UNIVERSITAS INDONESIA

ALGORITMA PELABELAN TOTAL SIMPUL AJAIB PADA GRAF *FRIENDSHIP*, KIPAS, DAN JAHANGIR YANG DIPERUMUM

SKRIPSI

BUDI UTAMI
0606067326

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI SARJANA MATEMATIKA
DEPOK
JULI 2010

UNIVERSITAS INDONESIA

ALGORITMA PELABELAN TOTAL SIMPUL AJAIB PADA GRAF *FRIENDSHIP*, KIPAS, DAN JAHANGIR YANG DIPERUMUM

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains

BUDI UTAMI
0606067326

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI SARJANA MATEMATIKA
DEPOK
JULI 2010

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama	: Budi Utami
NPM	: 0606067326
Tanda Tangan	:
Tanggal	: 7 Juli 2010

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :
Nama : Budi Utami
NPM : 0606067326
Program Studi : Sarjana Matematika
Judul Skripsi : Algoritma Pelabelan Total Simpul Ajaib pada Graf *Friendship*, Kipas, dan Jahangir yang Diperumum

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia

DEWAN PENGUJI

Pembimbing	: Dra. Denny R. Silaban, M.Kom	( )
Pembimbing	: Dra. Siti Aminah, M.Kom	( )
Penguji	: Prof. Dr. Belawati H Widjaja	( )
Penguji	: Mila Novita, M.Si	( )

Ditetapkan di : Depok
Tanggal : 7 Juli 2010

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Sains Jurusan Matematika pada Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia. Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

(1) Orang tua tercinta yang tak pernah lelah memberikan dukungan, semangat, dan terutama doa bagi penulis.

(2) Dra. Denny Riama Silaban, M.Kom dan Dra. Siti Aminah, M.Kom selaku pembimbing skripsi yang telah banyak meluangkan waktu, tenaga, dan pikiran serta memberikan saran dan motivasi pada penulis selama penyusunan skripsi ini.

(3) Dra. Suarsih Utama selaku pembimbing akademik penulis selama menjalani masa studi yang telah memberikan banyak bantuan dan nasehat, dan terima kasih kepada seluruh dosen Departemen FMIPA UI yang tidak hanya memberikan pelajaran kuliah, tapi juga pelajaran hidup.

(4) Dr. Kiki A. Sugeng, Dhian Widya, S.Si, M.Kom, Prof. Dr. Djati Kerami, Bevina D. Handari, PhD., dan Dra. Nora Hariadi, M.Si atas saran-saran yang telah diberikan.

(5) Staf karyawan Departemen Matematika FMIPA UI, khususnya Ibu Santi.

(6) Budi dan Mas Ari yang telah membantu penulis dalam urusan-urusan yang berkaitan dengan komputer dan laptop.

(7) Alfa, Mella, dan Milla selaku teman seperjuangan dalam " kelompok algoritma pelabelan".

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

(8) Farah dan Yuko yang telah bersedia meminjamkan kosannya dan juga printernya.

(9) Dian, Oppie, dan Soni (anggota kelompok "abadi") yang telah banyak membantu penulis pada masa-masa awal kuliah.

(10) Widya, Indah, Lena, Nadya, Putri Helmet, Ita, Yunita, Rita, Lidya, Yuri, Annisa, Bekti, Indra, Latief, Sevenhot, Michael, Reza, dan Rifza yang telah berjuang bersama dalam penyusunan skripsi.

(11) Puspa, Rahmanita, Kiki, dan teman-teman angkatan 2006 atas dukungan dan semangatnya.

(12) Kakak angkatan 2004, 2005 (terutama Kak Desti dan Kak Yanu), teman-teman angkatan 2007 (terutama Arif yang telah membantu membuat transkrip akademik penulis) dan 2008.

(13) Teman-teman begadang penulis "para manusia malam" yang tetap membuat penulis terjaga ketika mengerjakan skripsi tengah malam.

"Ketika kuingin menyerah dan pasrah, ketika kuingin pergi melupakan semua masalah, kudapati diriku bertahan dan terus melangkah. Itu semua karena kalian kawan, yang telah mengajariku arti perjuangan tak kenal lelah. Terima kasih." Penulis juga mengucapkan terima kasih kepada semua pihak yang telah membantu penyusunan skripsi ini yang tidak bisa disebutkan satu-persatu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Penulis

2010

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Budi Utami
NPM	: 0606067326
Program Studi	: Sarjana Matematika
Departemen	: Matematika
Fakultas	: Matematika dan Ilmu Pengetahuan Alam
Jenis karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*) atas karya ilmiah saya yang berjudul :
Algoritma Pelabelan Total Simpul Ajaib pada Graf *Friendship*, Kipas, dan Jahangir yang Diperumum.

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 7 Juli 2010
Yang menyatakan

(Budi Utami)

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

ABSTRAK

Nama : Budi Utami
Program Studi : Matematika
Judul : Algoritma Pelabelan Total Simpul Ajaib pada Graf *Friendship*, Kipas, dan Jahangir yang Diperumum

Misalkan G adalah graf dengan himpunan simpul $V = V(G)$ dan himpunan busur $E = E(G)$, dimana $|E|$ dan $|V|$ menyatakan banyaknya busur dan simpul pada G. Suatu pemetaan bijektif λ dari $V \cup E$ ke himpunan $\{1,2, \dots, |E| + |V|\}$ disebut pelabelan total simpul ajaib (PTSA) jika terdapat konstanta k sedemikian sehingga untuk setiap $x \in V$ berlaku $w(x) = \lambda(x) + \sum_{y \in N(x)} \lambda(xy) = k$, dimana $N(x)$ adalah himpunan simpul yang bertetangga dengan x. Nilai $w(x)$ disebut bobot x. Algoritma pelabelan sembarang graf secara umum adalah bersifat *NP-complete*. Dalam skripsi ini diberikan algoritma-algoritma untuk menghasilkan semua PTSA yang tidak isomorfik pada graf *friendship*, kipas, dan jahangir yang diperumum. Algoritma-algoritma tersebut kemudian diimplementasikan dalam bentuk program. Diberikan juga simulasi banyak PTSA yang berbeda untuk setiap nilai *k* yang mungkin dari ketiga kelas graf tersebut untuk beberapa nilai *n* dan *m*. Untuk graf kipas dengan $3 \leq n \leq 7$ dan graf jahangir yang diperumum dengan $n = 2$ dan $2 \leq m \leq 5$, $n = 3,4$ dan $m = 2$, serta $n = 3$ dan $m = 3$.

Kata Kunci : pelabelan total simpul ajaib, graf *friendship*, graf kipas, graf jahangir yang diperumum.
xv+77 halaman ; 60 gambar; 5 tabel
Daftar Pustaka : 6 (2002-2010)

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

ABSTRACT

Name : Budi Utami
Program Study : Mathematics
Title : Algorithms of Vertex Magic Total Labeling on Friendship, Fan, and Generalized Jahangir Graphs

Let G be a graph with vertex set $V = V(G)$ and edge set $E = E(G)$, where $|E|$ and $|V|$ are the number of edges and vertices on G. A one-to-one map λ from $V \cup E$ to the integers $\{1,2,\dots,|E|+|V|\}$ is called vertex magic total labeling (VMTL) if there is a constant k so that for every $x \in V$, $w(x) = \lambda(x) + \sum_{y \in N(x)} \lambda(xy) = k$, where $N(x)$ is a vertex set which adjacent with x and $w(x)$ is the weight of x. In general, the labeling algorithm for any graph is NP-complete. This *skripsi* gives the VMTL algorithms on friendship, fan, and generalized jahangir graphs. The algorithms generate all non isomorphic VMTLs on the corresponding classes of graphs for any possible value of *k*. This *skripsi* also gives the implementation and simulation of the algorithms. From the simulation we get all non isomorphic VMTLs on friendship graph and all non isomorphic VMTLs on fan and generalized jahangir graphs for some *n*. For fan graph with $3 \le n \le 7$ and generalized jahangir graph with $n = 2$ and $2 \le m \le 5$, $n = 3, 4$ and $m = 2$, and also $n = 3$ and $m = 3$.

Key Words : vertex magic total labeling, friendship graph, fan graph, generalized Jahangir graph.
xv+77 pages ; 60 pictures; 5 tables
Bibliography : 6 (2002-2010)

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

DAFTAR ISI

HALAMAN PERNYATAAN ORISINALITAS iii
HALAMAN PENGESAHAN iv
KATA PENGANTAR v
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS vii
ABSTRAK viii
ABSTRACT ix
DAFTAR ISI x
DAFTAR TABEL xi
DAFTAR GAMBAR xii
DAFTAR LAMPIRAN xv
1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Permasalahan 3
1.3 Tujuan 3
1.4 Pembatasan Masalah 3
1.5 Sistematika Penulisan 3
2 LANDASAN TEORI 4
2.1 Teori Graf 4
2.2 Jenis-Jenis Graf 7
2.3 Pelabelan Graf 9
2.4 Hasil yang Telah Diketahui 12
3 ALGORITMA PTSA 14
3.1 Algoritma PTSA Graf Friendship 15
3.2 Algoritma PTSA Graf Kipas 32
3.3 Algoritma PTSA Graf Jahangir yang Diperumum 44
4 IMPLEMENTASI DAN SIMULASI 58
4.1 Implementasi dan Simulasi Algoritma PTSA Graf *Friendship* 58
4.2 Implementasi dan Simulasi Algoritma PTSA Graf Kipas 65
4.3 Implementasi dan Simulasi Algoritma PTSA Graf Jahangir yang Diperumum 69
5 KESIMPULAN 75
DAFTAR PUSTAKA 77

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

DAFTAR TABEL

Tabel 4. 1 Banyaknya PTSA tidak isomorfik untuk graf *friendship* F_n dengan n = 2, 3 61

Tabel 4. 2 Banyaknya PTSA tidak isomorfik untuk graf kipas f_n dengan $3 \leq n \leq 7$ 67

Tabel 4. 3 Hasil simulasi program PTSA untuk graf $J_{n,m}$ dengan $n = 2$ dan $2 \leq m \leq 5$ 72

Tabel 4. 4 Hasil simulasi program PTSA untuk graf $J_{n,m}$ dengan $n = 3, 4$ dan $m = 2$ serta $n = 3$ dan $m = 3$ 72

Tabel 5. 1 Hasil yang yang diketahui dan hasil yang diperoleh 75

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

DAFTAR GAMBAR

Gambar 2. 1 Graf G 5
Gambar 2. 2 Contoh graf isomorfik 6
Gambar 2. 3 Contoh graf berlabel yang tidak isomorfik 6
Gambar 2. 4 (a) graf lintasan P_3 (b) graf lingkaran C_3 (c) graf roda W_3 7
Gambar 2. 5 (a) graf kipas f_{10} (b) graf *friendship* F_3 (c) graf jahangir $J_{8,2}$ 8
Gambar 2. 6 (a) pelabelan simpul pada W_3 (b) pelabelan busur pada W_3 9
Gambar 2. 7 Contoh PTSA pada f_4 dengan k=22 10
Gambar 2. 8 Contoh kasus rotasi pada W_3 12
Gambar 2. 9 Contoh kasus refleksi pada W_3 12

Gambar 3. 1 Graf F_3 16
Gambar 3. 2 Proses pelabelan graf *friendship* F_n 18
Gambar 3. 3 Proses pelabelan pada graf *friendship* F_n 19
Gambar 3. 4 Proses pelabelan graf *friendship* F_n 20
Gambar 3. 5 Kasus rotasi pada PTSA F_3 22
Gambar 3. 6 Contoh kasus isomorfisma sebagian pada F_3 23
Gambar 3. 7 Contoh kasus refleksi jenis pertama pada PTSA F_3 24
Gambar 3. 8 Contoh kasus refleksi jenis kedua pada PTSA F_3 25
Gambar 3. 9 Kasus refleksi gabungan refleksi jenis pertama dan kedua 26
Gambar 3. 10 Contoh Penerapan Algoritma PTSA graf *friendship* F_2 dengan 28
Gambar 3. 11 Contoh Penerapan Algoritma PTSA graf *friendship* F_2 dengan 28
Gambar 3. 12 Contoh Penerapan Algoritma PTSA graf *friendship* F_2 dengan 29
Gambar 3. 13 Contoh Penerapan Algoritma PTSA graf *friendship* F_2 dengan.... 30
Gambar 3. 14 Contoh Penerapan Algoritma PTSA graf *friendship* F_2 dengan 31
Gambar 3. 15 Graf Kipas f_6 33
Gambar 3. 16 Kasus refleksi pada PTSA graf f_5 dan f_6 35
Gambar 3. 17 Kasus refleksi pada PTSA f_3 (a) dan (b) 36
Gambar 3. 18 Proses pelabelan graf kipas f_n 37
Gambar 3. 19 Proses pelabelan graf kipas f_n 38

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

Gambar 3. 21 Contoh penerapan algoritma PTSA graf kipas f_4 dengan..... 40
Gambar 3. 22 Contoh penerapan algoritma PTSA graf kipas f_4 dengan..... 41
Gambar 3. 23 Contoh penerapan algoritma PTSA graf kipas f_4 dengan..... 41
Gambar 3. 24 Contoh penerapan algoritma PTSA graf kipas f_4 dengan..... 42
Gambar 3. 25 Contoh penerapan algoritma PTSA graf kipas f_4 dengan..... 43
Gambar 3. 26 PTSA pada graf f_4 dengan k = 27 43
Gambar 3. 27 Graf $J_{3,3}$ 44
Gambar 3. 28 Kasus rotasi pada PTSA graf $J_{3,3}$ 46
Gambar 3. 29 Kasus rotasi pada PTSA graf $J_{3,3}$ 47
Gambar 3. 30 Kasus refleksi sebagian pada PTSA graf $J_{2,2}$ 48
Gambar 3. 31 Proses pelabelan graf jahangir yang diperumum $J_{n,m}$..... 48
Gambar 3. 32 Proses pelabelan graf jahangir yang diperumum $J_{n,m}$..... 51
Gambar 3. 33 Proses pelabelan graf jahangir yang diperumum $J_{n,m}$..... 52
Gambar 3. 34 Contoh penerapan algoritma PTSA graf $J_{n,m}$ dengan n = 2,..... 54
Gambar 3. 35 Contoh penerapan algoritma PTSA graf $J_{n,m}$ dengan n = 2,..... 55
Gambar 3. 36 Contoh penerapan algoritma PTSA graf $J_{n,m}$ dengan n = 2,..... 56
Gambar 3. 37 Contoh penerapan algoritma PTSA graf $J_{n,m}$ dengan n = 2,..... 56
Gambar 3. 38 PTSA tidak isomorfik untuk graf $J_{2,3}$ dengan k = 29 57

Gambar 4. 1 (a) Tampilan awal pada *Command Window* (b) Tampilan keluaran pada *Command Window* (c) Tampilan keluaran pada berkas..... 59
Gambar 4. 2 PTSA graf F_2 dengan k=18 60
Gambar 4. 3 Seluruh PTSA tidak isomorfik pada graf F_2 61
Gambar 4. 4 Seluruh PTSA tidak isomorfik untuk graf F_3 dengan $k = 28$ 62
Gambar 4. 5 Seluruh PTSA tidak isomorfik untuk graf F_3 dengan $k = 29$ 64
Gambar 4. 6 Seluruh PTSA tidak isomorfik untuk graf F_3 dengan $k = 30$ 64
Gambar 4. 7 (a) Tampilan awal pada *Command Window* (b) Tampilan keluaran pada *Command Window* (c) Tampilan keluaran pada berkas..... 65
Gambar 4. 8 PTSA tidak isomorfik untuk graf kipas f_4 66
Gambar 4. 9 Grafik hasil simulasi untuk graf kipas f_n dengan $3 \leq n \leq 7$ 68
Gambar 4. 10 (a) Tampilan awal pada *Command Window* (b) Tampilan keluaran pada *Command Window* (c) Tampilan keluaran pada berkas 70

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

Gambar 4. 11 PTSA tidak isomorfik untuk graf $J_{2,3}$ dengan k = 29..... 71
Gambar 4. 12 Grafik hasil simulasi untuk graf $J_{2,m}$ dengan $2 \leq m \leq 5$ 73
Gambar 4. 13 Grafik hasil simulasi untuk graf $J_{n,2}$ dengan $2 \leq n \leq 4$ 74

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

DAFTAR LAMPIRAN

Lampiran 1 *Listing* program PTSA graf *friendship* ..... *CD*
Lampiran 2 *Listing* program PTSA graf kipas ..... *CD*
Lampiran 3 *Listing* program PTSA graf Jahangir yang diperumum..... *CD*

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Teori graf adalah bagian dari matematika diskrit yang banyak digunakan sebagai alat bantu untuk menggambarkan atau menyatakan suatu persoalan agar lebih mudah dimengerti dan diselesaikan. Banyak persoalan akan lebih jelas untuk diterangkan apabila persoalan tersebut dapat dimodelkan dalam bentuk graf.

Suatu graf *G* didefinisikan sebagai pasangan himpunan (*V*, *E*), dimana *V* adalah himpunan tak kosong simpul dan *E* adalah himpunan pasangan tak terurut dari dari simpul-simpul yang disebut busur. Suatu graf bisa berhingga atau tak berhingga, tergantung pada himpunan *V*, apakah berhingga atau tidak. Dalam skripsi ini hanya akan dibahas graf berhingga. Banyaknya simpul pada *G* dinotasikan dengan $|V|$ dan banyaknya busur dinotasikan dengan $|E|$.

Graf-graf yang memiliki ciri khusus biasa dikelompokkan menjadi suatu kelas graf dan diberi nama tersendiri. Misalnya, graf lingkaran, roda, *friendship*, kipas, dan jahangir yang diperumum. Graf lingkaran C_n dibentuk dari n simpul $(v_1, v_2, \dots, v_n)$ dan n busur $(v_1v_2, v_2v_3, \dots, v_{n-1}v_n, v_nv_1)$. Busur v_1v_2 menghubungkan simpul v_1 dan v_2, busur v_2v_3 menghubungkan simpul v_2 dan v_3, dan seterusnya. Dengan menambahkan satu simpul tengah *c* pada graf lingkaran dan menghubungkan simpul *c* tersebut dengan $v_1, v_2, \dots, v_n$, akan didapatkan graf roda W_n. Bagian lingkaran pada graf roda ini biasa disebut rim. Sementara busur yang menghubungkan simpul *c* dengan simpul *v* biasa disebut jari-jari. Jika satu busur pada bagian rim dari graf roda dihapus, maka akan didapatkan graf kipas f_n. Dari graf roda dengan ukuran n genap, jika busur pada bagian rim dihapus seba-nyak $n/2$, secara selang-seling, maka diperoleh graf *friendship* F_n. Sementara itu, jika yang dihapus adalah jari-jari pada graf roda, secara selang-seling, maka akan didapatkan graf jahangir yang diperumum $J_{n,m}$.

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

Pelabelan graf adalah pemetaan himpunan simpul atau busur, atau keduanya ke himpunan bilangan bulat, dengan kondisi tertentu. Jika domain dari pelabelan tersebut himpunan simpul, maka disebut pelabelan simpul. Jika domainnya himpunan busur, maka disebut pelabelan busur. Sedangkan, jika domainnya merupakan himpunan simpul dan busur, maka pelabelannya disebut pelabelan total. Dalam pelabelan total, terdapat istilah bobot simpul. Bobot simpul adalah jumlah label dari suatu simpul dengan label-label dari busur yang berhubungan dengan simpul tersebut. Jika bobot simpul dari semua simpul pada suatu pelabelan total sama dengan suatu konstanta k, maka pelabelan tersebut disebut pelabelan total simpul ajaib. Konstanta k disebut sebagai konstanta ajaib untuk pelabelan tersebut. Dalam skripsi ini, pelabelan yang akan dibahas adalah pelabelan total simpul ajaib (PTSA).

Dua pelabelan λ dan λ' untuk suatu graf G dikatakan isomorfik jika terdapat pemetaan bijektif dari λ ke λ' sedimikian sehingga simpul-simpul berlabel yang dipetakan bertetangga jika dan hanya jika simpul-simpul berlabel tersebut bertetangga.

Selama ini telah banyak hasil penelitian yang menunjukkan eksistensi pelabelan pada setiap kelas graf. Misalnya MacDougall dkk (2002) menunjukkan bahwa graf *friendship* F_n tidak memiliki PTSA untuk $n > 3$, graf kipas f_n tidak memiliki PTSA untuk $n > 10$, menurut Gallian (2009) graf jahangir $J_{n,t+1}$ memiliki PTSA untuk $n = 3$ dan $t = 1$ dan tidak memiliki PTSA untuk $n \geq 7t + 11$ dan $t \geq 1$. Hasil yang diberikan tersebut belum mencakup semua PTSA yang mungkin (jika ada) untuk kelas graf yang bersangkutan. Hanya graf dengan ukuran tertentu saja atau untuk nilai k tertentu saja. Membangun semua pelabelan yang mungkin (jika ada) dan tidak isomorfik pada suatu kelas graf bukanlah sesuatu yang mudah. Salah satu cara yang dapat dilakukan untuk membangun semua pelabelan tersebut adalah dengan menggunakan algoritma pelabelan. Masalahnya, algoritma pelabelan sembarang graf secara umum bersifat *NP-complete*. Sehingga algoritma pelabelan biasanya dibangun hanya untuk kelas graf tertentu saja. Baker dan Sawada (2008) dalam makalahnya telah membahas mengenai algoritma PTSA untuk graf lingkaran dan graf roda. Menggunakan algoritma tersebut, dapat diperoleh semua PTSA yang mungkin dan tidak isomorfik untuk kedua graf tersebut. Dalam skripsi

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

ini ide dari pembentukan algoritma PTSA untuk graf roda akan digunakan untuk mendapatkan algoritma-algoritma PTSA untuk kelas graf lain yang berkaitan dengan graf roda, seperti graf *friendship*, kipas, dan jahangir yang diperumum.

