah yang cukup banyak, tetapi dalam prosesnya telah dipilih beberapa artikel terbaik sesuai dengan hasil *review*. Untuk mempermudah dan mempercepat dalam proses *review* dan penyuntingan, kami mengharapkan kepada penulis untuk selalu mengikuti *template* dan / atau petunjuk Penulisan. Naskah atau Artikel yang dikirimkan tetapi sesuai dengan template maka akan dikembalikan sebelum masuk dalam proses *review*. Edisi terbitan kali ini memuat 8 Artikel.

Penghargaan setinggi-tingginya kami berikan kepada Penulis, Mitra Bestari, Tim editor dan semua Pihak yang terlibat dalam penyusunan serta penerbitan Jurnal Ilmiah Matrik untuk Edisi Volume 21 Nomor 2 Bulan Agustus 2019. Dalam upaya perbaikan dan peningkatan kualitas baik dari isi maupun tampilan jurnal, kami mengharapkan saran dan kritik membangun untuk perbaikan Edisi Berikutnya.

Tim Redaksi

Vol. 21 No.2, Agustus 2019 **ISSN : 1411-1624**
E-ISSN: 2621-8089

Jurnal Ilmiah MATRIK

DAFTAR ISI

ISSN : 1411-1624
e-ISSN: 2621-8089

ALGORITMA "HANCURKAN SEMUA SIKEL" UNTUK MENENTUKAN POHON PERENTANG MINIMUM DARI SUATU GRAF BERBOBOT

Ricky Aditya
Dosen Universitas Sanata Dharma
Paingan, Maguwoharjo, Depok, Sleman, Daerah Istimewa Yogyakarta
Sur-el : rickyaditya@usd.ac.id

Abstract: The minimum spanning tree is one of the applications of graph theory in various fields. There are several algorithms for determining the minimum spanning tree of a weighted graph, such as Kruskal's algorithm and Prim's algorithm. These two algorithms are not really easy to teach to students in general. Therefore in this paper presented an alternative algorithm called the algorithm "Destroy All Sikel", which is more intuitive and easier to understand. Furthermore, there are also examples of implementation and comparison with two other algorithms.

Keywords: *Kruskal's algorithm, Prim's algorithm, weighted graph, minimum spanning tree*

Abstrak : *Pohon perentang minimum merupakan salah satu penerapan dari teori graf dalam berbagai bidang. Ada beberapa algoritma untuk menentukan pohon perentang minimum daru suatu graf berbobot, antara lain algoritma Kruskal dan algoritma Prim. Dua algoritma ini tidak benar-benar mudah untuk diajarkan kepada mahasiswa secara umum. Oleh karena itu dalam makalah ini disajikan suatu algoritma alternatif yang disebut algoritma "hancurkan semua sikel", yang lebih bersifat intuitif dan lebih mudah dipahami. Lebih lanjut, disajikan pula contoh implementasinya beserta perbandingannya dengan dua algoritma lain.*

Kata kunci*: algoritma Kruskal, algoritma Prim, graf berbobot, pohon perentang minimum*

1. PENDAHULUAN

Teori graf merupakan salah satu bidang dalam matematika diskrit yang memiliki sejumlah penerapan dalam kehidupan sehari-hari. Dengan memodelkan suatu masalah dalam bentuk graf, yang terdiri dari kumpulan titik *(vertex)* dan kumpulan rusuk (*edge*) yang menghubungkannya, solusi optimal dari masalah tersebut dapat dicari menggunakan metode dalam teori graf.[1].

Salah satu penerapan dari teori graf adalah pohon perentang minimum. [1]. Pohon perentang minimum dari suatu graf berbobot (graf yang masing-masing rusuknya memiliki suatu nilai "bobot" yang non-negatif) adalah suatu pohon yang menghubungkan semua titik pada graf tersebut dengan total bobot pada masing-masing rusuk dari pohon tersebut seminimum mungkin.

