KEEFEKTIFAN PENGGUNAAN ALGORITMA BORUVKA, ALGORITMA PRIM, ALGORITMA KRUSKAL, DAN ALGORITMA SOLLIN DALAM MENENTUKAN POHON MERENTANG MINIMUM

SKRIPSI

Oleh:
MUFIDATUL KHOIROH
NIM.06510037

JURUSAN MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI (UIN)MAULANA MALIK IBRAHIM MALANG
2010

KEEFEKTIFAN PENGGUNAAN ALGORITMA BORUVKA, ALGORITMA PRIM, ALGORITMA KRUSKAL, DAN ALGORITMA SOLLIN DALAM MENENTUKAN POHON MERENTANG MINIMUM

SKRIPSI

Diajukan Kepada:

Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Sains (S.Si)

Oleh:
MUFIDATUL KHOIROH
NIM.06510037

JURUSAN MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI (UIN)MAULANA MALIK IBRAHIM MALANG
2010

KEEFEKTIFAN PENGGUNAAN ALGORITMA BORUVKA, ALGORITMA PRIM, ALGORITMA KRUSKAL, DAN ALGORITMA SOLLIN DALAM MENENTUKAN POHON MERENTANG MINIMUM

SKRIPSI

Oleh:
MUFIDATUL KHOIROH
NIM.06510037

Telah Diperiksa dan Disetujui untuk Diuji

Tanggal: 23 Juni 2010

Dosen Pembimbing I,	Dosen Pembimbing II,
Abdussakir, M.Pd	**Dr. Munirul Abidin, M.Ag**
NIP.19751006 200312 1 001	**NIP. 19720420 200212 1 003**

Mengetahui,
Ketua Jurusan Matematika

Abdussakir, M.Pd
NIP. 19751006 200312 1 001

KEEFEKTIFAN PENGGUNAAN ALGORITMA BORUVKA, ALGORITMA PRIM, ALGORITMA KRUSKAL, DAN ALGORITMA SOLLIN DALAM MENENTUKAN POHON MERENTANG MINIMUM

SKRIPSI

Oleh:
MUFIDATUL KHOIROH
NIM.06510037

Telah Dipertahankan di Depan Dewan Penguji Skripsi dan Dinyatakan Diterima sebagai Salah Satu Persyaratan untuk Memperoleh Gelar Sarjana Sains (S.Si)

Tanggal: 21 Juli 2010

Susunan Dewan Penguji		Tanda Tangan
1. Penguji Utama	: Wahyu Henky Irawan, M.Pd NIP. 19710420 200003 1 003	( )
2. Ketua	: Abdul Aziz, M.Si NIP. 19760318 200604 1 002	( )
3. Sekretaris	: Abdussakir, M.Pd NIP. 19751006 200312 1 001	( )
4. Anggota	: Dr. Munirul Abidin, M.Ag NIP. 19720420 200212 1 003	( )

Mengetahui dan Mengesahkan,
Ketua Jurusan Matematika

Abdussakir, M.Pd
NIP. 19751006 200312 1 001

SURAT PERNYATAAN
ORISINALITAS SKRIPSI

Saya yang bertanda tangan dibawah ini:

Nama : MUFIDATUL KHOIROH

NIM : 06510037

Jurusan : Matematika

Fakultas : Sains dan Teknologi

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan hasil tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 23 Juni 2010

Yang membuat pernyataan

Mufidatul Khoiroh

NIM.06510037

MOTTO

"Jika Allah menolong kamu, maka tidak ada yang dapat mengalahkan kamu dan jika Allah mau mengalahkan kamu, maka siapakah yang akan menolong kamu selain daripada Allah? Dan kepada Allah-lah hendaknya orang-orang mu'min itu bertawakkal"

(QS: Al Imran:160)

Belajarlah untuk menghayati hal-hal yang menakutkan, niscaya rasa takut itu akan sirna.

Orang berakal tidak akan bosan untuk meraih manfaat berpikir, tidak putus asa dalam menghadapi keadaan, dan tidak akan pernah berhenti dari berpikir dan berusaha.

PENULIS PERSEMBAHKAN

Dengan iringan doa dan rasa syukur yang teramat besar

Karya tulis ini penulis persembahkan kepada

Ayah dan ibu tercinta:

Bapak Huda dan Ibu Munadhifah yang tanpa lelah memberikan dorongan moril, spirituil, finansial dan tak henti-hentinya mencurahkan kasih sayangnya.

dan,

Adik-adik tersayang:

Dik Rosyid, Dik Haris, dan Dik Nia

yang selalu memberikan dukungan moril dan spirituil.

Kejarlah cita-cita kalian sampai kalian meraihnya..!!

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Puji syukur Kehadirat Allah SWT yang telah melimpahkan rahmad, taufiq dan hidayahNya sehingga penulis dapat menyelesaikan penulisan skripsi yang berjudul "*Keefektifan Penggunaan Algoritma Boruvka, Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin dalam Menentukan Pohon Merentang Minimum* " ini dengan baik. Shalawat serta salam semoga senantiasa tercurahkan kepada junjungan Nabi besar Muhammad Saw yang telah membimbing dan memberikan jalan terang.

Penulis menyadari bahwa penulisan skripsi ini tidak akan terselesaikan dengan baik tanpa adanya saran, arahan, bimbingan serta do'a dan bantuan dari semua pihak. Oleh karena itu patutlah penulis haturkan ucapan terima kasih yang sebesar-besarnya, terutama kepada:

1. Prof. Dr. H. Imam Suprayogo, selaku Rektor Universitas Islam Negeri (UIN) Maliki Malang .
2. Prof. Drs. Sutiman Bambang Sumitro, SU, D.Sc, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri (UIN) Maliki Malang.
3. Abdussakir, M.Pd, selaku Ketua Jurusan Matematika Fakultas Sains dan Teknologi Universitas Islam Negeri (UIN) Maliki Malang dan juga selaku dosen pembimbing, yang telah meluangkan waktunya untuk memberikan pengarahan selama penulisan skripsi ini.
4. Dr. Munirul Abidin, M.Ag, selaku dosen pembimbing keagamaan, yang telah memberikan saran dan bantuan selama penulisan skripsi ini.
5. Seluruh Dosen Fakultas Sains dan Teknologi UIN Maliki Malang yang telah memberikan ilmu pengetahuan kepada penulis selama di bangku kuliah, serta seluruh karyawan dan staf UIN Maliki Malang.
6. Bapak dan Ibu tercinta, yang selalu memberikan semangat dan motivasi baik moril maupun spirituil dan perjuangannya yang tak pernah kenal lelah dalam mendidik dan membimbing penulis hingga penulis sukses dalam meraih cita-

cita serta ketulusan do'anya kepada penulis sampai dapat menyelesaikan skripsi ini.

7. Adik-adik tersayang, yang selalu memberikan bantuan, semangat dan do'a selama kuliah serta dalam menyelesaikan skripsi ini.
8. Teman-teman Matematika angkatan 2006, terima kasih atas doa serta kenangan yang kalian berikan.
9. Semua pihak yang tidak mungkin penulis sebut satu persatu, atas keikhlasan bantuan moril dan sprituil penulis ucapkan terima kasih.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karena itu, penulis mengharapkan saran dan kritik yang bersifat membangun. Akhir kata, semoga skripsi ini dapat memberikan manfaat dan menambah wawasan keilmuan khususnya matematika. Amin.

Malang, 23 Juni 2010

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Khoiroh, Mufidatul. 2010. **Keefektifan Penggunaan Algoritma Boruvka, Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin dalam Menentukan Pohon Merentang Minimum.** Skripsi, Jurusan Matematika Fakultas Sains dan Teknologi Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang. Pembimbing: (I) Abdussakir, M.Pd. (II) Dr.Munirul Abidin, M.Ag.

Kata Kunci: Teori Graf, Pohon Merentang Minimum, Boruvka, Prim, Kruskal, Sollin

Teori graf merupakan cabang matematika sekaligus pokok bahasan yang memiliki banyak terapan saat ini. Graf adalah satu alat yang digunakan untuk untuk mencari solusi dari permasalahan diskret yang ditemui dalam dunia nyata. Pada skripsi ini menghadirkan graf dengan konsep pohon untuk memecahkan masalah yaitu mencari algoritma yang paling efektif dari algoritma Boruvka, algoritma Prim, algoritma Kruskal, dan algoritma Sollin. Sedangkan tujuan penulisan skripsi ini adalah menentukan algoritma manakah yang paling efektif digunakan dalam menentukan pohon merentang minimum.

Algoritma Boruvka, algoritma Prim, algoritma Kruskal, dan algoritma Sollin adalah algoritma yang digunakan dalam membangun pohon merentang minimum. Bagaimanakah langkah-langkah keempat algoritma ini dalam menentukan pohon merentang minimum dari graf yang disajikan di dalam skripsi ini dan bagaimana perbandingan antara keempatnya. Data yang digunakan adalah graf berbobot yang pada skripsi ini disajikan 4 graf berbobot yang setiap graf berbobot memuat 8 titik dengan jumlah sisi yang berbeda. Maka dalam skripsi ini jenis penelitian yang digunakan adalah studi kepustakaan (*Library Research*).

Hasil penelitian ini merupakan pendeskripsian langkah-langkah dalam menentuka pohon merentang minimum dengan menggunakan empat algoritma. Setelah itu dilanjutkan dengan analisis perbandingan dari empat algoritma tersebut. Hasil penelitian menunjukkan bahwa bentuk pohon merentang dan jumlah bobot merentangnya mempunyai kesamaan untuk setiap graf berbobot tersebut. Yang membedakan antara algoritma Boruvka, algoritma Prim, algoritma Kruskal, dan algoritma Sollin adalah algoritmanya berbeda-beda sehingga jumlah langkah yang digunakan keempat algoritma adalah berbeda-beda. Untuk graf G dengan jumlah sisi = $2(p - 1)$ algoritma Sollin paling efektif dan efisien dibandingkan algoritma Boruvka, algoritma Prim, dan algoritma Kuskal. Untuk graf G dengan jumlah sisi = $2(p - 1)$ namun terdapat sisi yang memiliki bobot yang sama algoritma Prim dan algoritma Sollin paling efektif dan efisien dibandingkan algoritma Boruvka, dan algoritma Kuskal. Untuk graf G dengan jumlah sisi < $2(p - 1)$ algoritma Sollin paling efektif dan efisien dibandingkan algoritma Boruvka, algoritma Prim, dan algoritma Kuskal. Untuk graf G dengan jumlah sisi > $2(p - 1)$ algoritma Kruskal paling efektif dan efisien dibandingkan algoritma Boruvka, algoritma Prim, dan algoritma Sollin. Pembahasan mengenai pohon merentang minimum ini masih dapat dilanjutkan untuk penelitian pohon merentang minimum pada jenis graf yang lain.

ABSTRACT

Khoiroh, Mufidatul. 2010. **Usage Effectiveness of Boruvka Algorithm, Prim Algoritma, Kruskal Algorithm, and Sollin Algoritma in Determining Minimum Spanning Tree**. Thesis, Mathematic Department Sains and Technology Faculty Islamic State University (UIN) Maulana Malik Ibrahim Malang. Counsellor: (I) Abdussakir, M.Pd. (II) Dr. Munirul Abidin, M.Ag.

Keywords: Graph Theory, Minimum Spanning Tree, Boruvka, Prim, Kruskal, Sollin

Graph theory is a branch of mathematics and a main topic which has many applications at the moments. Graph is one equipment applied for finding for solution from problems of discrete met in the real world. At this thesis presents graph with tree concept to solve problem that is looking for the most effective algorithm between Boruvka algorithm, Prim algorithm, Kruskal algorithm, and Sollin algorithm. While purpose of this thesis is determining which algorithm of is the most effective applied in determining.

Boruvka Algorithm, Prim algorithm, Kruskal algorithm, and Sollin algorithm are algorithms applied in building minimum spanning tree. How the steps of the four algorithms in determining minimum spanning tree from graph presented in this thesis and how comparison between them. Data applied is weighted graph which at this thesis presented 4 weighted graphs which every weighted graph loads 8 points with different sides number. Hence the research type of this thesis applied is bibliography study (Library Research).

The result of this research is description of steps in determining minimum spanning tree by using four algorithms. Then is continued with comparative analysis out of the four algorithms. The result of this research indicates that the form of spanning tree and the number of weight spanning it is having equality for every weighted graph. What differentiates between Boruvka algorithm, Prim algorithm, Kruskal algorithm, and Sollin algorithm are different so algorithm the that number of steps applied by fourth of algorithms are different. For graph G with number of sides = 2(p - 1), algoritma Sollin is the most efficient and effectively compared to Boruvka algoritma, Prim algorithm, and Kruskal algoritma. For graph G with number of sides = 2(p - 1) but there is side having the same weight, Prim algoritma and Sollin algoritma are the most efficient and effectively compared to Boruvka algoritma, and Kruskal algoritma. For graph G with number of sides < 2(p - 1), Sollin algoritma is more efficient and effectively compared to Boruvka algorithm, Prim algorithm, and Kruskal algorithm. For graph G with number of sides > 2(p - 1), algorithm Kruskal is the most efficient and effectively compared to Boruvka algorithm, Prim algorithm, and Sollin algorithm. Discussion about the minimum spanning tree admits of continued for research of minimum spanning tree other graphs type.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Alam semesta memuat bentuk-bentuk dan konsep matematika, meskipun alam semesta tercipta sebelum matematika itu ada. Alam semesta serta segala isinya diciptakan oleh Allah dengan ukuran-ukuran yang cermat dan teliti, dengan perhitungan-perhitungan yang mapan, dan dengan rumus-rumus serta persamaan yang seimbang dan rapi (Abdusysyakir,2007:79-80). Maka tidak diragukan lagi bahwa Al-Quran merupakan peletak dasar kemajuan ilmu pengetahuan dan teknologi bagi umat Islam. Allah berfirman dalam surat Al-Qamar ayat 49 sebagai berikut:

إِنَّا كُلَّ شَيْءٍ خَلَقْنَٰهُ بِقَدَرٍ ﴿٤٩﴾

Artinya : Sesungguhnya Kami menciptakan segala sesuatu menurut ukuran.(Q.S. Al-Qamar : 49)

Perkembangan ilmu pengetahuan dan teknologi yang sangat pesat, tidak lepas dari peran ilmu matematika, suatu ilmu yang merupakan ilmu bantu dalam menyelesaikan berbagai permasalahan yang terjadi di dalam kehidupan di dunia. Ilmu matematika ini merupakan alat untuk menyederhanakan penyajian dan pemahaman masalah. Karena dalam bahasan matematika, suatu masalah dapat menjadi lebih sederhana untuk disajikan, dipahami, dianalisis, dan dipecahkan. Untuk keperluan tersebut, maka pertama dicari pokok masalahnya, kemudian

dibuat rumusan atau model matematikanya, sehingga masalah lebih mudah dipecahkan (Purwanto, 1998:1). Matematika mempunyai sifat yang fleksibel, dalam arti dapat dimanfaatkan dan diaplikasikan dalam berbagai cabang disiplin ilmu lainnya.

Dewasa ini semakin banyak muncul penggunaan model matematika maupun penalaran matematika sebagai alat bantu dalam menyelesaikan permasalahan yang dihadapi dalam berbagai disiplin ilmu. Teori graf merupakan salah satu cabang matematika yang penting dan banyak manfaatnya karena teori-teorinya dapat diterapkan untuk memecahkan masalah dalam kehidupan sehari-hari. Dengan mengkaji dan menganalisis model atau rumusan teori graf dapat diperlihatkan peranan dan kegunaannya dalam memecahkan permasalahan. Permasalahan yang dirumuskan dengan teori graf dibuat sederhana, yaitu diambil aspek-aspek yang diperlukan dan dibuang aspek-aspek lainnya (Purwanto, 1998:1).

Dalam kehidupan sehari-hari terdapat permasalahan mengenai optimasi yang dapat diselesaikan menggunakan pohon merentang minimum, misalnya masalah mencari jarak terpendek, biaya termurah, dan tenaga seminimal mungkin dalam pembangunan jalan, jaringan telepon selular, maupun jaringan listrik. Terkait dengan pernyataan di atas, maka perlu adanya pemecahan untuk masalah-masalah tersebut. Salah satu teori yang dapat diaplikasikan dalam menyelesaikan permasalahan-permasalahan tersebut adalah dengan penerapan teori graf.

Penyelesaian masalah-masalah tersebut di atas, pada dasarnya menentukan terjadinya semua pohon merentang minimum yang mungkin dan

memperhitungkan pohon merentang minimum. Di dalam sebuah graf mungkin saja terdapat lebih dari satu pohon merentang, harus dicari pohon merentang yang mempunyai jumlah jarak terpendek, dengan kata lain harus dicari pohon merentang minimum. Mencari minimum dari suatu pohon merentang merupakan suatu masalah yang sudah cukup dikenal dalam pokok bahasan graf dan mempunyai terapan yang luas dalam praktek.

Terkait dengan pernyataan di atas, dalam menentukan algoritma yang paling efektif dalam menentukan pohon merentang minimum, penulis menganalogikannya dengan suatu sikap yang harus diambil agar tidak terjadi sikap yang berlebih-lebihan. Allah SWT menganjurkan kepada seluruh umat Islam untuk melakukan segala sesuatu secara efektif dan efisien. Di dalam agama Islam, sangatlah diwajibkan untuk hidup sederhana, dengan kata lain tidak berlebih-lebihan karena Allah SWT melarang melakukan pemborosan dalam segala hal seperti pemborosan waktu, pemborosan harta, dll. Seperti yang telah difirmankan dalam Al-Qur'an surat Al-Israa' ayat 27 yang berbunyi :

إِنَّ ٱلْمُبَذِّرِينَ كَانُوٓاْ إِخْوَٰنَ ٱلشَّيَٰطِينِ ۖ وَكَانَ ٱلشَّيْطَٰنُ لِرَبِّهِۦ كَفُورًا ﴿٢٧﴾

Artinya : Sesungguhnya pemboros-pemboros itu adalah saudara-saudara syaitan dan syaitan itu adalah sangat ingkar kepada Tuhannya.(Q.S. Al- Israa': 27)

Dalam ayat tersebut di atas mengingatkan untuk tidak bersikap berlebih-lebihan, karena itu adalah perbuatan yang dibenci Allah SWT. Jadi analoginya

adalah untuk apa dalam mengupayakan suatu hal kebaikan harus memilih suatu cara atau jalan yang sulit atau panjang, padahal ada suatu cara atau jalan yang cepat, mudah dan baik.

Berdasarkan uraian di atas serta mengingat pentingnya aplikasi graf dalam menentukan pohon merentang minimum, untuk itu diperlukan suatu algoritma yang tepat untuk menentukan pohon merentang minimum dalam suatu graf terhubung, berbobot, dan tidak berarah. Peneliti merasa bahwa penelitian ini merupakan salah satu penelitian yang menarik untuk dikaji, karena terdapat beberapa macam algoritma yang dapat digunakan untuk mencari pohon merentang minimum. Di sini, peneliti meneliti 4 macam algoritma yang dapat digunakan dalam menentukan pohon merentang minimum yaitu algoritma Boruvka, algoritma Prim, algoritma Kruskal, dan algoritma Sollin, yang masing-masing algoritma tersebut memiliki aturan yang berbeda-beda dalam menentukan pohon merentang minimum, sehingga peneliti merasa perlu mengkaji algoritma manakah yang paling efektif dalam menentukan pohon merentang minimum. Oleh karena itu penulis merumuskan judul untuk skripsi ini, yakni *Keefektifan Penggunaan Algoritma Boruvka, Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin dalam Menentukan Pohon Merentang Minimum*.

1.2 Rumusan Masalah

Rumusan masalah dari penulisan skripsi ini adalah diantara keempat algoritma tersebut, algoritma manakah yang paling efektif digunakan dalam menentukan pohon merentang minimum?

1.3 Tujuan

Tujuan penulisan skripsi ini adalah untuk mengetahui algoritma manakah yang paling efektif digunakan dalam menentukan pohon merentang minimum.

1.4 Batasan Masalah

Agar penulisan skripsi ini tidak meluas, penulis terlebih dahulu akan menegaskan bahwa maksud dari keefektifan disini adalah mengenai banyak kemungkinan pohon merentang yang dapat dihasilkan, serta kecepatan dalam memperoleh pohon merentangnya. Selanjutnya penulis mengaskan bahwa graf yang digunakan untuk mencari pohon merentang minimum antara lain:

1. Graf yang memuat titik yang banyaknya 8 dan sisi yang banyaknya 14.
2. Graf yang memuat titik yang banyaknya 8 dan sisi yang banyaknya tetap 14 namun terdapat sisi yang memilki bobot sama.
3. Graf yang memuat titik yang banyaknya 8 dan sisi yang banyaknya < 14 yakni 11.
4. Graf yang memuat titik yang banyaknya 8 dan sisi yang banyaknya > 14 yakni 20.

Dengan alasan masing-masing dari 4 algoritma tersebut memiliki cara yang berbeda-beda dalam mencari pohon merentang minimum, sehingga tidak menutup kemungkinan setiap graf berbobot yang disajikan maka algoritma pencari pohon merentang minimum yang paling efektif akan berbeda-beda.

1.5 Manfaat Penelitian

1. Bagi Penulis

   Penelitian ini digunakan sebagai tambahan informasi dan wawasan pengetahuan tentang teori graf, khususnya tentang pohon merentang minimum, algoritma Boruvka, algoritma Prim, algoritma Kruskal, dan algortma Sollin.

2. Bagi Lembaga

   Hasil penelitian ini dapat digunakan untuk bahan kepustakaan yang dijadikan sarana pengembangan wawasan keilmuan khususnya di jurusan matematika untuk mata kuliah teori graf.

3. Bagi Pengembangan Ilmu

   Hasil penelitian ini dapat digunakan untuk bahan pembanding bagi pihak yang ingin mengetahui lebih banyak tentang pohon merentang minimum.

1.6 Metode Penelitian

Jenis dari penelitian ini adalah deskriptif kualitatif. Pendekatan yang digunakan adalah pendekatan kualitatif dengan metode kepustakaan. Dalam pendekatan deskriptif kualitatif ini maka penulis menggunakan metode penelitian kepustakaan (*Library Research*). Metode penelitian kepustakaan yaitu penelitian yang dilakukan di dalam perpustakaan untuk mengumpulkan data dan informasi kemudian dilanjutkan dengan menyusun, mengolah, menganalisis, menarik kesimpulan, menafsirkan, dan menguji hipotesis didasarkan dari hasil pengolahan data sehingga diperoleh ringkasan/kesimpulan data. Pengumpulan data dan

informasi tersebut dapat dilakukan dengan bantuan bermacam materi yang terdapat di ruang perpustakaan seperti buku-buku dan dokumen yang ada.

Dalam skripsi ini membahas tentang algoritma Boruvka, algoritma Prim, algoritma Kruskal, dan algoritma Sollin beserta langkah-langkahnya dalam menentukan pohon merentang minimum dalam suatu graf sederhana terhubung, berbobot, dan tidak berarah.

Beberapa langkah yang harus dilakukan untuk menyelesaikan masalah pohon merentang minimum adalah:

1. Penentuan titik-titik dalam pembentukan graf
2. Penentuan bobot dari setiap sisi
3. Penghitungan pohon merentang minimum dari hasil pembentukan graf menggunakan algoritma Boruvka.
4. Penghitungan pohon merentang minimum dari hasil pembentukan graf menggunakan algoritma Prim.
5. Penghitungan pohon merentang minimum dari hasil pembentukan graf menggunakan algoritma Kruskal.
6. Penghitungan pohon merentang minimum dari hasil pembentukan graf menggunakan algoritma Sollin.
7. Perbandingan hasil yang diperoleh dari penghitungan menggunakan algoritma Boruvka, algoritma Prim, algoritma Kruskal, dan algoritma Sollin.

1.7 Sistematika Penulisan

Agar penulisan skripsi ini lebih terarah, mudah ditelaah dan dipahami, maka digunakan sistematika pembahasan yeng terdiri dari empat bab. Masing-masing bab dibagi ke dalam beberapa subbab dengan rumusan sebagai berikut:

BAB I PENDAHULUAN

Pada bab pendahuluan ini meliputi: latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, metode penelitian dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bagian ini terdiri atas konsep-konsep (teori-teori) yang mendukung bagian pembahasan. Konsep-konsep tersebut antara lain membahas tentang pengertian graf, graf terhubung, graf sederhana, serta hal-hal lain yang erat kaitannya dengan bagian yang diuraikan itu.

BAB III PEMBAHASAN

Pembahasan berisi tentang menentukan order minimum pohon merentang minimum dari suatu graf *G* dengan menggunakan algoritma Boruvka, algoritma Prim, algoritma Kruskal, dan algoritma Sollin, yang kemudian menentukan algoriema manakah yang lebih efektif digunakan dalam menentukan pohon merentang minimum, serta Kajian Agama Islam tentang menentukan algoritma yang paling efektif digunakan dalam menentukan pohon merentang minimum.

BAB IV PENUTUP

Pada bab ini dibahas tentang kesimpulan dan saran.

BAB II

KAJIAN TEORI

2.1 Definisi Keefektifan

Kalau menurut definisi, efektif adalah tepat guna atau berhasil guna atau tepat sasaran atau dapat dikatakan suatu kegiatan itu dapat dikatakan tercapai tujuannya. Sementara itu efisien adalah berdaya guna atau mampu memanfaatkan sumber daya yang ada dengan seoptimal mungkin. (Daryanto, 1997:181). Definisi keefektifan lebih ditekankan pada prosesnya sedangkan efisien lebih ditekankan pada waktu.

2.2 Definisi Graf

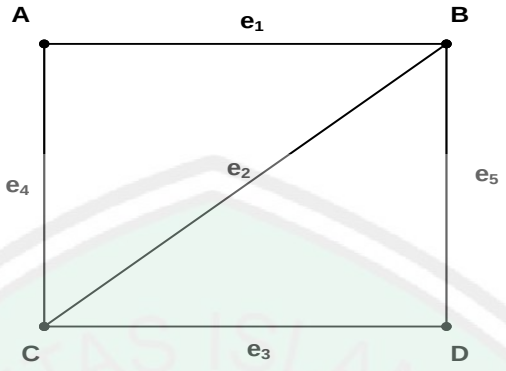
Graf G terdiri dari himpunan tidak kosong dari elemen-elemen yang disebut titik dan himpunan dari elemen-elemen yang disebut sisi. Graf G adalah pasangan himpunan (V(G),E(G)) yang dinotasikan dalam bentuk G={V(G),E(G)} dengan:

V(G) adalah himpunan titik yang tidak kosong yang jumlahnya berhingga, dan

E(G) adalah himpunan sisi yang dapat merupakan himpunan kosong

(Chartrand dan Ortrud, 1993).

Contoh 2.1:



Gambar 2.1 Graf *G*

Gambar 2.1 menunjukkan bahwa graf $G = G(V,E)$, di mana V terdiri dari titik A, B, C, D dan E terdiri dari lima sisi yaitu e_1 = {A, B}, e_2 = {B, C}, e_3 = {C, D}, e_4 = {A, C}, e_5 = {B, D}.

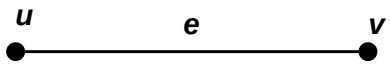
2.3 Terminologi dalam Graf

Terminology (istilah) yang berkaitan dengan graf akan sering digunakan. Di bawah ini didefinisikan beberapa terminologi yang sering dipakai dan yang berhubungan dengan pohon merentang minimum.

2.3.1 *Adjacent* dan *Incident*

Sisi $e = (u, v)$ dikatakan menghubungkan titik u dan v. Jika $e = (u, v)$ adalah sisi di graf G, maka u dan v disebut terhubung langsung (*adjacent*), u dan e serta v dan e disebut terkait langsung (*incident*). (Chartrand dan Lesniak, 1986:4). Dari definisi di atas, maka dapat digambarkan sebagai berikut :

Contoh 2.2 :



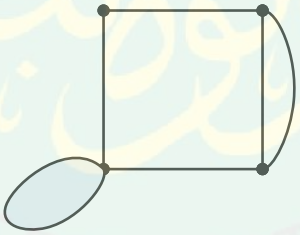
Gambar 2.2 *Adjacent* dan *Incident*

Gambar 2.2 Sisi $e = (u, v)$ yang menghubungkan titik u dan v. Karena $e = (u, v)$ sisi di G, maka u dan v disebut terhubung langsung (*adjacent*), sedangkan e dan u serta e dan v disebut terkait langsung (*incident*).