1.2 Permasalahan

Bagaimanakah cara membangun algoritma untuk mendapatkan semua PTSA yang tidak isomorfik untuk kelas graf yang berkaitan dengan graf roda?

1.3 Tujuan

Membentuk algoritma-algoritma PTSA untuk kelas graf yang berkaitan dengan graf roda. Mengimplementasikan algoritma-algoritma tersebut dengan program MATLAB dan melakukan simulasi untuk mendapatkan semua PTSA yang mungkin untuk setiap kelas graf tersebut.

1.4 Pembatasan Masalah

Dalam skripsi ini hanya akan dibentuk algoritma-algoritma PTSA untuk graf *friendship*, kipas, dan jahangir yang diperumum. Algoritma-algoritma tersebut kemudian diimplementasikan dengan program MATLAB. Simulasi dilakukan untuk setiap nilai k yang mungkin, sehingga banyaknya PTSA yang mungkin dan tidak isomorfik untuk setiap kelas graf tersebut dapat diperoleh.

1.5 Sistematika Penulisan

Skripsi ini terbagi menjadi 5 bab. Dalam bab 2 akan diberikan definisi-definisi dan konsep-konsep dasar yang berkaitan dengan graf dan pelabelan. Dalam bab 3, diberikan algoritma PTSA untuk graf *friendship*, kipas, dan jahangir yang diperumum. Implementasi algoritma dan simulasi akan diberikan pada bab 4. Terakhir, akan diberikan kesimpulan mengenai hasil yang diperoleh pada bab 5.

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

BAB 2
LANDASAN TEORI

Pada bab ini akan diberikan beberapa definisi dan konsep dasar dalam teori graf, serta definisi pelabelan yang akan digunakan. Definisi yang digunakan dalam bab ini secara umum diambil dari (Hartsfield & Ringel, 2003).

2.1 Teori Graf

Suatu graf G adalah pasangan himpunan (V, E) dimana V adalah himpunan tak kosong dari simpul, dan E adalah himpunan pasangan tak terurut dari simpul-simpul yang disebut busur. Himpunan simpul dari graf G biasa dinyatakan dengan $V(G)$ dan himpunan busurnya dinyatakan dengan $E(G)$. Himpunan busur $E(G)$ bisa merupakan suatu himpunan kosong. Jika demikian, maka G disebut graf kosong. Suatu graf G bisa berhingga atau tak berhingga, tergantung pada $V(G)$ berhingga atau tak berhingga. Banyaknya anggota himpunan $V(G)$ biasa dinotasikan dengan $|V|$ dan untuk himpunan $E(G)$ dinotasikan dengan $|E|$.

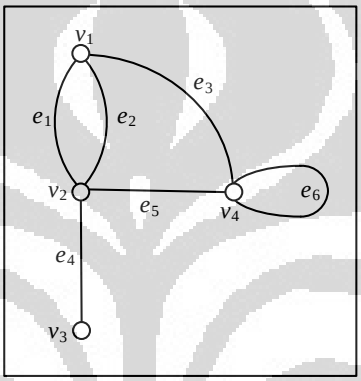
Busur pada suatu graf menghubungkan dua simpul (boleh sama) pada graf tersebut. Simpul-simpul tersebut disebut simpul akhir (*endpoint*) dari busur yang menghubungkannya. Jika simpul akhir dari suatu busur merupakan simpul yang sama, maka busur tersebut disebut gelung (*loop*). Dua simpul pada suatu graf dapat dihubungkan dengan lebih dari satu busur, busur-busur seperti ini disebut busur berganda. Graf yang tidak mengandung gelung dan busur berganda disebut graf sederhana.

Pada suatu graf G, simpul u dikatakan bertetangga (*adjacent*) dengan simpul v jika terdapat busur yang menghubungkan kedua simpul tersebut. Busur tersebut biasa dinotasikan dengan uv. Simpul v dikatakan hadir (*incident*) pada busur e jika v adalah simpul akhir dari busur e. Begitu juga busur e dikatakan hadir pada simpul v ketika simpul v menjadi simpul akhir dari busur e.

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

Derajat dari suatu simpul *v* adalah banyaknya busur yang hadir pada simpul *v*. Derajat dari simpul *v* biasa dinotasikan dengan *d*(*v*). Simpul dengan derajat 0 disebut simpul terisolasi. Simpul dengan derajat 1 disebut simpul terminal.

Graf *G* dapat direpresentasikan dalam bentuk diagram. Simpul pada graf *G* direpresentasikan dengan titik, sementara busurnya direpresentasikan dengan garis yang menghubungkan simpul. Simpul biasanya ditulis sebagai $v_1, v_2, \ldots, v_n$. Sedangkan busur dapat ditulis dengan menyatakan kedua simpul akhirnya v_iv_j, dimana $v_iv_j \in E$ dan $i, j \in \{1, \ldots, n\}$, atau dengan menulisnya sebagai $e_1, e_2, \ldots, e_m$.



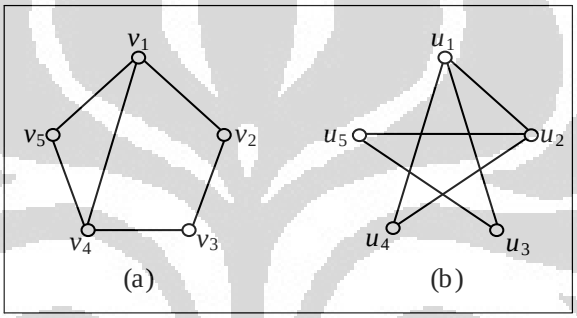
Gambar 2. 1 Graf *G*

Contoh graf *G* dengan himpunan simpul $V = \{v_1, v_2, v_3, v_4\}$ dan himpunan busur $E = \{v_1v_2, v_1v_2, v_1v_4, v_2v_3, v_2v_4, v_4v_4\} = \{e_1, e_2, e_3, e_4, e_5, e_6\}$ ditunjukkan pada Gambar 2.1. Banyaknya simpul dan busur pada graf tersebut adalah $|V| = 4$ dan $|E| = 6$. Gelung ditunjukkan oleh e_6 dan busur berganda ditunjukkan oleh e_1, e_2. Karena graf ini mengandung gelung dan busur berganda, maka graf ini bukan graf sederhana. Derajat dari setiap simpul pada graf tersebut adalah $d(v_1) = 3$, $d(v_2) = 3$, $d(v_3) = 1$, dan $d(v_4) = 4$.

Pada suatu graf *G*, deretan busur-busur yang membentuk sambungan tidak putus pada graf *G* disebut jalan (*walk*). Jika jalan tersebut tidak mengandung busur yang berulang, maka jalan tersebut disebut jalur (*trail*). Jika jalur ini tidak mengandung simpul yang berulang, maka disebut lintasan (*path*). Dengan kata lain, lintasan pada suatu graf *G* adalah jalan yang tidak mengandung simpul dan busur yang berulang. Suatu graf *G* dikatakan terhubung jika untuk sembarang dua

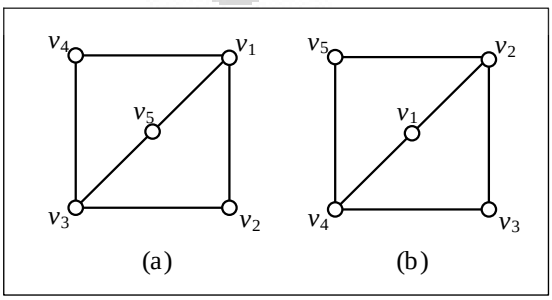
Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

simpul *u* dan *v* pada graf *G* terdapat lintasan dari simpul *u* ke *v*. Dalam skripsi ini graf yang akan digunakan adalah graf berhingga, sederhana, dan terhubung.
Dua graf yang tampak berbeda bisa jadi merupakan graf yang sama. Graf pada Gambar 2.2 merupakan graf yang sama, atau biasa disebut isomorfik. Dua graf $G = (V, E)$ dan $H = (W, F)$ dimana $|V| = |W|$ dan $|E| = |F|$ dikatakan isomor-fik jika terdapat pemetaan bijektif $f: V \rightarrow W$ sedemikian sehingga untuk semua simpul $v, w \in V: vw \in E \Leftrightarrow f(v)f(w) \in F$ (Graph Isomorphism). Pemetaan yang menyatakan kedua graf pada Gambar 2.2 adalah graf yang isomorfik adalah sebagai berikut. Simpul v_1 dipetakan ke simpul u_1, v_2 ke u_3, v_3 ke u_5, v_4 ke u_2, dan v_5 ke u_4.



Gambar 2. 2 Contoh graf isomorfik

Namun, dua graf yang tampak sama bisa jadi merupakan graf yang berbeda. Pada Gambar 2.3, graf (a) jelas memiliki struktur yang sama dengan graf (b). Tetapi, jika nama pada simpul (label) pada graf tersebut diperhitungkan, maka kedua graf tersebut bukanlah graf yang isomorfik. Isomorfisma graf dengan memperhatikan label pada graf tersebut dikenal dengan istilah isomorfisma pelabelan. Hal ini akan dibahas lebih lanjut pada subbab pelabelan graf.

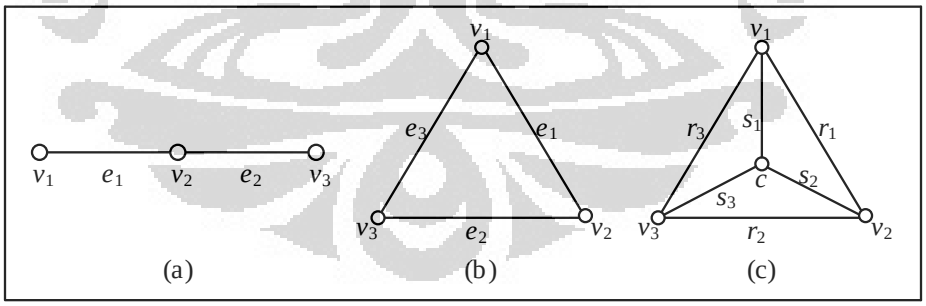


Gambar 2. 3 Contoh graf berlabel yang tidak isomorfik

Pada subbab selanjutnya akan diberikan definisi dari beberapa jenis graf yang akan digunakan dalam skripsi ini.

2.2 Jenis-Jenis Graf

Graf Lintasan P_n adalah graf dengan n simpul $V = \{v_1, v_2, \dots, v_n\}$ dan $n - 1$ busur $E = \{v_1v_2, v_2v_3, \dots, v_{n-1}v_n\}$, dimana v_1 merupakan simpul awal dan v_n merupakan simpul akhir dari graf ini. Graf lingkaran C_n adalah graf yang didapatkan dengan menambahkan satu busur yang menghubungkan simpul akhir dan simpul awal dari suatu graf lintasan P_n. Graf lingkaran C_n terdiri dari n simpul dan n busur. Dengan menambahkan satu simpul tengah (c) dan menghubungkannya dengan setiap simpul pada graf lingkaran, maka diperoleh graf roda W_n. Simpul dan busur graf roda pada bagian luar (yang membentuk graf lingkaran) disebut rim. Busur yang menghubungkan simpul tengah dengan simpul pada bagian rim disebut jari-jari (*spoke*). Busur pada bagian rim biasa dinotasikan dengan r, sedangkan jari-jari dinotasikan dengan s. Nilai n pada graf roda menunjukkan ba-nyaknya simpul pada bagian rim, jadi banyaknya simpul dan busur dari graf roda adalah $|V| = n + 1$ dan $|E| = 2n$. Gambar 2.4 menunjukkan contoh graf lintasan, lingkaran, dan roda, masing-masing dengan $n = 3$.



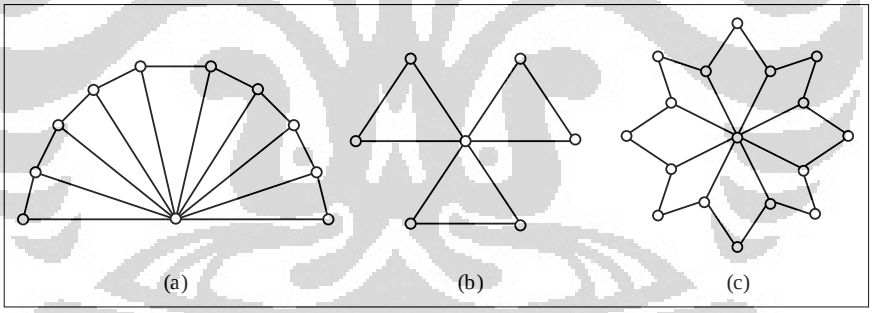
Gambar 2. 4 (a) graf lintasan P_3 (b) graf lingkaran C_3 (c) graf roda W_3

Ada beberapa jenis graf yang bisa didapatkan dari graf roda dengan menghapus satu atau beberapa busurnya dengan cara tertentu. Diantaranya adalah graf kipas, graf *friendship*, dan graf jahangir yang diperumum.

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

Graf kipas f_n diperoleh dengan menghapus satu busur pada bagian rim graf roda. Graf kipas juga dapat diperoleh dengan menghubungkan semua simpul pada graf lintasan P_n dengan satu simpul tengah (c) (MacDougall *et al*., 2002). Sama halnya dengan graf roda, nilai n pada graf kipas menunjukkan banyaknya simpul pada bagian rim, sehingga banyaknya simpul pada graf kipas f_n akan sama dengan graf roda W_n, yaitu $|V| = n + 1$. Sementara banyak busur pada graf kipas adalah $|E| = 2n - 1$.

Graf *friendship* F_n didapat dengan cara menghapus sebanyak $n/2$ busur pada bagian rim secara berselang-seling. Oleh karena itu, graf *friendship* hanya bisa didapatkan dari graf roda dengan n genap. Graf *friendship* juga dapat dipandang sebagai gabungan dari graf C_3 dengan satu simpul bersama (c) (MacDougall *et al*., 2002). Berbeda dengan graf roda dan kipas, nilai n pada graf *friendship* menyatakan banyaknya graf C_3 yang membentuk graf *friendship*. Dengan demikian, simpul dan busur yang membentuk graf *friendship* F_n adalah sebanyak $|V| = 2n + 1$ dan $|E| = 3n$.



Gambar 2. 5 (a) graf kipas f_{10} (b) graf *friendship* F_3 (c) graf jahangir $J_{8,2}$

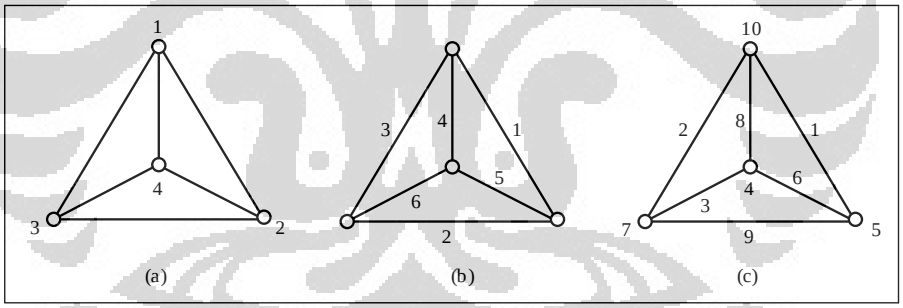
Lain halnya dengan graf kipas dan *friendship*, graf jahangir yang diperumum $J_{n,m}$ didapatkan dengan menghapus jari-jari dari graf roda secara berselang-seling. Nilai n pada $J_{n,m}$ menyatakan banyaknya simpul pada bagian rim yang terhubung dengan simpul tengah. Sementara nilai m menyatakan jarak antar simpul pada rim yang terhubung dengan simpul tengah. Jadi, graf jahangir yang diperumum adalah graf yang terdiri dari $nm + 1$ simpul yang terdiri dari graf lingkaran C_{nm} dan satu simpul tambahan yang bertetangga dengan n simpul pada C_{nm}, dimana jarak antar simpul pada C_{nm} yang bertetangga dengan simpul tambahan

tersebut adalah *m* (Gallian, 2009). Jadi, jelas bahwa banyaknya simpul pembentuk graf ini adalah $|V| = nm + 1$, sedangkan busur yang membentuk graf ini adalah sebanyak $|E| = nm + n$.

Gambar 2.5 menunjukkan contoh graf kipas dengan *n* = 10, graf *friendship* dengan *n* = 3, dan graf jahangir yang diperumum dengan *n* = 8 dan *m* = 2.

2.3 Pelabelan Graf

Pelabelan adalah pemetaan himpunan simpul, busur, atau keduanya ke himpunan bilangan bulat positif, dengan kondisi tertentu. Jika domainnya merupakan himpunan simpul, maka pelabelan tersebut disebut pelabelan simpul. Jika domainnya merupakan himpunan busur, maka disebut pelabelan busur. Sementara jika domainnya merupakan gabungan dari himpunan simpul dan busur, maka disebut pelabelan total. Contoh pelabelan simpul, busur, dan total pada graf roda ditunjukkan pada Gambar 2.6.



Gambar 2. 6 (a) pelabelan simpul pada W_3 (b) pelabelan busur pada W_3 (c) pelabelan total pada W_3

Bilangan yang diberikan pada simpul disebut label simpul, dan pada busur disebut label busur. Jumlah label yang terkait dengan suatu elemen pada graf disebut bobot. Pada pelabelan total, jumlah label simpul dengan seluruh label busur yang hadir pada simpul tersebut disebut bobot simpul. Sedangkan jumlah label busur dengan label simpul yang merupakan simpul akhir dari busur tersebut disebut bobot busur. Jika seluruh bobot simpul atau busur pada suatu pelabelan total bernilai sama, maka pelabelan tersebut disebut pelabelan total simpul ajaib atau pelabelan total busur ajaib.

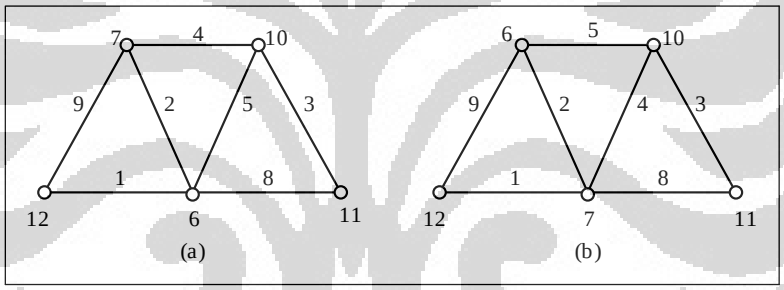
Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

Dalam skripsi ini, jenis pelabelan yang dibahas adalah pelabelan total simpul ajaib (PTSA), yaitu pelabelan total dimana seluruh bobot simpulnya bernilai sama. Contoh PTSA untuk graf kipas f_4 dengan k = 22 dapat dilihat pada Gambar 2.7.

Definisi 2.1 (MacDougall *et al*., 2002) Pemetaan bijektif λ dari $E \cup V$ ke himpunan bilangan bulat $\{1,2,\ldots,|E|+|V|\}$ disebut pelabelan total simpul ajaib (PTSA) jika terdapat konstanta k sedemikian sehingga untuk setiap simpul v,

$$w(v) = \lambda(v) + \sum_{u \in N(v)} \lambda(vu) = k$$

dimana $N(v)$ adalah himpunan simpul yang bertetangga dengan simpul v. Nilai $w(v)$ disebut bobot dari simpul v dan konstanta k disebut sebagai konstanta ajaib untuk λ.