Contoh penerapan pohon Perentang minimum adalah dalam pemasangan sambungan jaringan listrik, dimana dalam model grafnya, lokasi-lokasi yang akan dialiri listrik menjadi titik-titiknya dan jalur kabel yang menghubungkan lokasi-lokasi tersebut menjadi rusuk-rusuknya, dengan bobot dari tiap rusuk adalah panjang kabel atau biaya pemasangan kabel pada rusuk tersebut. Untuk meminimalisir biaya pemasangan kabel, maka haruslah dibuat suatu model jaringan yang menghubungkan semua lokasi dengan total biaya serendah mungkin, yang ekuivalen dengan menentukan

ISSN : 1411-1624
e-ISSN: 2621-8089

pohon perentang minimum dari model graf tersebut.

Ada beberapa algoritma untuk menentukan pohon perentang minimum dari suatu graf berbobot yang umum digunakan, antara lain algoritma Kruskal dan algoritma Prim. [2]. Dua algoritma inilah yang umum diajarkan dalam mata kuliah Matematika Diskrit dan Teori Graf pada jenjang S1. Akan tetapi, berdasarkan pengalaman penulis dalam mengajar, kedua algoritma ini bukan merupakan hal yang mudah untuk dipahami oleh sebagian besar mahasiswa. Oleh karena itu, penulis mencoba untuk memberikan metode alternatif untuk menentukan pohon perentang minimum dari suatu graf berbobot, yang disebut dengan algoritma "hancurkan semua sikel".

Algoritma ini lebih menggunakan pendekatan intuitif, yang kemungkinan besar dapat lebih mudah dipahami oleh mahasiswa. Algoritma tersebut akan dibicarakan lebih lanjut pada bagian berikutnya. Lebih lanjut, akan dibandingkan pula hasil dari algoritma ini dengan hasil menggunakan algoritma Kruskal dan algoritma Prim, beserta kelebihan dan kekurangannya.

2. METODOLOGI PENELITIAN

Metodologi penelitian dalam makalah ini cukup sederhana, yaitu :

1. Merumuskan ide ke dalam bentuk algoritma formal.
2. Mengimplementasikan algoritma pada suatu contoh kasus.
3. Membandingkan hasil algoritma dengan algoritma-algoritma lain yang sudah ada sebelumnya.

Dalam hal ini, algoritma "hancurkan semua sikel" akan dirumuskan secara formal. Pembuktian secara formal tidak diberikan secara rinci mengingat kesahihan dari algoritma ini cukup jelas secara intuitif. Untuk menguatkan hal tersebut, algoritma tersebut diujikan untuk suatu kasus dan dibandingkan hasilnya dengan algoritma lain yang sudah umum digunakan, dalam hal ini algoritma Kruskal dan Prim. Hal ini tentu saja sekadar justifikasi, bukan pembuktian secara matematis.

Dalam menerapkan metodologi penelitian ini, beberapa makalah terkait juga dikaji sebagai bahan perbandingan, antara lain perbandingan berbagai macam algoritma seperti pada publikasi yang berjudul *An Efficient Greedy Minimum Spanning Tree Algorithm Based on Vertex Associative Cycle Detection Method* [3] dan *Negligence Minimum Spanning Tree Algorithm* [4], yang juga membandingkan dengan algoritma *reverse-delete*, yang cukup mirip dengan algoritma yang dibahas dalam makalah ini. Selain itu penelitian lain yang berkaitan dengan penelitian ini berjudul *An Empirical Analysis of Algorithms for Constructing a Minimum Spanning Tree* yang membahas tentang perbandingan kecepatan komputasi algoritma .[5].

3. HASIL DAN PEMBAHASAN

Dalam bagian pembahasan ini, pertama akan ditinjau beberapa konsep dasar dalam teori graf. Selanjutnya, algoritma "hancurkan semua sikel" akan dijelaskan secara rinci beserta contoh

ISSN : 1411-1624
e-ISSN: 2621-8089

implementasinya. Sebagai pembanding, ditunjukkan pula bagaimana perbandingan hasilnya dengan algoritma-algoritma lain, seperti algoritma Kruskal dan Prim.