2.3.2 *Loop*

Loop adalah garis yang berawal dan berakhir pada suatu titik yang sama. (Wilson dan Watkins, 1990:43).

Contoh 2.3 :



Gambar 2.3 Graf yang Memuat *Loop* dan *Multiple Edge*

2.4 Graf Terhubung, Graf Berbobot, dan Subgraf

2.4.1 Graf Terhubung (*Connected*)

Misalkan u dan v titik berbeda pada graf G. Maka titik u dan v dapat dikatakan terhubung (*connected*), jika terdapat lintasan u–v di G. Sedangkan suatu graf G dapat dikatakan terhubung (*connected*), jika untuk setiap titik u dan v di G terhubung (Chartrand dan Lesniak, 1986:28).

Keterhubungan adalah sifat yang dimiliki graf. Graf terhubung dapat dilihat atau dibuktikan dari keterhubungan antara u dan v. Untuk lebih menguatkan kondisi $(u,v) \in E(G)$, sebut u dan v bersisian atau u dan v dihubungkan oleh satu sisi (Lih Hsing Hsu dan Cheng Kuan Lin, 2008:25).

Contoh 2.4 :

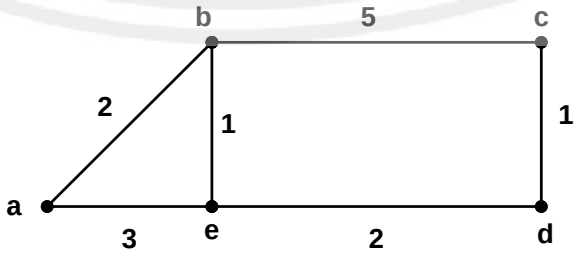
Gambar 2.4 Graf yang Terhubung

Graf G pada Gambar 2.4 dikatakan terhubung karena setiap titiknya terhubung dengan titik yang lain.

2.4.2 Graf Berbobot

Graf berbobot adalah graf yang setiap sisinya diberi sebuah (bobot). Bobot pada tiap dapat berbeda-beda bergantung pada masalah yang dimodelkan dengan graf (Munir. 2005:376).

Contoh 2.5 :



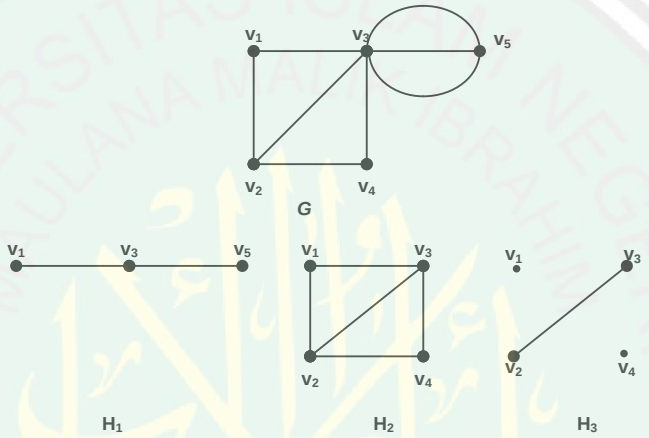
Gambar 2.5 Graf yang Berbobot

Graf G pada Gambar 2.5 dikatakan berbobot karena setiap sisinya diberi sebuah bobot.

2.4.3 Subgraf

Graf *H* disebut subgraf jika setiap titik dari graf *H* juga merupakan titik dari graf *G* dan setiap garis pada *H* juga merupakan garis pada graf *G*. Yang dinotasikan, *H* subgaraf dari *G* jika $V(H) \subset V(G)$ dan $E(H) \subset E(G)$ (Roman, 1989).

Contoh 2.6 :



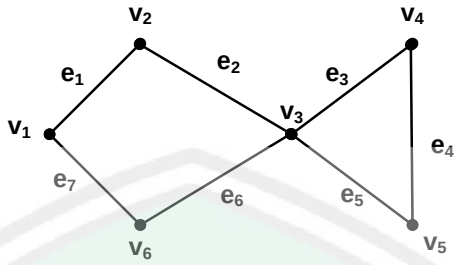
Gambar 2.6 Graf *G* dengan 3 Subgraf *H*

2.5 *Walk, Trail,* Sirkuit*, Path*

2.5.1 *Walk*

Walk dari titik u menuju titik v adalah urutan alternatif dari titik-titik dan garis-garis dari G yang dimulai dari titik u dan berakhir pada titik v , sehingga setiap sisi *incident* dengan titik yang terdahulu dan titik yang berikutnya. (Roman, 1989:33).

Contoh 2.7 :



Gambar 2.7 *Walk,Trail,Sirkuit, dan Path*

Dari Gambar 2.7 didapatkan walk dari v_1 sampai v_5, urutannya adalah (v_1, e_7, v_6, e_6, v_3, e_3, v_4, e_4, v_5).

2.5.2 *Trail*

Trail dari titik u menuju ke v adalah *walk* dari u menuju ke v di mana tidak ada sisi yang dilalui lebih dari satu kali atau dapat juga didefinisikan jalan *u–v* yang semua sisinya berbeda disebut *trai u-v* (Roman, 1989:33).

Dari gambar 2.7 dihasilkan *trail* dengan urutannya adalah (v_2, e_2, v_3, e_3, v_4, e_4, v_5, e_5, v_3)

2.5.3 Path

Path dari titik u menuju ke titik v adalah *walk* dari titik u menuju titik v dimana tidak ada titik yang dilalui lebih dari satu kali (Roman, 1989:34).Dari Gambar 2.7 dihasilkan path dengan urutannya adalah (v_1, e_1, v_2, e_2, v_3, e_5, v_5).

2.5.4 *Circuit* atau *Cycle*

Path yang berawal dan berakhir pada titik yang sama disebut *circuit* atau *cycle* (Munir, 2005:306).

Dari gambar 2.7 dihasilkan *circuit* atau *cycle* dengan urutannya adalah (v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_5, e_5, v_3, e_6, v_6, e_7, v_1).

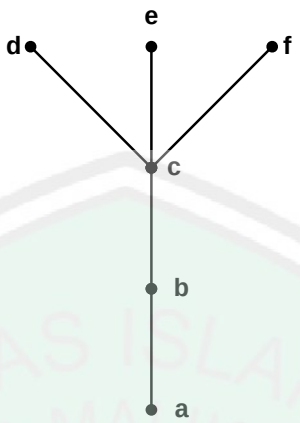
2.6 Tree, Forest

2.6.1 Pohon (*Tree*)

Sejumlah masalah yang berhubungan dengan graf yang ditemukan manusia dalam kehidupan nyata menimbulkan penemuan konsep-konsep pemecahan masalah graf. Konsep pohon pernah diterapkan pada tahun 1870-an oleh matematikawan Inggris yang bernama Arthur Cayley dalam penghitungan molekul kimia. Karya yang lebih baru membuktikan bahwa pohon telah digunakan di banyak bidang, mulai dari linguistik sampai komputer (Wilson dan Watkins, 1990: 54).

Pohon adalah graf tak berarah terhubung yang tidak mengandung sirkuit. Menurut definisi tersebut, ada dua sifat penting pada pohon yaitu terhubung dan tidak mengandung sirkuit (Chartrand dan Lesniak, 1986:67).

Di dalam suatu pohon, titik berderajat 1 dinamakan daun (*leaf*) atau titik terminal (*terminal node*), sedangkan titik yang berderajat lebih dari 1 dinamakan titik cabang (*branch node*) atau titik internal (*internal node*). Misal $G = \{V(G), E(G)\}$ dengan $V(G)$ = {a, b, c, d, e, f} dan $E(G)$ = {ab, ad, cd, de, df}. Graf G tersebut jika digambarkan seperti ditunjukkan pada Gambar 2.8



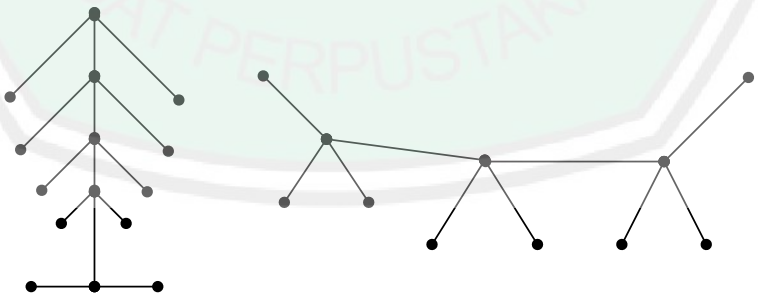
Gambar 2.8 Pohon

Dari gambar di atas, titik-titik *a, d, e,* dan *f* adalah daun, sedangkan titik-titik *b* dan *c* adalah titik cabang.

2.6.2 Hutan(*Forest*)

Forest adalah graf *acyclic* dimana setiap komponen terhubungnya merupakan pohon (*tree*) (Roman, 1989:37).

Contoh 2.8 :



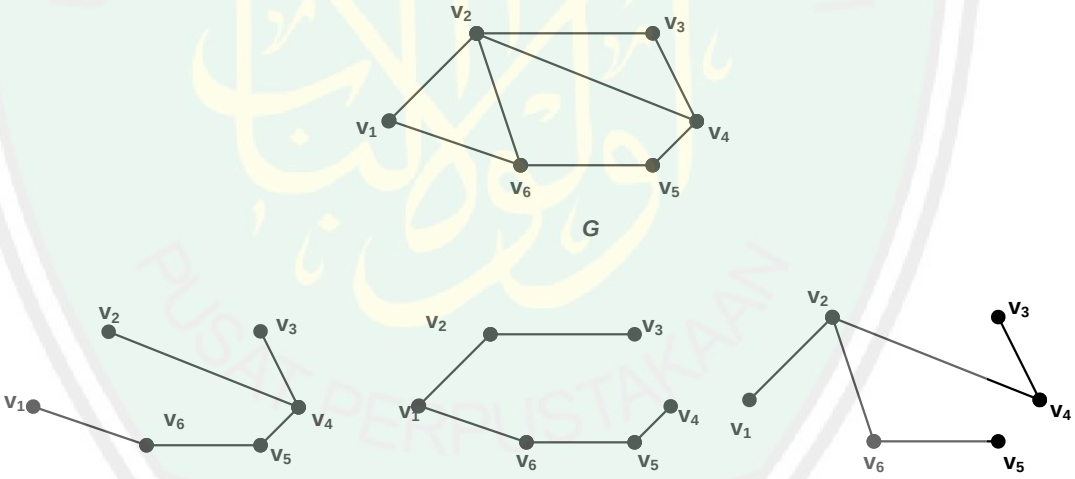
Gambar 2.9 *Forest*

2.7 Pohon Merentang dan Pohon Merentang Minimum

2.7.1 Definisi Pohon Merentang (*Spanning Tree*)

Pohon rentang suatu graf terhubung *G* adalah subgraf *G* yang merupakan pohon dan semua memuat titik dalam *G*. Disebut pohon merentang minimum karena semua simpul pada pohon T sama dengan semua simpul pada graf G, dan sisi-sisi pada pohon T $\subseteq$ sisi-sisi pada graf G. Dengan kata lain $V_1 = V$ dan $E_1 \subseteq E$. Pada contoh berikut ini akan diberikan bagaimana cara menentukan pohon merentang dari sebuah graf.

Contoh 2.10 :



Gambar 2.10 Graf *G* dan tiga Pohon Merentang *H* dari Graf *G*

Catatan:

Pohon perentang didefinisikan hanya untuk graf terhubung, karena pohon selalu terhubung. Pada graf tak terhubung dengan *n* buah titik tidak dapat ditemukan subgraf terhubung dengan *n* buah titik. Tiap komponen dari graf tak terhubung mempunyai satu buah pohon rentang. Dengan demikian, graf tak

terhubung k komponen mempunyai hutan merentang yang terdiri dari k buah pohon merentang. (Munir, 2005: 449).

2.7.2 Pohon Merentang Minimum (*Minimal Spanning Tree*)

Jika G adalah graf berbobot, maka bobot pohon merentang T dari G didefinisikan sebagai jumlah bobot semua sisi di T, pohon merentang yang berbeda mempunyai bobot yang berbeda pula. Di antara semua pohon merentang di G, adalah pohon merentang yang berbobot minimum dinamakan pohon merentang minimum **(**Munir**,** 2005:450**)**.

2.8 Algoritma Boruvka

Algoritma pertama untuk mencari pohon merentang minimum dari suatu graf ditemukan oleh Otakar Borůvka pada tahun 1926. Untuk menentukan pohon merentang minimum dari sebuah graf dengan menggunakan Algoritma Boruvka maka diperlukan langkah-langkah sebagai berikut:

Untuk mencari pohon merentang minimum pada graf *G*, maka

Langkah 1 : Salin titik dari *G* ke graf baru *L* yang kosong.

Langkah 2 : Sedangkan *L* tidak terhubung (artinya hutan lebih dari satu pohon)

- Untuk setiap pohon di *L*, hubungkan sebuah titik ke titik yang lain pada pohon yang lain di *L* dengan menambahkan sisi yang berbobot minimum

(Chartrand dan Ortrud, 1993:67).

2.9 Algoritma Prim

Masalah pohon merentang minimum dapat dipecahkan dengan bantuan suatu pohon yang ditemukan oleh Prim (1957). Algoritma ini biasa disebut dengan Algoritma Prim (Bondy dan Murty, 1976:146). Algoritma Prim adalah suatu Algoritma di dalam teori graf yang bertujuan menentukan suatu pohon merentang minimum dari suatu graf terhubung yang berbobot. Metode ini digunakan untuk menemukan suatu subset dari sisi yang membentuk suatu pohon yang melibatkan tiap-tiap titik, dimana total bobot dari semua sisi di dalam pohon adalah minimum. Secara terurut algotritma Prim dapat dituliskan sebagai berikut: (Munir, 2005: 451).

1. Menentukan sebarang titik awal dan dilanjutkan mengambil sisi dari graf *G* yang berbobot minimum dari titik awal yang di pilih tadi, masukkan ke dalam *T* yang kosong.
2. Pilih sisi *e* yang mempunyai bobot minimum berikutnya dan bersisian dengan titik di *T*, tetapi *e* tidak membentuk sirkuit di *T*. Masukkan *e* ke dalam *T*.
3. Ulangi langkah 2 hingga terbentuk pohon merentang minimumnya.

2.10 Algoritma Kruskal

Algoritma Kruskal adalah suatu Algoritma di dalam teori graf yang digunakan untuk mengkonstruksi pohon merentang minimum di dalam graf berbobot terhubung secara berurutan dari sisi yang berbobot kecil sampai berbobot besar hingga tidak terbentuk sikel. Algoritma Kruskal dapat diasumsikan dengan memilih sisi dari graf secara berurutan berdasarkan bobotnya dari bobot

kecil ke bobot besar. Secara terurut algotritma Prim dapat dituliskan sebagai berikut: (Munir, 2005: 455).

1. Urutkan sisi-sisi graf dari kecil ke besar. *T* merupakan himpunan kosong.
2. Pilih sisi *e* dengan bobot minimum yang tidak membentuk sirkuit di *T*, tambahkan *e* ke dalam *T*
3. Ulangi langkah 2 hingga terbentuk pohon merentang minimum

2.11 Algoritma Sollin

Algoritma Sollin adalah suatu Algoritma di dalam teori graf yang digunakan untuk menentukan pohon merentang minimum di dalam graf berbobot terhubung dengan cara melakukan penghapusan sisi-sisi yang tidak menyebabkan graf menjadi tidak berhubung atau membentuk sirkuit. Penghapusan tersebut dimulai dari sisi atau busur yang memiliki bobot terbesar hingga terkecil (Http://www.informatika.org/rinaldi/Matdis/20082009/Makalah2008/Makalah0809-061.pdf). Untuk menentukan pohon merentang minimum dari sebuah graf dengan menggunakan Algoritma Sollin maka diperlukan langkah-langkah sebagai berikut.

1. Urutkan sisi-sisi pada graf berdasarkan bobotnya dari besar ke kecil
2. Lakukan penghapusan setiap sisi yang tidak menyebabkan graf menjadi tidak terhubung
3. Ulangi langkah 2 hingga diperoleh pohon merentang minimum

2.12 Pembuktian dalam Pandangan Islam

Al-Quran merupakan kitab suci yang banyak menyimpan rahasia-rahasia baik dalam dunia nyata maupun ghaib, baik kehidupan masa sekarang ataupun masa yang akan datang, dan kini mulai banyak dikaji oleh para ilmuwan. Karena tanpa disadari bahwa Al-Quran sebenarnya menjadi acuan dalam berbagai hal bukan hanya sekedar sebagai pelengkap. Dari Al-Quran banyak ilmu-ilmu yang dapat digali diantaranya ilmu matematika, contohnya hukum yang menerangkan banyak jalan menuju kebaikan dan tidak berlebihan dalam melakukan ketaatan.

Dalam Al-Qur'an surat Al-Baqarah ayat 185 disebutkan:

يُرِيدُ ٱللَّهُ بِكُمُ ٱلۡيُسۡرَ وَلَا يُرِيدُ بِكُمُ ٱلۡعُسۡرَ ١٨٥

Artinya : Allah menghendaki kemudahan bagimu, dan tidak menghendaki kesukaran bagimu(Q.S. Al-Baqarah:185).

Ayat di atas menceritakan bahwa segala sesuatu yang baik jika dapat ditempuh dengan jalan yang mudah maka tidak perlu ditempuh dengan jalan yang sulit. Allah SWT menghendaki kemudahan bagi semua hambaNya dalam menempuh segala hal yang baik, karena apapun cara yang dapat ditempuh untuk mencapai suatu tujuan, kembali lagi pada firman Allah di atas untuk menempuh dengan cara yang mudah dari berbagai cara yang dapat ditempuh.

Dalam surat Al-Israa' ayat 27 juga disebutkan:

إِنَّ ٱلۡمُبَذِّرِينَ كَانُوٓاْ إِخۡوَٰنَ ٱلشَّيَٰطِينِۖ وَكَانَ ٱلشَّيۡطَٰنُ لِرَبِّهِۦ كَفُورٗا ٢٧

Artinya : Sesungguhnya pemboros-pemboros itu adalah saudara-saudara syaitan dan syaitan itu adalah sangat ingkar kepada Tuhannya.(Q.S. Al-Israa':27)

Berdasarkan kedua ayat itulah hendaknya dalam upaya untuk menempuh sesuatu hal yang baik jika ditemui banyak cara untuk menempuhnya, hendaknya ditempuh dengan cara termudah.

Dalam suatu hadits disebutkan:

وَفِي رِوَايَةٍ أُخْرَى لِبُخَارِيٍّ : { وَقَارِبُوْا وَاغْدُوْا وَرُوْحُوْا, وَشَيْءٌ مِنَ الدُّلْجَةِ, القَصْدَ تَبْلُغُوْا }.

Artinya : Berusahalah mendekati yang lurus, pergunakan waktu pagi, sore, dan pengujung malam. Lakukanlah yang sedang-sedang, lakukanlah yang sedang-sedang, niscaya kamu akan sampai."

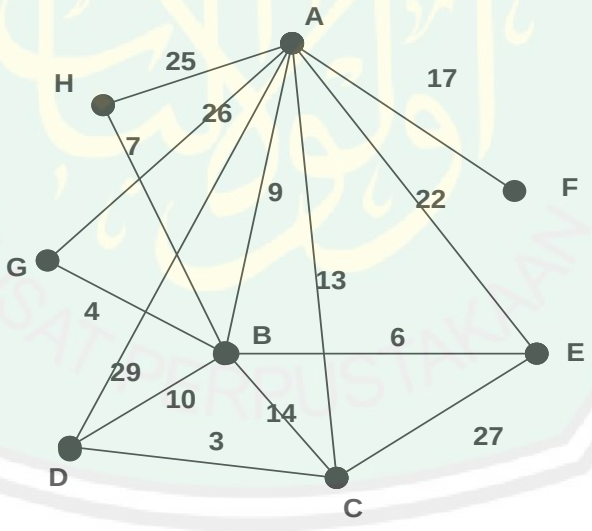
Hadits di atas menerangkan bahwa berbicara dan berbuat yang tidak berlebihan akan dapat mengantarkan kepada ridha Allah SWT. Karena orang yang berlebihan dalam berbicara dan berbuat pasti akan menuai kehancuran. Begitu pula sama halnya jika dalam mencari keefektifan suatu algoritma untuk mencari pohon merentang minimum, maka lebih baik gunakan algoritma yang diketahui paling efektif digunakan dalam mencari pohon merentang minimum.

BAB III

HASIL DAN PEMBAHASAN

Pada bab ini akan dibahas tentang pencarian pohon merentang minimum dengan menggunakan algoritma Boruvka, algoritma Prim, algoritma Kruskal, dan algoritma Sollin, yang kemudian dilanjutkan dengan menentukan algoritma manakah yang lebih efektif digunakan dalam menentukan pohon merentang minimum.

Berikut ini akan ditunjukkan graf berbobot *G* yang memuat 8 titik dan 14 sisi

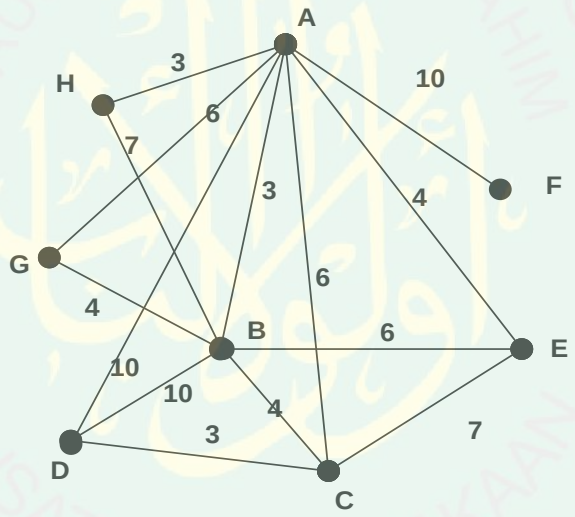


Gambar 3.1 Graf G dengan Banyak Sisi = 2(*p* – 1)

Gambar graf yang dimaksud di atas adalah *G* dengan banyak sisi = 2(*p* - 1) dengan *p* adalah banyak titik, dan diketahui banyak titik adalah 8 sehingga diperoleh graf *G* di mana banyak sisinya 14. Untuk mencari dan mendapatkan pohon merentang minimum dari graf tersebut, digunakan algoritma Boruvka

algoritma Prim, algoritma Kruskal, dan algoritma Sollin. Keempat algoritma tersebut mempunyai metodologi yang berbeda tetapi keempatnya memang dikonstruksikan untuk mendapatkan pohon merentang minimum. Setelah diperoleh pohon merentang minimum, maka akan diperoleh jarak minimum terpendek yang menghubungkan antara titik yang satu dengan yang lain.

Berikut ini akan ditunjukkan graf berbobot *G* yang memuat 8 titik dan 14 sisi namun terdapat sisi yang memiliki bobot sama.

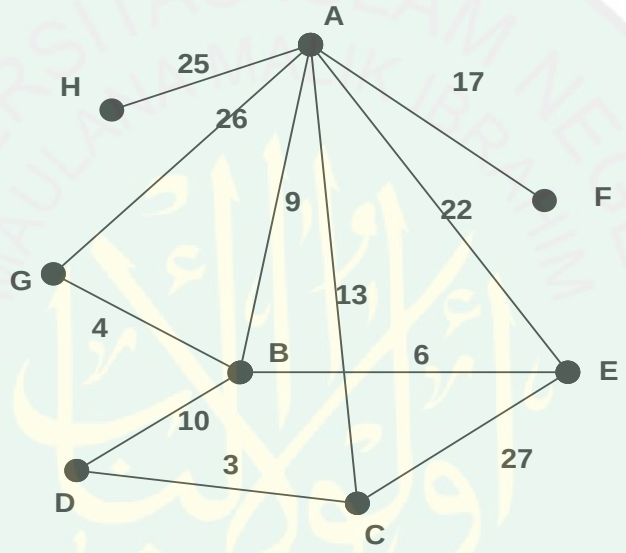


Gambar 3.2 Graf G dengan Banyak Sisi = 2(*p* – 1) dan Terdapat Sisi yang memiliki bobot Sama

Gambar graf yang dimaksud di atas adalah *G* dengan banyak sisi = 2(*p* - 1) dan terdapat sisi yang memiliki bobot yang sama, dengan *p* adalah banyak titik, dan diketahui banyak titik adalah 8 sehingga diperoleh graf *G* di mana banyak sisinya 14. Untuk mencari dan mendapatkan pohon merentang minimum dari graf tersebut, digunakan algoritma Boruvka algoritma Prim, algoritma Kruskal, dan algoritma Sollin. Keempat algoritma tersebut mempunyai metodologi yang berbeda tetapi keempatnya memang dikonstruksikan untuk mendapatkan pohon

merentang minimum. Setelah diperoleh pohon merentang minimum, maka akan diperoleh jarak minimum terpendek yang menghubungkan antara titik satu dengan yang lain.

Berikut ini akan ditunjukkan graf berbobot *G* yang memuat 8 titik dan 11 sisi.

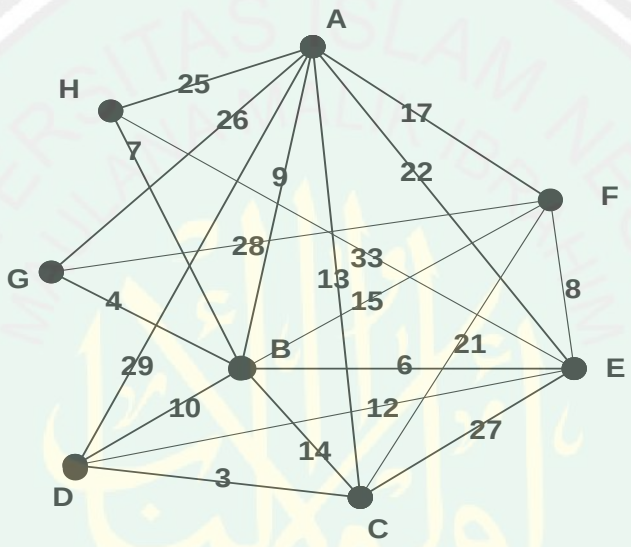


Gambar 3.3 Graf G dengan Banyak Sisi < 2(p – 1)

Gambar graf yang dimaksud di atas adalah *G* dengan banyak sisi < 2(*p* - 1) dengan *p* adalah banyak titik, dan diketahui banyak titik adalah 8 sehingga diperoleh graf *G* di mana banyak sisinya kurang dari 14, dan pada graf ini memuat 8 titik dan 11 sisi. Untuk mencari dan mendapatkan pohon merentang minimum dari graf tersebut, digunakan algoritma Boruvka algoritma Prim, algoritma Kruskal, dan algoritma Sollin. Keempat algoritma tersebut mempunyai metodologi yang berbeda tetapi keempatnya memang dikonstruksikan untuk mendapatkan pohon merentang minimum. Setelah diperoleh pohon merentang

minimum, maka akan diperoleh jarak minimum terpendek yang menghubungkan antara titik satu dengan yang lain.

Berikut ini akan ditunjukkan graf berbobot *G* yang memuat 8 titik dan 20 sisi.



Gambar 3.4 Graf G dengan Banyak Sisi > 2(*p* – 1)

Gambar graf yang dimaksud di atas adalah *G* dengan banyak sisi > 2(*p* - 1) dengan *p* adalah banyak titik, dan diketahui banyak titik adalah 8 sehingga diperoleh graf *G* di mana banyak sisinya lebih dari 14, dan pada graf ini memuat 8 titik dan 20 sisi. Untuk mencari dan mendapatkan pohon merentang minimum dari graf tersebut, digunakan Algoritma Boruvka Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin. Keempat algoritma tersebut mempunyai metodologi yang berbeda tetapi keempatnya memang dikonstruksikan untuk mendapatkan pohon merentang minimum. Setelah diperoleh pohon merentang minimum, maka akan

diperoleh jarak minimum terpendek yang menghubungkan antara titik satu dengan yang lain.