Gambar 2. 7 Contoh PTSA pada f_4 dengan k = 22

Konstanta ajaib k yang mungkin untuk PTSA suatu graf bisa didapatkan dengan menggunakan perhitungan dasar (*basic counting*). Dalam PTSA, label setiap simpul berkontribusi sekali pada bobot setiap simpul. Sementara label busur berkontribusi dua kali, yaitu pada bobot setiap simpul akhirnya. Sehingga, jumlah dari seluruh bobot simpul pada graf akan sama dengan penjumlahan seluruh label simpul dan dua kali seluruh label busur. Secara matematis

$$\sum_{i=1}^{|V|} w_i(v) = S_v + 2S_e = |V|k \qquad (2.1)$$

dimana S_v adalah penjumlahan dari semua label simpul dan S_e adalah penjumlahan dari semua label busur. Penjumlahan semua label simpul dan semua label busur sama dengan penjumlahan $\{1,2,\ldots,|E|+|V|\}$, atau

$$S_v + S_e = \sum_{i=1}^{|E|+|V|} i \qquad (2.2)$$

Batas bawah untuk k didapatkan dengan melabel semua busur dengan $|E|$ label terkecil. Sedangkan, batas atasnya bisa didapatkan dengan melabel semua busur dengan $|E|$ label terbesar.

$$\sum_{i=1}^{|E|} i \leq S_e \leq \sum_{i=|V|+1}^{|E|+|V|} i \tag{2.3}$$

Dari (2.1), (2.2), dan (2.3) diperoleh nilai k yang mungkin untuk PTSA dari graf dengan $|V|$ simpul dan $|E|$ busur, yaitu

$$\sum_{i=1}^{|E|} i + \sum_{i=1}^{|E|+|V|} i \leq |V|k \leq \sum_{i=|V|+1}^{|E|+|V|} i + \sum_{i=1}^{|E|+|V|} i$$

atau,

$$\frac{|E|(|E|+1)}{2} + \frac{(|E|+|V|)(|E|+|V|+1)}{2} \leq |V|k \leq \frac{|E|(|E|+2|V|+1)}{2} + \frac{(|E|+|V|)(|E|+|V|+1)}{2} \tag{2.4}$$

Dari persamaan terakhir ini, batas konstanta ajaib untuk PTSA suatu graf bisa didapatkan dengan mensubstitusi nilai $|V|$ dan $|E|$ dari graf tersebut. Bobot simpul didapatkan dari penjumlahan label yang merupakan bilangan bulat. Jadi, bobot simpul pasti merupakan bilangan bulat. Begitu juga konstanta ajaib *k*. Jika batas konstanta ajaib *k* yang didapatkan bukan bilangan bulat, maka dilakukan pembulatan. Jika batas bawah yang berbentuk pecahan, maka dibulatkan ke bilangan bulat terbesar terdekat. Sedangkan, jika batas atas, maka dibulatkan ke bilangan bulat terkecil terdekat. Misalnya untuk graf roda W_n. Banyaknya simpul dan busur untuk graf tersebut adalah $|V| = n + 1$ dan $|E| = 2n$. Dengan mensubstitusi nilai $|V|$ dan $|E|$ tersebut pada persamaan (2.4), maka diperoleh batas konstanta ajaib *k* untuk PTSA pada graf roda yaitu

$$\frac{13n^2+11n+2}{2(n+1)} \leq k \leq \frac{17n^2+15n+2}{2(n+1)}$$

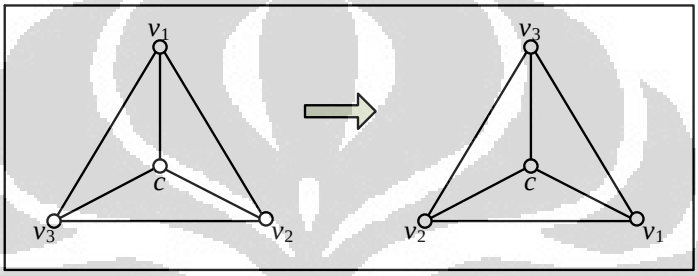
Contohnya untuk graf roda dengan $n = 3$. Batas *k* untuk PTSA pada W_3 adalah $19 \leq k \leq 25$.

Graf (a) dan (b) pada Gambar 2.7 merupakan graf yang isomorfik berdasarkan definisi graf isomorfik pada Subbab pertama. Namun, jika label pada kedua graf tersebut diperhatikan dalam menentukan apakah keduanya isomorfik, maka keduanya tidak isomorfik. Simpul dengan label 7 pada gambar (a) bertetangga dengan simpul berlabel 12, 6, dan 10. Sedangkan pada gambar (b), simpul dengan label 7 bertetangga dengan simpul berlabel 12, 6, 10, dan 11. Karena itu, kedua graf pada Gambar 2.7 tidak isomorfik. Seperti telah disinggung sebelumnya, penentuan apakah dua graf isomorfik dengan memperhatikan tidak hanya

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

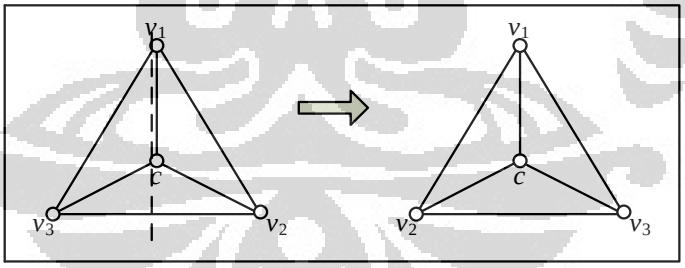
struktur tetapi juga label disebut isomorfisma pelabelan. Jadi, dua graf $G = (V, E)$ dan $H = (W, F)$, dimana $|V| = |W|$ dan $|E| = |F|$, dengan pelabelan $l: V \rightarrow L$ dan $l': W \rightarrow L$ dikatakan memiliki pelabelan yang isomorfik jika terdapat pemetaan bijektif $f: V \rightarrow W$ sedemikian sehingga untuk setiap $v, w \in V: vw \in E \Leftrightarrow f(v)f(w) \in F$ dan untuk setiap $v \in V: l(v) = l'(f(v))$ (Graph Isomorphism).

Isomorfisma pelabelan pada graf dengan struktur yang sama bisa terjadi karena rotasi atau refleksi. Contoh kasus pelabelan isomorfik karena rotasi bisa terlihat pada Gambar 2.8.



Gambar 2. 8 Contoh kasus rotasi pada W_3

Gambar 2.9 menunjukkan contoh pelabelan isomorfik karena kasus refleksi. Garis putus-putus menunjukkan sumbu simetri dari refleksi yang terjadi.



Gambar 2. 9 Contoh kasus refleksi pada W_3

2.4 Hasil yang Telah Diketahui

Hasil penelitian mengenai eksistensi PTSA pada suatu kelas graf telah banyak dipublikasikan, diantaranya MacDougall dkk (2002) telah menunjukkan bahwa graf kipas f_n tidak memiliki PTSA untuk $n > 10$ dan graf *friendship* F_n

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

tidak memiliki PTSA untuk $n > 3$. Begitu juga eksistensi PTSA untuk graf jahangir yang diperumum, menurut Gallian (2009) Rahim dkk membuktikan bahwa graf jahangir yang diperumum $J_{n,t+1}$ memiliki PTSA untuk $n = 3$ dan $t = 1$ dan tidak memiliki PTSA untuk $n \geq 7t + 11$ dan $t \geq 1$.

Hasil penelitian tersebut hanya menunjukkan eksistensi PTSA untuk nilai *k* tertentu saja, belum mencakup setiap nilai *k* yang mungkin. Salah satu cara yang bisa dilakukan untuk mendapatkan PTSA untuk setiap nilai *k* yang mungkin (jika ada) dari suatu graf adalah dengan menggunakan algoritma seperti pada Baker dan Sawada (2008). Mereka membangun algoritma PTSA untuk graf lingkaran dan graf roda. Menggunakan algoritma tersebut bisa diperoleh seluruh PTSA tidak isomorfik untuk masing-masing kelas graf.

Dalam skripsi ini, ide dari pembangunan algoritma PTSA untuk graf roda akan digunakan untuk membangun algoritma PTSA untuk kelas graf lain yang berkaitan dengan graf roda, yaitu graf *friendship*, kipas, dan jahangir yang diperumum.

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

BAB 3
ALGORITMA PTSA

Dalam bab ini akan dibahas mengenai algoritma PTSA untuk beberapa kelas graf yang berkaitan dengan graf roda. Telah disebutkan pada bab sebelumnya bahwa Baker dan Sawada (2008) telah membangun suatu algoritma PTSA untuk graf lingkaran dan graf roda. Menggunakan algoritma tersebut, semua PTSA yang mungkin yang berbeda (tidak isomorfik) dari masing-masing kelas graf bisa diketahui.

Setiap kelas graf memiliki struktur tersendiri. Karena itu algoritma pelabelan sembarang graf secara umum merupakan masalah *NP-complete*. Sehingga, algoritma pelabelan biasanya dibangun secara khusus untuk tiap kelas graf tertentu saja. Salah satunya adalah algoritma pelabelan yang dibangun oleh Baker dan Sawada yaitu algoritma PTSA untuk graf roda. Dengan menggunakan ide dari algoritma tersebut, bisa didapatkan algoritma-algoritma PTSA untuk kelas graf yang berkaitan dengan graf roda, diantaranya graf *friendship*, kipas, dan jahangir yang diperumum.

Secara umum, ide dari algoritma-algoritma PTSA untuk graf *friendship*, kipas, dan jahangir yang diperumum adalah melabel semua simpul dan busur dengan label yang mungkin secara iteratif, hingga seluruh simpul memiliki bobot yang sama. Setiap label hanya dapat digunakan satu kali karena pelabelan merupakan pemetaan satu-satu. Label busur akan diberikan sesuai dengan label yang tersedia. Sementara label simpul ditentukan melalui pengurangan konstanta ajaib *k* dengan jumlah label dari busur yang hadir pada simpul tersebut. Jika label simpul yang diperoleh tidak tersedia, maka elemen graf yang terakhir dilabel (busur atau simpul) akan diganti labelnya dengan label lain yang tersedia, sampai label simpul diperoleh. Jika label simpul tetap tidak diperoleh, maka dilakukan *backtracking* untuk mengubah label elemen graf yang dilabel sebelumnya. Hal ini dilakukan hingga diperoleh label simpul yang sesuai dengan label yang tersedia. Setelah satu PTSA terbentuk (jika ada), untuk mencari PTSA yang lain yang mungkin, juga dilakukan dengan proses *backtracking*. Elemen graf yang terakhir dilabel

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

diganti labelnya. Kemudian elemen kedua terakhir yang dilabel, sampai elemen pertama graf yang dilabel. Label elemen-elemen graf tersebut diganti dengan kemungkinan label lain yang tersedia pada tahap tersebut.

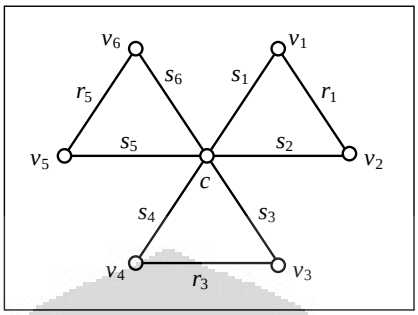
Lalu, untuk menjamin pelabelan yang dihasilkan (jika ada) tidak isomorfik, diberikan syarat tertentu dalam algoritma tersebut. Syarat yang diberikan tergantung dari kasus isomorfik yang mungkin terjadi pada setiap kelas graf. Memeriksa apakah sembarang dua pelabelan adalah isomorfik merupakan masalah *NP-Hard*. Dibandingkan dengan membangun pelabelan, kemudian memeriksa pelabelan yang dihasilkan isomorfik atau tidak, pemberian syarat yang menjamin pelabelan yang dihasilkan tidak isomorfik tentunya merupakan cara yang lebih efisien. Algoritma-algoritma PTSA tersebut bertujuan menghasilkan semua PTSA yang mungkin dan tidak isomorfik untuk masing-masing kelas graf. Karena itu, masukan (*input*) yang perlu diberikan adalah nilai *n* untuk graf *friendship* dan kipas, nilai *n* dan *m* untuk graf jahangir yang diperumum, serta nilai konstanta ajaib *k*. Keluarannya (*output*) adalah label untuk semua simpul dan busur dari graf yang berkaitan yang memenuhi ketentuan PTSA. Pembahasan lengkap mengenai algoritma -algoritma PTSA untuk graf *friendship*, kipas, dan jahangir yang diperumum akan diberikan pada subbab 3.1, 3.2, dan 3.3.

3.1 Algoritma PTSA Graf *Friendship*

Graf *friendship* F_n dapat diperoleh dari graf roda dengan menghapus busur pada bagian rim secara berselang-seling. Graf ini juga dapat dipandang sebagai gabungan dari *n* graf segitiga dengan satu simpul bersama, disebut *c*. Banyaknya simpul dan busur dari graf segitiga, masing-masing adalah 3. Karena satu simpul pada segitiga pembentuk graf *friendship* menjadi simpul bersama, maka banyaknya simpul pada graf *friendship* F_n adalah $|V| = 2n + 1$. Sementara banyaknya busur adalah $|E| = 3n$. Jika $n = 1$, maka graf *friendship* akan sama dengan graf lingkaran C_3. Dalam skripsi ini, graf *friendship* yang dibahas adalah graf *friendship* dengan $n \geq 2$. Seperti telah disinggung pada Bab 2, busur pada bagian rim dinotasikan dengan *r*. Sementara itu, busur yang menghubungkan simpul pada bagian rim dengan simpul tengah biasa disebut jari-jari (*spoke*) dan dinotasikan

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

dengan *s*. Gambar 3.1 menunjukkan graf F_3 beserta penamaan untuk setiap simpul dan busurnya.



Gambar 3. 1 Graf F_3

Batasan konstanta ajaib *k* yang mungkin untuk PTSA pada graf *friendship* dapat diperoleh dengan mensubstitusi nilai $|V|$ dan $|E|$ ke (2.4) yang telah didapatkan pada Bab 2, yaitu

$$\frac{|E|(|E|+1)}{2}+\frac{(|E|+|V|)(|E|+|V|+1)}{2} \le |V|k \le \frac{|E|(|E|+2|V|+1)}{2}+\frac{(|E|+|V|)(|E|+|V|+1)}{2}.$$

Maka diperoleh,

$$\frac{3n(3n+1)}{2}+\frac{(5n+1)(5n+2)}{2} \le (2n+1)k \le \frac{3n(7n+3)}{2}+\frac{(5n+1)(5n+2)}{2},$$

atau,

$$\frac{17n^2+9n+1}{2n+1} \le k \le \frac{23n^2+12n+1}{2n+1}. \qquad (3.1)$$

Selain menggunakan penghitungan dasar seperti yang telah dibahas pada Bab 2, batas konstanta ajaib *k* juga dapat dihitung dengan memperhatikan struktur graf *friendship*, seperti pada (MacDougall *et al*., 2002). Misalkan *M* adalah banyaknya label yang mungkin untuk melabel graf *friendship* yang diberikan. Pada graf *friendship* $|V| = 2n+1$ dan $|E| = 3n$, maka $M = 5n+1$. Simpul dengan derajat terbesar pada graf *friendship* F_n adalah simpul tengah *c*, dengan $d(c) = 2n$. Label yang dibutuhkan untuk melabel simpul *c* dan busur yang *hadir pada* simpul tersebut adalah $2n+1$. Batas bawah untuk *k* tidak mungkin lebih kecil dari bobot terendah yang mungkin untuk simpul *c*. Secara matematis,

$$k \ge w(c) = \frac{(2n+1)(2n+2)}{2} = (2n+1)(n+1). \qquad (3.2)$$

Selanjutnya perhatikan jumlah seluruh bobot simpul pada bagian rim. Batas atas dari jumlahan tersebut bisa diperoleh dengan melabel busur pada

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

bagian rim dengan *n* label terbesar dan simpul pada bagian rim serta jari-jari dengan 4*n* label terbesar berikutnya.

$$\begin{aligned} w(v_1) + \cdots + w(v_n) &\leq 2\sum_{i=4n+2}^{5n+1} i + \sum_{i=2}^{4n+1} i \\ &= \sum_{i=2}^{5n+1} i + \sum_{i=4n+2}^{5n+1} i \\ &= \frac{5n(5n+3)}{2} + \frac{n(9n+3)}{2} \\ &= \frac{34n^2+18n}{2} \\ &= n(17n+9). \end{aligned}$$

Batas atas dari *k* tidak mungkin lebih besar dari bobot terbesar yang mungkin dimiliki simpul pada bagian rim. Karena simpul pada bagian rim ada sebanyak 2*n*, maka

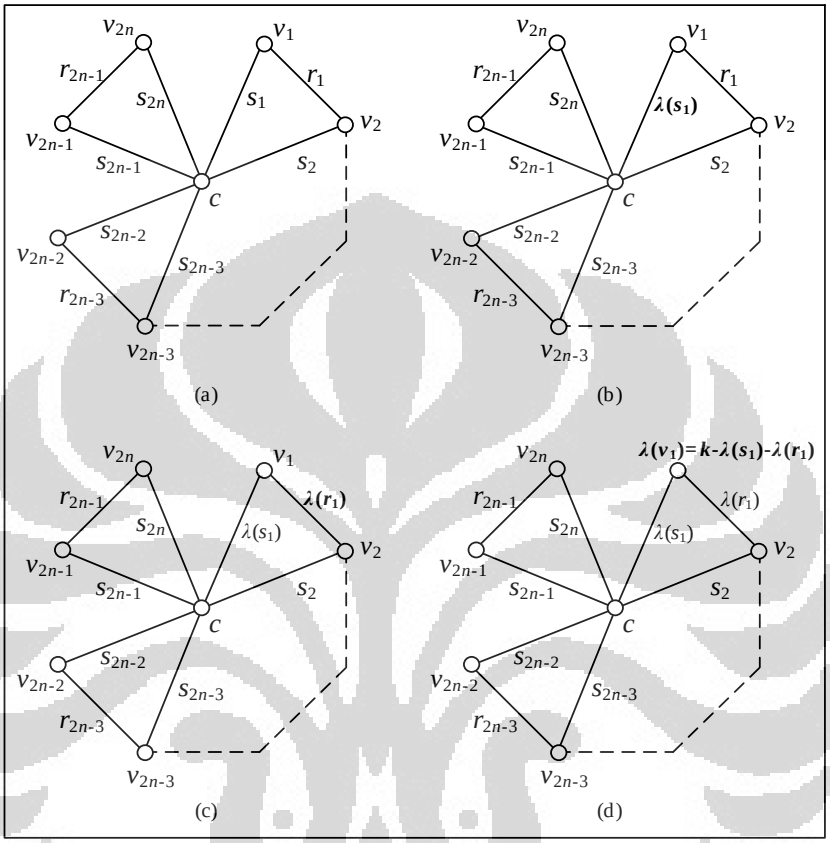
$$k \leq \frac{17n+9}{2}. \tag{3.3}$$

Saat $n = 4$, nilai batas atas *k* lebih besar dari batas bawahnya. Karena itu tidak mungkin terdapat PTSA untuk $n > 3$. Batas nilai *k* yang digunakan adalah nilai maksimal *k* dari (3.1) dan (3.2) sebagai batas bawah. Sementara itu, batas atas *k* adalah minimal nilai *k* dari (3.1) dan (3.3). Sama seperti yang telah disinggung pada Bab 2, jika batas bawah berbentuk pecahan, maka batas bawah tersebut dibulatkan keatas. Tetapi, jika batas atas, maka batas atas tersebut dibulatkan kebawah. Contohnya untuk graf *friendship* dengan $n = 2$. Batas *k* untuk graf ini berdasarkan (3.1) adalah $17.4 \leq k \leq 23.4$ dengan pembulatan menjadi $18 \leq k \leq 23$, sedangkan berdasarkan (3.2) dan (3.3) adalah $15 \leq k \leq 21.5$ dengan pembulatan menjadi $15 \leq k \leq 21$. Jadi, batas konstanta ajaib *k* untuk PTSA pada F_2 adalah $18 \leq k \leq 21$.

Seperti telah disebutkan sebelumnya, algoritma PTSA untuk graf ini bekerja secara iteratif. Untuk sembarang simpul *v*, busur-busur yang hadir pada simpul tersebut akan dilabel terlebih dahulu dengan label yang tersedia. Label yang mungkin dalam membangun PTSA untuk graf F_n adalah $\{1,2,\ldots,5n+1\}$. Karena pelabelan adalah pemetaan yang bijektif, label yang telah digunakan tidak dapat digunakan lagi untuk melabel busur dan simpul yan tersisa. Dengan kata lain, label yang telah digunakan tidak lagi tersedia untuk melabel busur dan simpul yang tersisa. Sementara itu, simpul *v* dan simpul tengah *c* dilabel dengan hasil pengurangan nilai *k* dengan label busur yang hadir pada *v* dan *c*. Karena pada

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

algoritma graf *friendship* label simpul bergantung pada label-label yang lain, maka label simpul ini biasa disebut *determined label*. Jika label simpul yang dihasilkan tidak tersedia, maka dilakukan *backtracking*.