3.1. Beberapa Pengertian Dasar dalam Teori Graf

Dalam sub-bagian ini akan dijelaskan beberapa pengertian dan konsep dasar dalam teori graf, antara lain walk, sikel, keterhubungan dan pohon.

Definisi 3.1. Diberikan suatu graf *G* dengan himpunan titik *V* dan himpunan rusuk *E*.

a. Suatu *walk* dari titik *v* ke titik *w* ($v, w \in V$) didefinisikan sebagai barisan selang-seling titik dan rusuk $v_0 e_1 v_1 e_2 \ldots v_{n-1} e_n v_n$, dengan $v_0, v_1, \ldots, v_n \in V$, $e_1, e_2, \ldots, e_n \in E$, $v_0 = v$, $v_n = w$ dan setiap rusuk e_i menghubungkan titik v_{i-1} dan v_i.
b. Suatu *walk* dikatakan tertutup jika titik awal dan titik akhirnya sama. *Walk* tertutup yang tidak melalui rusuk yang sama lebih dari satu kali disebut sikel.
c. Graf *G* dikatakan terhubung jika untuk sebarang dua titik $v, w \in V$, selalu terdapat *walk* dari *v* ke *w*.
d. Suatu graf terhubung disebut pohon jika graf tersebut tidak memuat sikel.

Penyebutan istilah "pohon" dalam teori graf analog dengan "pohon keluarga" (*family tree*) yang biasa digunakan untuk menggambarkan silsilah suatu keluarga. Dengan suatu modifikasi, suatu graf pohon dapat dinyatakan dalam bentuk *family tree*. Ketiadaan sikel dalam graf pohon juga sesuai dengan pohon dalam konteks biologi, yang mana normalnya tidak ada dua cabang/ranting berbeda dari pohon yang terhubung secara langsung sehingga membentuk "sikel ranting". Tentu saja tidak semua graf merupakan pohon, tetapi setiap graf terhubung akan memiliki graf bagian (subgraf) yang merupakan pohon, yang kemudian memunculkan konsep pohon perentang.

Definisi 3.2. Diberikan suatu graf terhubung *G*. Subgraf *T* dari *G* dikatakan pohon perentang dari *G* jika *T* merupakan pohon dan memuat semua titik pada graf *G*. Lebih lanjut, jika *G* merupakan graf berbobot, maka suatu pohon perentang *T* dari *G* disebut pohon perentang minimum jika total bobot dari semua rusuk pada *T* adalah yang terkecil dari semua kemungkinan pohon perentang dari *G*.

Konsep pohon perentang minimum dari suatu graf berbobot memiliki banyak penerapan, seperti telah dijelaskan pada bagian pendahuluan. Untuk menentukan pohon perentang minimum, diperlukan suatu karakterisasi dari graf pohon, seperti yang diberikan dalam teorema berikut.

Teorema 3.1. Jika *T* adalah sebuah graf dengan *n* titik, maka pernyataan-pernyataan berikut ekuivalen :

(i) *T* adalah sebuah pohon.
(ii) *T* tidak memuat sikel dan memiliki *n*-1 rusuk.
(iii) *T* terhubung dan memiliki *n*-1 rusuk.

Karakterisasi pohon dalam Teorema 2.1 berperan penting dalam memudahkan

ISSN : 1411-1624
e-ISSN: 2621-8089

pengkonstruksian pohon perentang minimum. Cukup dengan mengkonstruksikan suatu subgraf terhubung tanpa sikel dengan rusuk sebanyak *n*-1, maka akan terbentuk suatu pohon perentang dari graf tersebut. Yang menjadi masalah berikutnya adalah bagaimana menentukan pohon perentang minimum, yaitu yang total bobot dari semua rusuknya seminimum mungkin. Tentu saja tidak efisien jika total bobot dari semua kemungkinan pohon perentang dihitung satu per satu, lalu dicari yang paling kecil. Pada sub-bagian berikutnya akan dibahas suatu algoritma untuk menentukan pohon perentang minimum dari suatu graf berbobot.