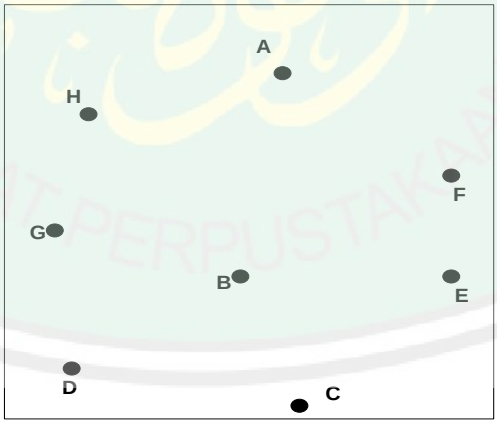
3.1 Penghitungan Pohon Merentang Minimum Menggunakan Algoritma Boruvka

3.1.1 Penghitungan Pohon Merentang Minimum pada Graf G dengan Banyak Sisi = 2(*p* - 1)

Untuk menentukan pohon merentang minimum dengan menggunakan algoritma Boruvka dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Langkah 1 : Salin titik dari *G* ke graf baru *L* yang kosong, dan diperoleh gambar seperti di bawah ini

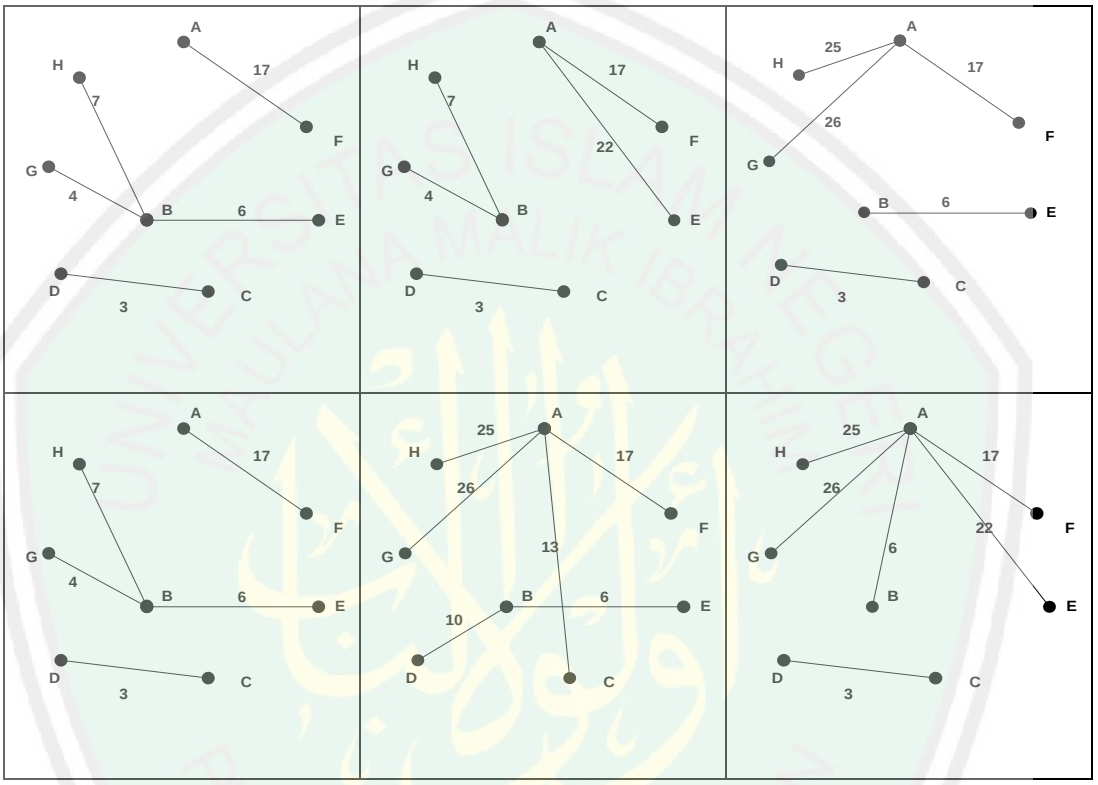
Gambar :



Gambar 3.5 Salinan Titik dari Graf G dengan Banyak Sisi = 2(*p* - 1) ke Graf L yang Kosong

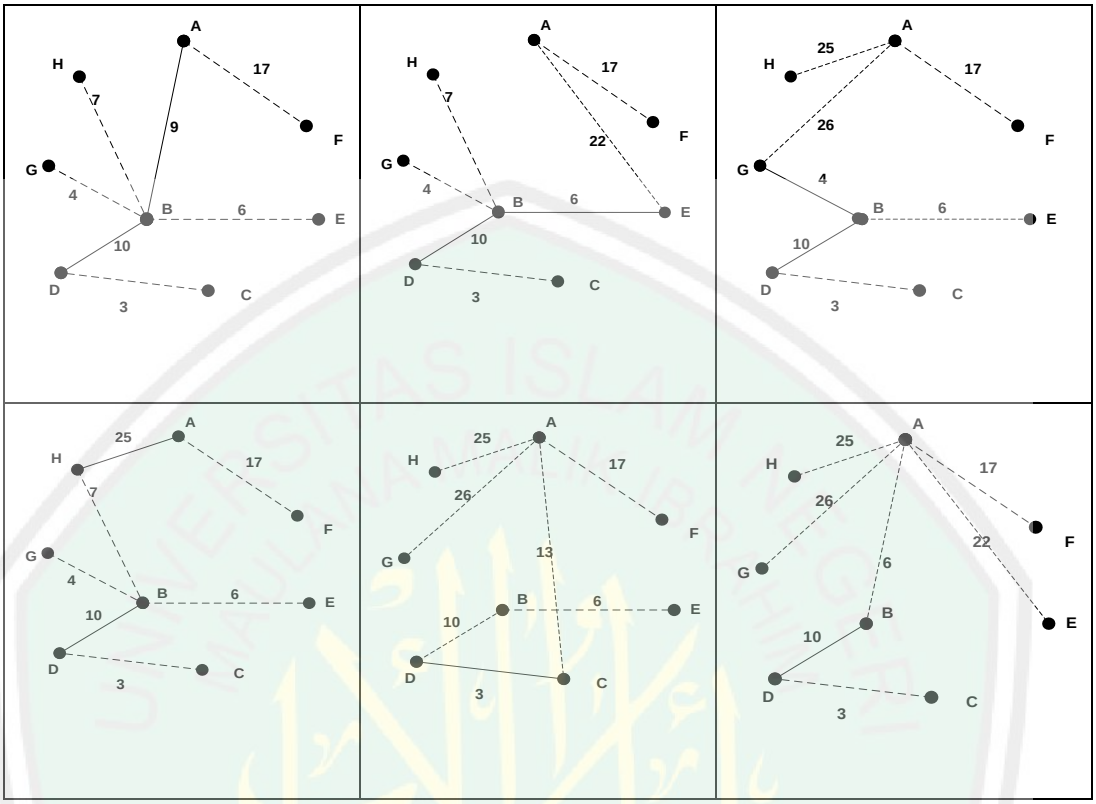
Langkah 2: Selanjutnya L adalah graf yang tidak terhubung yang berarti hutan yang terdiri dari beberapa pohon. Karena dalam penetapan hutan yang terdiri dari beberapa pohon tidak memperhatikan bobot dari

sisi yang dipilih. Jadi ada banyak kemungkinan hutan yang bisa ditetapkan, dan peneliti mencontohkan 6 kemungkinan dari banyak kemungkinan yang terjadi.



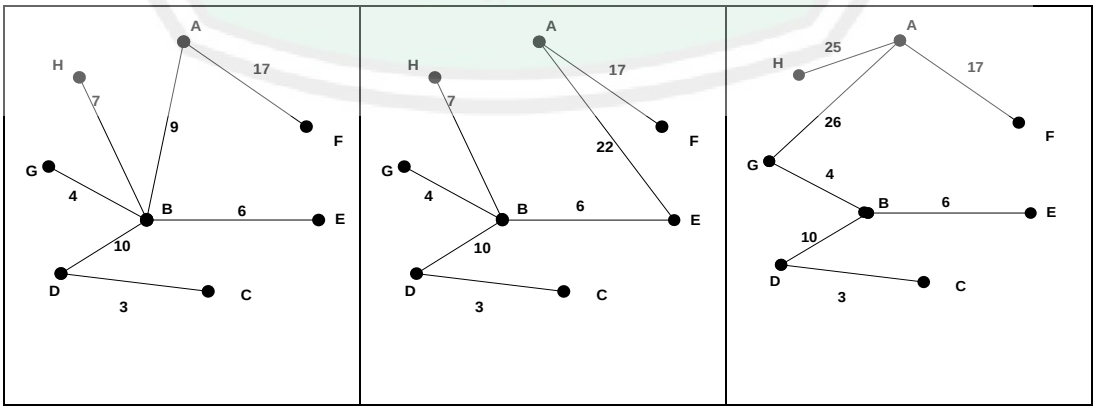
Gambar 3.6 Beberapa Hutan yang Diperoleh dari Graf G dengan Banyak Sisi = 2(p - 1)

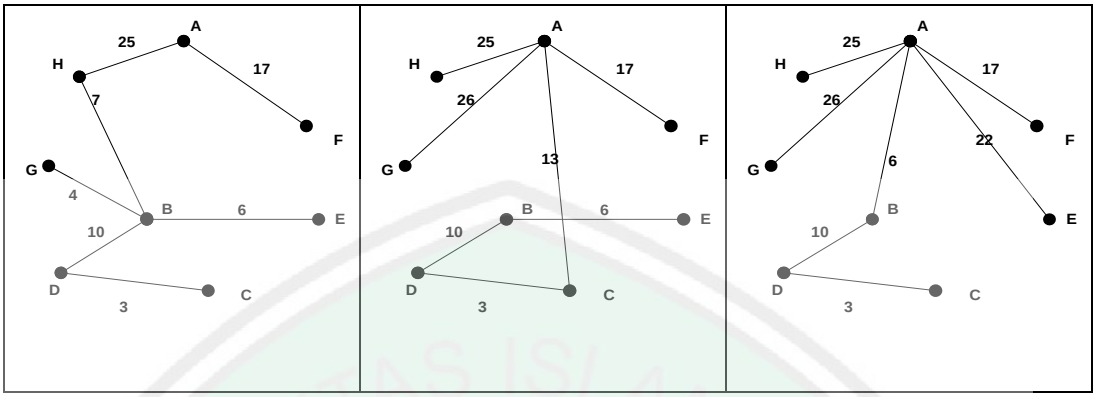
Langkah 3: Dari graf L di atas dapat dilihat bahwa graf L terdapat beberapa pohon. Selanjutnya untuk setiap pohon di L, sebuah titik dari setiap pohon dihubungkan dengan menambahkan sisi yang berbobot minimum dan diperoleh gambar seperti di bawah ini



Gambar 3.7 Beberapa Hutan Setelah Dihubungkan dengan Sisi Berbobot Minimum yang Diperoleh dari Graf G dengan Banyak Sisi = 2(p - 1)

Karena sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Boruvka pada gambar di bawah ini :





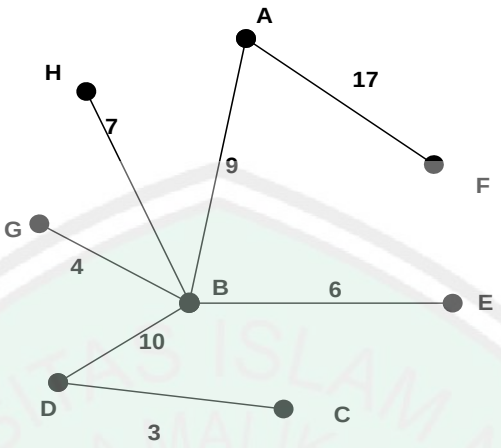
Gambar 3.8 Beberapa Pohon Merentang Minimum yang Diperoleh dari Graf G dengan Banyak Sisi = 2(p - 1)

Dari perhitungan Algoritma Boruvka di atas diperoleh pohon merentang minimum dengan banyak bobot sebagai berikut :

Tabel 3.1 Beberapa Bobot yang Diperoleh dari Graf G dengan Banyak Sisi = 2(p - 1)

Diperoleh pohon merentang minimum dengan bobot 56	Diperoleh pohon merentang minimum dengan bobot 69	Diperoleh pohon merentang minimum dengan bobot 90
Diperoleh pohon merentang minimum dengan bobot 72	Diperoleh pohon merentang minimum dengan bobot 100	Diperoleh pohon merentang minimum dengan bobot 112

Dari beberapa macam hutan yang dipilih di atas diperoleh pohon merentang minimum yang berbeda-beda, dan pohon merentang yang dipilih adalah pohon merentang dengan banyak bobot terkecil yaitu pohon merentang dengan bobot 56.



Gambar 3.9 Pohon Merentang Minimum dengan Algoritma Boruvka pada Graf G dengan Banyak Sisi = 2(p - 1)

Jadi diperoleh pohon merentang minimum dengan W = {A, B, C, D, E, F, G, H} dengan bobot 56, dan dari 8 titik serta 14 sisi, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 3.

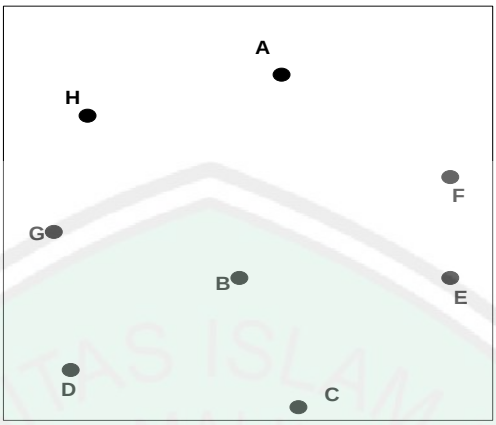
Catatan: Masih banyak kemungkinan hutan yang bisa ditetapkan, dan otomatis banyak pula kemungkinan pohon merentang minimum yang terbentuk.

3.1.2 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Banyak Sisi = 2(*p* - 1) dan Terdapat Sisi yang Memiliki Bobot Sama

Untuk menentukan pohon merentang minimum dengan menggunakan algoritma Boruvka dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

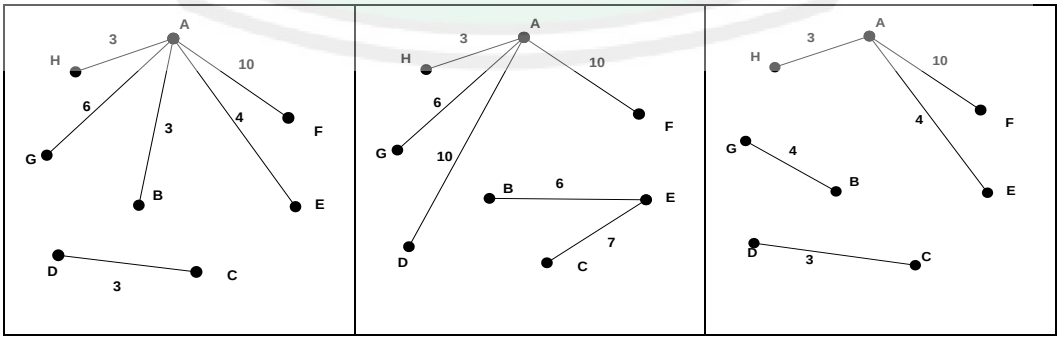
Langkah 1 : Salin titik dari *G* ke graf baru *L* yang kosong dan diperoleh gambar seperti di bawah ini

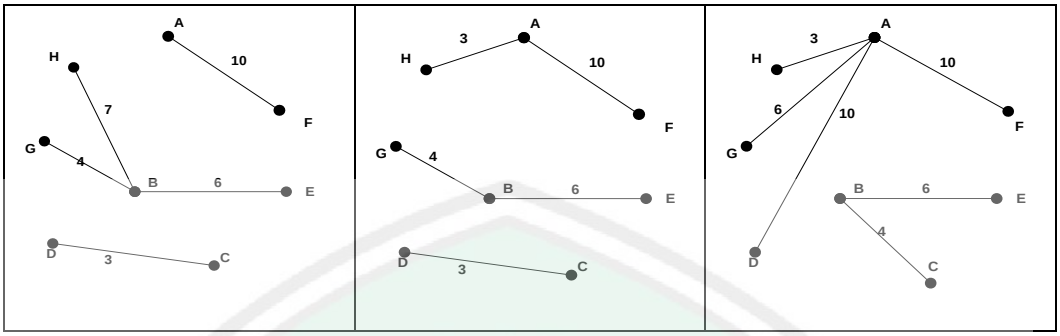
Gambar :



Gambar 3.10 Salinan Titik dari Graf G dengan Banyak Sisi = 2(p - 1) dan terdapat Sisi yang Memiliki Bobot Sama ke Graf L yang Kosong

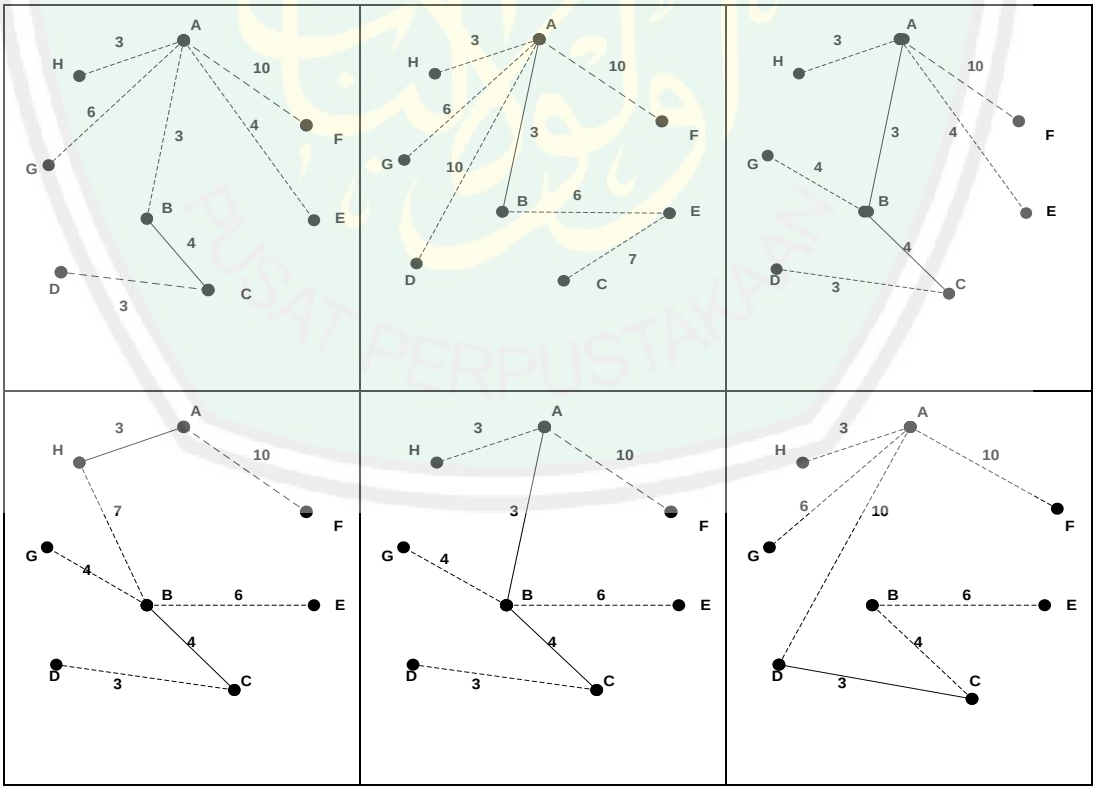
Langkah 2: Selanjutnya L adalah graf yang tidak terhubung yang berarti hutan yang terdiri dari beberapa pohon. Karena dalam penetapan hutan yang terdiri dari beberapa pohon tidak memperhatikan bobot dari sisi yang dipilih. Jadi ada banyak kemungkinan hutan yang bisa ditetapkan, dan peneliti mencontohkan 6 kemungkinan dari banyak kemungkinan yang terjadi.





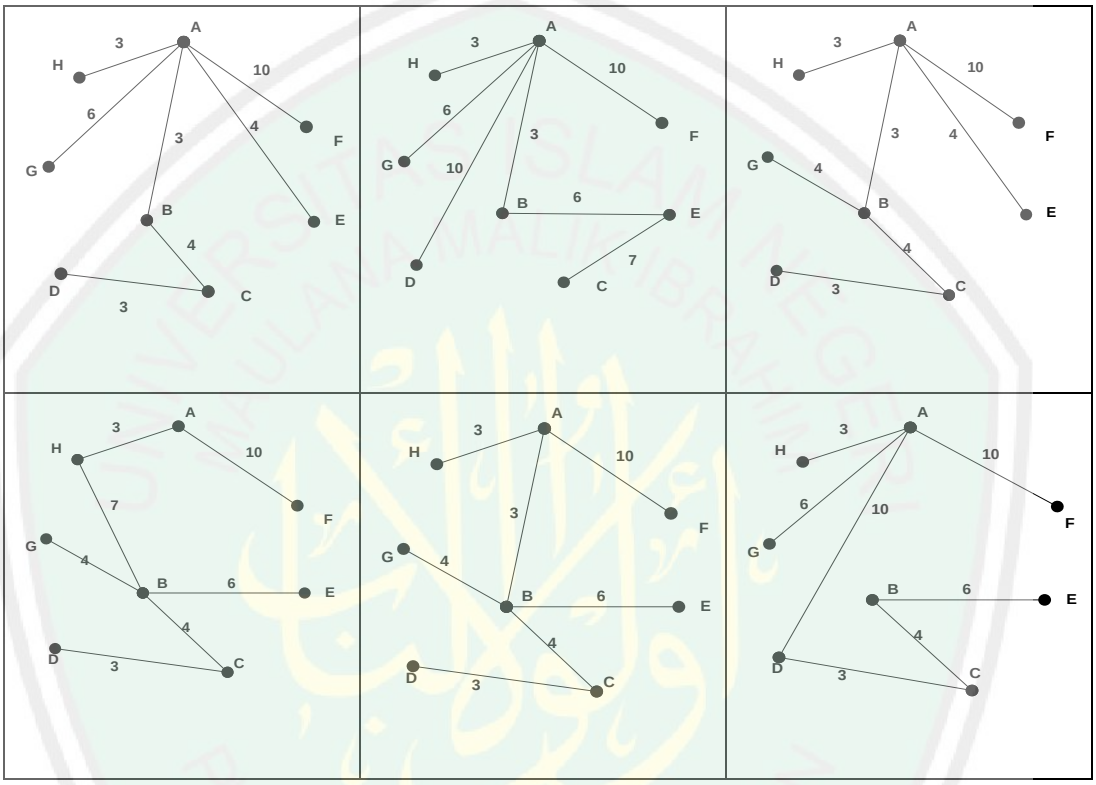
Gambar 3.11 Beberapa Hutan yang Diperoleh dari Graf G dengan Banyak Sisi = 2(p - 1) dan terdapat Sisi yang Memiliki Bobot Sama

Langkah 3: Dari graf L di atas dapat dilihat bahwa graf L terdapat beberapa pohon. Selanjutnya untuk setiap pohon di L, sebuah titik dari setiap pohon dihubungkan dengan menambahkan sisi yang berbobot minimum dan diperoleh gambar seperti di bawah ini



Gambar 3.12 Beberapa Hutan Setelah Dihubungkan dengan Sisi Berbobot Minimum yang Diperoleh dari Graf G dengan Banyak Sisi = 2(p - 1) dan terdapat Sisi yang Berbobot Sama

Karena sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Boruvka pada gambar di bawah ini :



Gambar 3.13 Beberapa Pohon Merentang Minimum yang Diperoleh dari Graf G dengan Banyak Sisi = 2(p - 1)dan terdapat Sisi yang Memiliki Bobot Sama

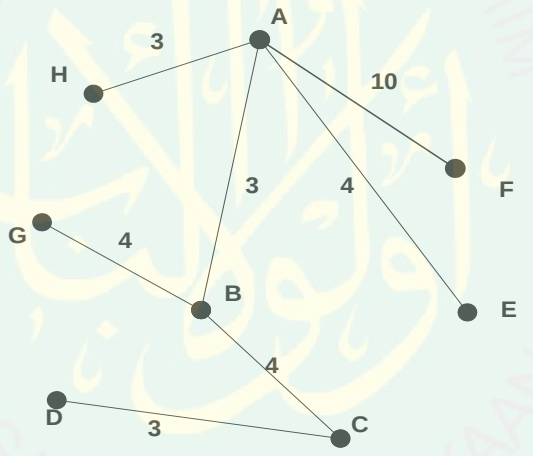
Dari perhitungan Algoritma Boruvka di atas diperoleh pohon merentang minimum dengan banyak bobot sebagai berikut :

Tabel 3.2 Beberapa Bobot yang Diperoleh dari Graf G dengan Banyak Sisi = 2(p - 1)dan terdapat Sisi yang Memiliki Bobot Sama

Diperoleh pohon merentang minimum dengan bobot 33	Diperoleh pohon merentang minimum dengan bobot 45	Diperoleh pohon merentang minimum dengan bobot 31

Diperoleh pohon merentang minimum dengan bobot 37	Diperoleh pohon merentang minimum dengan bobot 33	Diperoleh pohon merentang minimum dengan bobot 42

Dari beberapa macam hutan yang dipilih di atas diperoleh pohon merentang minimum yang berbeda-beda, dan pohon merentang yang dipilih adalah pohon merentang dengan banyak bobot terkecil yaitu pohon merentang dengan bobot 31.



Gambar 3.14 Pohon Merentang Minimum dengan Algoritma Boruvka pada Graf G Banyak Sisi = 2(p - 1) dan Terdapat Sisi yang Memiliki Bobot Sama

Jadi diperoleh pohon merentang minimum dengan W = {A, B, C, D, E, F, G, H} dengan bobot 31, dan dari 8 titik serta 14 sisi dan terdapat sisi yang memiliki bobot sama, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 3.