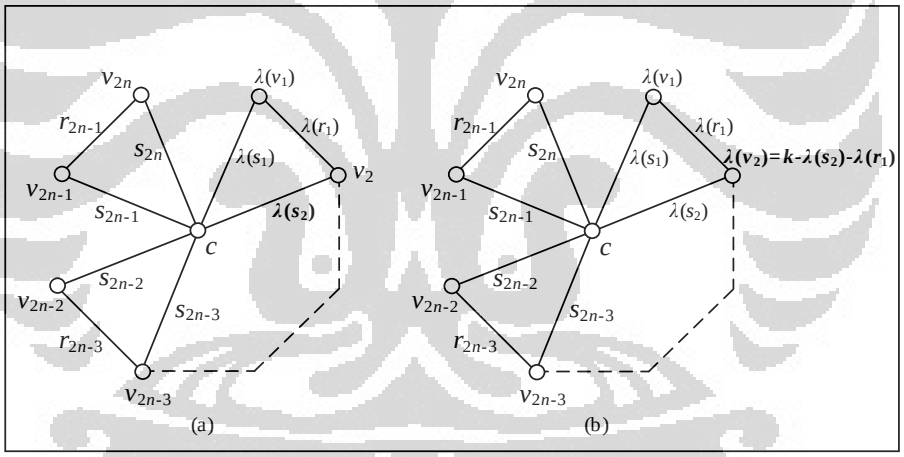


Gambar 3. 2 Proses pelabelan graf *friendship* F_n

Pada Gambar 3.2a terlihat bahwa simpul v_t (t ganjil) terhubung dengan busur s_t dan r_t, sedangkan simpul v_t (t genap) terhubung dengan busur s_t dan r_{t-1}. Misalkan pelabelan simpul dilakukan mengikuti urutan $v_1,v_2,\ldots,v_n$, baru kemudian simpul tengah c. Maka, untuk setiap tahap pelabelan simpul v_t (t ganjil) akan dilabel dua busur, yaitu s_t dan r_t. Sementara itu, untuk v_t (t genap) hanya perlu satu busur yaitu s_t. Busur r_{t-1} telah dilabel pada tahap pelabelan v_t (t ganjil). Untuk simpul tengah c, karena terhubung dengan semua jari-jari s maka label simpul c bisa diperoleh setelah semua jari-jari s dilabel.

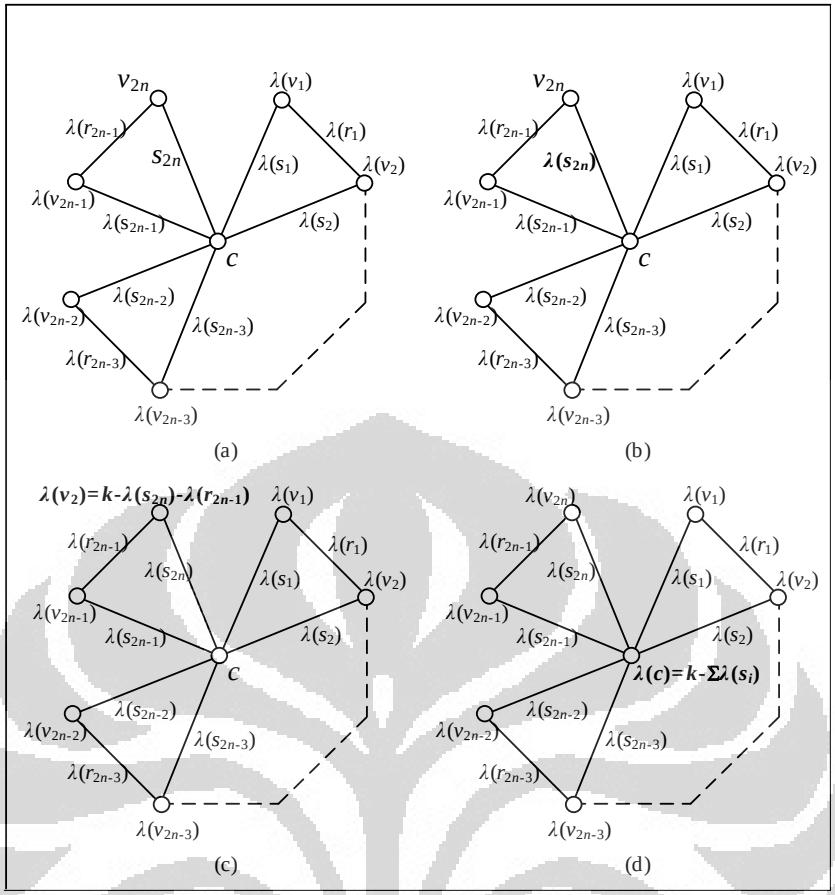
Pada tahap pelabelan simpul ganjil atau v_t (t ganjil) busur yang perlu dilabel adalah s_t dan r_t. Misalkan s_t dipilih untuk dilabel terlebih dahulu dengan $\lambda(s_t)$, baru kemudian r_t dengan $\lambda(r_t)$. Setelah kedua busur tersebut dilabel, maka

label v_t adalah $\lambda(v_t) = k - \lambda(s_t) - \lambda(r_t)$. Jika $\lambda(v_t)$ tidak tersedia, maka dilakukan penggantian label untuk r_t. Label untuk r_t yang telah digunakan akan kembali tersedia dan busur r_t dilabel dengan kemungkinan label lain yang tersedia. Proses ini dilakukan hingga didapatkan $\lambda(v_t)$. Jika untuk semua kemungkinan label untuk r_t tidak juga diperoleh $\lambda(v_t)$, maka akan dilakukan *backtracking*. Label r_t dan s_t yang telah digunakan sebelumnya kembali tersedia dan s_t dilabel dengan kemungkinan label lain yang tersedia. Jika semua kemungkinan label untuk s_t tidak menghasilkan $\lambda(v_t)$ yang tersedia, maka proses pelabelan kembali ke tahap $t - 1$ (tahap pelabelan simpul genap). Untuk kasus $t = 1$, proses pelabelan berhenti dan dapat disimpulkan bahwa tidak terdapat PTSA untuk nilai *n* dan *k* yang diberikan. Misalkan pada tahap ke-*t* ini $\lambda(v_t)$ berhasil diperoleh, maka proses pelabelan berlanjut ke tahap $t + 1$. Gambar 3.2b-3.2d menunjukkan tahap pelabelan simpul v_1, dimana diasumsikan $\lambda(v_1)$ yang dihasilkan tersedia.



Gambar 3. 3 Proses pelabelan pada graf *friendship* F_n

Langkah pelabelan simpul genap atau v_t (*t* genap) pada dasarnya sama dengan simpul ganjil. Bedanya, pada tahap pelabelan v_t (*t* genap) hanya satu busur yang perlu dilabel, yaitu s_t. Misalkan s_t dilabel dengan $\lambda(s_t)$. Label untuk v_t akan diperoleh dari $\lambda(v_t) = k - \lambda(s_t) - \lambda(r_{t-1})$. Proses pelabelan akan berlanjut jika $\lambda(v_t)$ yang dihasilkan tersedia. Namun jika terjadi sebaliknya, maka label simpul s_t akan diganti. Label s_t yang telah digunakan sebelumnya kembali tersedia dan s_t dilabel dengan kemungkinan label lain yang tersedia hingga diperoleh $\lambda(v_t)$ yang mungkin. Jika semua kemungkinan label untuk s_t tidak menghasilkan $\lambda(v_t)$ yang



Gambar 3. 4 Proses pelabelan graf *friendship* F_n

mungkin, maka dilakukan *backtracking*. Proses pelabelan kembali ke tahap $t-1$, label $r_{t\text{-}1}$ akan diganti. Sebelumnya, label $v_{t\text{-}1}$ dan $r_{t\text{-}1}$ yang telah digunakan kembali tersedia, lalu $r_{t\text{-}1}$ dilabel dengan kemungkinan label lain. Tahap pelabelan simpul v_2 dengan asumsi $\lambda(v_2)$ yang diperoleh tersedia bisa dilihat pada Gambar 3.3. Langkah selanjutnya sama seperti yang telah dijelaskan pada tahap pelabelan simpul ganjil. Pelabelan yang terbentuk setelah tahap $2n-1$ akan tempak seperti Gambar 3.4a.

Ketika *t* mencapai 2*n*, semua jari-jari telah dilabel (Gambar 3.4b). Karena itu, saat $t = 2n$ tidak hanya label v_{2n} (Gambar 3.4c) yang bisa diperoleh tetapi juga label simpul tengah *c* (Gambar 3.4d). Label simpul *c* diperoleh dari $\lambda(c) = k - \sum_{i=1}^{2n} \lambda(s_i)$. Jika $\lambda(c)$ yang diperoleh sesuai dengan label terakhir yang tersisa, maka satu PTSA berhasil dibentuk. PTSA yang terbentuk ini akan dicetak oleh fungsi print. Untuk mencari PTSA yang lain (jika ada) dilakukan *backtracking*. Label untuk simpul *c*, v_{2n}, dan busur s_{2n} yang telah digunakan sebelumnya kembali

tersedia. Lalu, s_{2n} kembali dilabel dengan kemungkinan label lain yang tersedia. Langkah selanjutnya adalah menentukan kembali label untuk v_{2n} dan c. Setelah semua kemungkinan label yang tersedia pada tahap ini dicoba untuk s_{2n}, maka proses pelabelan diulangi dari tahap $2n-1$. Label yang telah digunakan dari simpul c sampai v_{2n-1} dan r_{2n-1} kembali disediakan. Langkah yang dilakukan selanjutnya sama seperti yang dijelaskan pada tahap pelabelan simpul ganjil. Proses ini dilakukan sampai elemen graf yang pertama kali dilabel, yaitu s_1, selesai dilabel dengan semua kemungkinan label yang tersedia. Langkah *backtracking* ini juga dilakukan jika $\lambda(c)$ yang diperoleh tidak sesuai dengan satu label yang tersisa.

Jadi, secara umum proses pelabelan graf *friendship* dapat dibagi menjadi tiga fungsi. Fungsi pertama adalah fungsi untuk melabel simpul v_t dengan t ganjil. Dalam fungsi ini akan terdapat dua pengulangan (*loop*), karena ada dua busur yang perlu dilabel. Sedangkan fungsi kedua adalah fungsi untuk melabel simpul v_t dengan t genap, kecuali $t = 2n$. Dalam fungsi ini hanya akan terdapat satu pengulangan, karena hanya satu busur yang perlu dilabel. Dan fungsi yang ketiga adalah fungsi untuk menyelesaikan tahap akhir dari proses pembentukan PTSA graf *friendship*. Sama seperti fungsi kedua, pada fungsi ketiga ini terdapat satu pengulangan. Bedanya, pada fungsi ketiga ini tidak hanya label simpul v pada bagian rim yang ditentukan nilainya, tetapi juga label simpul tengah c.

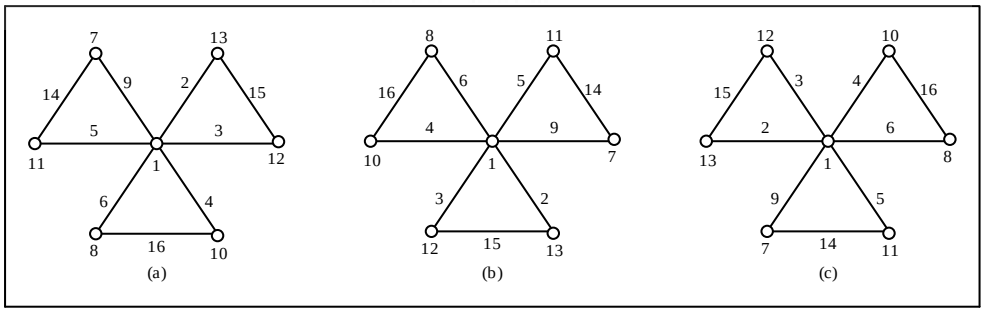
```
Algoritma 1(i).
function friendship1(t)
for each available label i do
    λ(s_t):= i
    avail [i]:= false
    for each available label j do
        λ(r_t):= j
        avail [j]:= false
        λ(v_t):= k - λ(s_t) - λ(r_t)
        if 0 < λ(v_t) ≤ 5n+1 and avail [λ(v_t)] then
            avail [λ(v_t)]:= false
            friendship2(t+1)
            avail [λ(v_t)]:= true
        end if
        avail [j]:= true
    end for
    avail [i]:= true
end for
```

Algoritma 1(ii).
function friendship2(*t*)
if $t = 2n$ **then**
  friendship3
else
  for each available label *i* **do**
    $\lambda(s_t) := i$
    avail [*i*]:= **false**
    $\lambda(v_t) := k - \lambda(s_t) - \lambda(r_{t-1})$
    if $0 < \lambda(v_t) \leq 5n+1$ **and** *avail* $[\lambda(v_t)]$
    then
      avail $[\lambda(v_t)]$:= **false**
      friendship1(*t*+1)
      avail $[\lambda(v_t)]$:= **true**
    end if
    avail [*i*]:= **true**
  end for
end if

Algoritma 1(iii).
function finalizefriendship
for each available label *i* **do**
  $\lambda(s_{2n}) := i$
  avail [*i*]:= **false**
  $\lambda(v_{2n}) := k - \lambda(s_t) - \lambda(r_{t-1})$
  if $0 < \lambda(v_{2n}) \leq 5n+1$ **and** *avail* $[\lambda(v_{2n})]$
  then
    avail $[\lambda(v_{2n})]$:= **false**
    $\lambda(c) := k - \sum_{i=1}^{2n} \lambda(s_i)$
    if $0 < \lambda(c) \leq 5n+1$ **and** *avail* $[\lambda(c)]$
    then
      print
    end if
    avail $[\lambda(v_{2n})]$:= **true**
  end if
  avail [*i*]:= **true**
end for

Pengulangan untuk melabel semua busur pada setiap fungsi dilakukan sebanyak label yang mungkin yang tersedia pada tahap tersebut. Ketiga fungsi yang membentuk algoritma PTSA untuk graf *friendship* dapat dilihat pada Algoritma 1(i), 1(ii), dan 1(iii). Perlu diperhatikan bahwa dalam algoritma ini fungsi friendship1 akan memanggil fungsi friendship2 dan sebaliknya (rekursif).

Seperti telah disebutkan sebelumnya, memeriksa apakah dua PTSA isomorfik atau tidak, bukanlah hal yang mudah. Karena itu, dibandingkan memeriksa seluruh PTSA yang telah didapatkan untuk mendapatkan PTSA yang tidak isomorfik, maka akan lebih mudah menambahkan syarat pada algoritma PTSA yang menjamin PTSA yang dihasilkan tidak isomorfik. Untuk mengetahui syarat apa yang perlu ditambahkan guna menghindari PTSA yang isomorfik, maka akan terlebih dahulu dibahas kasus isomorfik yang mungkin terjadi pada graf *friendship*.

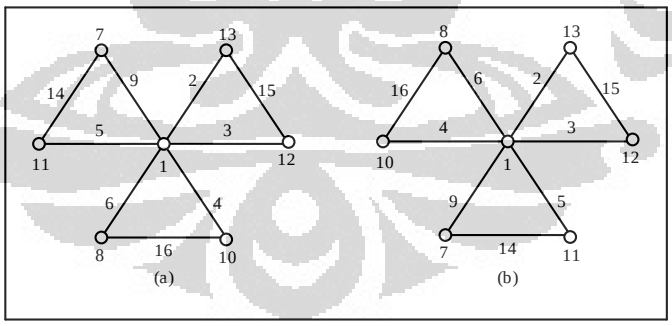


Gambar 3. 5 Kasus rotasi pada PTSA F_3

Pada Bab 2 telah disinggung bahwa kasus isomorfisma pelabelan bisa terjadi karena rotasi ataupun refleksi dari label simpul-simpul dan busur-busur yang berinsidensi. Kasus rotasi pada pelabelan graf *friendship* hanya terjadi pada simpul-simpul dibagian rim. Simpul tengah pada graf *friendship* tidak mungkin mengalami rotasi karena derajatnya selalu lebih besar dari simpul lainnya. Kasus rotasi pada PTSA graf *friendship* F_3 dapat dilihat pada Gambar 3.5.

Salah satu cara yang dapat dilakukan untuk menghindari PTSA hasil rotasi adalah dengan menetapkan segitiga mana pada graf *friendship* yang menjadi segitiga pertama. Misalkan segitiga yang mengandung s_1 dianggap sebagai segitiga pertama. Untuk membedakan segitiga ini dari segitiga pembentuk graf *friendship* yang lain, salah satu cara yang bisa dipilih adalah dengan melabel s_1 dengan label yang lebih kecil dibandingkan label s_t (t ganjil) yang lain (Gambar 3.5a). Jadi, syarat yang ditambahkan untuk menghindarkan kasus isomorfisma karena rotasi pada graf *friendship* adalah $\lambda(s_1) < \lambda(s_t)$ dengan t ganjil, $t > 1$. Karena syarat ini hanya berkaitan dengan t ganjil, maka syarat ini ditambahkan ke fungsi pertama.

Jika dicermati, kasus rotasi pada PTSA graf *friendship* bisa terjadi hanya pada sebagian segitiga pembentuknya. Gambar 3.6 menunjukkan contoh kasus ini pada PTSA graf F_3.



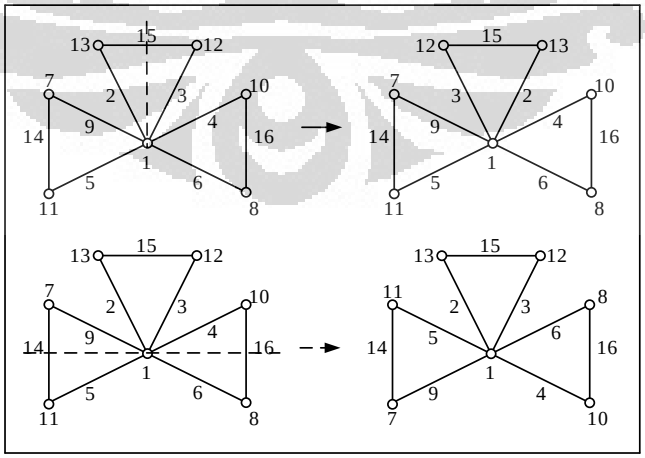
Gambar 3. 6 Contoh kasus isomorfisma sebagian pada F_3

PTSA karena kasus ini dapat dihindari dengan menetapkan urutan pada segitiga-segitiga pembentuk graf *friendship*. Misalnya, segitiga pertama adalah segitiga dengan label jari-jari s_t (t ganjil) terkecil. Segitiga kedua adalah segitiga dengan label s_t (t ganjil) terkecil berikutnya, dan seterusnya. Pada kasus rotasi sebelumnya, ditetapkan segitiga yang mengandung s_1 sebagai segitiga pertama,

dimana label s_1 lebih kecil dari label s_t (t ganjil) yang lain. Misalkan s_3 dipilih untuk mendapatkan label terkecil selanjutnya diantara s_t (t ganjil) yang lain, lalu s_5 dan seterusnya sampai s_{2n-1} mendapatkan label yang ter-besar. Maka, syarat yang perlu ditambahkan adalah $\lambda(s_1) < \lambda(s_3) < \ldots < \lambda(s_{2n-1})$. Syarat ini ditambahkan pada fungsi pertama karena hanya berkaitan dengan t ganjil (Algoritma 2(i) baris ke-16). Syarat ini otomatis mencakup syarat yang diberikan untuk kasus rotasi sebelumnya.

Algoritma 2(i).
function friendship1(*t*)

```
1.  if t = 1 then
2.    for each available label i do
3.      λ(s1):= i
4.      avail [i]:= false
5.      for each available label j do
6.        λ(r1):= j
7.        avail [j]:= false
8.        λ(v1):= k - λ(s1) - λ(r1)
9.        if 0 < λ(v1) ≤ 5n+1 and avail
          [λ(v1)] then
10.         avail [λ(v1)]:= false
11.         friendship2(2)
12.         avail [λ(v1)]:= true
          end if
13.       avail [j]:= true
        end for
14.     avail [i]:= true
      end for
15. else
16.   for each available label i> λ(st-2)
      do
17.     λ(st):= i
18.     avail [i]:= false
19.     for each available label j do
20.       λ(rt):= j
21.       avail [j]:= false
22.       λ(vt):= k - λ(st) - λ(rt)
23.       if 0 < λ(vt) ≤ 5n+1 and avail
          [λ(vt)] then
24.         avail [λ(vt)]:= false
25.         friendship2(t+1)
26.         avail [λ(vt)]:= true
          end if
27.       avail [j]:= true
        end for
28.     avail [i]:= true
    end for
  end if
```

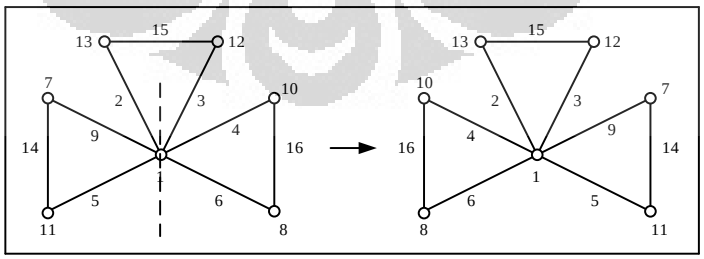


Gambar 3. 7 Contoh kasus refleksi jenis pertama pada PTSA F_3

Kasus refleksi pada pelabelan graf *friendship* dapat dibedakan menjadi dua jenis. Jenis yang pertama adalah refleksi yang terjadi di salah satu atau lebih segitiga pembentuk graf tersebut. Gambar 3.7 menunjukkan contoh kasus refleksi jenis pertama pada PTSA graf *friendship* F_3. Kasus refleksi jenis ini dapat dihindari dengan menetapkan jari-jari tertentu pada tiap segitiga mendapatkan label yang lebih kecil, apakah s_t dengan t genap atau t ganjil. Misalnya dipilih s_t dengan t ganjil pada tiap segitiga mendapatkan label lebih kecil dari s_t dengan t genap, maka $\lambda(s_1) < \lambda(s_2)$, $\lambda(s_3) < \lambda(s_4)$, dan seterusnya sampai $\lambda(s_{2n-1}) < \lambda(s_{2n})$. Dengan kata lain $\lambda(s_{t-1}) < \lambda(s_t)$ untuk t genap, $t > 1$. Syarat ini perlu ditambahkan pada fungsi kedua dan ketiga algoritma PTSA graf *friendship* (Algoritma 2(ii) baris ke-4 dan Algoritma 2(iii) baris ke-1).