3.2. Algoritma "Hancurkan Semua Sikel"

Ide dari metode untuk menentukan pohon perentang minimum ini sebenarnya sederhana, yaitu menggunakan definisi dari pohon : "graf terhubung sederhana tanpa sikel". Pohon perentang dari suatu graf terhubung merupakan subgraf yang memuat semua titik tetapi tidak memuat satu sikel pun. Dengan kata lain, untuk mendapatkan pohon perentang dari suatu graf terhubung, maka semua sikel harus "dihancurkan".

Untuk "menghancurkan" suatu sikel, tentunya tidak perlu menghapus terlalu banyak rusuk, cukup satu rusuk saja yang dihilangkan untuk membuat sikel tersebut hanya menjadi lintasan biasa. Jika setiap rusuk memiliki bobot dan yang dicari adalah pohon perentang dengan total bobot minimum, maka secara intuitif rusuk yang harus dihilangkan tentu saja yang memiliki bobot terbesar. Jika hal ini dilakukan terus menerus hingga semua sikel "hancur", maka akan diperoleh adalah suatu pohon perentang yang total bobotnya minimum.

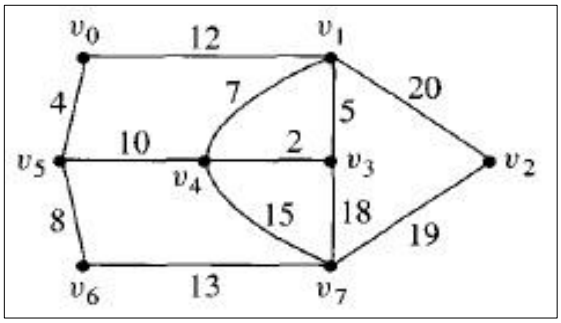
Secara garis besar, algoritma tersebut dapat ditulis sebagai berikut :

1. Input : Graf berbobot terhubung G.
2. Pilih sebarang sikel dalam graf G.
3. Pada sikel di langkah 2, cari rusuk dengan bobot terbesar, lalu hapus rusuk tersebut dari graf G.
4. Periksa apakah graf G masih memiliki sikel. Jika ya, ulangi langkah 2 dan 3. Jika tidak, pohon perentang minimum telah didapatkan.
5. Output : Pohon perentang minimum dari graf G.

Berbeda dengan algoritma Kruskal dan algoritma Prim, algoritma "hancurkan semua sikel" ini tidak memerlukan pengurutan bobot. Pemilihan sikel mana yang "dihancurkan" lebih dulu dapat dilakukan secara acak, selama rusuk yang dihapus adalah yang bobotnya paling besar. Mengingat pohon dengan *n* titik akan memiliki *n*-1 rusuk, maka untuk graf yang memiliki *m* rusuk, akan diperlukan *m*-*n*+1 iterasi.

3.3. Contoh Implementasi Algoritma

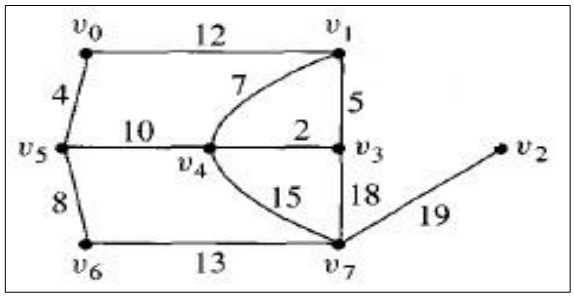
Algoritma pada bagian 3.2 akan diimplementasikan untuk menentukan pohon perentang minimum dari graf berbobot berikut :