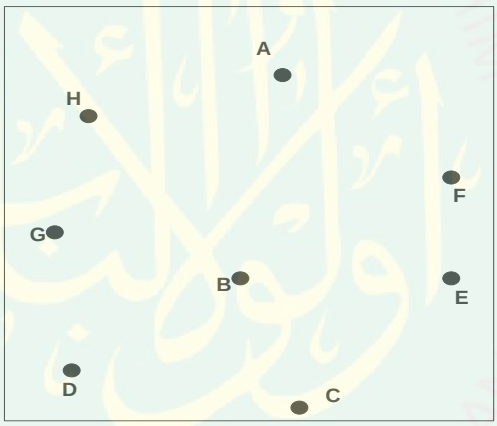
Catatan: Masih banyak kemungkinan hutan yang bisa ditetapkan, dan otomatis banyak pula kemungkinan pohon merentang minimum yang terbentuk

3.1.3 Penghitungan Pohon Merentang Minimum pada Graf G dengan Banyak Sisi < 2(*p* - 1)

Untuk menentukan pohon merentang minimum dengan menggunakan algoritma Boruvka dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

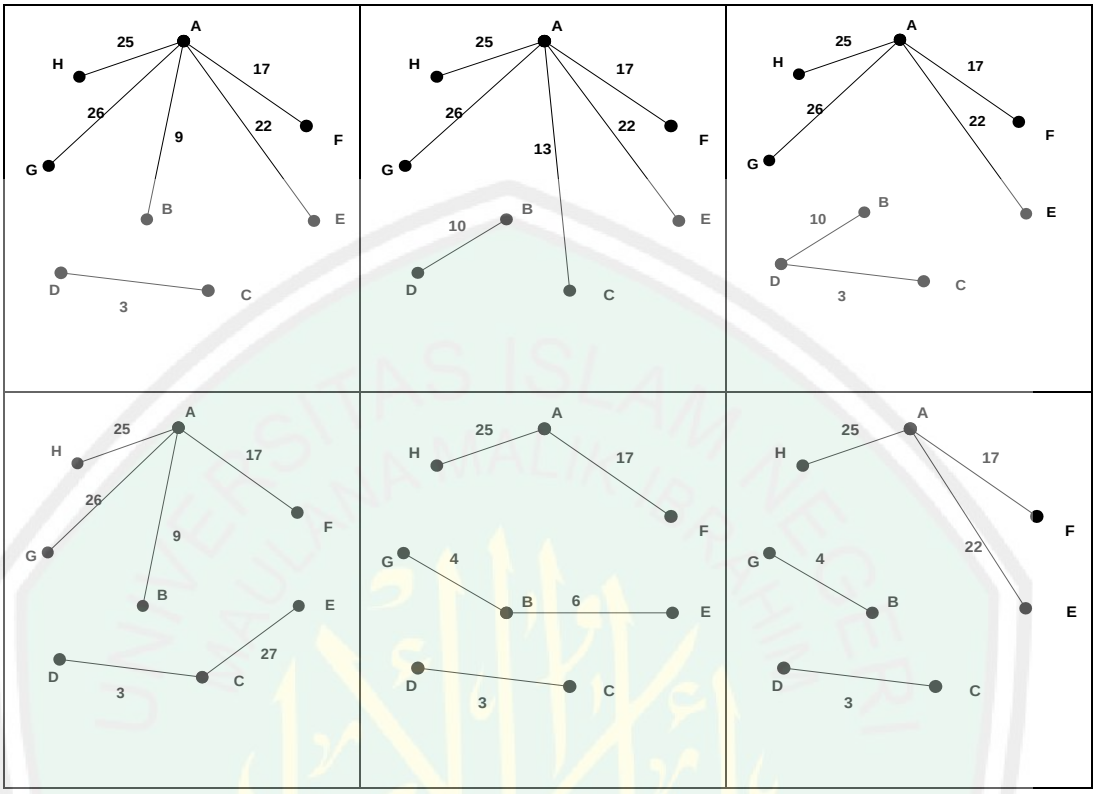
Langkah 1 : Salin titik dari *G* ke graf baru *L* yang kosong dan diperoleh gambar seperti di bawah ini

Gambar :



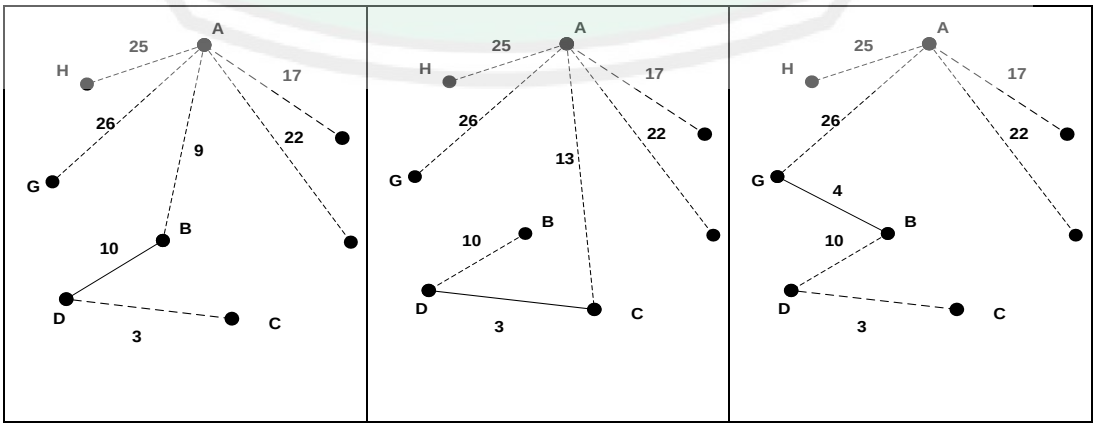
Gambar 3.15 Salinan Titik dari Graf G dengan Banyak Sisi < 2(*p* - 1)ke Graf L yang Kosong

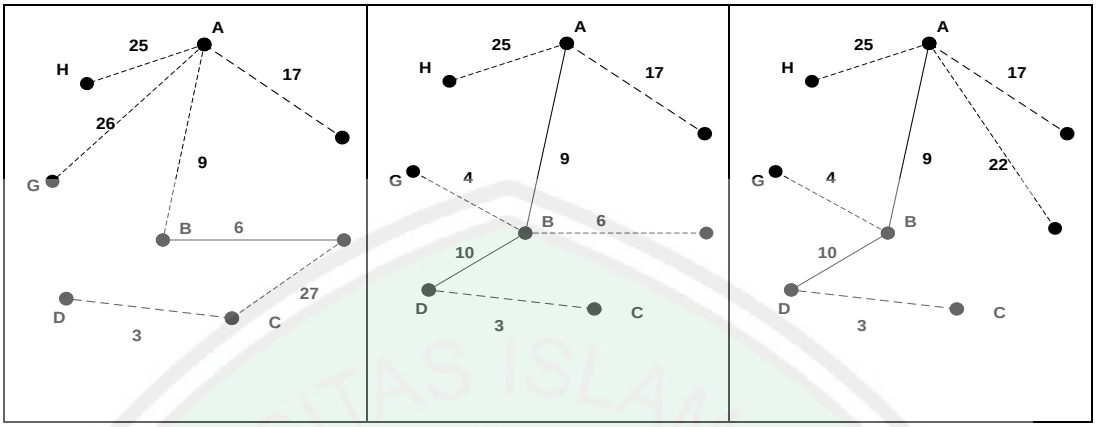
Langkah 2: Selanjutnya L adalah graf yang tidak terhubung yang berarti hutan yang terdiri dari beberapa pohon. Karena dalam penetapan hutan yang terdiri dari beberapa pohon tidak memperhatikan bobot dari sisi yang dipilih. Jadi ada banyak kemungkinan hutan yang bisa ditetapkan, dan penulis mencontohkan 6 kemungkinan dari banyak kemungkinan yang terjadi.



Gambar 3.16 Beberapa Hutan yang Diperoleh dari Graf G dengan Banyak Sisi < 2(p - 1)

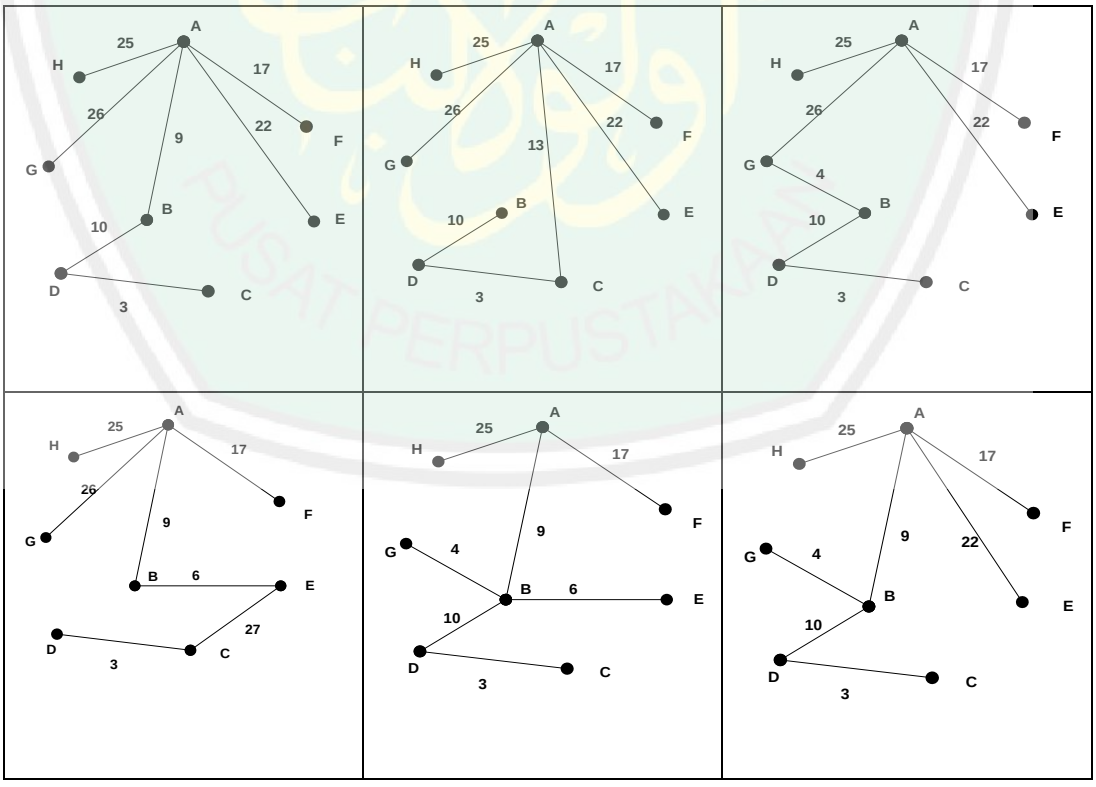
Langkah 3: Dari graf L di atas dapat dilihat bahwa graf L terdapat beberapa pohon. Selanjutnya untuk setiap pohon di L, sebuah titik dari setiap pohon dihubungkan dengan menambahkan sisi yang berbobot minimum, dan diperoleh gambar seperti di bawah ini





Gambar 3.17 Beberapa Hutan Setelah Dihubungkan dengan Sisi Berbobot Minimum yang Diperoleh dari Graf G dengan Banyak Sisi < 2(p - 1)

Karena sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Boruvka pada gambar di bawah ini



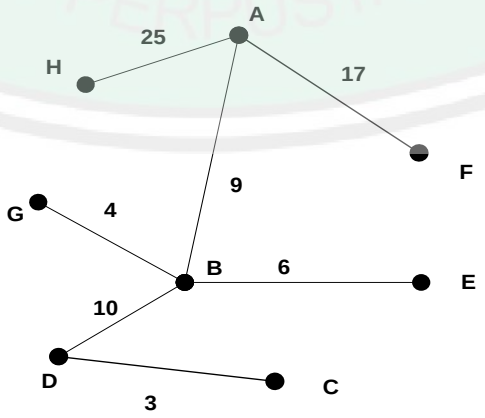
Gambar 3.18 Beberapa Pohon Merentang Minimum yang Diperoleh dari Graf G dengan Banyak Sisi < 2(p - 1)

Dari perhitungan Algoritma Boruvka di atas diperoleh pohon merentang minimum dengan banyak bobot sebagai berikut :

Tabel 3.3 Beberapa Bobot yang Diperoleh dari Graf G dengan Banyak Sisi < 2(p - 1)

Diperoleh pohon merentang minimum dengan bobot 112	Diperoleh pohon merentang minimum dengan bobot 116	Diperoleh pohon merentang minimum dengan bobot 107
Diperoleh pohon merentang minimum dengan bobot 113	Diperoleh pohon merentang minimum dengan bobot 74	Diperoleh pohon merentang minimum dengan bobot 90

Dari beberapa macam hutan yang dipilih di atas diperoleh pohon merentang minimum yang berbeda-beda, dan pohon merentang yang dipilih adalah pohon merentang dengan banyak bobot terkecil yaitu pohon merentang dengan bobot 74.



Gambar 3.19 Pohon Merentang Minimum dengan Algoritma Boruvka pada Graf G dengan Banyak Sisi < 2(p - 1)

Jadi diperoleh pohon merentang minimum dengan W = {A, B, C, D, E, F, G, H} dengan bobot 74, dan dari 8 titik serta 11 sisi, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 3.

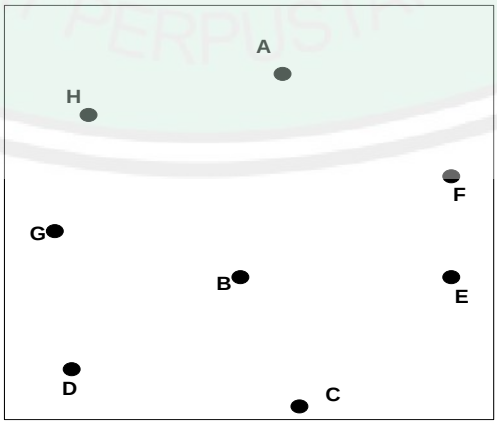
Catatan: Masih banyak kemungkinan hutan yang bisa ditetapkan, dan otomatis banyak pula kemungkinan pohon merentang minimum yang terbentuk

3.1.4 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Banyak Sisi > 2(*p* - 1)

Untuk menentukan pohon merentang minimum dengan menggunakan algoritma Boruvka dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

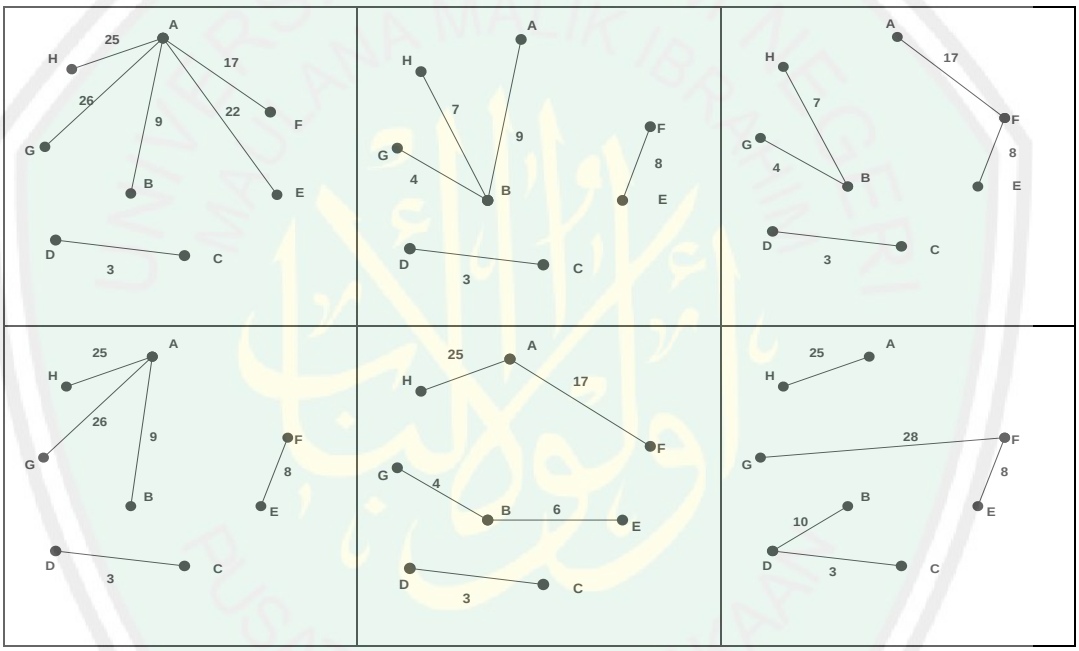
Langkah 1 : Salin titik dari *G* ke graf baru *L* yang kosong, dan diperoleh gambar seperti di bawah ini

Gambar :



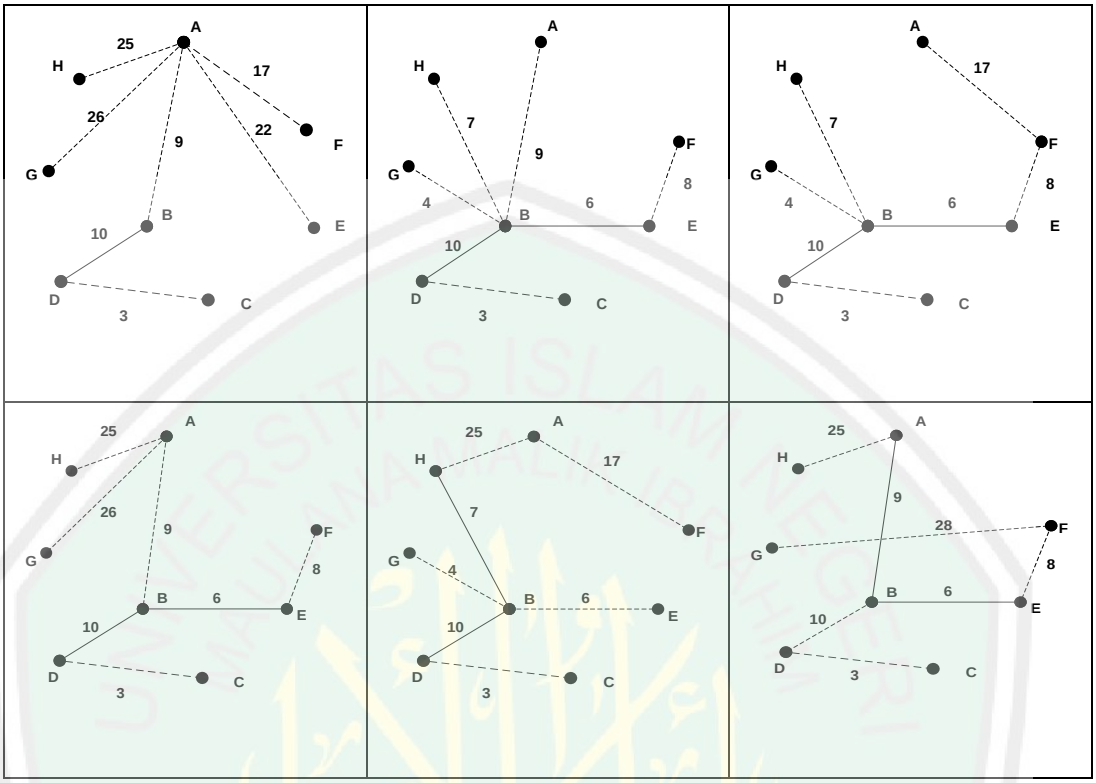
Gambar 3.20 Salinan Titik dari Graf G dengan Banyak Sisi > 2(*p* - 1)ke Graf L yang Kosong

Langkah 2: Selanjutnya L adalah graf yang tidak terhubung yang berarti hutan yang terdiri dari beberapa pohon. Karena dalam penetapan hutan yang terdiri dari beberapa pohon tidak memperhatikan bobot dari sisi yang dipilih. Jadi ada banyak kemungkinan hutan yang bisa ditetapkan, dan penulis mencontohkan 6 kemungkinan dari beberapa kemungkinan yang terjadi.



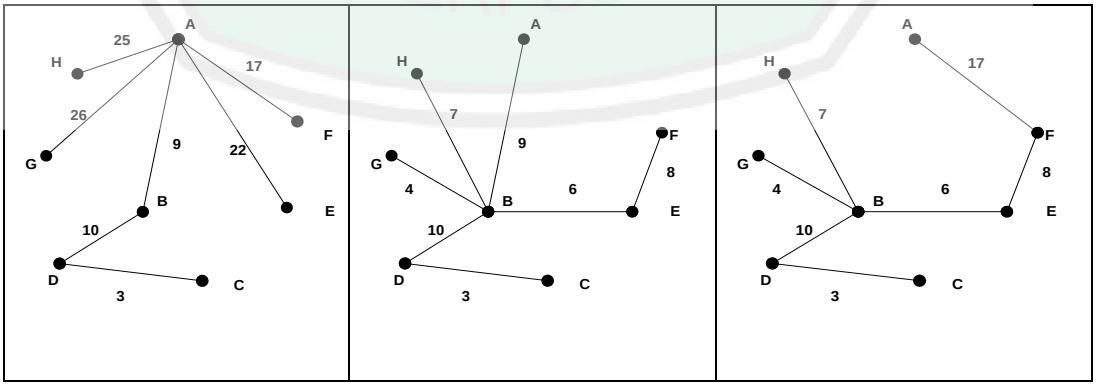
Gambar 3.21 Beberapa Hutan yang Diperoleh dari Graf G dengan Banyak Sisi > 2(p - 1)

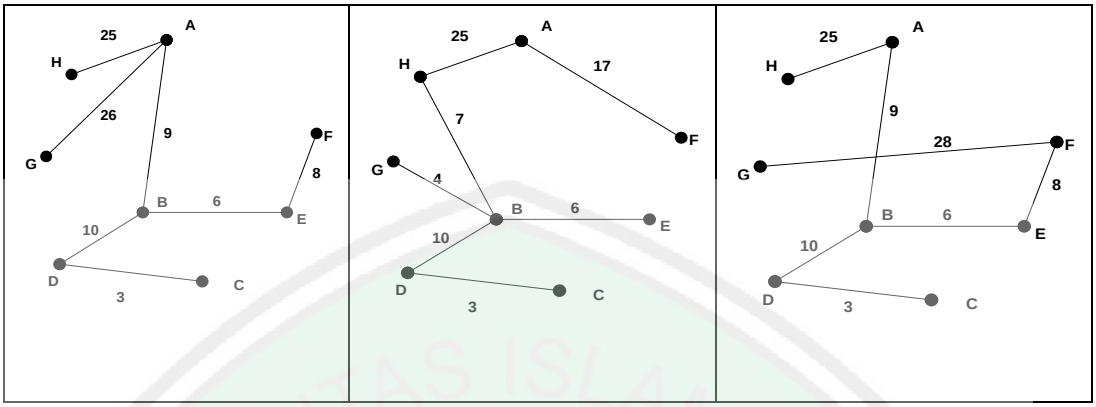
Langkah 3: Dari graf *L* di atas dapat dilihat bahwa graf *L* terdapat beberapa pohon. Selanjutnya untuk setiap pohon di *L*, sebuah titik dari setiap pohon dihubungkan dengan menambahkan sisi yang berbobot minimum, dan diperoleh gambar seperti di bawah ini



Gambar 3.22 Beberapa Hutan Setelah Dihubungkan dengan Sisi Berbobot Minimum yang Diperoleh dari Graf G dengan Banyak Sisi > 2(*p* - 1)

Karena sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Boruvka pada gambar di bawah ini :





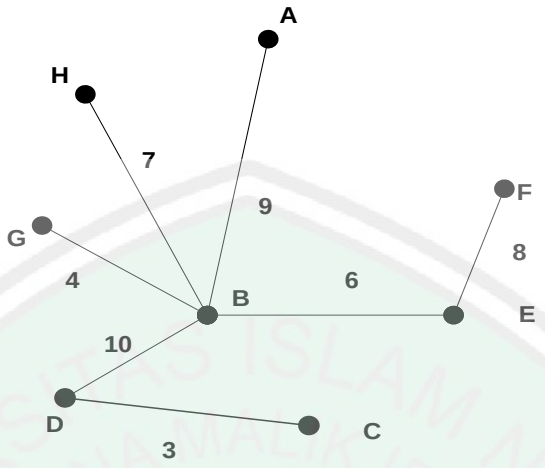
Gambar 3.23 Beberapa Pohon Merentang Minimum yang Diperoleh dari Graf G dengan Banyak Sisi > 2(p - 1)

Dari perhitungan Algoritma Boruvka di atas diperoleh pohon merentang minimum dengan banyak bobot sebagai berikut :

Tabel 3.4 Beberapa Bobot yang Diperoleh dari Graf G dengan Banyak Sisi > 2(p - 1)

Diperoleh pohon merentang minimum dengan bobot 112	Diperoleh pohon merentang minimum dengan bobot 47	Diperoleh pohon merentang minimum dengan bobot 55
Diperoleh pohon merentang minimum dengan bobot 87	Diperoleh pohon merentang minimum dengan bobot 72	Diperoleh pohon merentang minimum dengan bobot 89

Dari beberapa macam hutan yang dipilih di atas diperoleh pohon merentang minimum yang berbeda-beda, dan pohon merentang yang dipilih adalah pohon merentang dengan banyak bobot terkecil yaitu pohon merentang dengan bobot 47.



Gambar 3.24 Pohon Merentang Minimum dengan Algoritma Boruvka pada Graf G dengan Banyak Sisi > 2(p - 1)

Jadi diperoleh pohon merentang minimum dengan W = {A, B, C, D, E, F, G, H} dengan bobot 47, dan dari 8 titik serta 20 sisi, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 3.

Catatan: Masih banyak kemungkinan hutan yang bisa ditetapkan, dan otomatis banyak pula kemungkinan pohon merentang minimum yang terbentuk

3.2 Penghitungan Pohon Merentang Minimum Menggunakan Algoritma Prim

3.2.1 Penghitungan Pohon Merentang Minimum pada Graf G dengan Banyak Sisi = 2(p - 1)

Untuk menentukan pohon merentang minimum dengan menggunakan Algoritma Prim dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Langkah 1 : Pilih sebarang titik awal yaitu titik A lalu dilanjutkan mengambil sisi berbobot minimum dari graf G, masukkan ke dalam *T*.

Langkah 2 : Pilih sisi dengan bobot minimum berikutnya dan bersisian dengan titik sebelumnya di T. Misalkan sisi yang dipilih adalah BG. Letakkan sisi BG ke dalam *T*.

Langkah 3 : BE adalah sisi yang dipilih selanjutnya. Letakkan sisi BE ke dalam *T*.

Langkah 4 : Masih sama dengan langkah sebelumnya, Misalkan sisi yang dipilih adalah sisi BH. Letakkan sisi BH ke dalam *T*.

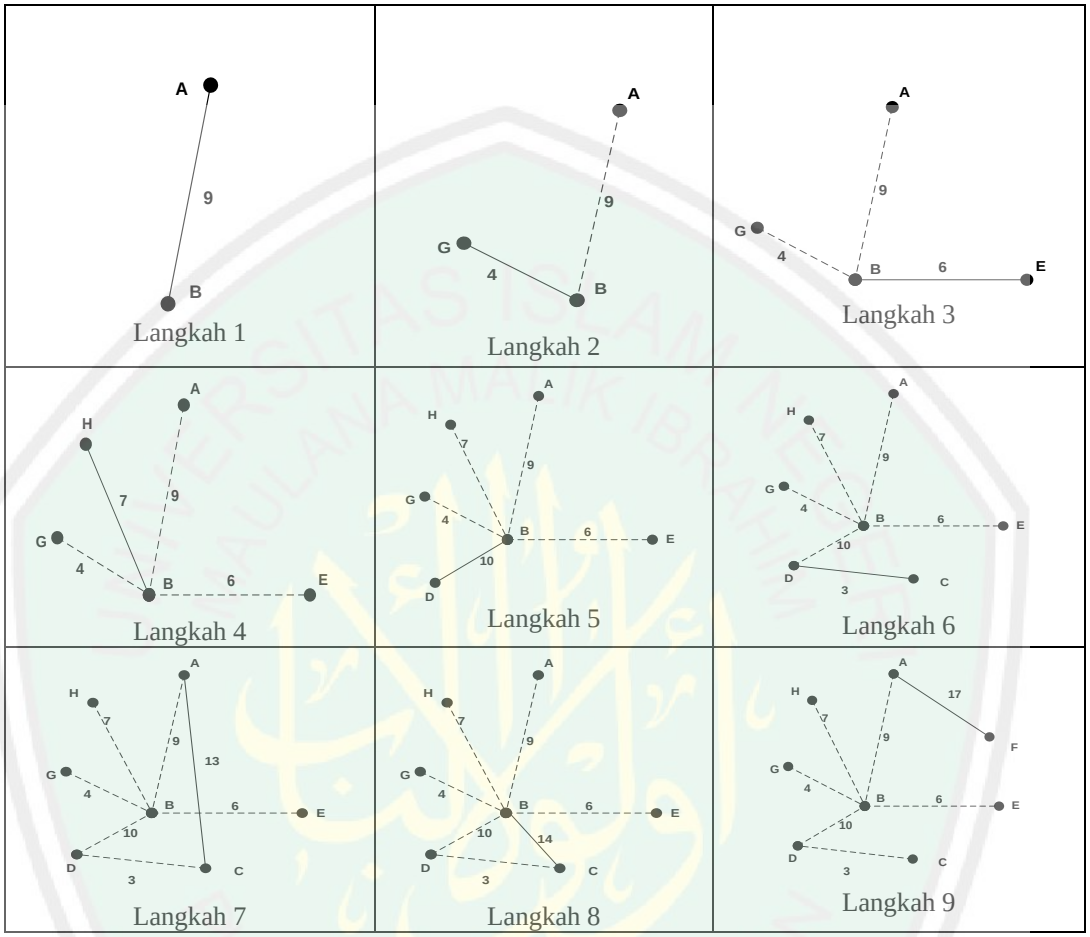
Langkah 5 : Pilih sisi dengan bobot minimal berikutnya. Misalkan sisi yang dipilih adalah BD. Letakkan sisi BD ke dalam *T*.

Langkah 6 : DC adalah sisi yang dipilih selanjutnya. Letakkan sisi DC ke dalam *T*.