Algoritma 2(i).
function friendship2(*t*)
1. **if** $t = 2n$ **then**
2. **friendship3**
3. **else**
4. **for** each available label $i > \lambda(s_{t-1})$ **do**
5. $\lambda(s_t) := i$
6. *avail* [*i*]:= **false**
7. $\lambda(v_t) := k - \lambda(s_t) - \lambda(r_{t-1})$
8. **if** $0 < \lambda(v_t) \leq 5n+1$ **and** *avail* $[\lambda(v_t)]$ **then**
9. *avail* $[\lambda(v_t)]$:= **false**
10. **friendship1(*t*+1)**
11. *avail* $[\lambda(v_t)]$:= **true**
end if
12. *avail* [*i*]:= **true**
end for
end if

Algoritma 2(iii).
function finalizefriendship
1. **for** each available label $i > \lambda(s_{t-1})$ **do**
2. $\lambda(s_{2n}) := i$
3. *avail* [*i*]:= **false**
4. $\lambda(v_{2n}) := k - \lambda(s_t) - \lambda(r_{t-1})$
5. **if** $0 < \lambda(v_{2n}) \leq 5n+1$ **and** *avail* $[\lambda(v_{2n})]$ **then**
6. *avail* $[\lambda(v_{2n})]$:= **false**
7. $\lambda(c) := k - \sum_{i=1}^{2n} \lambda(s_i)$
8. **if** $0 < \lambda(c) \leq 5n+1$ **and** *avail* $[\lambda(c)]$ **then**
9. **print**
end if
10. *avail* $[\lambda(v_{2n})]$:= **true**
end if
11. *avail* [*i*]:= **true**
end for

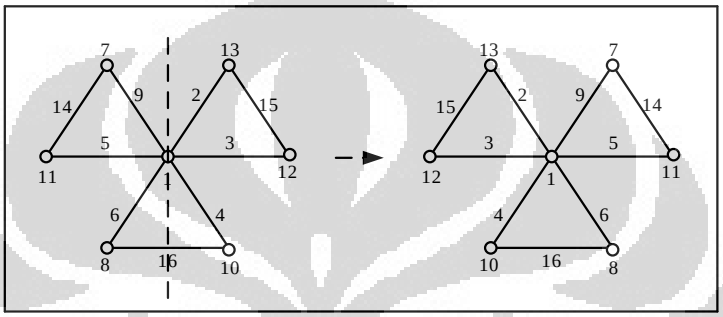


Gambar 3. 8 Contoh kasus refleksi jenis kedua pada PTSA F_3

Kasus refleksi jenis kedua adalah refleksi antar segitiga pembentuk graf *friendship*. Kasus refleksi jenis ini pada PTSA graf *friendship* F_3 dapat dilihat

pada Gambar 3.8. Syarat yang ditetapkan untuk kasus refleksi jenis pertama, yaitu $(s_{t-1}) < \lambda(s_t)$ untuk t genap dan $t > 1$, cukup untuk mencegah kasus ini terjadi.

Jika diperhatikan lebih lanjut, kasus refleksi yang mungkin terjadi pada PTSA graf *friendship* bisa saja merupakan gabungan dari kasus refleksi jenis pertama dan kedua. Gambar 3.9 menunjukkan contoh kasus ini pada PTSA F_3. Syarat yang diberikan untuk mencegah kasus refleksi jenis pertama dan kedua serta merta mencegah kasus ini terjadi.



Gambar 3. 9 Kasus refleksi gabungan refleksi jenis pertama dan kedua

Syarat $\lambda(s_t) > \lambda(s_{t-2})$ untuk t ganjil dan $\lambda(s_{t-1}) < \lambda(s_t)$ untuk t genap, dimana $t > 1$, akan menjamin PTSA yang dihasilkan dari algoritma yang dibentuk di atas tidak isomorfik. Selain itu, dengan adanya syarat ini maka waktu komputasi bisa dipangkas. Pada algoritma PTSA graf *friendship* ini, pengulangan yang dilakukan untuk melabel setiap busur adalah sebanyak label yang mungkin yang tersedia, yaitu sebanyak $5n + 1$ dikurangi label yang telah dipakai. Dengan adanya kedua syarat tersebut, maka pengulangan yang dilakukan bisa dikurangi.

Syarat pertama pada algoritma ini mengharuskan $\lambda(s_1) < \lambda(s_3) < \ldots < \lambda(s_{2n-1})$ dan syarat kedua mengharuskan $\lambda(s_1) < \lambda(s_2)$, $\lambda(s_3) < \lambda(s_4), \ldots, \lambda(s_{2n-1}) < \lambda(s_{2n})$, berarti $\lambda(s_1) < \lambda(s_2) < \lambda(s_3) < \ldots < \lambda(s_{2n})$, $\lambda(s_3) < \lambda(s_4) < \lambda(s_5) < \ldots < \lambda(s_{2n})$, dan seterusnya sampai $\lambda(s_{2n-1}) < \lambda(s_{2n})$. Sehingga, label yang mungkin untuk s_t dengan t ganjil adalah $\{1,2, \ldots, 3n + t + 1\}$. Contohnya untuk PTSA pada F_3, label yang mungkin untuk PTSA ini adalah $\{1,2,\ldots,16\}$. Label yang mungkin untuk s_1 adalah $\{1,2,\ldots,11\}$. Andaikan s_1 dilabel dengan 12, maka label yang mungkin yang lebih besar dari label s_1 ada sebanyak 4 label. Sementara itu, jari-jari yang perlu dilabel ada sebanyak 5, yaitu $s_2, s_3, \ldots, s_6$. Karena itu, s_1 tidak mungkin dilabel dengan label

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

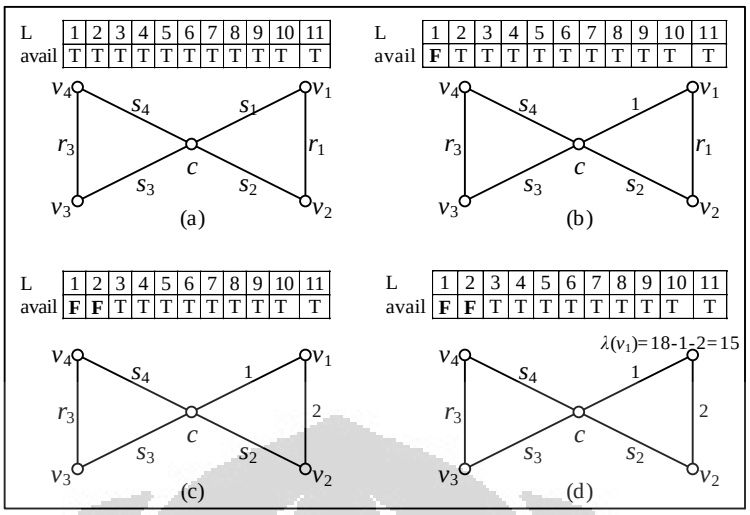
yang lebih besar dari 11. Hal ini juga mengurangi banyaknya komputasi yang perlu dilakukan untuk melabel s_1. Tanpa adanya syarat ini, pengulangan untuk melabel s_1 akan dilakukan sebanyak 16 kali. Sedangkan, dengan adanya syarat ini, pengulangan untuk melabel s_1 hanya perlu dilakukan sebanyak 11 kali.

Selain membatasi label yang mungkin untuk s_t dengan t ganjil, syarat tersebut juga membatasi label yang mungkin untuk s_t dengan t genap. Label yang mungkin untuk setiap s_t dengan t genap adalah label yang lebih besar dari label s_{t-1} sampai $5n + 1$. Menyambung contoh sebelumnya, yaitu untuk PTSA pada F_3, misalkan s_1 dilabel dengan label 11. Label yang mungkin untuk s_2 adalah $\{12,\ldots,16\}$, sehingga pengulangan yang perlu dilakukan untuk melabel s_2 hanya 5 kali.

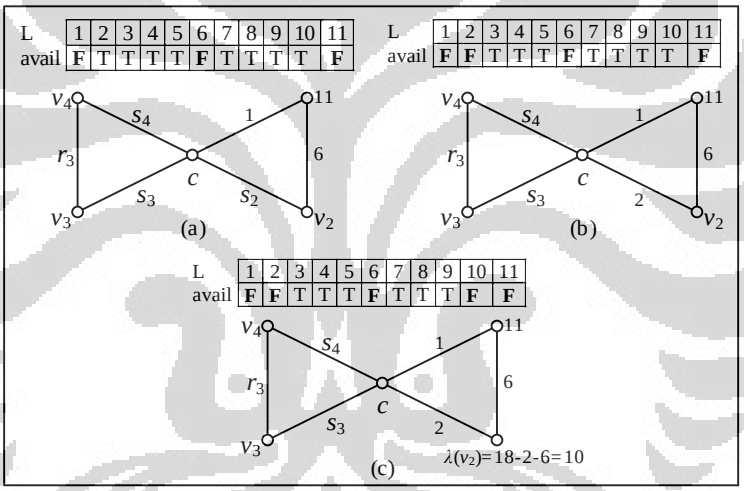
Selanjutnya, akan diberikan contoh penggunaan algoritma ini untuk membangun PTSA pada F_2. Seperti telah disebutkan sebelumnya, batas konstanta ajaib untuk graf ini adalah $18 \leq k \leq 21$. Misalkan nilai k yang dipilih adalah 18. Berarti masukan yang diberikan adalah $n = 2$ dan $k = 18$. Label yang mungkin untuk membangun PTSA untuk graf F_2 adalah $L=\{1,2,\ldots,11\}$. Pada dasarnya, pemilihan label yang digunakan tidak perlu berurutan. Tetapi, untuk memudahkan, maka label yang dipilih untuk melabel busur adalah label terkecil yang mungkin yang tersedia pada tahap tersebut. Jika label busur akan diganti, maka label yang dipilih adalah label terkecil berikutnya yang tersedia. Label i yang telah digunakan tidak dapat digunakan untuk melabel simpul dan busur yang tersisa. Pada algoritma, hal ini ditandai dengan *avail*[i]:=false. Sementara itu, *avail*[i]:=true menandakan bahwa label i kembali tersedia dalam L dan dapat digunakan kembali

Proses pelabelan dimulai dengan memanggil fungsi friendship1 (Algoritma 2(i)) dengan parameter $t = 1$. Parameter t ini menunjukkan busur dan simpul mana yang akan dilabel. Tahap pertama diawali dengan melabel s_1 dengan 1 (Gambar 3.10b). Selanjutnya busur r_1 dilabel dengan label yang mungkin yang tersedia $\{2,3,\ldots,11\}$. Sesuai dengan ketentuan yang telah disebutkan sebelumnya, maka r_1 akan dilabel dengan 2 (Gambar 3.10c). Setelah s_1 dan r_1 dilabel, maka diperoleh $\lambda(v_1) = 18 - 1 - 2 = 15$ (Gambar 3.10d).

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010



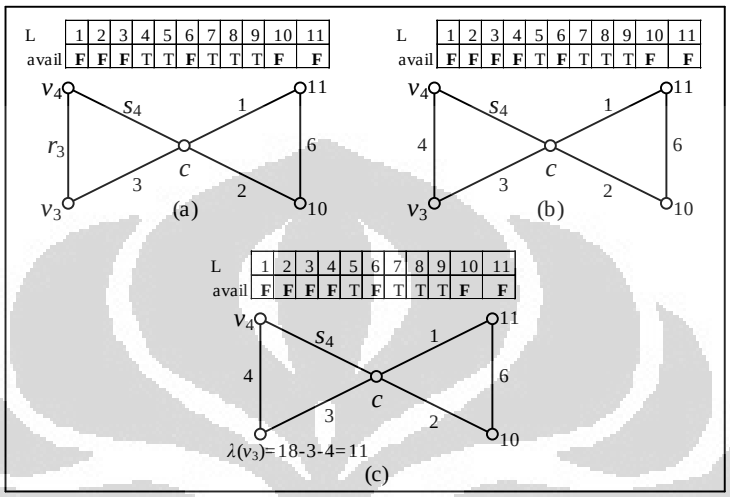
Gambar 3. 10 Contoh Penerapan Algoritma PTSA graf *friendship* F_2 dengan n = 18 (Tahap pelabelan simpul v_1)



Gambar 3. 11 Contoh Penerapan Algoritma PTSA graf *friendship* F_2 dengan n = 18 (Tahap pelabelan simpul v_2)

Karena label 15 tidak tersedia, maka v_1 tidak dapat dilabel dengan 15. Untuk mendapatkan label v_1 yang lain, maka $\lambda(r_1)$ akan diganti. Label r_1 sebelumnya, $\lambda(r_1) = 2$, kembali tersedia (baris ke-13) dan proses pelabelan kembali ke baris 5, yaitu r_1 dilabel dengan label lain yang tersedia. Melabel r_1 dengan 3, 4, atau 5 akan berakhir sama seperti melabel r_1 dengan 2 (label v_1 tidak tersedia). Maka selanjutnya, r_1 dilabel dengan 6. Label v_1 yang diperoleh adalah $\lambda(v_1) = 18 – 1 – 6 = 11$ (Gambar 3.11a). Setelah $\lambda(v_1)$ berhasil diperoleh, proses pelabelan berlanjut ke tahap pelabelan v_2 dengan memanggil fungsi friendship2 (Algoritma 2(ii)) dengan t = 2 seperti ditunjukkan pada baris ke-12.

Label yang masih tersedia pada tahap ini adalah {2,...,5,7,…,10}. Label yang mungkin untuk s_2 adalah yang lebih besar dari label s_1 (baris ke-4), sesuai syarat untuk mencegah kasus refleksi. Maka s_2 akan dilabel dengan 2 (Gambar 3.11b). Dengan demikian diperoleh $\lambda(v_2) = 18 - 2 - 6 = 10$ (Gambar 3.11c).

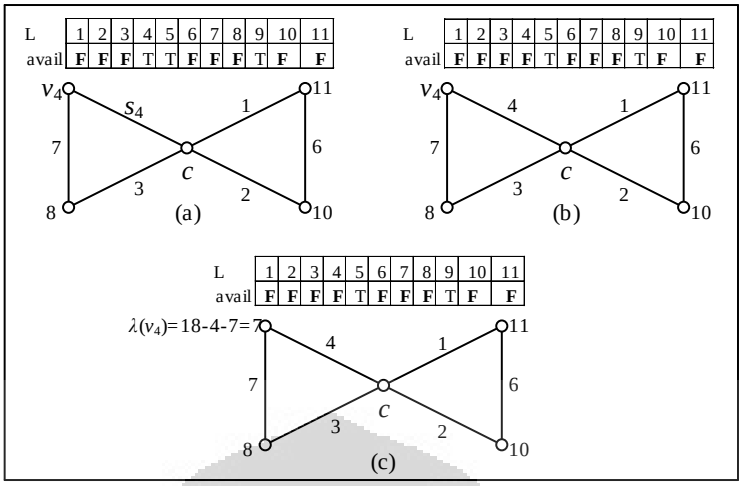


Gambar 3. 12 Contoh Penerapan Algoritma PTSA graf *friendship* F_2 dengan $n = 18$ (Tahap pelabelan simpul v_3)

Selanjutnya dipanggil fungsi friendship1 dengan $t = 3$ (baris ke-10) untuk tahap pelabelan v_3. Label yang tersedia pada tahap ini adalah {3,4,5,7,8,9}. Sama seperti tahap pelabelan v_1, ada dua busur yang perlu dilabel. Busur yang pertama adalah s_3. Untuk mencegah kasus rotasi dan refleksi jenis kedua, $\lambda(s_3) > \lambda(s_1)$, seperti ditunjukkan pada baris ke-16. Karena semua label yang tersisa pada tahap ini lebih besar dari $\lambda(s_1)$, berarti semua label tersebut mungkin untuk s_3. Maka, s_3 akan dilabel dengan label terkecil yang tersedia, yaitu 3 (Gambar 3.12a). Lalu, r_3 akan dilabel dengan 4 (Gambar 3.12b). Akibatnya, diperoleh $\lambda(v_3) = 18 - 3 - 4 = 11$ (Gambar 3.12c).

Karena label 11 tidak tersedia, maka v_3 tidak dapat dilabel dengan 11. Label r_3 akan diganti hingga diperoleh label yang mungkin untuk v_3. Langkah yang dilakukan sama seperti pada tahap pelabelan v_1 saat mengganti label r_1. Busur r_3 akan dilabel dengan 7, sehingga label untuk v_3 adalah $\lambda(v_3) = 18 - 3 - 7 = 8$ (Gambar 3.13a). Proses pelabelan berlanjut dengan memanggil fungsi friendship2 dengan $t = 4$ (baris ke-25).

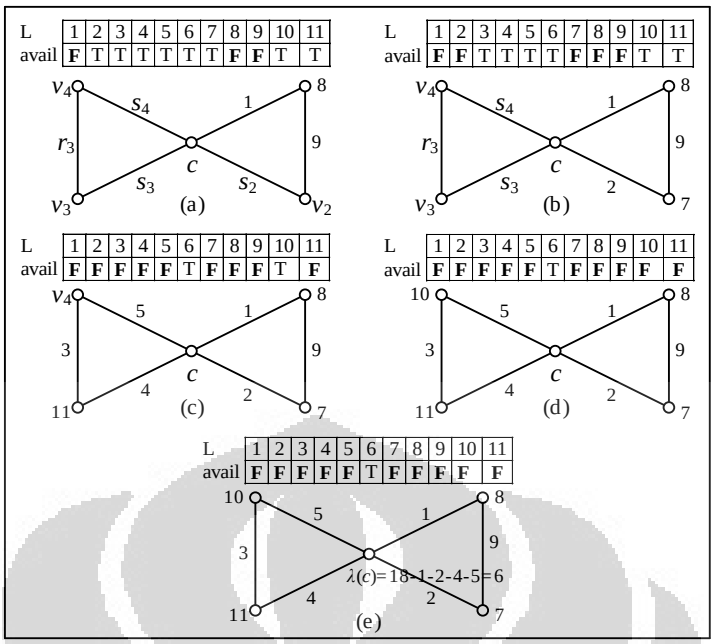
Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010



Gambar 3. 13 Contoh Penerapan Algoritma PTSA graf *friendship* F_2 dengan n = 18 (Tahap pelabelan simpul v_4)

Karena $t = 4 = 2n$, maka fungsi yang dipanggil adalah fungsi finalize-friendship (Algoritma 2(iii)). Label yang tersisa pada tahap ini adalah {4,5,9}. Ketiga label yang tersisa ini lebih besar dari $\lambda(s_3)$. Artinya, ketiga label ini mungkin untuk melabel s_4. Maka, busur s_4 akan dilabel dengan 4 (Gambar 3.13b). Label untuk v_4 diperoleh dari $\lambda(v_4) = 18 - 4 - 7 = 7$ (Gambar 3.13c). Simpul v_4 tidak dapat dilabel dengan 7 karena label ini tidak tersedia. Untuk mendapatkan label yang lain untuk v_4, $\lambda(s_4)$ akan diganti dengan label lain yang tersedia. Label s_4 yang telah digunakan sebelumnya, yaitu $\lambda(s_4) = 4$, kembali tersedia (baris ke-11) dan proses pelabelan diulangi dari baris pertama (s_4 dilabel dengan label lain yang mungkin).

Label yang mungkin untuk menggantinya adalah 5 dan 9. Namun, karena kedua label ini tidak menghasilkan $\lambda(v_4)$ yang mungkin dan tidak ada lagi label lain yang tersedia untuk s_4 pada tahap ini, maka proses pelabelan kembali ke tahap dimana fungsi ini dipanggil. Fungsi ini dipanggil pada tahap ketiga, dalam algoritma ditunjukkan pada baris ke-25. Selanjutnya $\lambda(v_3) = 8$ dan $\lambda(r_3) = 7$ kembali tersedia (baris ke-26 dan 27) dan proses pelabelan kembali ke baris 19, r_3 dilabel dengan label lain yang mungkin. Label yang tersedia sekarang adalah {4,5,7,8,9}. Sebelumnya r_3 telah dilabel dengan 7, maka sekarang r_3 akan dilabel dengan 8. Langkah yang dilakukan setelah ini sama dengan langkah yang dilakukan sebelumnya pada tahap pelabelan v_3.



Gambar 3. 14 Contoh Penerapan Algoritma PTSA graf *friendship* F_2 dengan n = 18 (*Backtracking* dan tahap pelabelan simpul v_4 dan c)

Melabel r_3 dengan 8 atau 9 akan berakhir sama seperti ketika melabel r_3 dengan 7. Karena itu, *backtracking* kembali dilakukan. Label s_3 yang telah digunakan sebelumnya kembali tersedia (baris ke-28) dan $\lambda(s_3)$ diganti dengan label lain yang tersedia. Proses pelabelan kembali dilakukan sama seperti sebelumnya. Untuk contoh ini, *backtracking* dilakukan sampai r_1. Karena mengganti $\lambda(s_3)$ dengan label lain yang tersedia pada tahap itu tidak menghasilkan $\lambda(v_4)$ yang mungkin. Begitu juga ketika proses berlanjut dengan mengganti $\lambda(s_2)$ dengan label lain yang tersedia pada tahap tersebut.