Gambar 2. Contoh Graf Berbobot

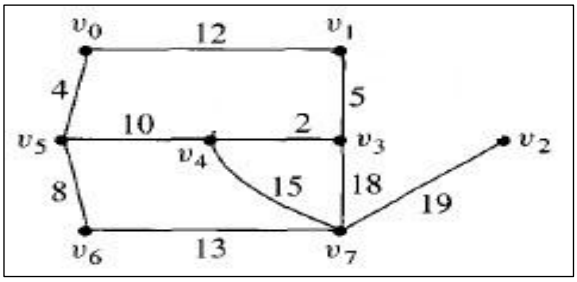
ISSN : 1411-1624
e-ISSN: 2621-8089

Iterasi 1 : Algoritma ini dapat dimulai dari sebarang sikel manapun. Pertama, ditinjau sikel yang merupakan "sisi luar" dari graf tersebut ($v_0 \to v_1 \to v_2 \to v_7 \to v_6 \to v_5 \to v_0$). Terlihat bahwa rusuk dengan bobot terbesar pada sikel ini adalah rusuk yang menghubungkan v_1 dan v_2 dengan bobot 20. Rusuk inilah yang kemudian dieliminasi sehingga graf tersebut menjadi :



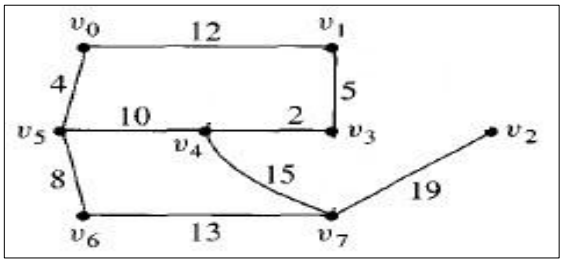
Gambar 3. Hasil setelah iterasi pertama

Iterasi 2 : Selanjutnya ditinjau sikel $v_1 \to v_3 \to v_4 \to v_1$. Rusuk yang dieliminasi adalah rusuk yang menghubungkan v_1 dan v_4 (nilai bobot 7), sehingga diperoleh :



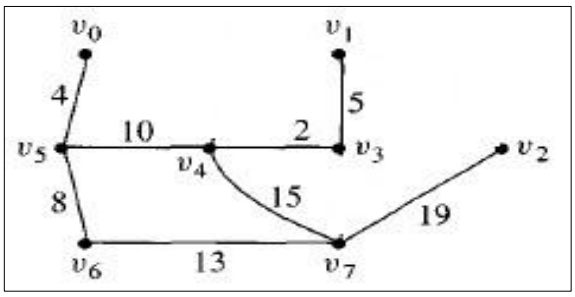
Gambar 4. Hasil setelah iterasi kedua

Iterasi 3 : Ditinjau sikel $v_3 \to v_4 \to v_7 \to v_3$, eliminasi rusuk dengan bobot 18 :



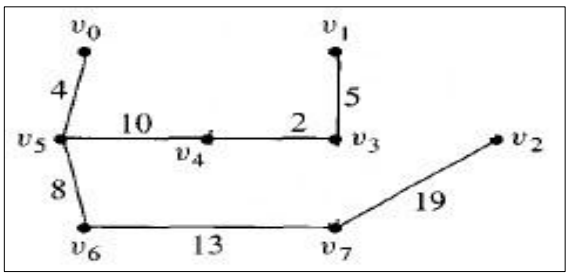
Gambar 5. Hasil setelah iterasi ketiga

Iterasi 4 : Ditinjau sikel $v_0 \to v_1 \to v_3 \to v_4 \to v_5 \to v_0$, eliminasi rusuk dengan bobot 12 :



Gambar 6. Hasil setelah iterasi keempat

Iterasi 5 : Ditinjau sikel $v_4 \to v_5 \to v_6 \to v_7 \to v_4$, eliminasi rusuk dengan bobot 15 :



Gambar 7. Hasil setelah iterasi kelima (Pohon Perentang Minimum)

Setelah lima iterasi diperoleh suatu subgraf yang tidak lagi memiliki sikel, yang berarti sudah merupakan pohon perentang. Subgraf pada Gambar 7 itulah yang merupakan pohon perentang minimum pada Gambar 2 dengan total bobot 61.