Langkah 7 : Pilih sisi dengan bobot minimal berikutnya. Misalkan sisi yang dipilih adalah CA dengan bobot 13. Karena jika dimasukkan ke dalam *T* akan membentuk lintasan, maka sisi CA tidak dapat dipilih

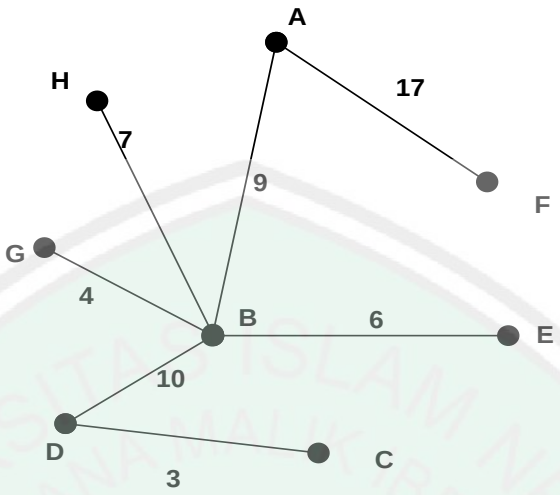
Langkah 8 : CB adalah sisi dengan bobot minimal berikutnya dengan bobot 14. Jika dimasukkan ke dalam *T* akan membentuk lintasan maka sisi CB tidak dapat dipilih.

Langkah 9 : Langkah terakhir yang digunakan adalah sama dengan langkah sebelumnya. Misalkan sisi yang dipilih AF. Letakkan sisi AF ke dalam *T*.



Gambar 3.25 Langkah-Langkah Algoritma Prim pada Graf G dengan Banyak Sisi = 2(p - 1)

Karena sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Prim pada gambar di bawah ini :



Gambar 3.26 Pohon Merentang Minimum dengan Algoritma Prim pada Graf G dengan Banyak Sisi = 2(p - 1)

Dari perhitungan Algoritma Prim di atas diperoleh pohon merentang minimum dengan jumlah bobot sebagai berikut :

W(A, B) + W(B, G) + W(B, E) + W(B, H) + W(B, D) + W(C, D) + W(A, F)

= 7 + 6 + 4 + 9 + 10 + 3 + 17 = 56

Jadi diperoleh pohon merentang minimum dengan W = {A, B, C, D, E, F, G, H} dengan bobot 56, dan dari 8 titik serta 14 sisi, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 9.

Catatan: Meskipun dalam menentukan titik awal adalah sebarang, banyak kemungkinan pohon merentang minimum yang terbentuk hanya satu.

3.2.2 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Banyak Sisi = 2(*p* - 1) dan Terdapat Sisi yang Memiliki Bobot Sama

Untuk menentukan pohon merentang minimum dengan menggunakan Algoritma Prim dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Langkah 1 : Pilih sebarang titik awal yaitu titik A lalu dilanjutkan mengambil sisi berbobot minimum dari graf G, masukkan ke dalam *T*. Misalkan sisi AB dengan bobot 3. Letakkan sisi AB ke dalam *T*.

Langkah 2 : Pilih sisi dengan bobot minimum berikutnya dan bersisian dengan titik sebelumnya di T. Misalkan sisi yang dipilih adalah AH dengan bobot 3. Letakkan sisi AH ke dalam *T*.

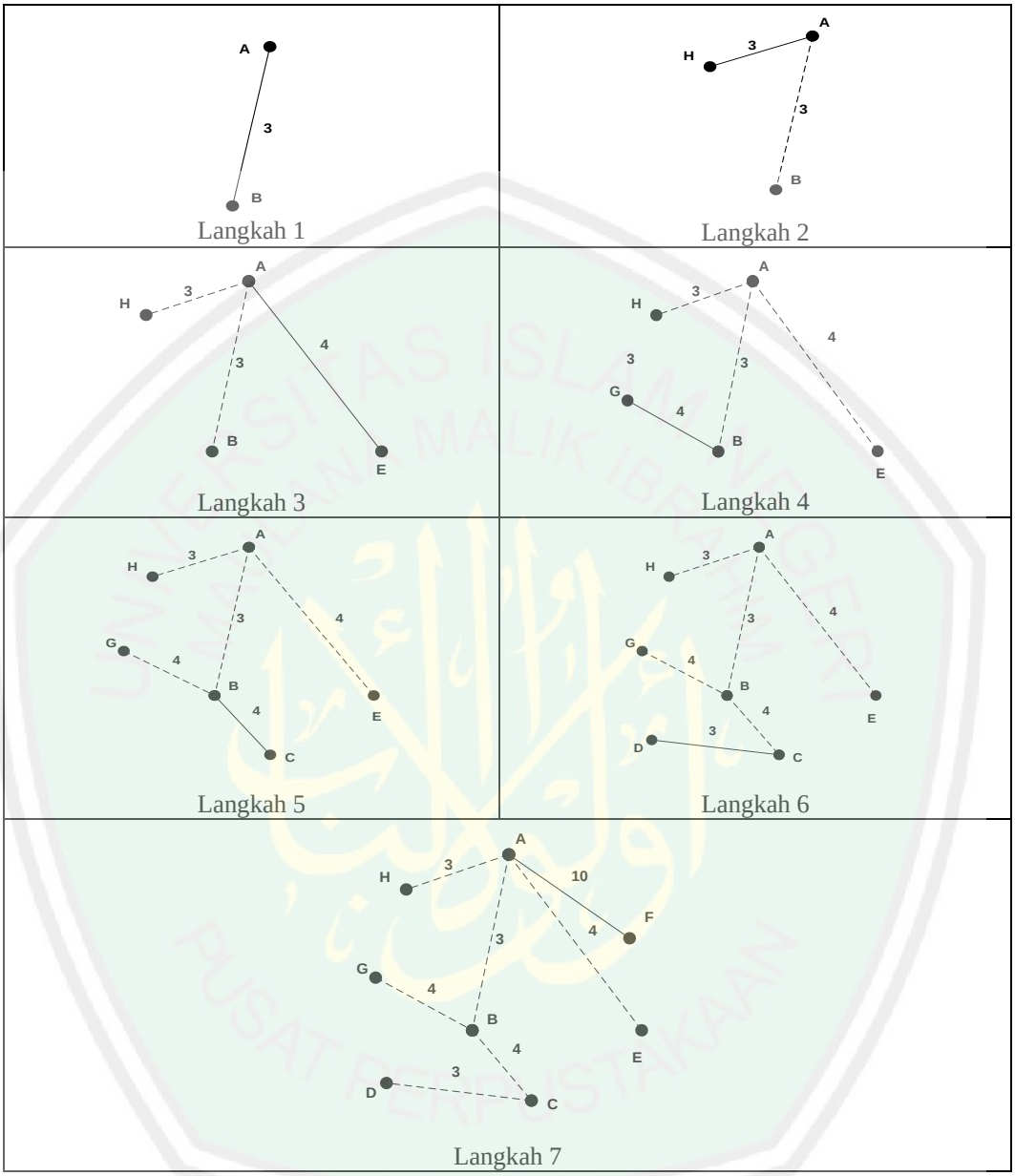
Langkah 3 : AE adalah sisi yang dipilih selanjutnya dengan bobot 4. Letakkan sisi AE ke dalam *T*.

Langkah 4 : BG adalah sisi yang dipilih selanjutnya dengan bobot 4. Letakkan sisi BG ke dalam *T*.

Langkah 5 : Masih sama dengan langkah sebelumnya, Misalkan sisi yang dipilih adalah sisi BC dengan bobot 4. Letakkan sisi BC ke dalam *T*.

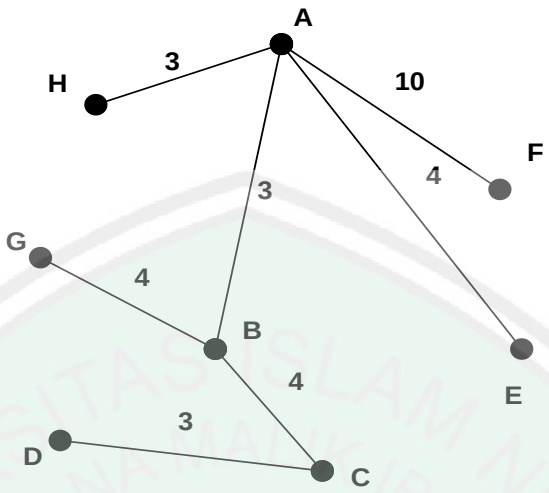
Langkah 6 : Pilih sisi dengan bobot minimal berikutnya. Misalkan sisi yang dipilih adalah CD dengan bobot 3. Letakkan sisi CD ke dalam *T*.

Langkah 7 : Langkah terakhir yang digunakan adalah sama dengan langkah sebelumnya. Misalkan sisi yang dipilih AF. Letakkan sisi AF ke dalam *T*.



Gambar 3.27 Langkah-Langkah Algoritma Prim pada Graf G dengan Banyak Sisi = 2(p - 1) dan terdapat Sisi yang Memiliki Bobot Sama

Karena sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Prim dapat dilihat pada gambar di bawah ini :



Gambar 3.28 Pohon Merentang Minimum dengan Algoritma Prim pada Graf G dengan Banyak Sisi = 2(p - 1) dan terdapat Sisi yang Memiliki Bobot Sama

Dari perhitungan Algoritma Prim di atas diperoleh pohon merentang minimum dengan jumlah bobot sebagai berikut :

W(A, B) + W(A, H) + W(A, E) + W(B, G) + W(B, C) + W(C, D) + W(A, F)

= 3 + 3 + 4 + 4 + 4 + 3 + 10 = 31

Jadi diperoleh pohon merentang minimum dengan W = {A, B, C, D, E, F, G, H} dengan bobot 31, dan dari 8 titik serta 14 sisi dan terdapat sisi yang memiliki bobot sama, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 7.

Catatan: Meskipun dalam menentukan titik awal adalah sebarang, banyak kemungkinan pohon merentang minimum yang terbentuk hanya satu.

3.2.3 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Banyak Sisi < 2(*p* - 1)

Untuk menentukan pohon merentang minimum dengan menggunakan Algoritma Prim dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Langkah 1 : Pilih sebarang titik awal yaitu titik A lalu dilanjutkan mengambil sisi berbobot minimum dari graf G, masukkan ke dalam *T*. Misalkan sisi AB dengan bobot 9. Letakkan sisi AB ke dalam *T*.

Langkah 2 : Pilih sisi dengan bobot minimum berikutnya dan bersisian dengan titik sebelumnya di T. Misalkan sisi yang dipilih adalah BG. Letakkan sisi BG ke dalam *T*.

Langkah 3 : BE adalah sisi yang dipilih selanjutnya. Letakkan sisi BE ke dalam *T*.

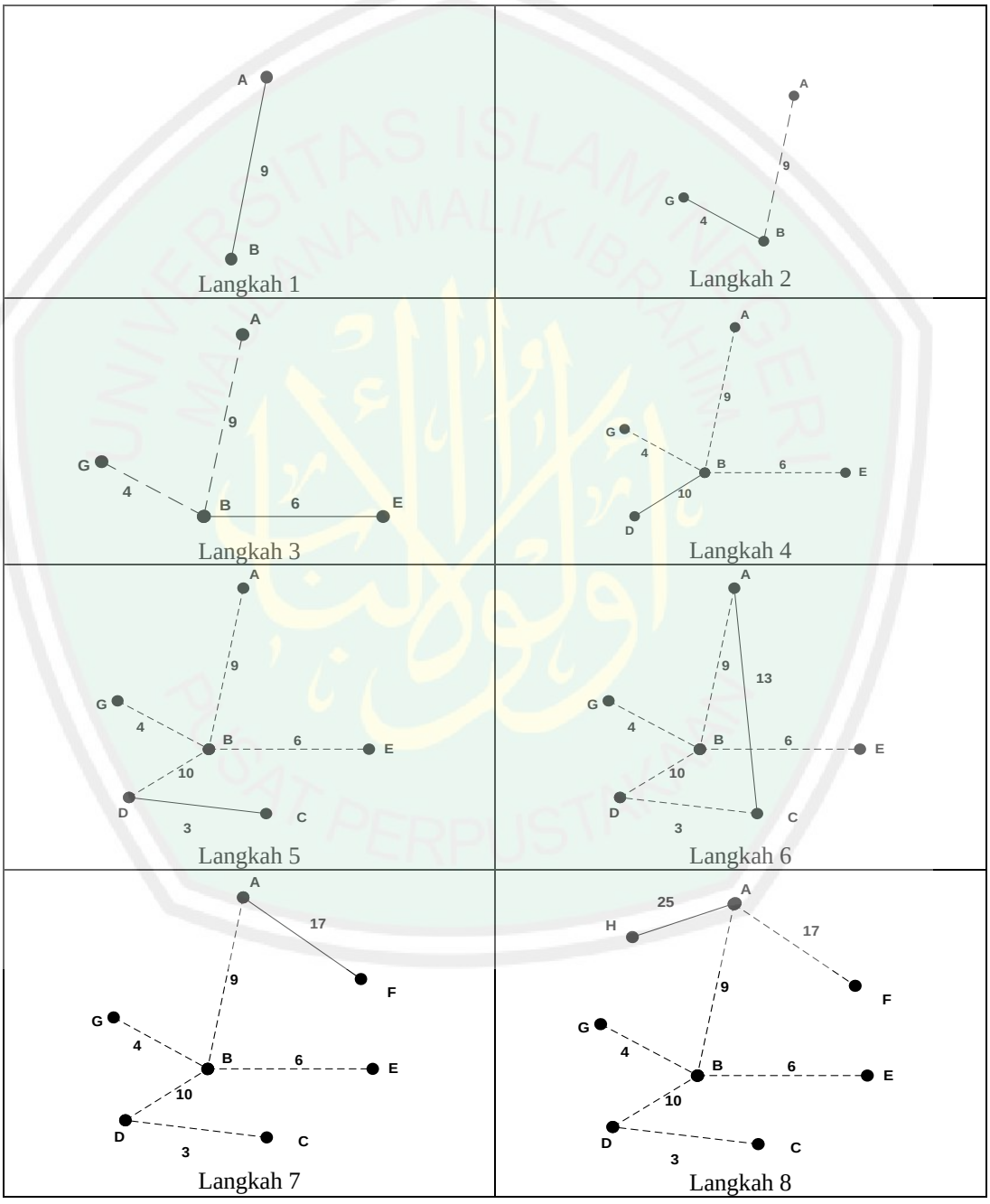
Langkah 4 : Pilih sisi dengan bobot minimal berikutnya. Misalkan sisi yang dipilih adalah BD. Letakkan sisi BD ke dalam *T*.

Langkah 5 : DC adalah sisi yang dipilih selanjutnya. Letakkan sisi DC ke dalam *T*.

Langkah 6 : Pilih sisi dengan bobot minimal berikutnya. Misalkan sisi yang dipilih adalah CA dengan bobot 13. Karena jika dimasukkan ke dalam *T* akan membentuk lintasan, maka sisi CA tidak dapat dipilih

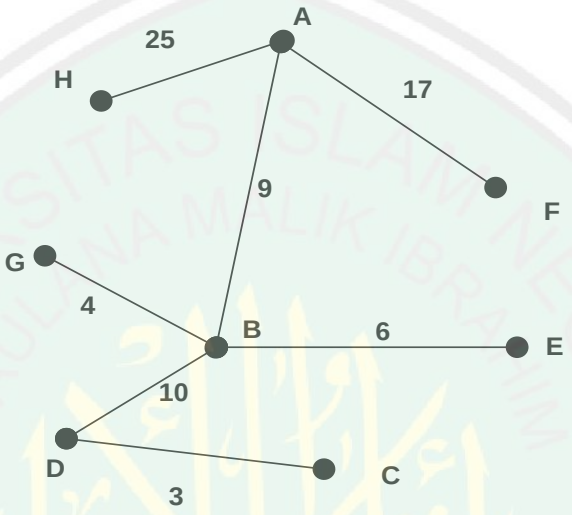
Langkah 7 : AF adalah sisi yang dipilih selanjutnya dengan bobot 17. Letakkan sisi AF ke dalam *T*.

Langkah 8 : Langkah terakhir yang digunakan adalah sama dengan langkah sebelumnya. Misalkan sisi yang dipilih AH. Letakkan sisi AH ke dalam *T*.



Gambar 3.29 Langkah-Langkah Algoritma Prim pada Graf G dengan Banyak Sisi < 2(*p* - 1)

Karena sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Prim pada gambar di bawah ini :



Gambar 3.30 Pohon Merentang Minimum dengan Algoritma Prim pada Graf G dengan Banyak Sisi < 2(p - 1)

Dari perhitungan Algoritma Prim di atas diperoleh pohon merentang minimum dengan jumlah bobot sebagai berikut :

W(A, B) + W(B, G) + W(B, E) + W(B, D) + W(D, C) + W(A, F) + W(A, H)

= 7 + 6 + 4 + 9 + 3 + 17 + 25= 74

Jadi diperoleh pohon merentang minimum dengan W = {A, B, C, D, E, F, G, H} dengan bobot 74, dan dari 8 titik serta 11 sisi, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 8.

Catatan: Meskipun dalam menentukan titik awal adalah sebarang, banyak kemungkinan pohon merentang minimum yang terbentuk hanya satu.

3.2.4 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Banyak Sisi > 2(*p* - 1)

Untuk menentukan pohon merentang minimum dengan menggunakan Algoritma Prim dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Langkah 1 : Pilih sebarang titik awal yaitu titik A lalu dilanjutkan mengambil sisi berbobot minimum dari graf G, masukkan ke dalam *T*. Misalkan sisi AB dengan bobot 9. Letakkan sisi AB ke dalam *T*.

Langkah 2 : Pilih sisi dengan bobot minimum berikutnya dan bersisian dengan titik sebelumnya di T. Misalkan sisi yang dipilih adalah BG. Letakkan sisi BG ke dalam *T*.

Langkah 3 : BE adalah sisi yang dipilih selanjutnya. Letakkan sisi BE ke dalam *T*.

Langkah 4 : Masih sama dengan langkah sebelumnya, Misalkan sisi yang dipilih adalah sisi BH. Letakkan sisi BH ke dalam *T*.

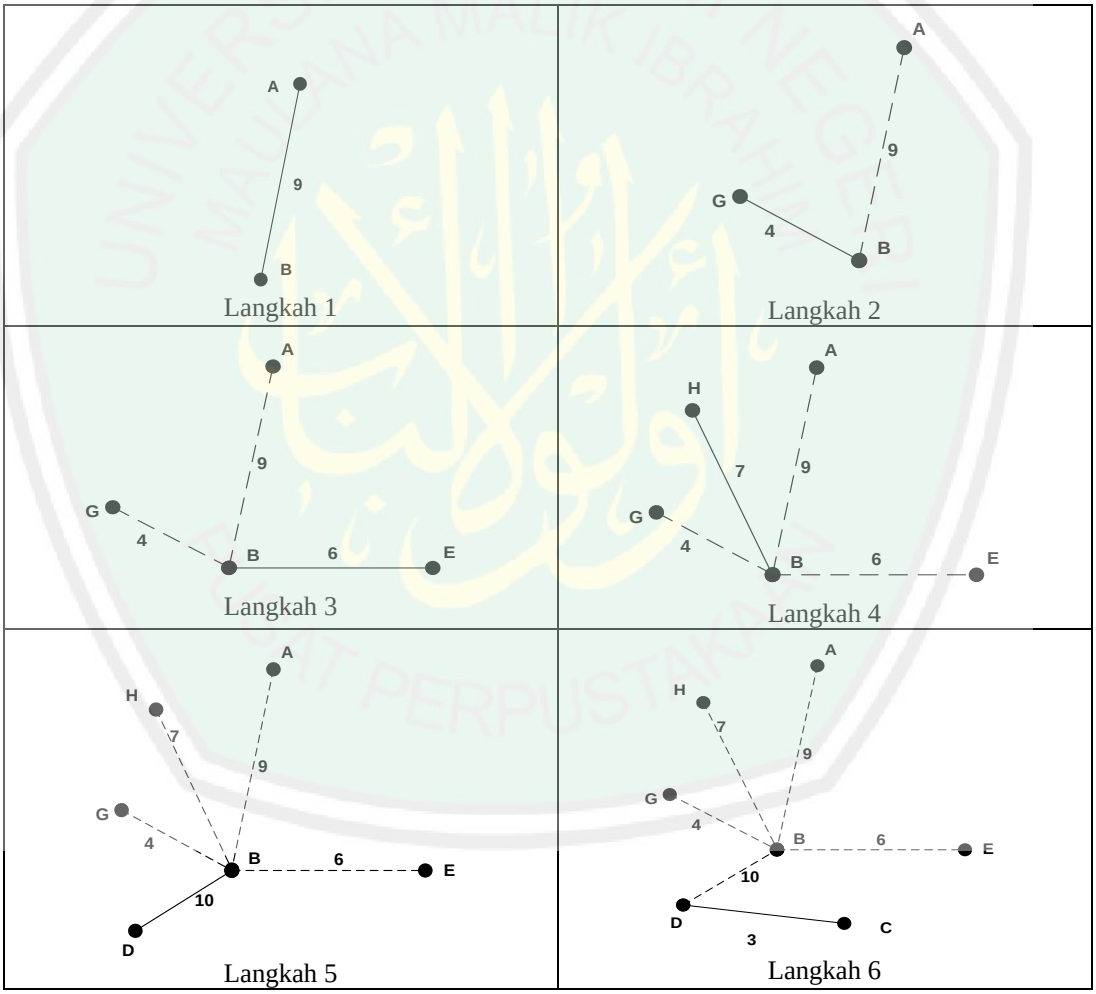
Langkah 5 : Pilih sisi dengan bobot minimal berikutnya. Misalkan sisi yang dipilih adalah BD. Letakkan sisi BD ke dalam *T*.

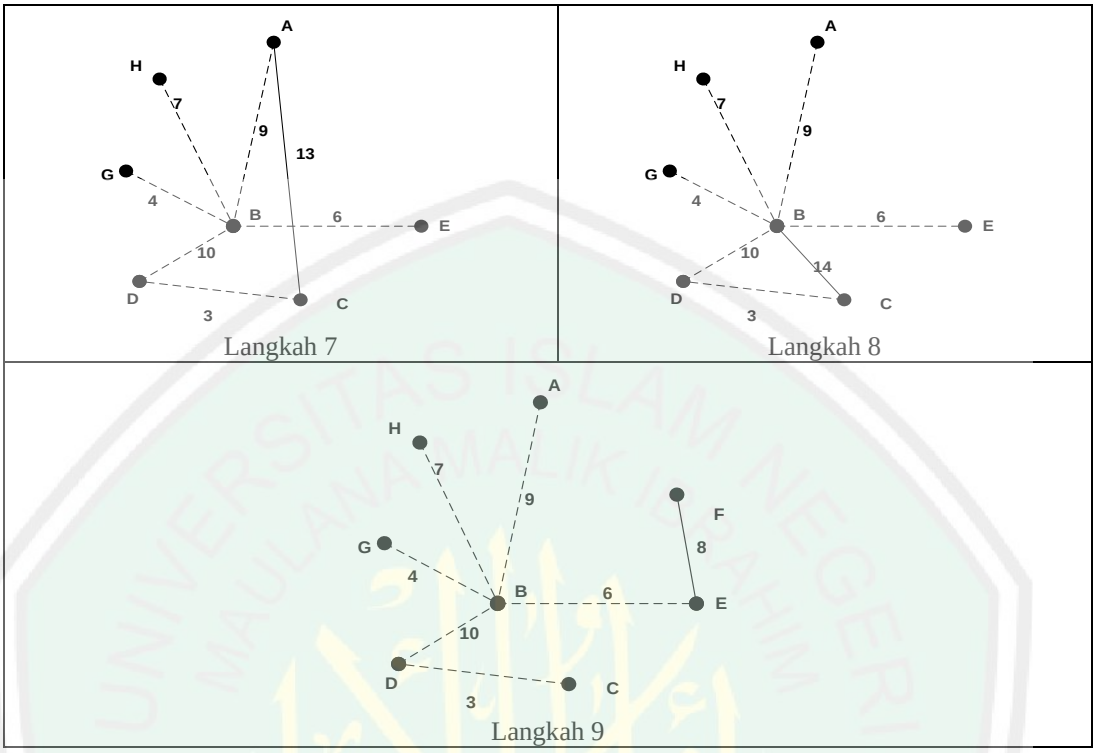
Langkah 6 : DC adalah sisi yang dipilih selanjutnya. Letakkan sisi DC ke dalam *T*.

Langkah 7 : Pilih sisi dengan bobot minimal berikutnya. Misalkan sisi yang dipilih adalah CA dengan bobot 13. Karena jika dimasukkan ke dalam *T* akan membentuk lintasan (C, A, B, D, C), maka sisi CA tidak dapat dipilih.

Langkah 8 : CB adalah sisi dengan bobot minimal berikutnya dengan bobot 14. Jika dimasukkan ke dalam *T* akan membentuk lintasan (C, B, D, C) maka sisi CB tidak dapat dipilih.

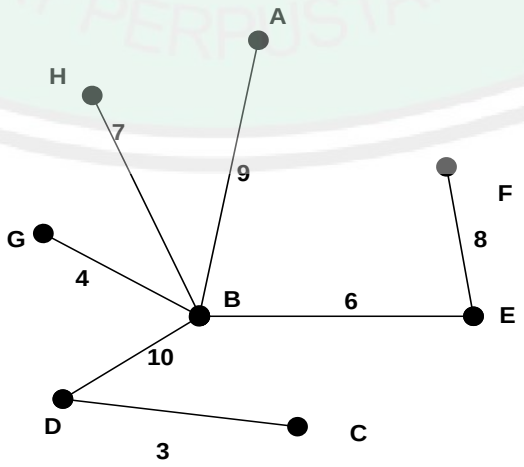
Langkah 9 : Langkah terakhir yang digunakan adalah sama dengan langkah sebelumnya. Misalkan sisi yang dipilih EF dengan bobot 8. Letakkan sisi EF ke dalam *T*.





Gambar 3.31 Langkah-Langkah Algoritma Prim pada Graf G dengan Banyak Sisi > 2(p - 1)

Karena sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Prim pada gambar di bawah ini :



Gambar 3.32 Pohon Merentang Minimum dengan Algoritma Prim pada Graf G dengan Banyak Sisi > 2(p - 1)

Dari perhitungan Algoritma Prim di atas diperoleh pohon merentang minimum dengan jumlah bobot sebagai berikut :

W(A, B) + W(B, G) + W(B, E) + W(B, H) + W(B, D) + W(C, D) + W(E, F)

= 7 + 6 + 4 + 9 + 10 + 3 + 8 = 47

Jadi diperoleh pohon merentang minimum dengan W = {A, B, C, D, E, F, G, H} dengan bobot 47, dan dari 8 titik serta 20 sisi, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 9.

Catatan: Meskipun dalam menentukan titik awal adalah sebarang, banyak kemungkinan pohon merentang minimum yang terbentuk hanya satu.

3.3 Penghitungan Pohon Merentang Minimum Menggunakan Algoritma Kruskal

3.3.1 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Banyak Sisi = 2(*p* - 1)

Untuk menentukan pohon merentang minimum dengan menggunakan Algoritma Kruskal dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Misalkan :

e_1 = (C, D), e_2 = (B, G), e_3 = (B, E), e_4 =(B, H), e_5 = (A, B), e_6 = (B, D), e_7 = (A, C), e_8 = (B, C), e_9 = (A, F), e_{13} = (A, E), e_{10} = (A, H), e_{11} = (A, G), e_{12} = (C, E), e_{14} = (A, D)

Langkah 1 : Dipilih sisi yang berbobot paling minimum, yaitu sisi CD karena berbobot 3

Langkah 2 : Pilih sisi berbobot minimum berikutnya. Misalkan sisi yang dipilih adalah sisi BG dengan bobot 4.

Langkah 3 : Masih sama dengan langkah sebelumnya. Misalkan sisi yang dipilih adalah sisi BE dengan bobot 6.

Langkah 4 : BH adalah sisi yang dipilih berikutnya karena mempunyai bobot minimum 7.