Label yang tersedia untuk melabel r_1 adalah {2,…,11}. Terakhir r_1 telah dilabel dengan 6, maka label penggantinya sekarang adalah 7. Namun, melabel r_1 dengan 7 atau 8 akan berujung sama dengan melabel r_1 dengan 6. Pada akhirnya tidak dapat diperoleh $\lambda(v_4)$ yang mungkin. Label selanjutnya yang mungkin untuk r_1 adalah 9. Melabel r_1 dengan 9 akan menghasilkan $\lambda(v_1) = 18 - 1 - 9 = 8$ (Gambar 3.14a). Setelah $\lambda(v_1)$ diperoleh, proses pelabelan berlanjut pada tahap pelabelan v_2. Label yang tersedia pada tahap ini adalah {2,..,7,10,11}. Busur s_2 akan dilabel dengan 2, sehingga diperoleh $\lambda(v_2) = 18 - 2 - 9 = 7$ (Gambar 3.14b). Langkah selanjutnya dilakukan sama seperti yang telah dijelaskan sebelumnya.

Pada tahap terakhir label yang tersisa adalah {5,6,10}. Busur s_4 akan dilabel dengan 5 (Gambar 3.14c). Akibatnya, label untuk simpul v_4 adalah $\lambda(v_4) = 18 - 5 - 3 = 10$ (Gambar 3.14d). Karena $\lambda(v_4)$ yang diperoleh sesuai dengan label yang tersedia, proses berlanjut dengan mencari label simpul c, yaitu $\lambda(c) = 18 - (1 + 2 + 4 + 5) = 6$ (Gambar 3.14e). Label terakhir yang tersisa sesuai dengan $\lambda(c)$ yang diperoleh, berarti satu PTSA selesai dibentuk. PTSA ini akan dicetak oleh fungsi print (baris ke-9).

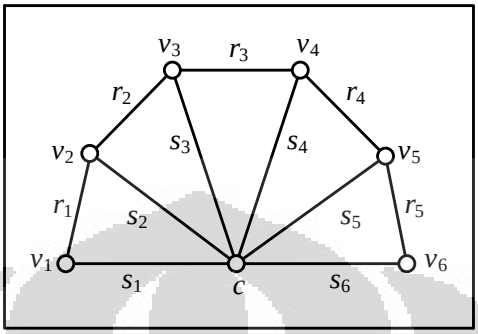
Setelah terbentuk satu PTSA, maka dilakukan *backtracking* untuk mendapatkan PTSA yang lain. Label v_4 dan s_4 yang telah digunakan sebelumnya kembali tersedia (baris ke-10 dan 11). Label s_4 diganti dengan label lain yang mungkin dan proses pelabelan dilanjutkan seperti telah dijelaskan sebelumnya. Setelah semua kemungkinan label yang ada pada tahap ini dicoba untuk melabel s_4, *backtracking* kembali dilakukan. Label v_4 sampai r_3 yang telah digunakan sebelumnya kembali tersedia. Lalu, busur r_3 dilabel dengan kemungkinan label lain yang ada pada tahap tersebut dan proses pelabelan dilanjutkan seperti biasa. Proses ini akan dilakukan sampai s_1 selesai dilabel dengan semua kemungkinan label yang ada untuk membentuk PTSA graf F_2. Untuk graf F_2 dengan $k = 18$, PTSA yang tidak isomorfik yang terbentuk hanya satu, yaitu seperti yang terlihat pada Gambar 3.14e.

3.2 Algoritma PTSA Graf Kipas

Pada Subbab 3.1 telah dibahas mengenai algoritma PTSA untuk graf *friendship*, yaitu suatu graf yang dapat dibentuk dengan menghapus $n/2$ busur pada bagian rim graf roda secara berselang-seling. Sekarang akan dibahas mengenai algoritma PTSA untuk graf kipas. Sama halnya dengan graf *friendship*, graf kipas juga dapat diperoleh dengan menghapus busur pada bagian rim graf roda. Beda-nya, busur yang dihapus hanya salah satu busur. Seperti telah dibahas pada Bab 2, graf kipas f_n dapat juga dipandang sebagai gabungan graf lintasan P_n dengan satu simpul tengah c, dimana setiap simpul pada graf P_n dihubungkan dengan simpul c. Jadi, banyaknya simpul dari graf kipas f_n adalah n simpul dari graf P_n dan satu simpul tengah c atau $|V| = n + 1$. Sementara itu, banyaknya busur adalah $n - 1$ busur dari graf P_n ditambah n busur yang menghubungkan setiap

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

simpul pada P_n dengan simpul *c*, atau $|E| = 2n - 1$. Jika $n = 2$, graf kipas akan sama dengan graf lingkaran C_3. Dalam skripsi ini, graf kipas yang dibahas hanyalah graf kipas dengan ukuran $n > 2$.



Gambar 3. 15 Graf Kipas f_6

Seperti pada graf *friendship*, simpul dan busur pada bagian luar graf kipas disebut rim. Busur pada bagian rim dinotasikan dengan *r* dan simpul pada bagian tersebut dinotasikan dengan *v*. Sementara itu, busur yang menghubungkan simpul *v* dengan *c* biasa disebut jari-jari (*spoke*) dan dinotasikan dengan *s*. Gambar 3.15 menunjukkan graf kipas dengan ukuran 6.

Pada Bab 2 telah diperoleh rumus umum untuk mendapatkan batas konstanta ajaib *k* yang mungkin untuk suatu graf berdasarkan perhitungan dasar. Batas *k* untuk membentuk PTSA graf kipas f_n bisa diperoleh dengan mensubstitusi nilai $|V|$ dan $|E|$ graf kipas ke rumus umum tersebut, yaitu (2)

$$\frac{|E|(|E|+1)}{2} + \frac{(|E|+|V|)(|E|+|V|+1)}{2} \leq |V|k \leq \frac{|E|(|E|+2|V|+1)}{2} + \frac{(|E|+|V|)(|E|+|V|+1)}{2}.$$

Maka diperoleh,

$$\frac{(2n-1)2n}{2} + \frac{3n(3n+1)}{2} \leq (n+1)k \leq \frac{(2n-1)(4n+2)}{2} + \frac{3n(3n+1)}{2}$$

atau,

$$\frac{13n^2+n}{2(n+1)} \leq k \leq \frac{17n^2+3n-2}{2(n+1)}. \qquad (3.4)$$

Sama halnya dengan graf *friendship* batas *k* untuk graf kipas juga bisa didapatkan dengan perhitungan yang didasarkan pada struktur grafnya. Pada graf kipas derajat dari simpul tengah *c* adalah $d(c) = n$. Sementara itu, derajat simpul awal v_1 dan simpul akhir v_n adalah 2, dan simpul *v* lainnya adalah 3. Jadi, $d(c)$ tidak pernah lebih kecil dari $d(v_i)$. Jika simpul *c* dan busur yang hadir pada simpul

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

tersebut dilabel dengan $n + 1$ label terkecil, maka akan didapatkan bobot terendah yang mungkin untuk simpul *c*, *w*(*c*). Batas bawah konstanta ajaib *k* tidak mungkin lebih kecil dari *w*(*c*) ini. Secara matematis,

$$k \geq w(c) = \sum_{i=1}^{n+1} i = \frac{(n+1)(n+2)}{2}. \qquad (3.5)$$

Sekarang perhatikan jumlahan dari bobot *n* simpul pada bagian rim. Label dari busur pada bagian rim berkontribusi dua kali pada bobot simpul, yaitu kedua simpul ujungnya. Karena itu, nilai maksimal dari jumlahan tersebut bisa didapatkan jika busur pada bagian rim dilabel dengan $n - 1$ label terbesar dan simpul pada bagian rim serta jari-jari dengan 2*n* label terbesar berikutnya.

$$\begin{aligned} w(v_1) + \cdots + w(v_n) &\leq 2\sum_{i=2n+2}^{3n} i + \sum_{i=2}^{2n+1} i \\ &= \sum_{i=2}^{3n} i + \sum_{i=2n+2}^{3n} i \\ &= \frac{(3n-1)(3n+2)}{2} + \frac{(n-1)(5n+2)}{2} \\ &= 7n^2 - 2. \end{aligned}$$

Batas *k* terbesar yang mungkin untuk PTSA graf kipas tidak mungkin melebihi bobot maksimal untuk simpul pada bagian rim. Karena simpul pada bagian rim ada sebanyak *n*, maka

$$nk \leq 7n^2 - 2$$

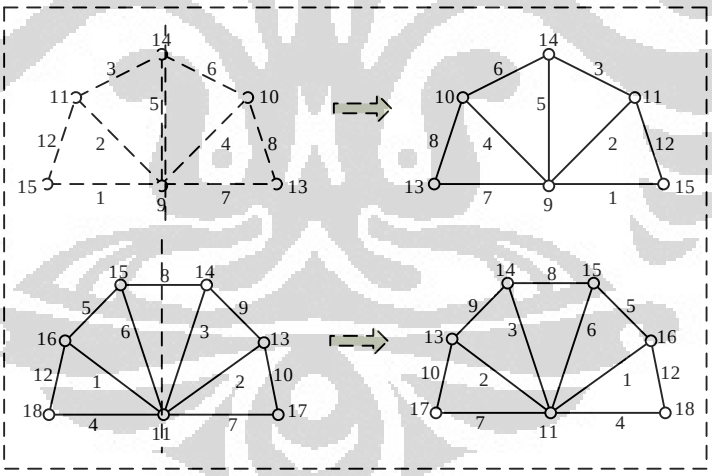
atau,

$$k \leq 7n - \frac{2}{n}. \qquad (3.6)$$

Saat $n = 11$, batas atas *k* yang didapat dari (3.6) lebih kecil dari batas bawahnya yang didapat dari (3.5). Artinya, tidak terdapat PTSA yang mungkin untuk graf kipas f_n dengan $n > 10$. Batas bawah nilai *k* yang akan digunakan dalam membagun PTSA graf kipas adalah nilai maksimal antara batas bawah *k* yang didapatkan dari (3.4) dan (3.5). Sementara itu, batas atas *k* yang digunakan adalah nilai minimal antara batas atas dari (3.4) dan (3.6). Jika nilai batas *k* yang didapatkan berbentuk pecahan, maka berlaku aturan pembulatan seperti yang telah dijelaskan pada Bab 2 dan Subbab 3.1. Contohnya untuk $n = 4$, batas *k* yang diperoleh dari (3.4) adalah $21.2 \leq k \leq 28.2$ dibulatkan menjadi $22 \leq k \leq 28$, sedangkan dari (3.5) dan (3.6) adalah $15 \leq k \leq 27.5$ dibulatkan menjadi $15 \leq k \leq 27$. Jadi, batas *k* yang mungkin untuk membangun PTSA graf f_4 adalah $22 \leq k \leq 27$. Selanjutnya akan dibahas mengenai algoritma PTSA untuk graf ini.

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

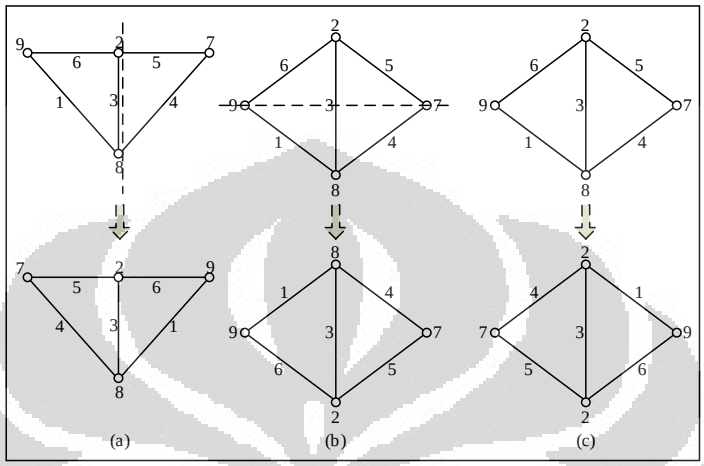
Pada dasarnya algoritma PTSA untuk graf kipas tidak jauh berbeda dengan algoritma pada graf *friendship*. Dalam membangun PTSA untuk graf kipas dengan ukuran n dan nilai k tertentu, busur dan simpul pada graf ini akan dilabel secara iteratif. Label yang tersedia untuk membangun PTSA graf f_n adalah $L = \{1,2,\ldots,3n\}$. Untuk menentukan label dari simpul v dan c, akan terlebih dulu dilabel busur yang hadir pada simpul v dan c tersebut. Label busur dipilih sesuai dengan label yang tersedia. Sementara label simpul diperoleh dari pengurangan nilai konstanta ajaib k dengan jumlah dari label busur yang hadir pada simpul tersebut. Karena pelabelan adalah pemetaan satu-satu, maka label yang telah digunakan tidak dapat digunakan kembali. Sebelum membahas mengenai algoritma PTSA untuk graf kipas lebih lanjut, terlebih dahulu dibahas mengenai kasus isomorfisma yang mungkin terjadi pada PTSA graf kipas. Hal ini penting untuk menentukan syarat apa yang perlu diberikan untuk mencegah terbentuknya PTSA yang isomorfik. Syarat ini merupakan salah satu perbedaan antara algoritma PTSA untuk graf kipas dengan graf *friendship*.



Gambar 3. 16 Kasus refleksi pada PTSA graf f_5 dan f_6

Telah disebutkan sebelumnya bahwa simpul awal dan simpul akhir dari graf kipas memiliki derajat yang sama, yaitu $d(v_1) = d(v_n) = 2$. Simpul tengah c berderajat n dan simpul lainnya berderajat 3. Karena itu, untuk $n > 3$, isomorfisma pelabelan pada graf ini tidak mungkin terjadi karena kasus rotasi. PTSA yang isomorfik mungkin terbentuk karena refleksi. Contoh kasus refleksi pada

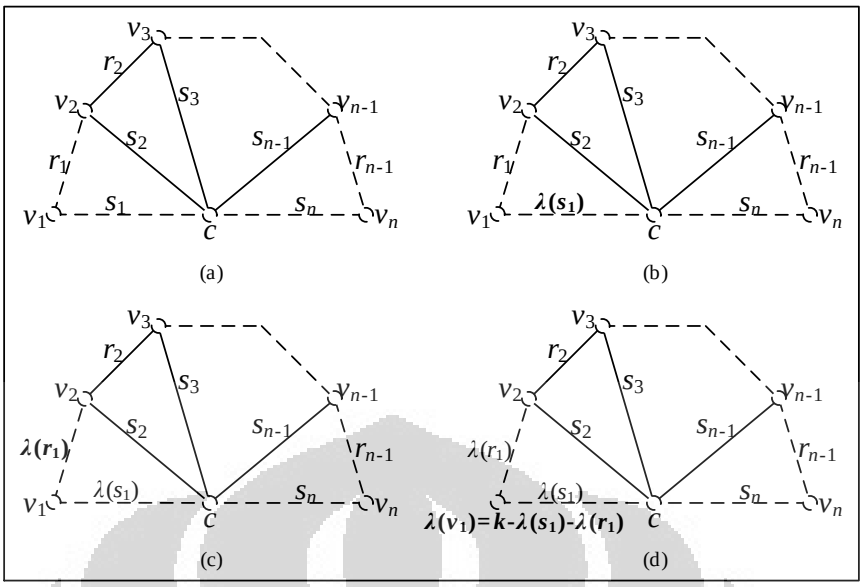
PTSA graf kipas f_5 dan f_6 dapat dilihat pada Gambar 3.14 (garis putus-putus menunjukkan sumbu simetri untuk refleksi yang terjadi). Untuk mencegah kasus refleksi ini, salah satu cara yang dapat dipilih adalah dengan menetapkan bahwa label s_n lebih besar dari label s_1 untuk setiap $n > 1$, atau $\lambda(s_n) > \lambda(s_1)$ dengan $n > 1$.



Gambar 3. 17 Kasus refleksi pada PTSA f_3 (a) dan (b) Kasus rotasi pada PTSA f_3 (c)

Perlu diperhatikan bahwa pada f_3 tidak hanya terjadi satu kasus refleksi seperti pada graf kipas lainnya. Pada graf f_3 $d(v_2) = d(c)$, karena itu mungkin terjadi refleksi dengan sumbu simetrinya adalah garis yang melalui simpul v_1 dan v_3. Karena alasan ini jugalah, rotasi mungkin terjadi pada PTSA f_3. Gambar 3.15 menunjukkan kasus refleksi dan rotasi yang terjadi pada PTSA f_3. Syarat $\lambda(s_n) > \lambda(s_1)$ tidak cukup untuk mencegah kasus refleksi dan rotasi yang terjadi pada f_3 (Gambar 3.13b dan c). Perlu ada syarat tambahan (khusus untuk pembentukan PTSA f_3) untuk mencegah terbentuknya PTSA yang isomorfik karena kasus ini. Syarat yang dapat dipilih salah satunya adalah $\lambda(c) > \lambda(v_2)$.

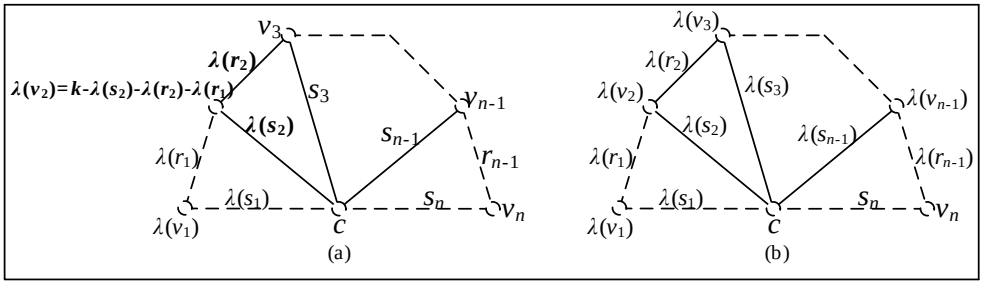
Jika dicermati, untuk graf f_n dengan $n > 3$, dari setiap PTSA yang terbentuk akan terdapat satu PTSA lain yang isomorfik karena hasil refleksi. Artinya, tanpa menggunakan syaratpun dapat diketahui banyaknya PTSA yang tidak isomorfik yang terbentuk. Yaitu, dengan mengeliminasi setengah dari PTSA yang terbentuk. Namun, seperti telah disinggung pada subbab 3.1, syarat yang mencegah terbentuknya PTSA yang isomorfik ini juga berguna untuk memangkas waktu komputasi yang diperlukan untuk membangun seluruh PTSA.



Gambar 3. 18 Proses pelabelan graf kipas f_n

Untuk mengetahui dimana syarat yang telah ditentukan tadi digunakan dalam algoritma, akan terlebih dahulu dijelaskan mengenai pembentukan algoritma PTSA graf kipas. Pada Gambar 3.18.a terlihat bahwa simpul v_1 terhubung dengan s_1 dan r_1. Simpul v_t (t = 2,..,n – 1) terhubung dengan s_t, r_{t-1}, dan r_t. Simpul v_n terhubung dengan s_n dan r_n. Dan, simpul tengah c terhubung dengan seluruh jari-jari s. Misalkan pelabelan simpul dilakukan mengikuti urutan v_1,v_2,…,v_n, dan c. Artinya, untuk setiap tahap pelabelan v_t (t = 1,…,n – 1) akan didahului dengan pelabelan dua busur, yaitu s_t dan r_t. Busur r_{t-1} yang hadir pada simpul v_t (t = 2,…,n – 1) telah dilabel pada tahap t-1, karenanya pada tahap pelabelan v_t (t = 2,…,n – 1) hanya dua busur yang perlu dilabel. Sementara itu, pada tahap pelabelan v_n hanya ada satu busur yang perlu dilabel, yaitu s_n. Karena pada tahap pelabelan v_n ini seluruh jari-jari s selesai dilabel, maka label untuk simpul c bisa ditentukan pada tahap ini.

Pada tahap pelabelan v_t (t = 1,..,n – 1) akan dilakukan dua kali pelabelan busur, yaitu r_t dan s_t. Misalkan s_t dilabel terlebih dahulu dengan $\lambda(s_t)$, baru kemudian r_t dengan $\lambda(r_t)$. Setelah kedua busur ini dilabel, maka label v_t bisa diperoleh dari $\lambda(v_t) = k - \lambda(s_t) - \lambda(r_t) - \lambda(r_{t-1})$. Khusus untuk t = 1, label v_1 diperoleh dari $\lambda(v_1) = k - \lambda(s_1) - \lambda(r_1)$. Tahap pelabelan untuk simpul v_1 dan v_2 dengan asumsi $\lambda(v_1)$ dan $\lambda(v_2)$ yang diperoleh tersedia bisa dilihat pada Gambar 3.18 dan 3.19a. Sementara akhir dari tahap pelabelan v_{n-1} akan tampak seperti Gambar 3.19b.