3.4. Perbandingan dengan Algoritma Kruskal dan Prim

Dua algoritma yang lebih umum dipakai, yakni algoritma Kruskal dan algoritma Prim, memiliki pendekatan yang berbeda dengan algoritma "hancurkan semua sikel". Misal suatu graf berbobot terhubung G memiliki n titik,

ISSN : 1411-1624
e-ISSN: 2621-8089

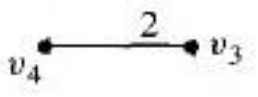
maka akan dicari *n*-1 rusuk yang dapat membentuk pohon perentang minimum. Dalam algoritma Kruskal, yang dijelaskan dalam [2], mula-mula semua rusuk diurutkan menurut bobotnya, dari terkecil ke terbesar. Selanjutnya, rusuk-rusuk tersebut dirangkai sesuai urutan bobotnya, dimulai dari rusuk dengan bobot terkecil. Rusuk dengan bobot terkecil berikutnya ditambahkan satu per satu, kecuali jika penambahan rusuk tersebut akan membentuk sikel. Demikian seterusnya hingga *n*-1 iterasi dan diperolehlah pohon perentang minimum.

Di sisi lain, algoritma Prim, yang dijelaskan dalam [2], juga menggunakan pengurutan bobot. Hanya saja, algoritma Prim lebih berorientasi ke mencakup semua titik pada suatu graf. Algoritma ini dapat dimulai dari sebarang titik pada graf tersebut, lalu dipilih rusuk dengan bobot terkecil yang terhubung dengan titik tersebut. Rusuk tersebut menghubungkan titik awal dengan suatu titik lain. Selanjutnya, dipilih rusuk dengan bobot terkecil yang terhubung dengan dua titik tersebut dan terhubunglah satu titik yang lain lagi, tentunya yang tidak membentuk sikel. Proses ini diteruskan hingga semua titik tercakup.

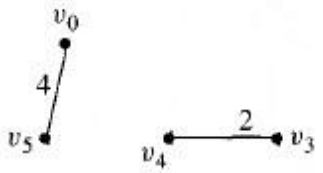
Kedua algoritma tersebut dapat diterapkan untuk menentukan pohon perentang minimum untuk graf pada Gambar 2. Hasilnya adalah sebagai berikut :

Algoritma Kruskal

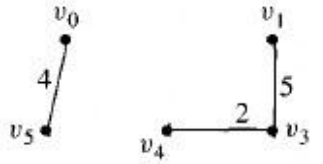
Iterasi 1 :



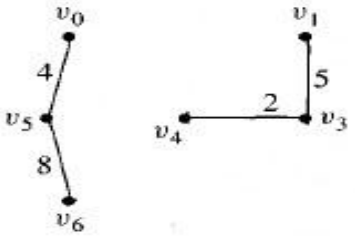
Iterasi 2 :



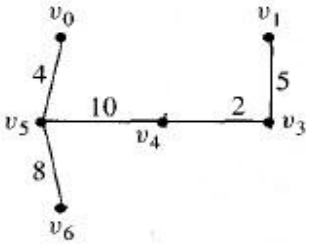
Iterasi 3 :



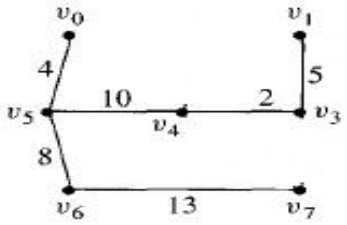
Iterasi 4 :



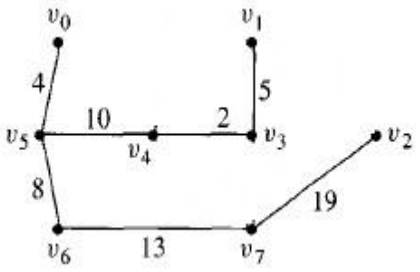
Iterasi 5 :



Iterasi 6 :



Iterasi 7 :

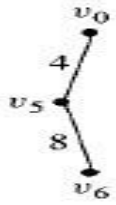


ISSN : 1411-1624
e-ISSN: 2621-8089

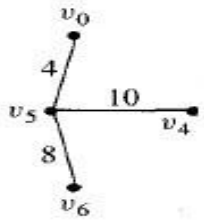
Algoritma Prim (Titik v_0 sebagai Awalan)

Iterasi 1 :