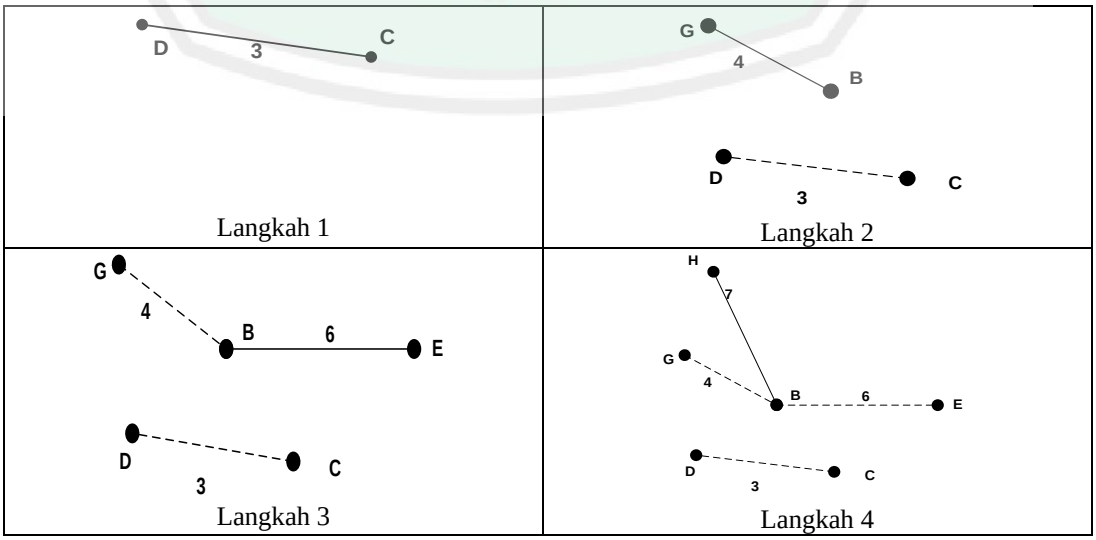
Langkah 5 : Selanjutnya sisi yang dipilih adalah AB dengan bobot 9.

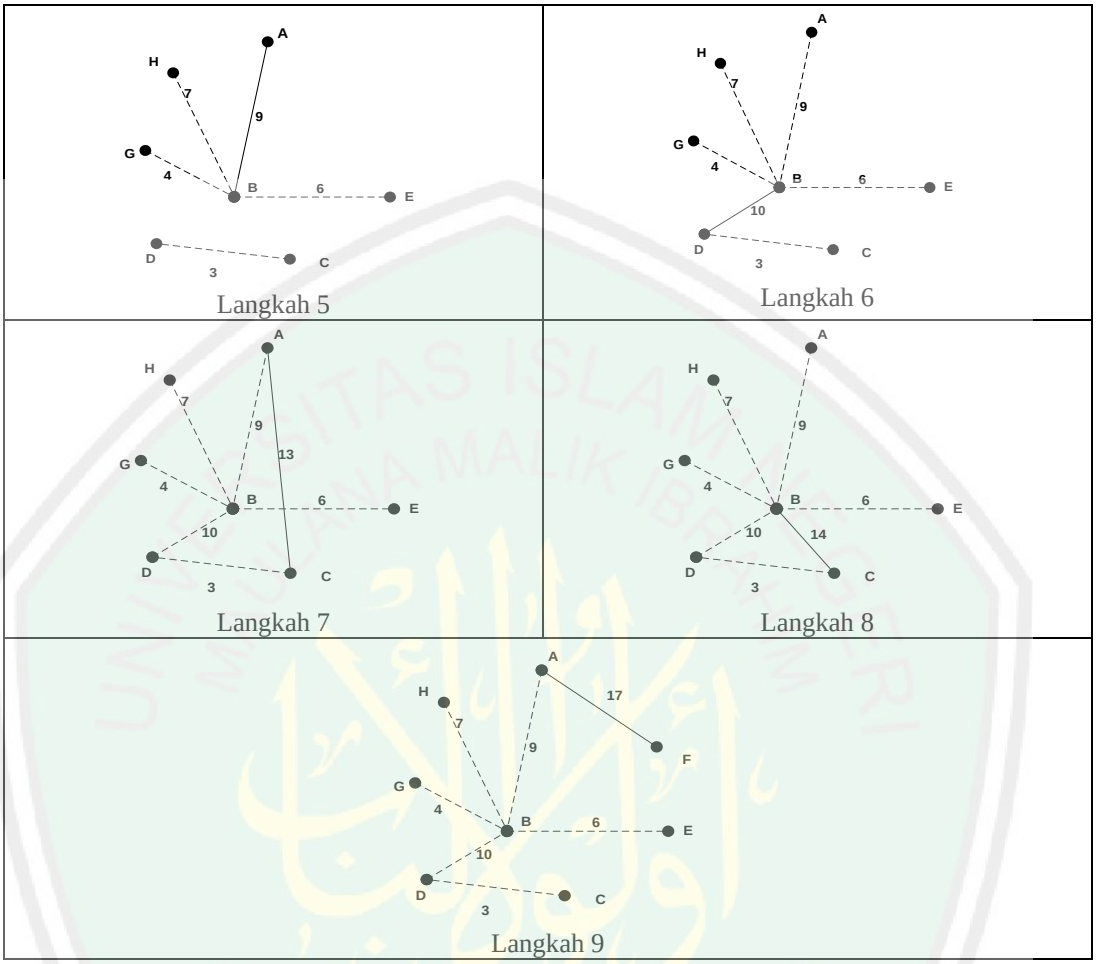
Langkah 6 : BD adalah sisi berbobot minimum berikutnya yang dipilih dengan bobot 10.

Langkah 7 : AC adalah sisi minimum berikutnya yang dipilih dengan bobot 13. Karena jika dimasukkan ke dalam *T* akan menghasilkan lintasan (A, C, D, B, A) maka sisi AC tidak dapat dipilih.

Langkah 8 : Sisi yang dipilih selanjutnya adalah sisi BC dengan bobot 14. Tetapi ternyata jika dimasukkan ke dalam *T* akan menghasilkan lintasan (B, C, D, B).

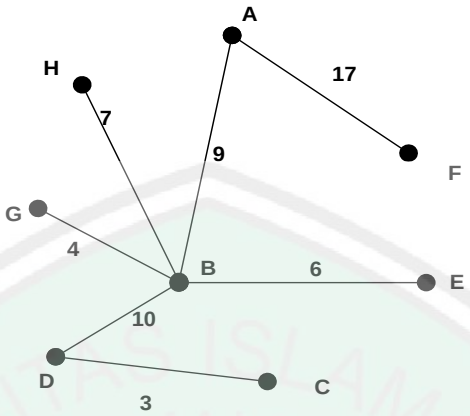
Langkah 9 : Selanjutnya sisi yang dipilih adalah AF dengan bobot 17.





Gambar 3.33 Langkah-Langkah Algoritma Kruskal pada Graf G dengan Banyak Sisi = 2(p - 1)

Karena sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Kruskal dapat dilihat pada gambar di bawah ini :



Gambar 3.34 Pohon Merentang Minimum dengan Algoritma Kruskal pada Graf G dengan Banyak Sisi = 2(p - 1)

Dari perhitungan Algoritma Kruskal di atas diperoleh pohon merentang minimum dengan jumlah bobot sebagai berikut :

$w(e_1) + w(e_2) + w(e_3) + w(e_4) + w(e_5) + w(e_6) + w(e_9) =$

$3 + 4 + 6 + 7 + 9 + 10 + 17 = 56$

Jadi diperoleh pohon merentang minimum dengan $T = \{ e_1, e_2, e_3, e_4, e_5, e_6, e_9 \}$ dengan bobot 56, dan dari 8 titik serta 14 sisi, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 9.

Catatan: Karena dalam mencari pohon merentang minimum langkahnya adalah dilakukan pengurutan terlebih dahulu pada setiap bobot dari terkecil hingga terbesar, dan dilanjutkan dengan menambahkan bobot yang telah diurutkan tadi hingga terbentuk pohon merentang minimum. Jadi banyak kemungkinan pohon merentang minimum yang dapat terbentuk hanya satu.

3.3.2 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Banyak Sisi = 2(*p* - 1) dan Terdapat Sisi yang Memiliki Bobot Sama

Untuk menentukan pohon merentang minimum dengan menggunakan Algoritma Kruskal dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Misalkan :

e_1 = (A, B), e_2 = (A, H), e_3 = (C, D), e_4 =(A, E), e_5 = (B, C), e_6 = (B, G), e_7 = (A, G), e_8 = (A, C), e_9 = (B, E), e_{10} = (B, H), e_{11} = (C, E), e_{12} = (A, F), e_{13} = (A, D), e_{14} = (B, D)

Langkah 1 : Dipilih sisi yang berbobot paling minimum, yaitu sisi AB karena berbobot 3

Langkah 2 : Pilih sisi berbobot minimum berikutnya. Misalkan sisi yang dipilih adalah sisi AH dengan bobot 3.

Langkah 3 : Masih sama dengan langkah sebelumnya. Misalkan sisi yang dipilih adalah sisi CD dengan bobot 3.

Langkah 4 : AE adalah sisi yang dipilih berikutnya karena mempunyai bobot minimum 4.

Langkah 5 : Selanjutnya sisi yang dipilih adalah BC dengan bobot 4.

Langkah 6 : BG adalah sisi berbobot minimum berikutnya yang dipilih dengan bobot 4.

Langkah 7 : AG adalah sisi minimum berikutnya yang dipilih dengan bobot 6. Karena jika dimasukkan ke dalam *T* akan menghasilkan lintasan (A, G, B, A) maka sisi AG tidak dapat dipilih.

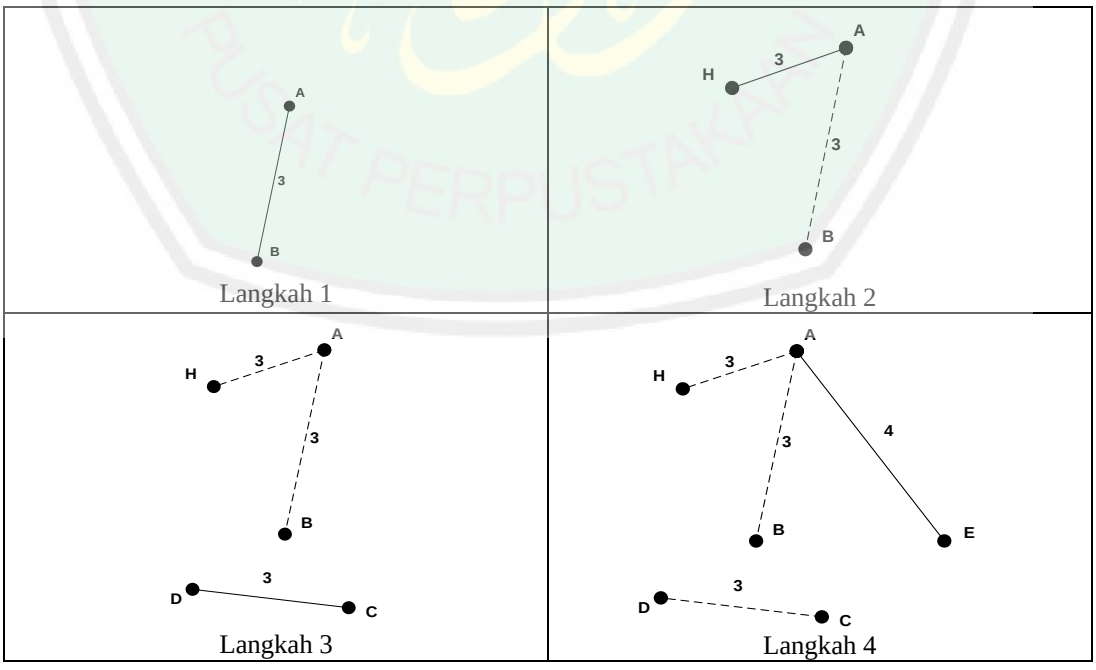
Langkah 8 : AC adalah sisi minimum berikutnya yang dipilih dengan bobot 6, tetapi ternyata jika dimasukkan ke dalam *T* akan menghasilkan lintasan (A, C, B, A) maka sisi AC Tidak dapat dipilih.

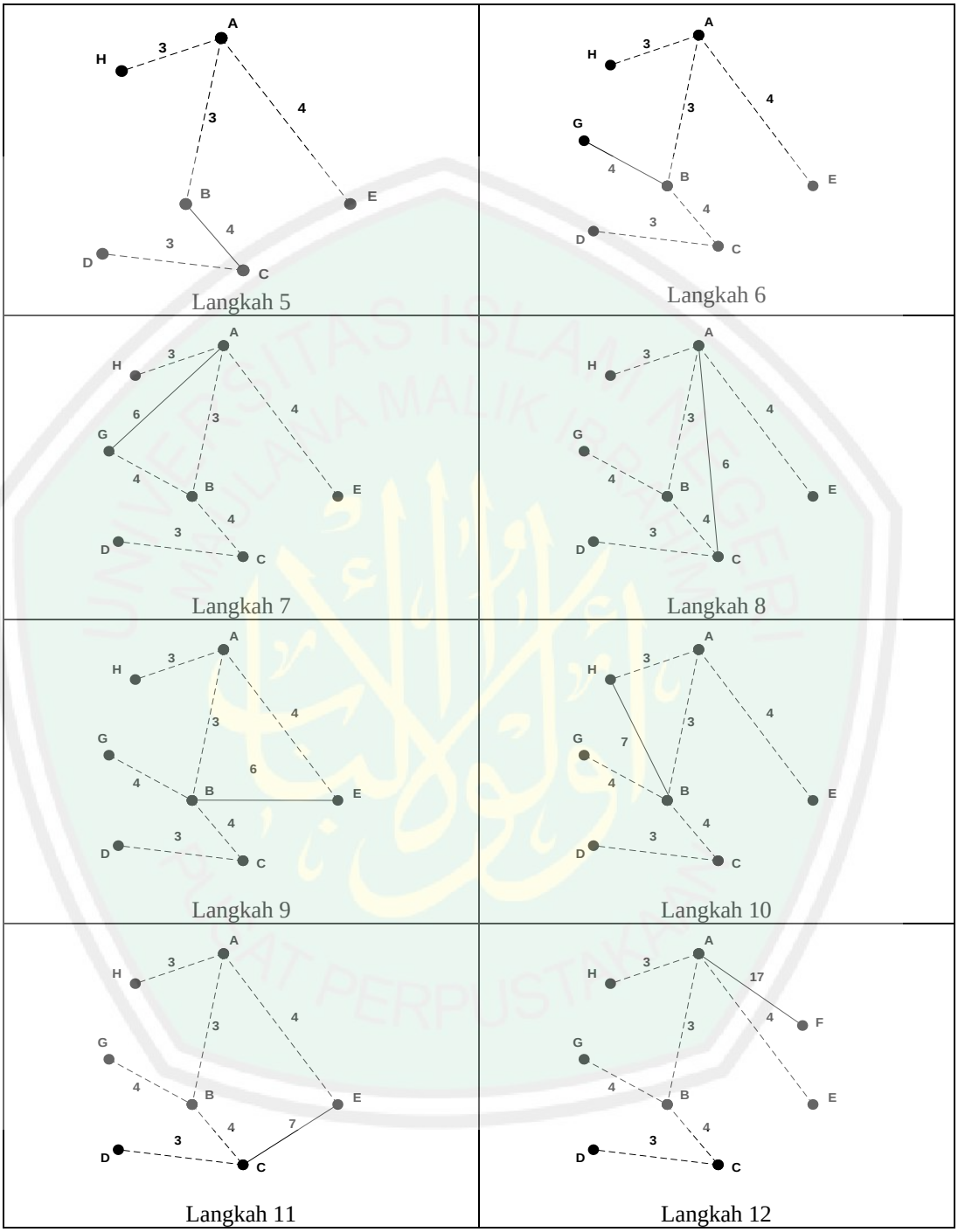
Langkah 9 : Selanjutnya adalah BE dengan bobot 6, masih tidak dapat dipilih karena jika dimasukkan ke dalam *T* akan menghasilkan lintasan (B, E, A, B).

Langkah 10 : Sisi yang dipilih selanjutnya adalah BH dengan bobot 7. Karena jika dimasukkan ke dalam *T* akan menghasilkan lintasan (B, H, A, B) maka sisi BH tidak dapat dipilih.

Langkah 11 : Selanjutnya adalah CE dengan bobot 7. Tetapi ternyata jika dimasukkan ke dalam *T* akan menghasilkan lintasan (C, E, A, B, C).

Langkah 12 : Sisi yang dipilih selanjutnya adalah AF dengan bobot 10.

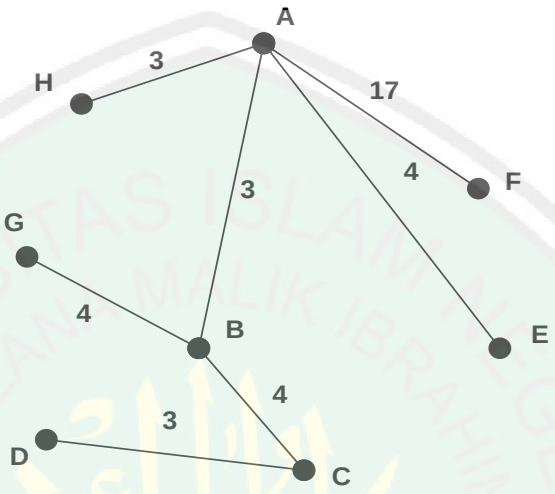




Gambar 3.35 Langkah-Langkah Algoritma Kruskal pada Graf G dengan Banyak Sisi = 2(p - 1) dan terdapat Sisi yang Memiliki Bobot Sama

Karena sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari

perhitungan menggunakan Algoritma Kruskal dapat dilihat pada gambar di bawah ini :



Gambar 3.36 Pohon Merentang Minimum dengan Algoritma Kruskal pada Graf G dengan Banyak Sisi = 2(p - 1) dan terdapat Sisi yang Memiliki Bobot Sama

Dari perhitungan Algoritma Kruskal di atas diperoleh pohon merentang minimum dengan jumlah bobot sebagai berikut :

$w(e_1) + w(e_2) + w(e_3) + w(e_4) + w(e_5) + w(e_6) + w(e_{12})$

$= 3 + 3 + 3 + 4 + 4 + 4 + 10 = 31$

Jadi diperoleh pohon merentang minimum dengan $T = \{ e_1, e_2, e_3, e_4, e_5, e_6, e_{12} \}$ dengan bobot 31, dan dari 8 titik serta 14 sisi dan terdapat sisi yang memiliki bobot sama, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 12.

Catatan: Karena dalam mencari pohon merentang minimum langkahnya adalah dilakukan pengurutan terlebih dahulu pada setiap bobot dari terkecil hingga terbesar, dan dilanjutkan dengan menambahkan bobot yang telah diurutkan tadi

hingga terbentuk pohon merentang minimum. Jadi banyak kemungkinan pohon merentang minimum yang dapat terbentuk hanya satu.

3.3.3 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Banyak Sisi $< 2(p - 1)$

Untuk menentukan pohon merentang minimum dengan menggunakan Algoritma Kruskal dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Misalkan :

e_1 = (C, D), e_2 = (B, G), e_3 = (B, E), e_4 = (A, B), e_5 = (B, D), e_6 = (A, C), e_7 = (A, F), e_8 = (A, E, e_9 = (A, H), e_{10} = (A, G), e_{11} = (C, E).

Langkah 1 : Dipilih sisi yang berbobot paling minimum, yaitu sisi CD karena berbobot 3

Langkah 2 : Pilih sisi berbobot minimum berikutnya. Misalkan sisi yang dipilih adalah sisi BG dengan bobot 4.

Langkah 3 : Masih sama dengan langkah sebelumnya. Misalkan sisi yang dipilih adalah sisi BE dengan bobot 6.

Langkah 4 : Selanjutnya sisi yang dipilih adalah AB dengan bobot 9.

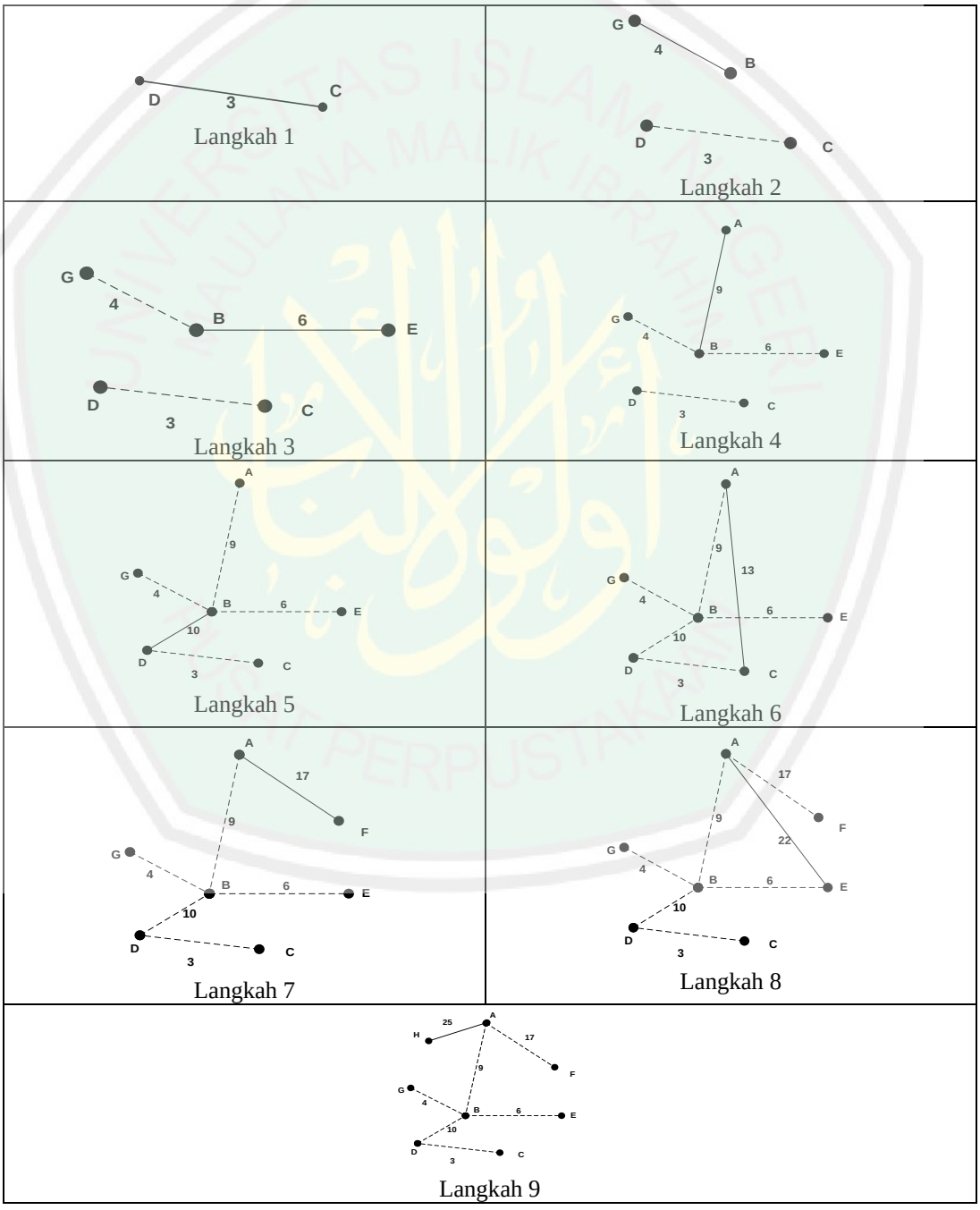
Langkah 5 : BD adalah sisi berbobot minimum berikutnya yang dipilih dengan bobot 10.

Langkah 6 : AC adalah sisi minimum berikutnya yang dipilih dengan bobot 13. Karena jika dimasukkan ke dalam *T* akan menghasilkan lintasan (A, C, D, B, A) maka sisi AC tidak dapat dipilih.

Langkah 7 : Selanjutnya sisi yang dipilih adalah AF dengan bobot 17.

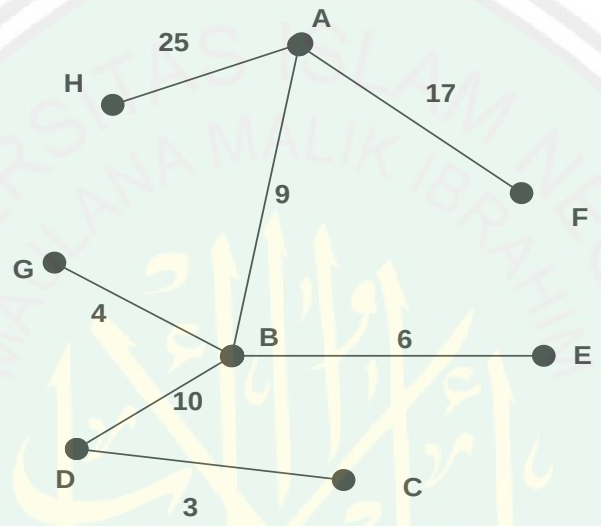
Langkah 8 : AE adalah sisi minimum berikutnya yang dipilih dengan bobot 22. Karena jika dimasukkan ke dalam *T* akan menghasilkan lintasan (A, E, B, A) maka sisi AE tidak dapat dipilih.

Langkah 9 : AH adalah sisi yang dipilih selanjutnya dengan bobot 25.



Gambar 3.37 Langkah-Langkah Algoritma Kruskal pada Graf G dengan Banyak Sisi < 2(*p* - 1)

Karena sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Kruskal dapat dilihat pada gambar di bawah ini :



Gambar 3.38 Pohon Merentang Minimum dengan Algoritma Kruskal pada Graf G dengan Banyak Sisi < 2(p - 1)

Dari perhitungan Algoritma Kruskal di atas diperoleh pohon merentang minimum dengan jumlah bobot sebagai berikut :

$w(e_1) + w(e_2) + w(e_3) + w(e_4) + w(e_5) + w(e_7) + w(e_9)$

$= 3 + 4 + 6 + 9 + 10 + 17 + 25 = 74$

Jadi diperoleh pohon merentang minimum dengan $T = \{ e_1, e_2, e_3, e_4, e_5, e_7, e_9 \}$ dengan bobot 74, dan dari 8 titik serta 11 sisi, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 9.

Catatan: Karena dalam mencari pohon merentang minimum langkahnya adalah dilakukan pengurutan terlebih dahulu pada setiap bobot dari terkecil hingga terbesar, dan dilanjutkan dengan menambahkan bobot yang telah diurutkan tadi hingga terbentuk pohon merentang minimum. Jadi banyak kemungkinan pohon merentang minimum yang dapat terbentuk hanya satu.

3.3.4 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Sisi yang Memiliki Bobot Sama

Untuk menentukan pohon merentang minimum dengan menggunakan Algoritma Kruskal dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Misalkan :

e_1 = (C, D), e_2 = (B, G), e_3 = (B, E), e_4 =(B, H), e_5 = (E, F), e_6 = (A, B), e_7 = (B, D), e_8 = (D, E), e_9 = (A, C), e_{10} = (B, C), e_{11} = (B, F), e_{12} = (A, F), e_{13} = (C, F), e_{14} = (A, E), e_{15} = (A, H), e_{16} = (A, G), e_{17} = (C, E), e_{18} = (F, G), e_{19} = (A, D), e_{20} = (E, H)

Langkah 1 : Dipilih sisi yang berbobot paling minimum, yaitu sisi CD karena berbobot 3

Langkah 2 : Pilih sisi berbobot minimum berikutnya. Misalkan sisi yang dipilih adalah sisi BG dengan bobot 4.