Gambar 3. 19 Proses pelabelan graf kipas f_n

Jika label yang diperoleh untuk v_t tidak tersedia, maka $\lambda(r_t)$ akan diganti. Label r_t yang telah digunakan sebelumnya tersedia kembali dalam L, lalu $\lambda(r_t)$ diganti dengan label lain yang tersedia pada tahap tersebut. Proses ini dilakukan hingga diperoleh $\lambda(v_t)$ yang sesuai dengan label yang tersedia. Setelah diperoleh $\lambda(v_t)$, proses pela-belan dilanjutkan ke tahap $t + 1$. Namun, jika semua label yang tersedia untuk r_t tidak menghasilkan $\lambda(v_t)$ yang sesuai, maka dilakukan *backtracking*. Label r_t dan s_t yang telah digunakan sebelumnya tersedia kembali dalam L. Kemudian $\lambda(s_t)$ diganti dengan label lain yang tersedia pada tahap tersebut. Proses yang dilakukan setelah ini sama dengan proses pelabelan yang telah dijelaskan. Jika semua label untuk s_t pada tahap ini tidak menghasilkan $\lambda(v_t)$ yang sesuai dengan label yang tersedia, maka *backtracking* kembali dilakukan. Pelabelan kembali ke tahap $t - 1$, $\lambda(r_{t-1})$ akan diganti dengan label lain yang tersedia. Langkah yang dilakukan selan-jutnya sama seperti langkah penggantian $\lambda(r_t)$ yang telah dijelaskan. Untuk kasus $t = 1$, jika semua label yang tersedia untuk s_1 tidak menghasilkan $\lambda(v_1)$ yang mung-kin, maka dapat ditarik kesimpulan bahwa tidak terdapat PTSA yang mungkin untuk nilai n dan k yang diberikan.

Berbeda dengan tahap pelabelan v_t ($t = 1,\ldots,n - 1$), pada tahap pelabelan simpul akhir hanya akan dilakukan satu pelabelan busur, yaitu s_n. Misalnya s_n dilabel dengan $\lambda(s_n)$, dimana $\lambda(s_n) > \lambda(s_1)$ untuk mencegah kasus refleksi (Gambar 3.20a). Label untuk v_n bisa diperoleh dari $\lambda(v_n) = k - \lambda(s_n) - \lambda(r_{n-1})$ (Gambar 3.20b). Jika $\lambda(v_n)$ sesuai dengan label yang tersedia, maka proses pelabelan dilanjutkan dengan menentukan label untuk simpul c (Gambar 3.20c). Label simpul c ini diperoleh dari $\lambda(c) = k - \Sigma\lambda(s_i)$. Jika λ(c) sesuai dengan satu label yang tersisa, artinya satu PTSA berhasil terbentuk. PTSA yang terbentuk ini akan dicetak oleh

fungsi print. Namun, bila tidak demikian, maka akan dilakukan *backtracking*. Busur terakhir yang dilabel, s_n, akan diganti labelnya dengan label lain yang tersedia pada tahap ini. Label v_n dan s_n yang telah digunakan sebelumnya tersedia kembali. Kemudian s_n dilabel dengan label lainnya dan proses pelabelan dilanjutkan seperti biasa. Jika semua label untuk sn pada tahap ini tidak mengha-silkan v_n atau c yang sesuai dengan label yang tersisa, maka *backtracking* kembali dilakukan. Proses pelabelan kembali ke tahap $n - 1$. Label busur r_{n-1}akan diganti. Langkah yang dilakukan sama seperti yang dijelaskan pada tahap pelabelan v_t ($t = 1,\ldots,n - 1$). Langkah *backtracking* ini juga dilakukan untuk mencari PTSA yang lain (jika ada) setelah satu PTSA berhasil dibentuk. Proses pembentukan seluruh PTSA graf kipas untuk nilai n dan k yang diberikan akan selesai setelah s_1 dilabel dengan seluruh label yang mungkin.

Algoritma 3(i).
function fan(*t*)

```
1.  if t = n then
2.    finalizefan
3.  else
4.    for each available label do
5.      λ(s_t):= i
6.      avail [i]:= false
7.      for each available label j do
8.        λ(r_t):= j
9.        avail [j]:= false
10.       if t = 1
11.         λ(v_t):= k - λ(s_t) - λ(r_t)
12.       else
13.         λ(v_t):= k - λ(s_t) - λ(r_t) - λ(r_{t-1})
14.         if 0 < λ(v_t) ≤ 3n and avail
            [λ(v_t)] then
15.           avail [λ(v_t)]:= false
16.           fan (t+1)
17.           avail [λ(v_t)]:= true
            end if
          end if
18.       avail [j]:= true
        end for
19.     avail [i]:= true
      end for
    end if
```

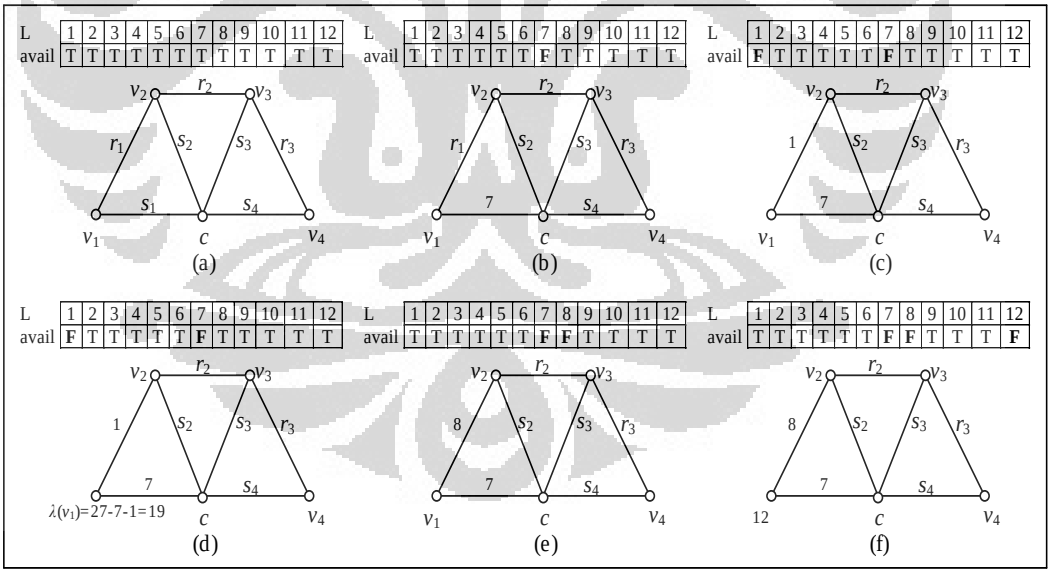
Algoritma 3(ii).
function finalizefan

```
1.  for each available label i > λ(s_1) do
2.    λ(s_n):= i
3.    avail [i]:= false
4.    λ(v_n):= k - λ(s_t) - λ(r_{t-1})
5.    if 0 < λ(v_n) ≤ 3n and avail [λ(v_n)]
      then
6.      avail [λ(v_n)]:= false
7.      λ(c):= k − Σ_{i=1}^{n} λ(s_i)
8.      if n = 3 then
9.        if λ(v_2) < λ(c) ≤ 3n and avail
          [λ(c)] then
10.         print
11.         avail [λ(v_n)]:= true
          end if
12.     else
13.       if 0 < λ(c) ≤ 3n and avail [λ(c)]
          then
14.         print
15.         avail [λ(v_n)]:= true
          end if
        end if
      end if
16.     avail [i]:= true
    end for
```

Jadi, algoritma PTSA graf kipas dapat dibentuk dari dua fungsi. Fungsi pertama (Algoritma 3(i)) adalah fungsi untuk melabel v_t ($t = 1,\ldots,n - 1$). Fungsi ini akan memanggil dirinya sendiri dengan parameter t yang menunjukkan bahwa

busur dan simpul ke-t akan dilabel. Fungsi kedua (Algoritma 3(ii)) adalah fungsi untuk melabel dua simpul terakhir, yaitu v_n dan c. Karena fungsi kedua ini hanya akan menyelesaikan pelabelan pada tahap akhir, fungsi ini akan dipanggil ketika $t = n$. Pada fungsi kedua ini s_n dan c dilabel, jadi syarat yang sebelumnya dipilih ($\lambda(s_n) > \lambda(s_1)$ dan $\lambda(c) > \lambda(v_2)$) untuk menjamin PTSA yang dihasilkan tidak isomorfik ditambahkan pada fungsi ini (Algoritma 3(ii) baris ke-1 dan ke-9). Dari syarat $\lambda(s_n) > \lambda(s_1)$ dapat disimpulkan bahwa label yang mungkin untuk s_1 adalah $1,\ldots,3n-1$. Andaikan s_1 dilabel dengan $3n$, maka tidak tersisa label yang mungkin untuk s_n. Karena itu, jari-jari s_1 tidak mungkin dilabel dengan $3n$.

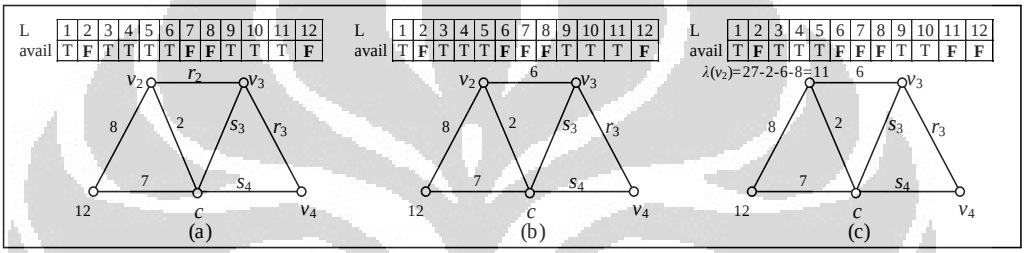
Selanjutnya akan diberikan contoh pembentukan PTSA graf kipas dengan $n = 4$ dan $k = 27$. Label yang tersedia untuk membentuk PTSA f_4 adalah $L = \{1,2,\ldots,12\}$. Seperti disinggung pada Subbab 3.1, pemilihan label untuk melabel busur tidak perlu berurutan dari label terkecil sampai terbesar, atau sebaliknya. Namun, untuk memudahkan proses pelabelan, biasanya pemilihan label dilakukan secara berurutan.



Gambar 3. 20 Contoh penerapan algoritma PTSA graf kipas f_4 dengan $k = 27$ (Tahap pelabelan simpul v_1)

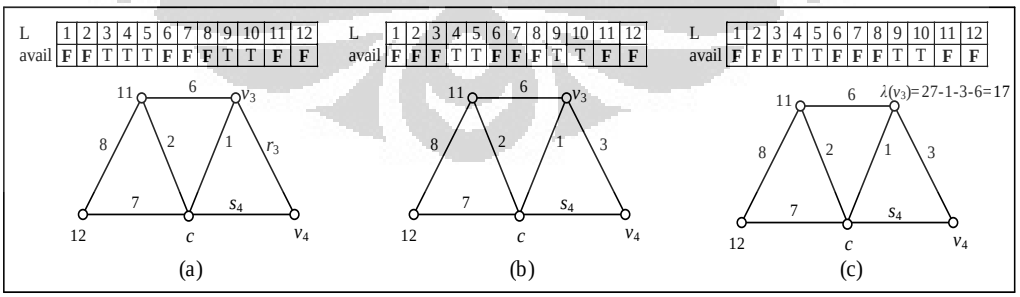
Proses pelabelan diawali dengan memanggil fungsi fan dengan $t = 1$. Label yang mungkin untuk s_1 adalah $1,\ldots,12$ (Gambar 3.21a). Misalnya s_1 dilabel dengan 7 (Gambar 3.21b). Lalu, busur r_1 dilabel dengan 1 (Gambar 3.21c). Maka,

label untuk v_1 adalah $\lambda(v_1) = 27 - 7 - 1 = 19$ (Gambar 3.21d). Karena label 19 tidak tersedia, v_1 tidak bisa dilabel dengan 19. Untuk mendapatkan label yang lain untuk v_1, maka $\lambda(r_1)$ akan diganti. Label r_1 akan diganti hingga didapatkan label yang mungkin untuk v_1. Melabel r_1 dengan 2,…,7 akan menghasilkan label yang tidak mungkin untuk v_1. Selanjutnya, r_1 akan dilabel dengan 8 (Gambar 3.21e). Dengan demikian akan diperoleh $\lambda(v_1) = 27 - 7 - 8 = 12$ (Gambar 3.21f). Karena $\lambda(v_1)$ yang diperoleh sesuai dengan label yang tersedia, maka proses pelabelan bisa dilanjutkan ke tahap berikutnya. Fungsi fan akan memanggil dirinya sendiri dengan parameter $t = 2$ (baris ke-16).



Gambar 3. 21 Contoh penerapan algoritma PTSA graf kipas f_4 dengan $k = 27$ (Tahap pelabelan simpul v_2)

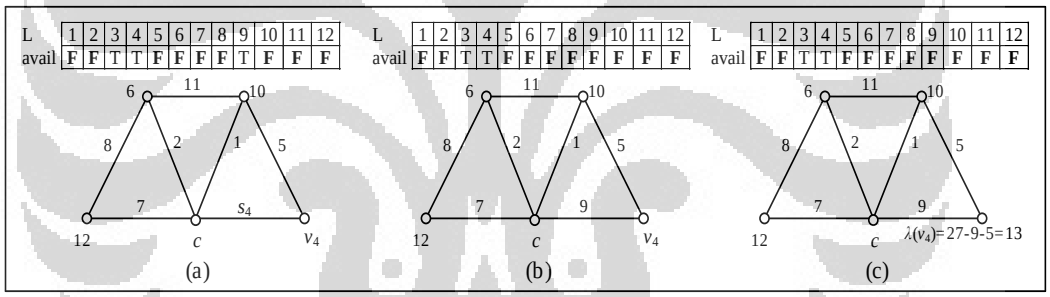
Langkah yang dilakukan pada tahap $t = 2$ ini serupa dengan tahap sebelumnya. Bedanya, label v_2 diperoleh dari $\lambda(v_2) = k - \lambda(s_2) - \lambda(r_2) - \lambda(r_1)$, pada algoritma ditunjukkan pada baris ke-13. Gambar 3.22 menunjukkan proses pada tahap ini.



Gambar 3. 22 Contoh penerapan algoritma PTSA graf kipas f_4 dengan $k = 27$ (Tahap pelabelan simpul v_3)

Setelah tahap $t = 2$ selesai, fungsi fan kembali dipanggil dengan $t = 3$. Label yang tersedia pada tahap ini adalah {1,3,4,5,9,10}. Jari-jari s_3 akan dilabel

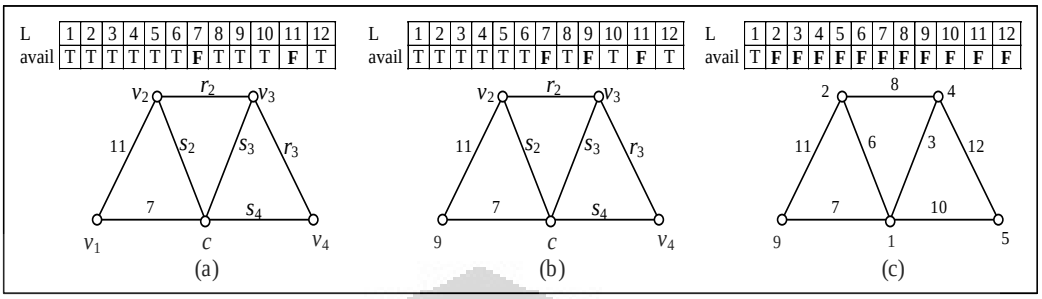
dengan 1 (Gambar 3.23a). Kemudian r_3 dilabel dengan 3 (Gambar 3.23b), sehingga diperoleh label untuk v_3, yaitu $\lambda(v_3) = 27 - 1 - 3 - 6 = 17$ (Gambar 3.23c). Karena label 17 tidak tersedia, maka v_3 tidak dapat dilabel dengan 17. Label r_3 akan diganti agar dihasilkan $\lambda(v_3)$ yang lain. Namun, semua label yang tersedia untuk melabel r_3 pada tahap ini tidak menghasilkan label yang mungkin untuk v_3. Karena itu, dilakukan *backtracking*. Mengganti label s_3 dengan label lain yang tersedia pada tahap ini juga akan ber-akhir dengan kegagalan dalam memperoleh $\lambda(v_3)$ yang mungkin. Oleh sebab itu, *backtracking* kembali dilakukan. Proses pelabelan kembali ke tahap $t = 2$, $\lambda(r_2)$ akan diganti dengan label lain. Sebelumnya, $\lambda(v_2)$ dan $\lambda(r_2)$ yang telah digunakan tersedia kembali (baris ke-17 dan ke-18). Lalu, proses pelabelan kembali dilakukan dari baris ke-7. Langkah yang dilakukan selanjutnya sama seperti yang telah dijelaskan. Gambar 3.24a menunjukkan pelabelan yang terbentuk setelah tahap ketiga.



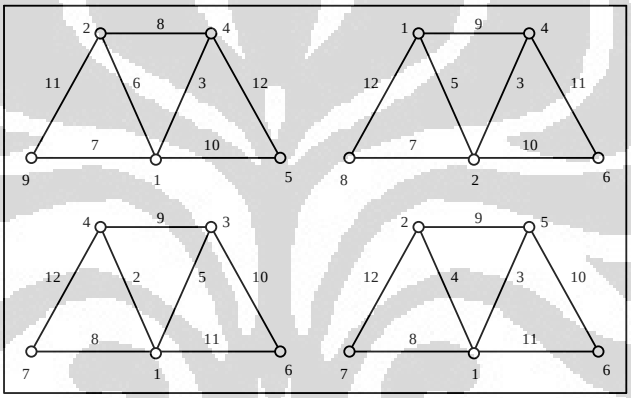
Gambar 3. 23 Contoh penerapan algoritma PTSA graf kipas f_4 dengan $k = 27$ (Tahap pelabelan simpul v_4)

Fungsi fan kembali dipanggil dengan $t = 4$. Tetapi, karena $t = 4$ merupakan tahap terakhir, maka fungsi finalizefan dipanggil (baris ke-2). Label yang tersedia pada tahap ini adalah {3,4,9}. Label yang mungkin untuk s_4 harus lebih besar dari $\lambda(s_1)$, sesuai dengan syarat untuk mencegah kasus refleksi (baris ke-1). Artinya, hanya label 9 yang mungkin dipilih untuk melabel s_4 (Gambar 3.24b). Akibatnya, diperoleh $\lambda(v_4) = 27 - 9 - 5 = 13$ (Gambar 3.24c). Label 13 tidak tersedia, sehingga v_4 tidak mungkin dilabel dengan label ini. Untuk mendapatkan $\lambda(v_4)$ yang lain, label s_4 akan diganti. Namun, karena tidak ada label lain yang mungkin untuk melabel s_4 pada tahap ini, maka dilakukan *backtracking*. Pelabelan kembali ke tahap

t = 3, label r_3 akan diganti. Proses yang dilakukan serupa dengan proses penggantian $\lambda(r_2)$ pada penjelasan sebelumnya.



Gambar 3. 24 Contoh penerapan algoritma PTSA graf kipas f_4 dengan k = 27 (*Backtracking* dan PTSA yang terbentuk)



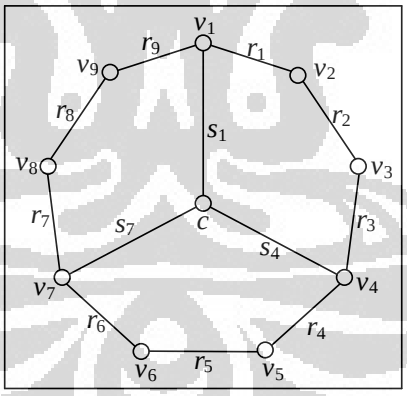
Gambar 3. 25 PTSA pada graf f_4 dengan k = 27

Untuk contoh ini, langkah *backtracking* akan dilakukan sampai r_1. Penggantian $\lambda(r_3)$ dengan label lain yang tersedia pada tahap tersebut, akan menghasilkan label yang tidak mungkin untuk v_4. Hal yang sama terjadi pada penggantian $\lambda(s_3)$, $\lambda(r_2)$, dan $\lambda(s_2)$ dengan label yang tersedia pada setiap tahap yang terkait. Proses pelabelan kembali ke baris 7 (Algoritma 3(i)). Label yang tersedia untuk melabel r_1 adalah {1,…,6,8,…,12}. Melabel r_1 dengan 9 atau 10 akan berakhir sama seperti ketika r_1 dilabel dengan 8. Selanjutnya r_1 akan dilabel dengan 11 (Gambar 3.25a), sehingga diperoleh $\lambda(v_1)$ = 27 – 7 – 11 = 9 (Gambar 3.25b). Karena $\lambda(v_1)$ yang diperoleh sesuai dengan label yang ada, proses pelabelan akan berlanjut ke tahap berikutnya. Langkah yang dilakukan sama seperti yang telah dijelaskan sebelumnya. PTSA yang akan terbentuk, dapat dilihat pada Gambar 3.25c. PTSA ini akan dicetak oleh fungsi print. Langkah *backtracking*

kembali dilakukan untuk mendapatkan PTSA yang lain. Gambar 3.26 menunjukkan semua PTSA yang tidak isomorfik yang terbentuk untuk f_4 dengan $k = 27$.