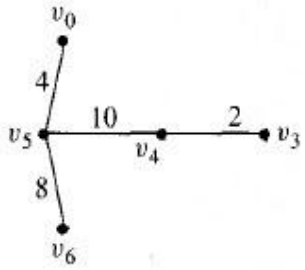
Iterasi 2 :



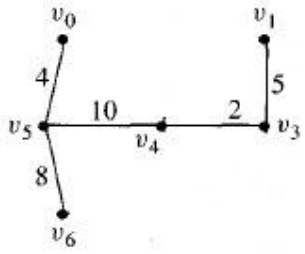
Iterasi 3 :



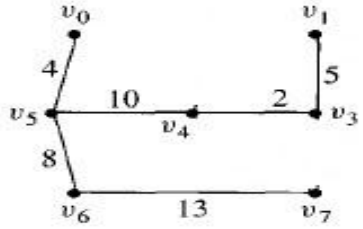
Iterasi 4 :



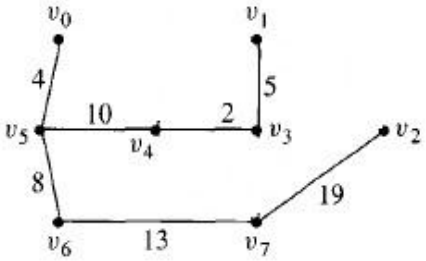
Iterasi 5 :



Iterasi 6 :



Iterasi 7 :



Terlihat bahwa hasil dari kedua algoritma ini akan sama dengan algoritma "hancurkan semua sikel", seperti pada Gambar 7. Hal ini tentunya bukan suatu kebetulan, karena berdasarkan idenya, secara intuitif memang akan diperoleh pohon perentang dengan bobot minimum.

4. KESIMPULAN

Berdasarkan pembahasan pada bagian sebelumnya, dapat diperoleh sejumlah simpulan,bahwa Algoritma "hancurkan semua sikel" merupakan suatu algoritma intuitif untuk menentukan pohon perentang minimum dari suatu graf berbobot dengan prinsip sederhana, yaitu menghapus rusuk dengan bobot terbesar dari setiap sikel yang ada. Algoritma "hancurkan semua sikel" akan mendapatkan hasil pohon perentang minimum yang sama (atau setidaknya ekuivalen secara bobot) dengan algoritma Kruskal dan algoritma Prim. Berdasarkan pengalaman penulis, algoritma ini juga lebih mudah untuk dipahami oleh mahasiswa secara umum.

Kajian yang dibahas dalam Penelitian ini masih belum menyentuh analisis efisiensi dari algoritma "hancurkan semua sikel". Untuk graf

ISSN : 1411-1624
e-ISSN: 2621-8089

dengan banyak titik dan rusuk yang relatif kecil, mungkin algoritma ini lebih efisien, tetapi belum diukur efisiensinya untuk graf dengan banyak titik dan rusuk yang cukup besar. Hal ini dapat menjadi masukan untuk penelitian yang lebih lanjut.

UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih kepada salah satu teman penulis, Nugroho Seto Saputra, yang memberikan ide dasar dari makalah ini.

DAFTAR PUSTAKA

[1] R. J. Wilson, *Pengantar Teori Graf*, Edisi Kelima, Jakarta : Penerbit Erlangga, 2010.
[2] S. S. Epp, *Discrete Mathematics with Applications*, 4th ed. Belmont : Brooks/Cole Publishing Co.
[3] P. Biswas, M. Goel, H. Negi, and M. Datta, "An Efficient Greedy Minimum Spanning Tree Algorithm based on Vertex Associative Cycle Detection Method," *Procedia Computer Science*, vol. 92, pp. 513-519, 2016.
[4] J. H. S. Alkhalissi, A. Sayli, "Negligence Minimum Spanning Tree Algorithm," *European Journal of Science and Technology*, vol. 14, pp. 70-76, 2018.
[5] B. M. E. Moret and H. D. Shapiro, "An Empirical Analysis of Algorithms for Constructing a Minimum Spanning Tree," in *2nd Workshop on Algorithm and Data Structures (WADS)*, 1991, pp. 400-411.