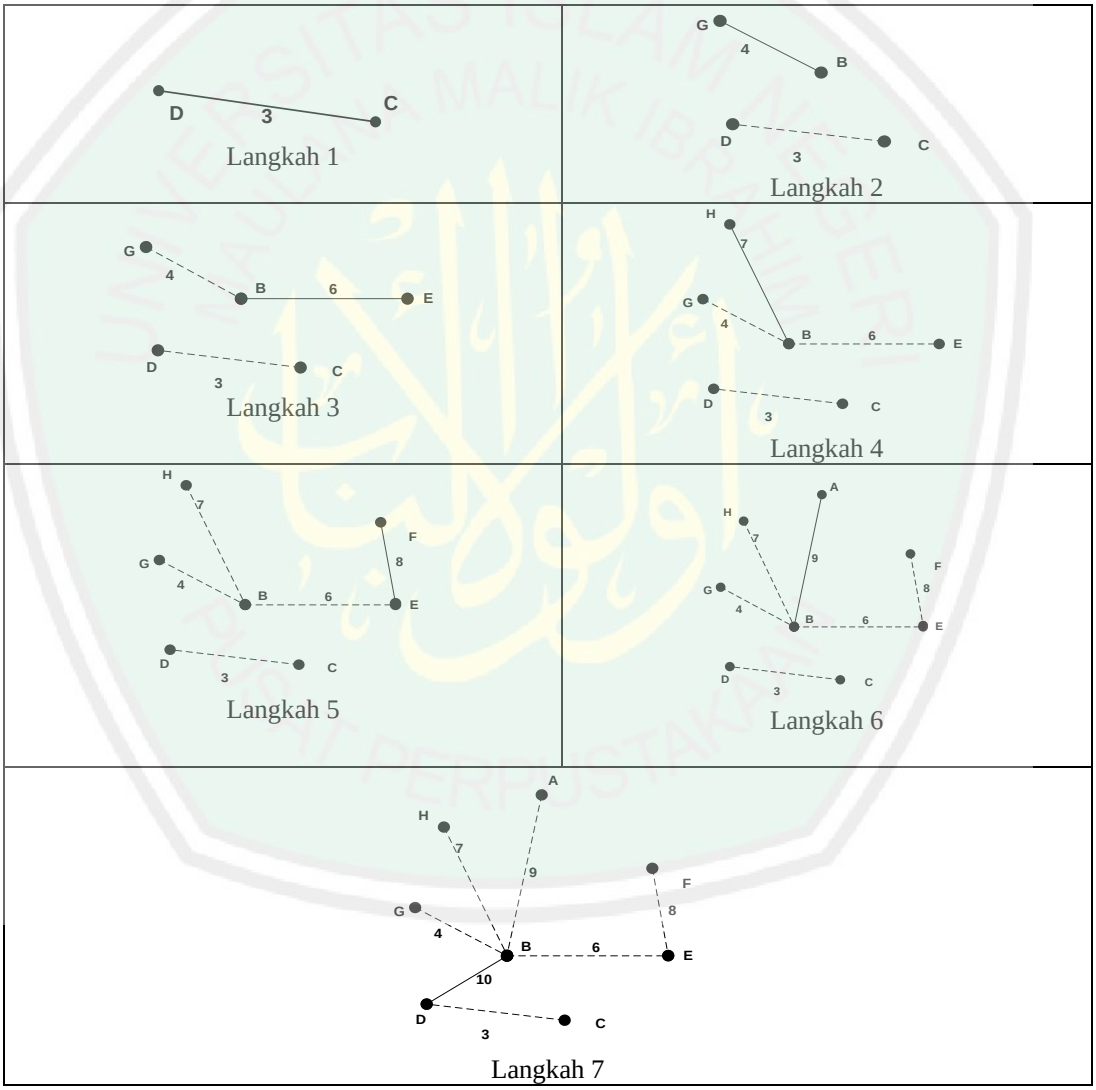
Langkah 3 : Masih sama dengan langkah sebelumnya. Misalkan sisi yang dipilih adalah sisi BE dengan bobot 6.

Langkah 4 : BH adalah sisi yang dipilih berikutnya karena mempunyai bobot minimum 7.

Langkah 5 : Selanjutnya sisi yang dipilih adalah EF dengan bobot 8.

Langkah 6 : Pilih sisi berbobot minimum berikutnya. Misalkan sisi yang dipilih adalah sisi AB dengan bobot 9.

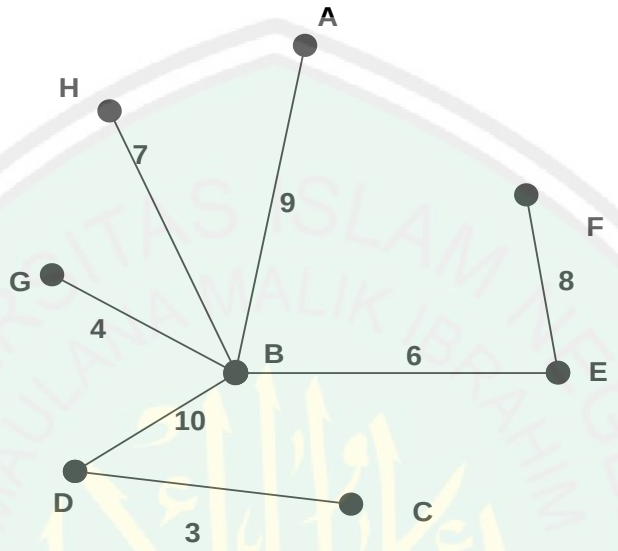
Langkah 7 : BD adalah sisi berbobot minimum berikutnya yang dipilih dengan bobot 10.



Gambar 3.39 Langkah-Langkah Algoritma Kruskal pada Graf G dengan Banyak Sisi > 2(p - 1)

Karena sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari

perhitungan menggunakan Algoritma Kruskal dapat dilihat pada gambar di bawah ini :



Gambar 3.40 Pohon Merentang Minimum dengan Algoritma Kruskal pada Graf G dengan Banyak Sisi >2(p - 1)

Dari perhitungan Algoritma Kruskal di atas diperoleh pohon merentang minimum dengan jumlah bobot sebagai berikut :

$w(e_1) + w(e_2) + w(e_3) + w(e_4) + w(e_5) + w(e_6) + w(e_7)$

$= 3 + 4 + 6 + 7 + 8 + 9 + 10 = 47$

Jadi diperoleh pohon merentang minimum dengan $T = \{ e_1, e_2, e_3, e_4, e_5, e_6, e_7 \}$ dengan bobot 47, dan dari 8 titik serta 20 sisi, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 7.

Catatan: Karena dalam mencari pohon merentang minimum langkahnya adalah dilakukan pengurutan terlebih dahulu pada setiap bobot dari terkecil hingga terbesar, dan dilanjutkan dengan menambahkan bobot yang telah diurutkan tadi

hingga terbentuk pohon merentang minimum. Jadi banyak kemungkinan pohon merentang minimum yang dapat terbentuk hanya satu.

3.4 Penghitungan Pohon Merentang Minimum Menggunakan Algoritma Sollin

3.4.1 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Sisi yang Memiliki Bobot Sama

Untuk menentukan pohon merentang minimum dengan menggunakan Algoritma Sollin dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Mula-mula kita buat dulu tabel yang berisikan sisi-sisi yang terurut dari besar ke kecil.

Tabel 3.5 Urutan Sisi dari Bobot Terbesar Hingga Terkecil Graf G dengan Banyak Sisi = 2(p - 1)

Bobot	Sisi
29	(A, D)
27	(C, E)
26	(A, G)
25	(A, H)
22	(A, E)
17	(A, F)
14	(B, C)
13	(A, C)
10	(B, D)

9	(A, B)
7	(B, H)
6	(B, E)
4	(B, G)
3	(C, D)

Kemudian kita lakukan penghapusan sisi dimulai dari yang memiliki bobot terbesar dan tidak membuat graf menjadi tidak terhubung.

Langkah 1 : Dihapus sisi yang berbobot paling maksimum, yaitu sisi AD karena berbobot 29.

Langkah 2 : Hapus sisi berbobot maksimum berikutnya. Misalkan sisi yang dihapus adalah sisi CE dengan bobot 27.

Langkah 3 : Masih sama dengan langkah sebelumnya. Misalkan sisi yang dihapus adalah AG dengan bobot 26.

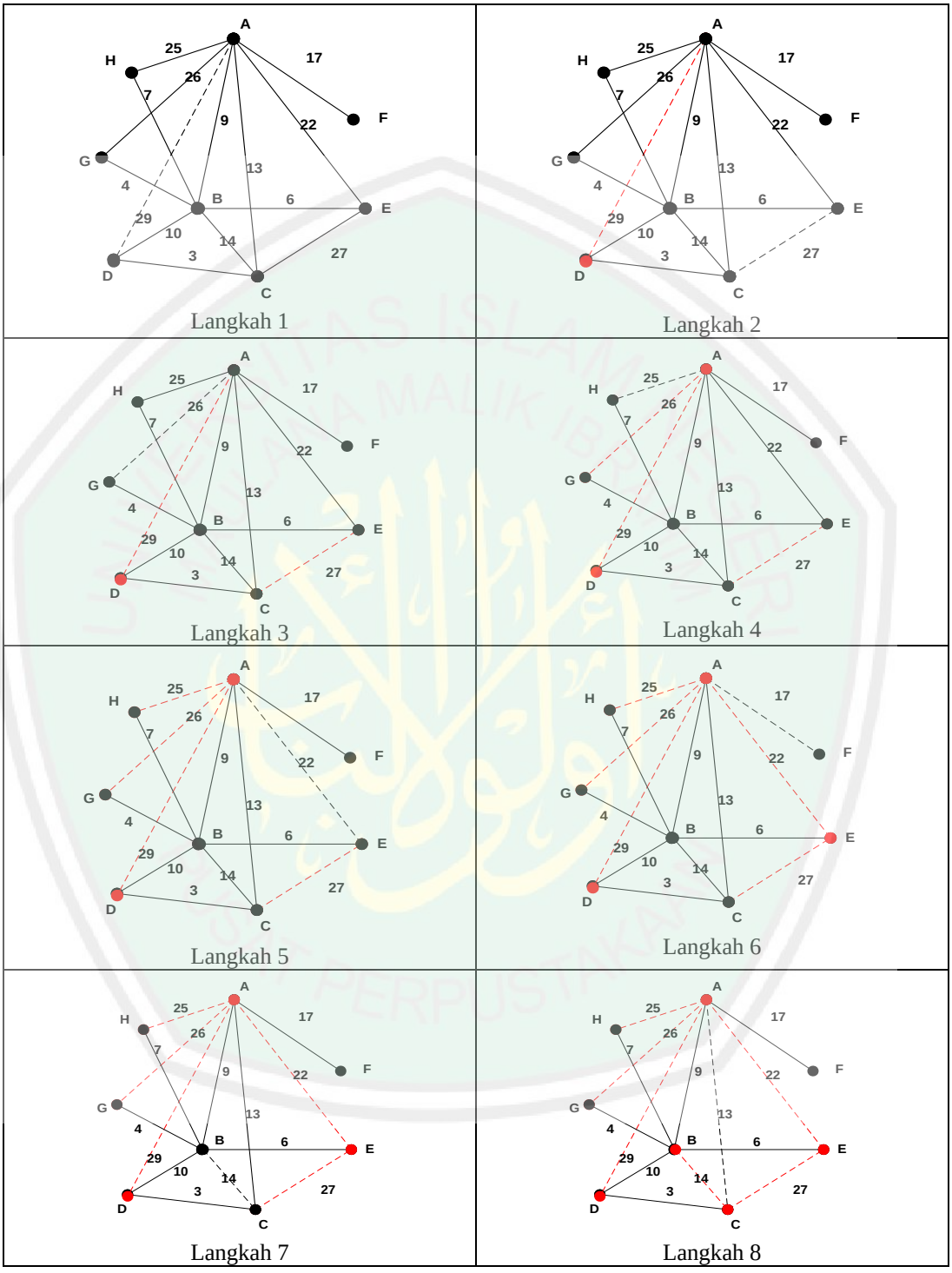
Langkah 4 : AH adalah sisi yang dihapus selanjutnya karena mempunyai bobot maksimum 25.

Langkah 5 : Selanjutnya sisi yang dihapus adalah AE dengan bobot 22.

Langkah 6 : AF adalah sisi maksimum selanjutnya dengan bobot 17, tetapi tidak bisa dihapus karena akan menyebabkan graf manjadi tidak terhubung (titik F akan menjadi titik terpencil).

Langkah 7 : Sisi yang dihapus selanjutnya adalah BC dengan bobot 14.

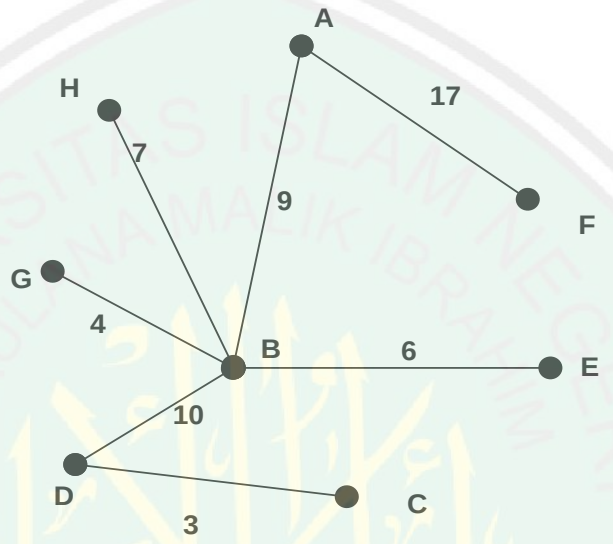
Langkah 8 : Hapus sisi berbobot maksimum berikutnya. Misalkan sisi yang dihapus adalah sisi AC dengan bobot 13.



Gambar 3.41 Langkah-Langkah Algoritma Sollin pada Graf G dengan Banyak Sisi = 2(p - 1)

Karena sudah tidak terdapat lagi sisi yang dapat dihapus dan itu artinya semua titik sudah terhubung dan sudah diperoleh pohon merentang minimumnya,

maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Sollin dapat dilihat pada gambar di bawah ini :



Gambar 3.42 Pohon Merentang Minimum dengan Algoritma Sollin pada Graf G dengan Banyak Sisi =2(p - 1)

Dari perhitungan Algoritma Sollin di atas diperoleh pohon merentang minimum dengan jumlah bobot sebagai berikut :

W(C, D) + W(B, G) + W(B, E) + W(B, H) + W(A, B) + W(B, D) + W(A, F)

= 3 + 4 + 6 + 7 + 9 + 10 + 17 = 56

Jadi diperoleh pohon merentang minimum dengan T = { A, B, C, D, E, F, G, H } dengan bobot 56, dan dari 8 titik serta 14 sisi, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 8.

Catatan: Karena dalam mencari pohon merentang minimum langkahnya adalah dilakukan pengurutan terlebih dahulu pada setiap bobot dari terbesar hingga

terkecil, dan dilanjutkan dengan menghapus bobot yang telah diurutkan tadi hingga terbentuk pohon merentang minimum. Jadi banyak kemungkinan pohon merentang minimum yang dapat terbentuk hanya satu.

3.4.2 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Banyak Sisi = 2(*p* - 1) dan Terdapat Sisi yang Memiliki Bobot Sama

Untuk menentukan pohon merentang minimum dengan menggunakan Algoritma Sollin dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Mula-mula kita buat dulu tabel yang berisikan sisi-sisi yang terurut dari besar ke kecil.

Tabel 3.6 Urutan Sisi dari Bobot Terbesar Hingga Terkecil Graf G dengan Banyak Sisi = 2(*p* - 1) dan terdapat Sisi yang Memiliki Bobot Sama

Bobot	Sisi
10	(A, D)
10	(A, F)
10	(B, D)
7	(B, H)
7	(C, E)
6	(A, C)
6	(A, G)
6	(B, E)
4	(A, E)

4	(B, C)
4	(B, G)
3	(A, B)
3	(A, H)
3	(C, D)

Kemudian kita lakukan penghapusan sisi dimulai dari yang memiliki bobot terbesar dan tidak membuat graf menjadi tidak terhubung.

Langkah 1 : Dihapus sisi yang berbobot paling maksimum, yaitu sisi AD karena berbobot 10.

Langkah 2 : Sisi AF tidak dapat dihapus karena akan menyebabkan graf menjadi tidak terhubung.

Langkah 3 : Hapus sisi berbobot maksimum berikutnya. Misalkan sisi yang dihapus adalah sisi BD dengan bobot 10.

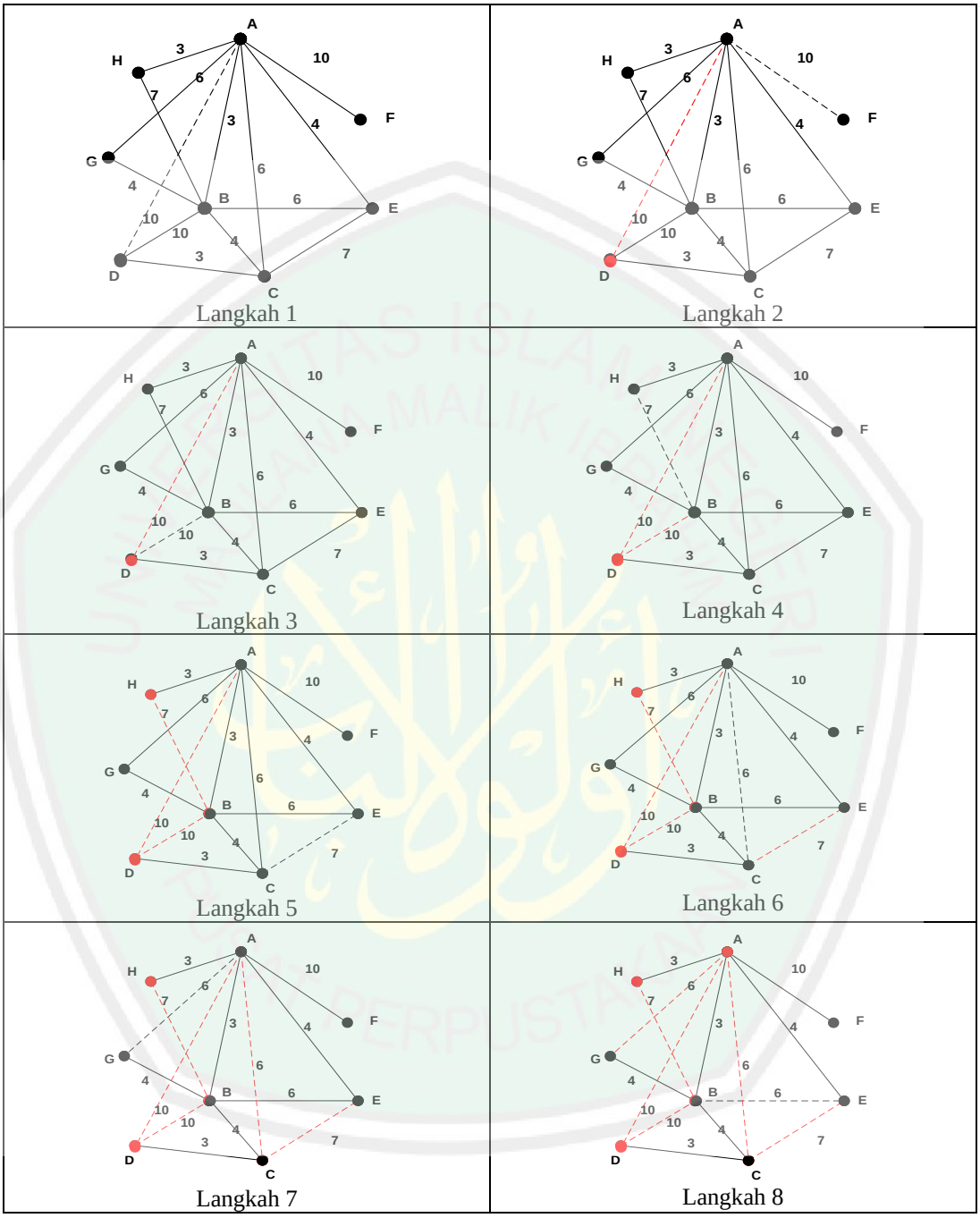
Langkah 4 : Masih sama dengan langkah sebelumnya. Misalkan sisi yang dihapus adalah BH dengan bobot 7.

Langkah 5 : CE adalah sisi yang dihapus selanjutnya karena mempunyai bobot maksimum 7.

Langkah 6 : Selanjutnya sisi yang dihapus adalah AC dengan bobot 6.

Langkah 7 : Hapus sisi berbobot maksimum berikutnya. Misalkan sisi yang dihapus adalah sisi AG dengan bobot 6.

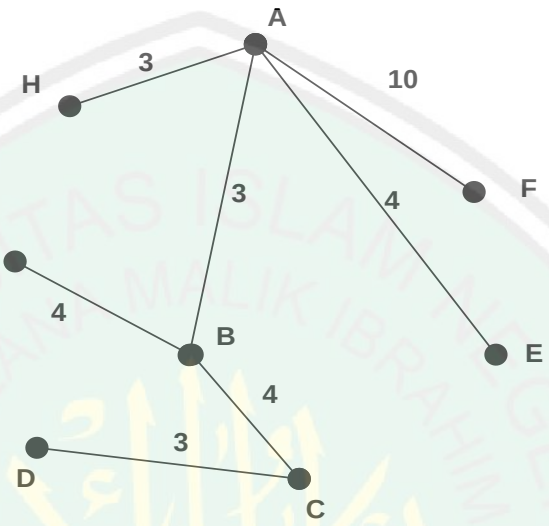
Langkah 8 : Masih sama dengan langkah sebelumnya. Misalkan sisi yang dihapus adalah BE dengan bobot 6.



Gambar 3.43 Langkah-Langkah Algoritma Sollin pada Graf G dengan Banyak Sisi = 2(p - 1) dan terdapat Sisi yang Memiliki Bobot Sama

Karena sudah tidak terdapat lagi sisi yang dapat dihapus dan itu artinya semua titik sudah terhubung dan sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh

dari perhitungan menggunakan Algoritma Sollin dapat dilihat pada gambar di bawah ini :



Gambar 3.44 Pohon Merentang Minimum dengan Algoritma Sollin pada Graf G dengan Banyak Sisi =2(p - 1) dan terdapat Sisi yang Memiliki Bobot Sama

Dari perhitungan Algoritma Sollin di atas diperoleh pohon merentang minimum dengan jumlah bobot sebagai berikut :

W(C, D) + W(A, H) + W(A, B) + W(B, G) + W(B, C) + W(A, E) + W(A, F)

= 3 + 3 + 3 + 4 + 4 + 4 + 10 = 31

Jadi diperoleh pohon merentang minimum dengan T = {A, B, C, D, E, F, G, H} dengan bobot 31, dan dari 8 titik serta 14 sisi dan terdapat sisi yang memiliki bobot sama, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 8.

Catatan: Karena dalam mencari pohon merentang minimum langkahnya adalah dilakukan pengurutan terlebih dahulu pada setiap bobot dari terbesar hingga terkecil, dan dilanjutkan dengan menghapus bobot yang telah diurutkan tadi

hingga terbentuk pohon merentang minimum. Jadi banyak kemungkinan pohon merentang minimum yang dapat terbentuk hanya satu.

3.4.3 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Banyak Sisi < 2(*p* - 1)

Untuk menentukan pohon merentang minimum dengan menggunakan Algoritma Sollin dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Mula-mula kita buat dulu tabel yang berisikan sisi-sisi yang terurut dari besar ke kecil.

Tabel 3.7 Urutan Sisi dari Bobot Terbesar Hingga Terkecil Graf G dengan Banyak Sisi < 2(*p* - 1)

Bobot	Sisi
27	(C, E)
26	(A, G)
25	(A, H)
22	(A, E)
17	(A, F)
13	(A, C)
10	(B, D)
9	(A, B)
6	(B, E)
4	(B, G)

3	(C, D)

Kemudian kita lakukan penghapusan sisi dimulai dari yang memiliki bobot terbesar dan tidak membuat graf menjadi tidak terhubung.

Langkah 1 : Dihapus sisi yang berbobot paling maksimum, yaitu sisi CE karena berbobot 27.

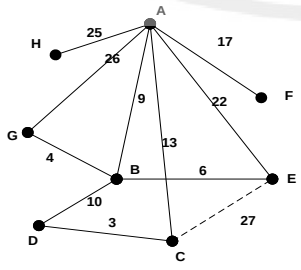
Langkah 2 : Hapus sisi berbobot maksimum berikutnya. Misalkan sisi yang dihapus adalah sisi AG dengan bobot 26.

Langkah 3 : AH adalah sisi maksimum selanjutnya dengan bobot 25, tetapi tidak bisa dihapus karena akan menyebabkan graf manjadi tidak terhubung

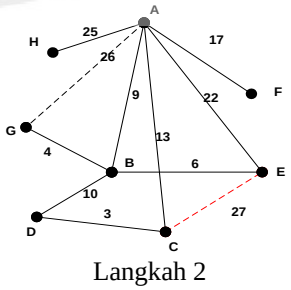
Langkah 4 : Selanjutnya sisi yang dihapus adalah AE dengan bobot 22.

Langkah 5 : AF adalah sisi maksimum selanjutnya dengan bobot 17, tetapi tidak bisa dihapus karena akan menyebabkan graf manjadi tidak terhubung.

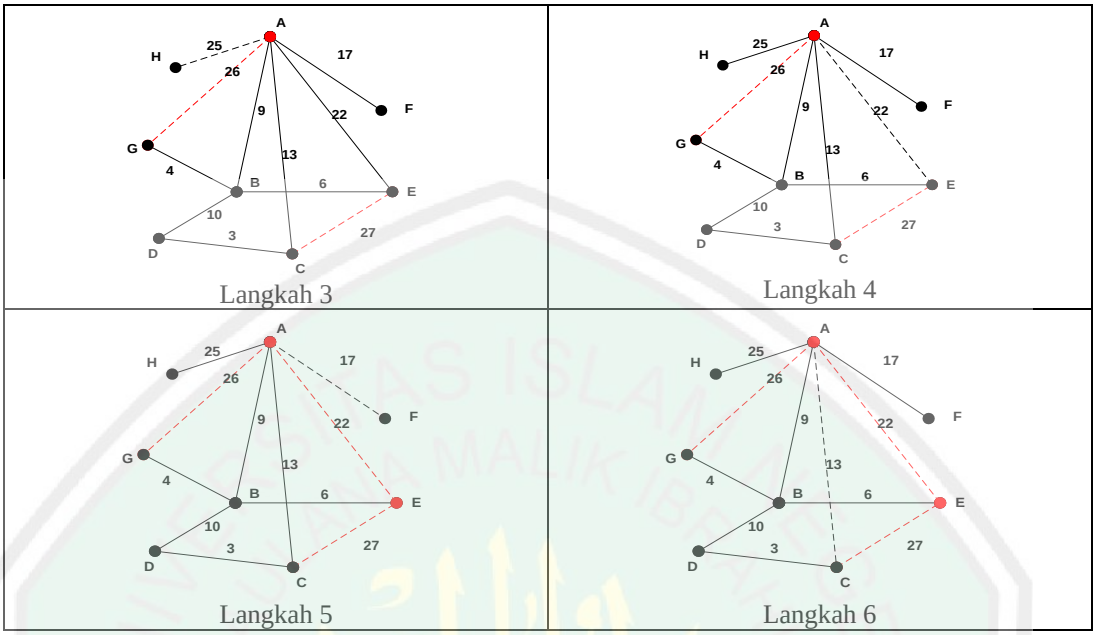
Langkah 6 : Hapus sisi berbobot maksimum berikutnya. Misalkan sisi yang dihapus adalah sisi AC dengan bobot 13.



Langkah 1

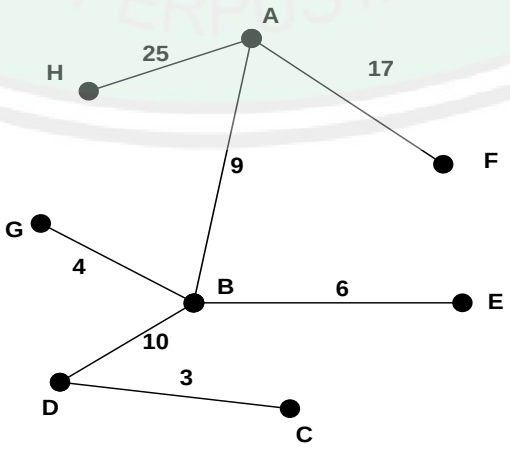


Langkah 2



Gambar 3.45 Langkah-Langkah Algoritma Sollin pada Graf G dengan Banyak Sisi < 2(p - 1)

Karena sudah tidak terdapat lagi sisi yang dapat dihapus dan itu artinya semua titik sudah terhubung dan sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Sollin dapat dilihat pada gambar di bawah ini :



Gambar 3.46 Pohon Merentang Minimum dengan Algoritma Sollin pada Graf G dengan Banyak Sisi < 2(p - 1)

Dari perhitungan Algoritma Sollin di atas diperoleh pohon merentang minimum dengan jumlah bobot sebagai berikut :

W(C, D) + W(B, G) + W(B, E) + W(A, B) + W(B, D) + W(A, F) + W (A, H)

= 3 + 4 + 6 + 7 + 9 + 10 + 17 + 25 = 74

Jadi diperoleh pohon merentang minimum dengan T = { A, B, C, D, E, F, G, H } dengan bobot 74, dan dari 8 titik serta 11 sisi, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 6.

Catatan: Karena dalam mencari pohon merentang minimum langkahnya adalah dilakukan pengurutan terlebih dahulu pada setiap bobot dari terbesar hingga terkecil, dan dilanjutkan dengan menghapus bobot yang telah diurutkan tadi hingga terbentuk pohon merentang minimum. Jadi banyak kemungkinan pohon merentang minimum yang dapat terbentuk hanya satu.

3.4.4 Penghitungan Pohon Merentang Minimum pada Graf *G* dengan Sisi yang Memiliki Bobot Sama

Untuk menentukan pohon merentang minimum dengan menggunakan Algoritma Sollin dapat dilakukan dengan mengikuti prosedur dalam langkah-langkah di bawah ini :

Mula-mula kita buat dulu tabel yang berisikan sisi-sisi yang terurut dari besar ke kecil.

Tabel 3.8 Urutan Sisi dari Bobot Terbesar Hingga Terkecil Graf G dengan Banyak Sisi > 2(p - 1)

Bobot	**Sisi**
33	(E, H)
29	(A, D)
28	(F, G)
27	(C, E)
26	(A, G)
25	(A, H)
22	(A, E)
21	(C, F)
17	(A, F)
15	(B, F)
14	(B, C)
13	(A, C)
12	(D, E)
10	(B, D)
9	(A, B)
8	(E, F)
7	(B, H)
6	(B, E)
4	(B, G)
3	(C, D)

Kemudian kita lakukan penghapusan sisi dimulai dari yang memiliki bobot terbesar dan tidak membuat graf menjadi tidak terhubung.

Langkah 1 : Dihapus sisi yang berbobot paling maksimum, yaitu sisi EH karena berbobot 33.

Langkah 2 : Hapus sisi berbobot maksimum berikutnya. Misalkan sisi yang dihapus adalah sisi AD dengan bobot 29.

Langkah 3 : Masih sama dengan langkah sebelumnya. Misalkan sisi yang dihapus adalah FG dengan bobot 28.

Langkah 4 : CE adalah sisi yang dihapus selanjutnya karena mempunyai bobot maksimum 27.

Langkah 5 : Selanjutnya sisi yang dihapus adalah AG dengan bobot 26.

Langkah 6 : AH adalah sisi maksimum selanjutnya dengan bobot 25.

Langkah 7 : Sisi yang dihapus selanjutnya adalah AE dengan bobot 22.

Langkah 8 : Hapus sisi berbobot maksimum berikutnya. Misalkan sisi yang dihapus adalah sisi CF dengan bobot 21.

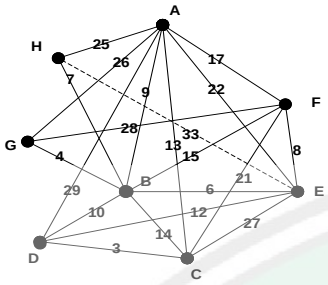
Langkah 9 : AF adalah sisi maksimum selanjutnya dengan bobot 17.

Langkah 10 : Masih sama dengan langkah sebelumnya. Misalkan sisi yang dihapus adalah BF dengan bobot 15.

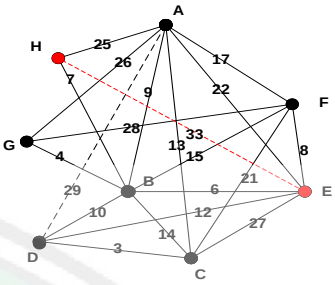
Langkah 11 : Sisi yang dihapus selanjutnya adalah BC dengan bobot 14.

Langkah 12 : Hapus sisi berbobot maksimum berikutnya. Misalkan sisi yang dihapus adalah sisi AC dengan bobot 13

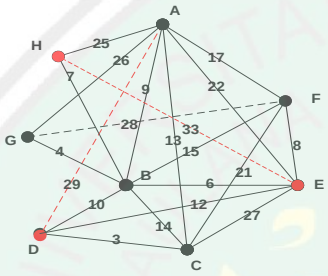
Langkah 13 : DE adalah sisi yang dihapus selanjutnya karena mempunyai bobot maksimum 12.



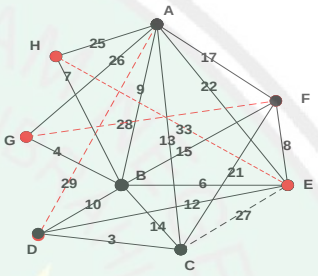
Langkah 1



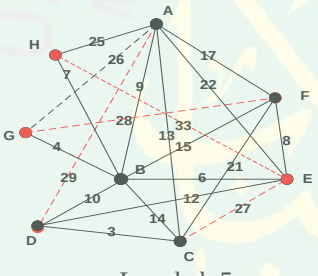
Langkah 2



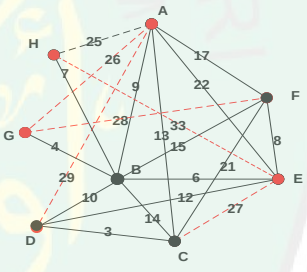
Langkah 3



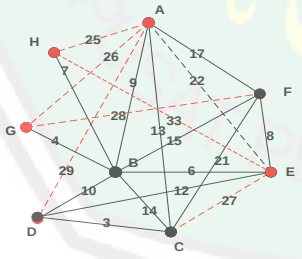
Langkah 4



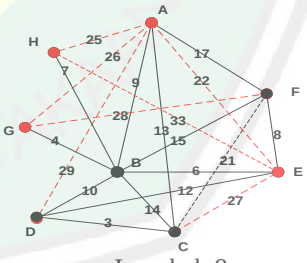
Langkah 5



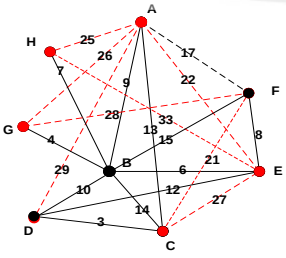
Langkah 6



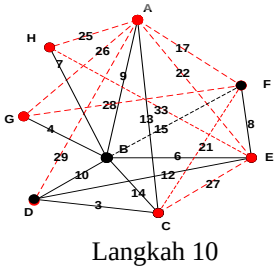
Langkah 7



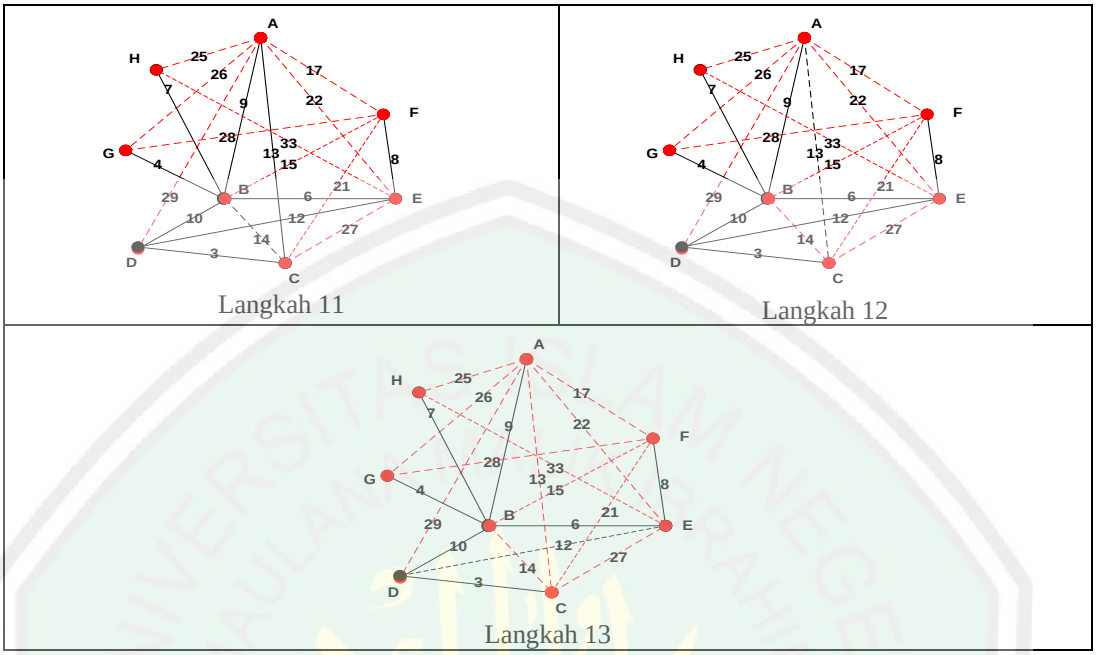
Langkah 8



Langkah 9

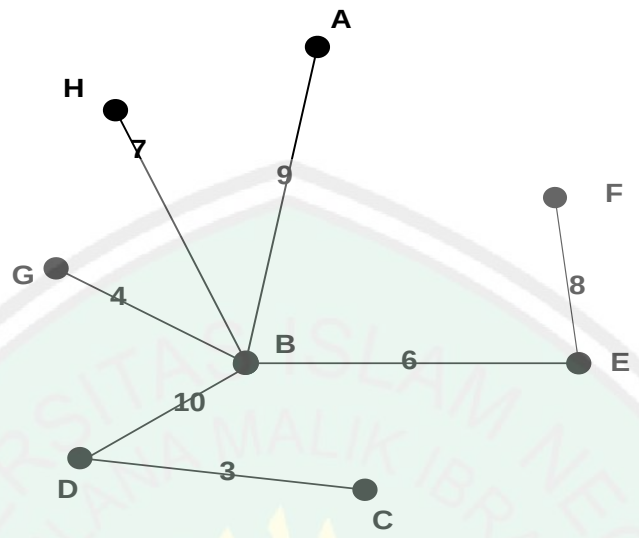


Langkah 10



Gambar 3.47 Langkah-Langkah Algoritma Sollin pada Graf G dengan Banyak Sisi > 2(p - 1)

Karena sudah tidak terdapat lagi sisi yang dapat dihapus dan itu artinya semua titik sudah terhubung dan sudah diperoleh pohon merentang minimumnya, maka langkah dapat dihentikan. Hasil pohon merentang minimum yang diperoleh dari perhitungan menggunakan Algoritma Sollin dapat dilihat pada gambar di bawah ini :



Gambar 3.48 Pohon Merentang Minimum dengan Algoritma Sollin pada Graf G dengan Banyak Sisi > 2(p - 1)

Dari perhitungan Algoritma Sollin di atas diperoleh pohon merentang minimum dengan jumlah bobot sebagai berikut :

W(C, D) + W(B, G) + W(B, E) + W(B, H) + W(E, F) + W(A, B) + W(B, D)

= 3 + 4 + 6 + 7 + 8 + 9 + 10 = 47

Jadi diperoleh pohon merentang minimum dengan T = { A, B, C, D, E, F, G, H } dengan bobot 47, dan dari 8 titik serta 20 sisi, setelah diperoleh pohon merentang minimumnya diperoleh adalah 8 titik dan 7 sisi, dan banyak langkah yang ditempuh adalah 12.

Catatan: Karena dalam mencari pohon merentang minimum langkahnya adalah dilakukan pengurutan terlebih dahulu pada setiap bobot dari terbesar hingga terkecil, dan dilanjutkan dengan menghapus bobot yang telah diurutkan tadi hingga terbentuk pohon merentang minimum. Jadi banyak kemungkinan pohon merentang minimum yang dapat terbentuk hanya satu.

3.5 Perbandingan

Dari perhitungan di atas dapat dikatakan bahwasannya penggunaan Algoritma Boruvka, Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin mempunyai perbedaan yang mendasar, yaitu :

1. Pada Algoritma Boruvka pencarian pohon merentang minimum dapat dilakukan dengan cara menentukan hutan pada graf baru yang kosong dari graf *G*. Dalam penetapan hutan tidak memperhatikan bobot dari sisi yang dipilih. Kemudian menambahkan sisi yang berbobot minimum dari setiap pohon hingga terbentuk pohon merentang.
2. Pada Algoritma Prim langkah pertama untuk menentukan titik yang dipilih adalah bebas, kemudian dari titik yang dipilih tersebut dapat mudah menentukan sisi berbobot minimum selanjutnya. 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.
3. Pada Algoritma Kruskal pencarian pohon merentang minimum didasarkan pada sisi yang mempunyai bobot minimum dan sisi tersebut tidak membentuk sirkuit dengan sisi-sisi yang telah terpilih sebagai pohon.
4. Pada Algoritma Sollin pencarian pohon merentang minimum didasarkan pada sisi yang mempunyai bobot maksimum, kemudian dari sisi-sisi yang berbobot maksimum tersebut dilakukan penghapusan sisi-sisi yang tidak menyebabkan graf menjadi tidak terhubung atau membentuk sirkuit.

Penghapusan dapat dihentikan saat tidak ada lagi sisi yang dapat dihapus, karena jika sisi tersebut dihapus akan menyebabkan graf menjadi tidak terhubung.

Berdasarkan perhitungan dari keempat algoritma di atas, maka dapat ditampilkan sebuah tabel perbandingan dari Algoritma Boruvka, Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin sebagai berikut:

Tabel 3.9 Perbandingan Hasil Algoritma Boruvka, Algoritma Prim, Algoritma Kruskal, Algoritma Sollin

	8;14				8;14 (sisi kembar)				8;11				8;20			
	AB	AP	AK	AS	AB	AP	AK	AS	AB	AP	AK	AS	AB	AP	AK	AS
Jumlah	56	56	56	56	31	31	31	31	74	74	74	74	47	47	47	47
Sisi	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
Langkah	3	9	9	8	3	7	12	7	3	8	9	6	3	9	7	12
kemungkinan	n	n	n	n	1	1	1	1	1	1	1	1	1	1	1	1
Gambar Graf																

Keterangan :

AB : Algoritma Boruvka

AP : Algoritma Prim

AK : Algoritma Kruskal

AS : Algoritma Sollin

Jumlah : Jumlah bobot setelah terbentuk pohon merentang minimum

Sisi : Banyaknya sisi setelah terbentuk pohon merentang minimum

Langkah : Banyaknya langkah setelah terbentuk pohon merentang minimum

Gambar Graf : Gambar pohon merentang minimum yang diperoleh

Kemuungkinan: Banyak kemungkinan membentuk pohon merentang minimum

n : Anggota himpunann bilangan bulat positif ($n \in z^{+}$)

Dari tabel di atas dapat dilihat perbandingan antara Algoritma Boruvka, Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin, yaitu:

1. Jumlah bobot minimum yang dihasilkan untuk setiap graf pada Algoritma Boruvka, Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin adalah sama.
    a. Dari hasil perhitungan keempat algoritma untuk graf G dengan banyak sisi = $2(p - 1)$ diperoleh jumlah bobot pohon merentang minimum adalah 56.
    b. Dari hasil perhitungan keempat algoritma untuk graf G dengan banyak sisi = $2(p - 1)$ dan terdapat sisi yang memiliki bobot sama pada beberapa sisi diperoleh jumlah bobot pohon merentang minimum adalah 31.
    c. Dari hasil perhitungan keempat algoritma untuk graf G dengan banyak sisi < $2(p - 1)$ diperoleh jumlah bobot pohon merentang minimum adalah 74.
    d. Dari hasil perhitungan keempat algoritma untuk graf G dengan banyak sisi > $2(p - 1)$ diperoleh jumlah bobot pohon merentang minimum adalah 47.
2. Banyaknya sisi yang terbentuk setelah diperoleh pohon merentang untuk setiap graf pada algoritma Boruvka, algoritma Prim, algoritma Kruskal, dan algoritma Sollin adalah sama yakni 7.

3. Banyaknya langkah yang ditempuh oleh setiap graf pada algoritma Boruvka, algoritma Prim, algoritma Kruskal, dan algoritma Sollin hingga terbentuk pohon merentang minimum adalah berbeda-beda..
4. Banyaknya kemungkinan membentuk pohon merentang minimum dari keempat algoritma adalah berbeda. Pada algoritma Boruvka dapat diperoleh banyak kemungkinan pohon merentang minimum karena sebelum diperoleh pohon merentang minimum, langkah awal algoritma Boruvka adalah menetapkan hutan dengan syarat tidak memperhatikan bobot dari sisi yang dipilih. Sedangkan banyak kemungkinan pohon merentang minimum yang terbentuk dengan menggunakan algoritma Prim, algoritma Kruskal, dan algoritma Sollin adalah 1.
5. Pohon merentang minimum yang dihasilkan untuk setiap graf memiliki bentuk yang sama.
    a. Gambar graf pohon merentang minimum yang diperoleh pada keempat algoritma untuk graf G dengan banyak sisi = $2(p - 1)$ adalah 
    b. Gambar graf pohon merentang minimum yang diperoleh pada keempat algoritma untuk graf G dengan banyak sisi = $2(p - 1)$ dan terdapat sisi yang memiliki bobot sama adalah 
    c. Gambar graf pohon merentang minimum yang diperoleh pada keempat algoritma untuk graf G dengan banyak sisi < $2(p - 1)$ adalah 

d. Gambar graf pohon merentang minimum yang diperoleh pada keempat algoritma untuk graf *G* dengan banyak sisi > 2(*p* - 1) adalah 

3.6 Kajian Agama Berdasarkan Hasil Pembahasan

Berdasarkan hasil pembahasan, maka dapat diketahui bahwa dari masing-masing algoritma yang digunakan untuk menentukan pohon merentang minimum pada empat kasus graf adalah berbeda-beda. Tapi dari adanya perbedaan itu dapat dikatakan bahwa masing-masing algoritma memiliki cara yang berbeda-beda dalam membentuk pohon merentang minimum. Ada algoritma yang sangat efektif digunakan pada suatu graf tertentu, tapi jika algoritma itu digunakan pada graf lain bisa jadi algoritma itu sangat tidak efektif digunakan.

Dari penjelasan di atas, jika direlevansikan dengan kajian agama adalah pada suatu ayat yang menyebutkan bahwa Allah SWT menghendaki kemudahan bagi semua hambaNya dalam menempuh segala hal yang baik. Sebagaimana yang tertera pada surat Al-Baqarah ayat 185:

يُرِيدُ ٱللَّهُ بِكُمُ ٱلْيُسْرَ وَلَا يُرِيدُ بِكُمُ ٱلْعُسْرَ ﴿١٨٥﴾

Artinya : Allah menghendaki kemudahan bagimu, dan tidak menghendaki kesukaran bagimu(Q.S. Al-Baqarah:185).

Ada pepatah arab menyebutkan

لاَ تَحْتَقِرْ مَنْ دُوْنَكَ فَلِكُلِّ شَيْءٍ مَزِيَّةٌ

Artinya : Janganlah kamu menghina orang lain, karena segala sesuatu memiliki kelebihan.

Maksud dari pepatah itu adalah segala sesuatu di dunia ini pasti memiliki kelebihan dan kekurangannya masing-masing. Sama halnya dengan algoritma yang digunakan untuk menentukan pohon merentang minimum. Tiap algoritma tersebut memiliki cara yang berbeda-beda dalam mencari pohon merentang minimum. Ada algoritma yang lebih efektif digunakan untuk kasus graf tertentu dan bila algoritma ini digunakan untuk kasus graf lain menjadi tidak efektif jadi digunakan algoritma lain yang lebih efektif, itu semua tergantung dari bentuk grafnya.

Aplikasi dalam kehidupan sehari-hari adalah misalkan pada pemasangan kabel listrik pada 4 lokasi dan setiap lokasi tersebut terdapat 4 rumah yang akan dipasang listrik. Ada hal-hal yang harus diperhatikan, yaitu pihak PLN harus meminimalkan biaya, waktu dan tenaga. Maka dalam memecahkan persoalan ini, ilmu graf dapat diterapkan yaitu mengenai masalah mencari pohon merentang minimum dengan menggunakan algoritma yang paing efektif. Pada lokasi pertama dengan 4 rumah dan jalur kabel yang bisa dilewati ada 7 jalur dengan jarak masing-masing jalur adalah berbeda, pada lokasi kedua dengan 4 rumah dan jalur kabel yang bisa dilewati ada 7 namun ada yang beberapa jalur yang berjarak sama, lokasi ketiga dengan 4 rumah dan jalur kabel yang bisa dilewati ada 3 jalur dengan jarak masing-masing jalur adalah berbeda, dan pada lokasi keempat dengan 4 rumah dan jalur kabel yang bisa dilewati ada 10 jalur dengan jarak

masing-masing jalur adalah berbeda. Maka dari permasalahan tersebut di atas, algoritma yang paling efektif dari masing-masing kasus bisa dijalankan.

Sebagai akhir dari analisis tentang relevansi antara konsep salah satu cabang matematika yaitu teori graf khususnya, maka masalah pencarian algoritma yang paling efektif digunakan dalam menentukan pohon merentang minimum pada suatu graf dengan kajian Islam yang sekaligus merupakan hal yang utama yang dapat dijadikan sebagi refleksi dari semuanya dapat disimpulkan. Ternyata setelah banyak mempelajari matematika yang merupakan ilmu hitung-menghitung serta banyak mengetahui mengenai masalah yang terdapat dalam matematika yang dapat direlevansikan dalam agama Islam sesuai dengan konsep-konsep yang ada dalam Al-Qur'an, maka akan dapat menambah keyakinan diri akan kebesaran Allah SWT selaku sang pencipta yang serba Maha, salah satunya adalah Maha Matematika. Karena Dialah sang raja yang sangat cepat dan teliti dalam semua masalah perhitungan (Abdusysyakir, 2007: 83).

BAB IV

PENUTUP

4.1 Kesimpulan

Berdasarkan hasil pembahasan Bab III, maka dapat diambil kesimpulan, bahwa keempat algoritma yang digunakan memiliki cara yang berbeda-beda dalam mencari pohon merentang minimum, dan dari beragam cara yang berbeda-beda itu diperoleh beberapa kemungkinan pohon merentang minimum yang dapat terbentuk. Pada algoritma Boruvka dapat diperoleh banyak kemungkinan pohon merentang minimum, karena sebelum diperoleh pohon merentang minimum langkah awal algoritma Boruvka adalah menetapkan hutan dengan syarat tidak memperhatikan bobot dari sisi yang dipilih. Sedangkan kemungkinan pohon merentang minimum yang terbentuk dengan menggunakan algoritma Prim, algoritma Kruskal, dan algoritma Sollin adalah 1. Jadi jika dilihat dari segi berapa banyak kemungkinan pohon merentang minimum yang dapat terbentuk, algoritma Prim, algoritma Kruskal, dan algoritma Sollin adalah yang paling tepat digunakan untuk mencari pohon merentang minimum.

Dari penjelasan di atas, jika dilihat dari segi kecepatan langkah hingga terbentuk pohon merentang minimum dengan mengabaikan algoritma Boruvka, maka diperoleh:

a. Untuk graf dengan banyak sisi = $2(p - 1)$ algoritma Sollin paling efektif dibandingkan algoritma Prim, dan algoritma Kruskal.

b. Untuk graf dengan banyak sisi = 2(p - 1) namun terdapat sisi yang memiliki bobot yang sama algoritma Prim dan algoritma Sollin lebih efektif dibandingkan dan algoritma Kruskal.

c. Untuk graf dengan banyak sisi < 2(p - 1) algoritma Sollin paling efektif dibandingkan algoritma Prim, dan algoritma Kruskal.

d. Untuk graf dengan banyak sis$i$ > 2(p - 1) Algoritma Kruskal paling efektif dibandingkan algoritma Prim, dan algoritma Sollin.

4.2 Saran

Pada skripsi ini masih banyak jenis graf lain yang dapat dicari pohon merentang minimumnya. Maka dari itu, untuk penelitian selanjutnya, penulis menyarankan kepada pembaca untuk melanjutkan p enelitian menggunakan jenis-jenis graf lainnya.

DAFTAR PUSTAKA

Abdusysyakir. 2007. *Ketika Kyai Mengajar Matematika*. Malang: UIN Malang Press.

Bondy, J.A, and Murty, U.S.R. 1976. *Graph Theory With Applications*. London: MacMillan Press.

Chartrand, G. dan Lesniak, L. 1986. *Graph and Digraph 2 nd Edition.* California: Wadsworth. Inc.

Chartrand, G. dan Ortrud, R. O. 1993. *Applied and Algorithmic Graph Theory*. New York: McGraw-Hill, Inc.

Hsu, Lih-Hsing and Cheng Kuan-Lin. 2008. *Graph Theory and Interconnection Networks*. New York: CRC Press.

Http://www.informatika.org/rinaldi/Matdis/Makalah2008/Makalah0809-061.pdf. *Menentukan Pohon Merentang Minimum dengan Algoritma Sollin*. Tanggal akses: 19 September 2009.

Munir, Rinaldi. 2005. *Matematika Diskrit*. Bandung: Informatika.

Nawawi, Imam. 2005. *Sharah dan Terjemah Shalihin*. Jakarta: Al-I'tishom.

Nawawi, Imam. 2006. *Shahih Riyadhush-Shalihin 1*. Jakarta: Pustaka Azzam.

Purwanto. 1998. *Matematika Diskrit*. Malang: Institut Keguruan dan Ilmu Pendidikan Malang.

Roman, S. 1989. *An Introduction To Discrete Mathematics*. Second Edition. New York: McGraw-Hill, Inc.

Wilson, R. J dan Watkins, J. J. 1990. *Graph An Introductory Approach a First Course In Discrete Mathematics*. Canada: John Whiley and Sons, Inc.

KEMENTRIAN AGAMA RI
UNIVERSITAS ISLAM NEGERI (UIN) MAULANA MALIK IBRAHIM MALANG
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Gajayana No. 50 Dinoyo Malang (0341)551345 Fax. (0341)572533

BUKTI KONSULTASI SKRIPSI

Nama : Mufidatul Khoiroh
NIM : 06510037
Fakultas/ jurusan : Sains Dan Teknologi/ Matematika
Judul skripsi : Keefektifan Penggunaan Algoritma Boruvka, Algoritma Prim, Algoritma Kruskal, dan Algoritma Sollin dalam Menentukan Pohon Merentang Minimum
Pembimbing I : Abdussakir, M.Pd
Pembimbing II : Dr. Munirul Abidin, M.Ag

No	Tanggal	HAL	Tanda Tangan	
1	09 Oktober 2009	Konsultasi Masalah	1.	
2	15 Maret 2010	Konsultasi BAB II, III		2.
3	29 Maret 2010	Revisi BAB II dan III	3.	
4	19 April 2010	Konsultasi Keagamaan		4.
5	21 April 2010	Konsultasi BAB I, II, III	5.	
6	24 April 2010	Revisi BAB I, II, III		6.
7	26 April 2010	Konsultasi BAB I, II, III	7.	
8	26 April 2010	Revisi Keagamaan		8.
9	05 Mei 2010	Revisi BAB I, II, III	9.	

10	05 Mei 2010	Konsultasi keagamaan		10.
11	12 Mei 2010	Konsultasi keseluruhan	11.	
12	24 Mei 2010	Revisi keseluruhan		12.
13	25 Mei 2010	Revisi keagamaan	13.	
14	01 Juni 2010	Konsultasi Keseluruhan		14.
15	23 Juni 2010	ACC Keseluruhan	15.	

Malang, 23 Juni 2010
Mengetahui,
Ketua Jurusan Matematika

Abdussakir, M.Pd
NIP.19751006 200312 1 001