3.3 Algoritma PTSA Graf Jahangir yang Diperumum

Dua graf yang telah dibahas sebelumnya, graf *friendship* dan kipas, bisa diperoleh dari graf roda dengan menghapus satu atau lebih busur pada bagian rim. Berbeda dengan kedua graf tersebut, graf jahangir yang diperumum $J_{n,m}$ diperoleh dari graf roda dengan menghapus beberapa jari-jari graf roda secara berselang-seling. Seperti telah dijelaskan pada Bab 2, nilai n pada $J_{n,m}$ menyatakan banyaknya simpul pada bagian rim yang terhubung dengan simpul tengah, sedangkan nilai m menyatakan jarak antar simpul pada rim yang terhubung dengan simpul tengah c. Sama seperti dua graf sebelumnya, simpul dan busur pada bagian luar disebut rim dan dinotasikan dengan v dan r. Sementara busur yang menghubungkan v dan c biasa disebut jari-jari (*spoke*) dan dinotasikan dengan s. Gambar 3.27 menunjukkan contoh graf jahangir yang diperumum $J_{3,3}$.



Gambar 3. 26 Graf $J_{3,3}$

Batas konstanta ajaib k untuk PTSA graf $J_{n,m}$ ini bisa diperoleh dengan dua cara yang sama seperti dua graf sebelumnya. Pertama, dengan mensubstitusi nilai $|V|$ dan $|E|$ ke rumus umum batas konstanta ajaib k yang diperoleh berdasarkan perhitungan dasar pada (2), yaitu

$$\frac{|E|(|E|+1)}{2} + \frac{(|E|+|V|)(|E|+|V|+1)}{2} \leq |V|k \leq \frac{|E|(|E|+2|V|+1)}{2} + \frac{(|E|+|V|)(|E|+|V|+1)}{2}.$$

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

Banyaknya simpul pada graf ini adalah nm pada bagian rim dan satu simpul tengah, atau $|V| = nm + 1$. Sementara banyaknya busur adalah *nm* pada bagian rim dan *n* jari-jari, atau $|E| = nm + n$. Dengan mensubstitusi nilai $|V|$ dan $|E|$ ini ke (2), maka akan diperoleh

$$\frac{(nm+n)(nm+n+1)}{2} + \frac{(2nm+n+1)(2nm+n+2)}{2} \leq (nm+1)k \leq \frac{(nm+n)(3nm+n+3)}{2} + \frac{(2nm+n+1)(2nm+n+2)}{2}$$

atau,

$$\frac{5n^2m^2+6n^2m+7nm+2n^2+4n+2}{2(nm+1)} \leq k \leq \frac{7n^2m^2+8n^2m+9nm+2n^2+6n+2}{2(nm+1)}. \quad (3.7)$$

Cara yang kedua adalah dengan perhitungan yang memperhatikan struktur grafnya. Batas bawah konstanta ajaib *k* bisa diperoleh dengan melabel simpul dengan derajat terbesar dan busur-busur yang terhubung dengan simpul tersebut dengan label-label terkecil. Pada graf $J_{n,m}$ dengan $n > 2$, derajat dari simpul tengah *c* tidak lebih kecil dari simpul lainnya. Karena itu, dengan melabel simpul tengah *c* dan jari-jari dengan $n + 1$ label terkecil akan diperoleh batas bawah *k*, yaitu

$$k \geq \sum_{i=1}^{n+1} i = \frac{(n+1)(n+2)}{2}. \quad (3.8)$$

Untuk graf $J_{2,m}$, simpul yang memiliki derajat terbesar adalah simpul pada bagian rim yang terhubung dengan simpul tengah. Simpul tersebut berderajat tiga. Jadi batas bawah *k* yang mungkin untuk membentuk PTSA graf $J_{2,m}$ akan sama dengan $J_{3,m}$. Sementara itu, batas atas *k* untuk membangun PTSA graf ini tidak mungkin lebih besar dari bobot maksimal yang mungkin untuk simpul pada bagian rim. Jumlah maksimal dari bobot *nm* simpul pada bagian rim bisa diperoleh dengan melabel busur pada bagian rim dengan *nm* label terbesar dan simpul pada bagian rim serta jari-jari dengan *nm* + *n* label terbesar berikutnya. Hal ini dikarenakan label busur pada bagian rim berkontribusi dua kali pada bobot simpul pada bagian rim. Secara matematis,

$$\begin{aligned} w(v_1) + \cdots + w(v_{nm}) &\leq 2\sum_{i=nm+n+2}^{2nm+n+1} i + \sum_{i=2}^{nm+n+1} i \\ &= \sum_{i=2}^{2nm+n+1} i + \sum_{i=nm+n+2}^{2nm+n+1} i \\ &= \frac{(2nm+n)(2nm+n+3)}{2} + \frac{nm(3nm+2n+3)}{2} \\ &= \frac{7n^2m^2+6n^2m+9nm+n^2+3n}{2}. \end{aligned}$$

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

Karena simpul pada bagian rim ini ada sebanyak *nm*, maka

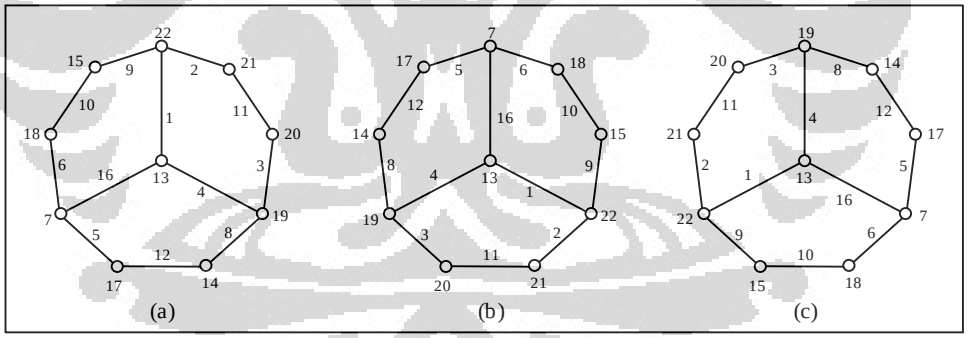
$$nmk \leq \frac{7n^2m^2+6n^2m+9nm+n^2+3n}{2}$$

atau,

$$k \leq \frac{7nm+6n+9}{2} + \frac{n+3}{2m}. \qquad (3.9)$$

Sama seperti dua graf sebelumnya, batas *k* yang mungkin untuk graf ini diperoleh dari irisan hasil yang diperoleh dari perhitungan cara pertama dan kedua, yaitu dari (3.7), (3.8), dan (3.9). Pembulatan yang dilakukan terhadap nilai *k* yang diperoleh juga sama seperti dua graf sebelumnya. Contohnya untuk graf $J_{2,3}$, batas konstanta ajaib *k* yang mungkin untuk membangun PTSA graf ini adalah $23 \leq k \leq 30$ berdasarkan (3.7). Berdasarkan (3.8) dan (3.9), batas yang diperoleh adalah $10 \leq k \leq 32$. Jadi, batas konstanta ajaib *k* yang mungkin untuk membangun PTSA graf $J_{2,3}$ adalah $23 \leq k \leq 30$.

Sebelum masuk pada penjelasan algoritma PTSA untuk graf jahangir yang diperumum, akan diberikan terlebih dahulu penjelasan mengenai kasus isomorfisma yang mungkin terjadi pada PTSA graf ini.



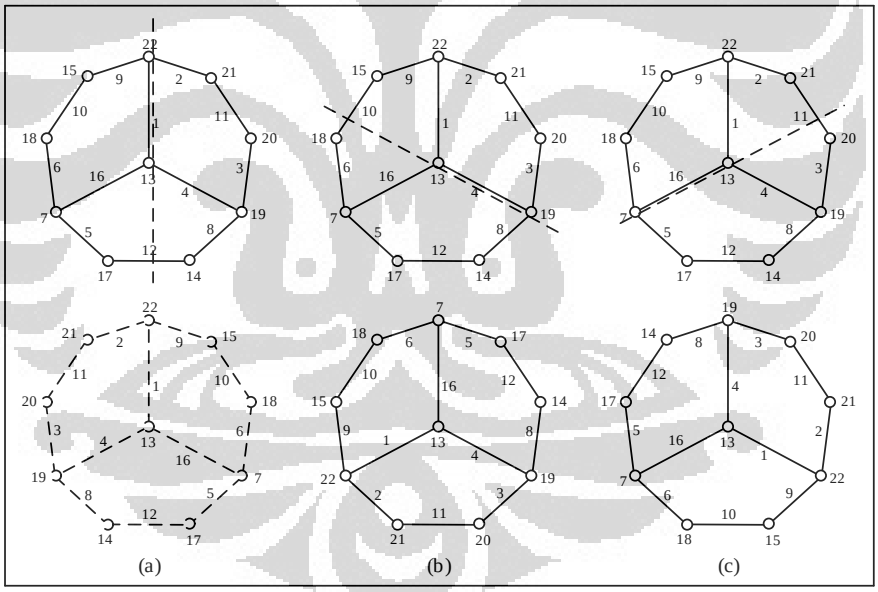
Gambar 3. 27 Kasus rotasi pada PTSA graf $J_{3,3}$

Berbeda dari PTSA graf kipas f_n ($n > 3$) yang tidak mungkin mengalami rotasi, PTSA graf $J_{n,m}$ mungkin mengalami rotasi. Dari satu PTSA graf $J_{n,m}$ bisa diperoleh $n - 1$ PTSA yang isomorfik dengan PTSA tersebut. Contohnya, pada PTSA graf $J_{3,3}$ (Gambar 3.28a), dari PTSA tersebut bisa diperoleh dua PTSA yang isomorfik dengan melakukan rotasi (Gambar 3.28b dan c). PTSA yang terbentuk karena kasus rotasi dapat dieliminasi, salah satunya dengan memilih s_1 sebagai

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

jari-jari yang dilabel dengan label terkecil dibandingkan jari-jari yang lain, atau $\lambda(s_t) > \lambda(s_1)$ dimana $t = 2,\ldots,nm$.

Selain karena rotasi, PTSA yang isomorfik juga mungkin terjadi karena kasus refleksi. Sumbu simetri dari kasus refleksi yang terjadi pada graf ini selalu melalui simpul tengah *c*. Gambar 3.29 menunjukkan contoh kasus refleksi pada PTSA $J_{3,3}$ (garis putus-putus menunjukkan sumbu simetri dari refleksi yang terjadi). Jika diperhatikan beberapa PTSA yang terbentuk karena kasus refleksi ini (Gambar 3.29b dan c) dapat dieliminasi dengan syarat yang dipilih untuk kasus rotasi sebelumnya, yaitu $\lambda(s_t) > \lambda(s_1)$ dimana $t = 2,\ldots,n$. Namun, tidak semua PTSA yang terbentuk karena kasus refleksi ini tereliminasi, yaitu PTSA yang terbentuk karena refleksi terhadap sumbu yang melalui s_1. Perlu ada syarat lain yang diberikan. Salah satu yang bisa dipilih adalah melabel r_n dengan label yang lebih kecil dari label r_1, $\lambda(r_n) < \lambda(r_1)$.

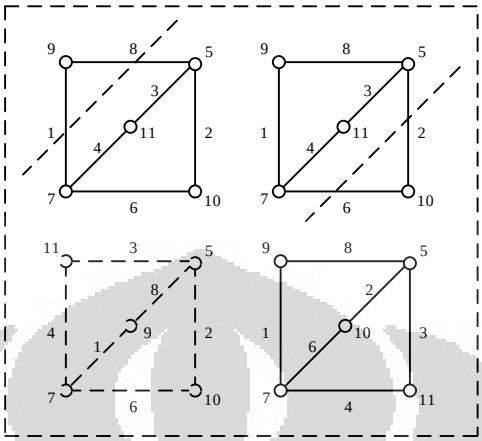


Gambar 3. 28 Kasus rotasi pada PTSA graf $J_{3,3}$

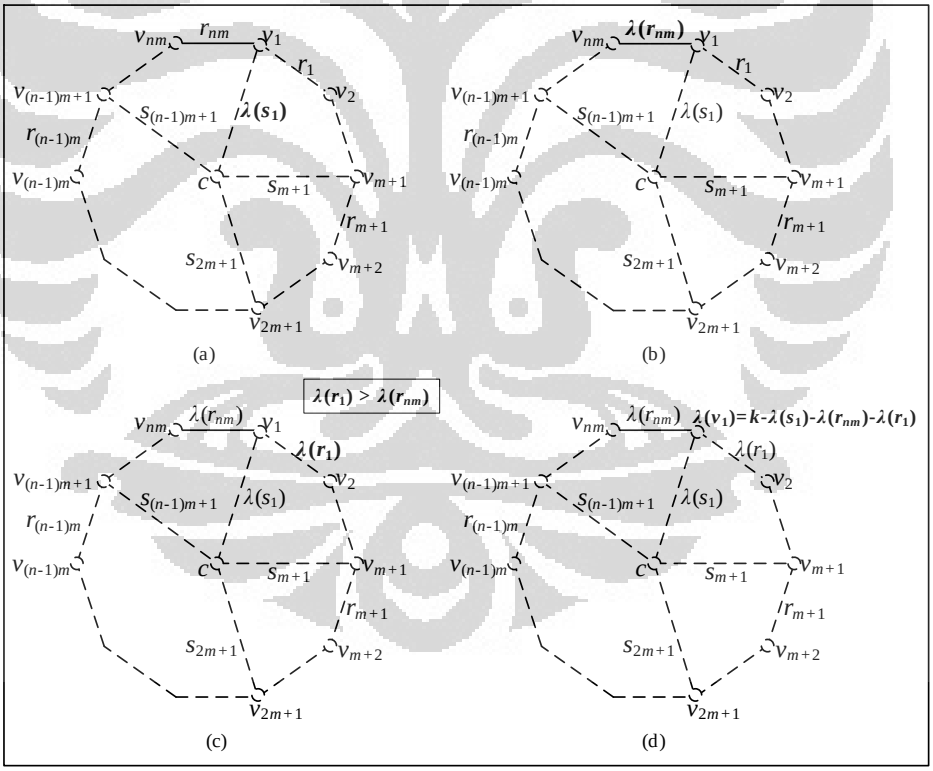
Untuk PTSA graf $J_{2,2}$, PTSA yang isomorfik tidak hanya terjadi karena kasus rotasi dan refleksi seperti yang telah dijelaskan sebelumnya. Pada graf $J_{2,2}$ refleksi mungkin terjadi secara sebagian. Lebih jelasnya, perhatikan Gambar 3.30. Dua syarat yang telah ditentukan sebelumnya, tidak cukup untuk mengeliminasi PTSA yang isomorfik pada graf $J_{2,2}$. Perlu ada syarat tambahan yang diberikan

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

untuk mengeliminasi PTSA isomorfik pada graf ini. Salah satu syarat yang bisa dipilih adalah dengan menetapkan $\lambda(v_2) < \lambda(c) < \lambda(v_4)$.



Gambar 3. 29 Kasus refleksi sebagian pada PTSA graf $J_{2,2}$



Gambar 3. 30 Proses pelabelan graf jahangir yang diperumum $J_{n,m}$

Selanjutnya akan dibahas mengenai algoritma PTSA untuk graf jahangir yang diperumum. Misalkan pelabelan simpul dilakukan mengikuti urutan $v_1, v_2, \ldots, v_{nm}$, dan terakhir c. Pada graf $J_{n,m}$, simpul v_1, $v_{m+1}, \ldots, v_{(n-1)m+1}$ terhubung dengan

tiga busur, simpul $v_2,...,v_m,\ldots,v_{nm}$ terhubung dengan dua busur, dan simpul c terhubung dengan semua jari-jari. Dengan mengikuti urutan pelabelan simpul yang telah ditentukan, maka tahap pelabelan simpul v_1 akan diawali dengan pelabelan tiga busur, sedangkan $v_{m+1},\ldots,v_{(n-1)m+1}$ hanya akan diawali dengan pelabelan dua busur. Berbeda dengan dua graf sebelumnya, semua jari-jari pada graf $J_{n,m}$ akan telah dilabel pada tahap pelabelan $v_{(n-1)m+1}$. Jadi, pada tahap pelabelan $v_{(n-1)m+1}$ label dari simpul c bisa diperoleh. Sementara itu, tahap pelabelan simpul lainnya selain v_{nm} akan diawali dengan pelabelan satu busur. Tahap pelabelan v_{nm} tidak perlu diawali dengan pelabelan busur karena busur yang terhubung dengan v_{nm} telah dilabel pada tahap sebelumnya.

Simpul v_1 terhubung dengan busur s_1, r_1, dan r_{nm}, maka tahap pelabelan v_1 diawali dengan pelabelan tiga busur. Pada tahap pelabelan v_1, misalkan s_1 dilabel terlebih dahulu dengan $\lambda(s_1)$. Kemudian, r_{nm} dilabel dengan $\lambda(r_{nm})$ dan terakhir r_1 dilabel dengan $\lambda(r_1)$. Maka, label v_1 diperoleh dari $\lambda(v_1) = k - \lambda(s_1) - \lambda(r_1) - \lambda(r_{nm})$. Tahap pelabelan simpul v_1 dengan asumsi $\lambda(v_1)$ yang diperoleh sesuai dengan label yang tersedia bisa dilihat pada Gambar 3.31. Jika $\lambda(v_1)$ yang diperoleh sesuai dengan label yang tersedia, maka proses pelabelan berlanjut ke tahap pelabelan v_2. Namun, jika yang terjadi sebaliknya, maka $\lambda(r_1)$ akan diganti dengan label lain yang tersedia hingga diperoleh $\lambda(v_1)$ yang mungkin. Jika semua kemungkinan label yang ada untuk r_1 tidak menghasilkan label yang mungkin untuk v_1, maka dilakukan *backtracking*. Label r_{nm} akan diganti dengan label lain yang tersedia. Seandainya semua label yang mungkin untuk r_{nm} juga tidak menghasilkan label yang mungkin untuk v_1, *backtacking* kembali dilakukan. Label s_1 akan diganti dengan label lain yang tersedia hingga diperoleh $\lambda(v_1)$ yang sesuai dengan label yang ada. Jika proses ini juga tidak berakhir dengan $\lambda(v_1)$ yang mungkin, maka dapat disimpulkan bahwa tidak terdapat PTSA untuk graf $J_{n,m}$ dengan k yang diberikan.

Semua simpul pada bagian rim yang tidak terhubung dengan simpul c, yaitu $v_2,\ldots,v_m,v_{m+2},\ldots,v_{2m},\ldots,v_{2m+2},\ldots,v_{nm}$, hanya terhubung dengan dua busur, yaitu r_t dan r_{t-1} dimana $t = 2, m, m+2, 2m, \ldots, nm$. Tahap pelabelan simpul-simpul ini akan diawali dengan pelabelan satu busur yaitu r_t, kecuali v_{nm} (tahap pelabelan simpul v_{nm} akan dilakukan pada tahap akhir). Pada tahap pelabelan v_t, kecuali v_{nm},

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

pertama akan dilabel busur r_t dengan $\lambda(r_t)$. Lalu, label untuk v_t bisa diperoleh dari $\lambda(v_t) = k - \lambda(r_t) - \lambda(r_t)$. Jika $\lambda(v_t)$ yang diperoleh sesuai dengan label yang ada, maka proses pelabelan berlanjut ke tahap selanjutnya. Jika tidak, maka $\lambda(r_t)$ akan diganti dengan label lain yang tersedia hingga diperoleh $\lambda(v_t)$ yang mungkin. Apabila penggantian $\lambda(r_t)$ dengan semua label yang tersedia pada tahap tersebut tidak menghasilkan $\lambda(v_t)$ yang mungkin, maka dilakukan *backtracking*.

Algoritma 4(i).
function jahangir1

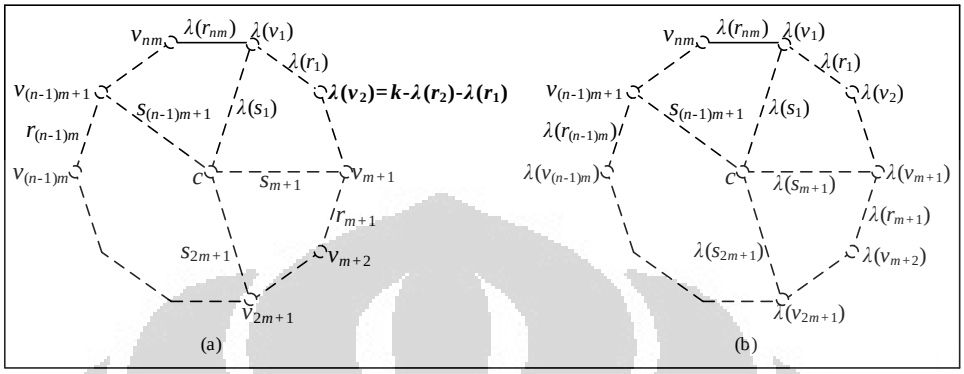
```
1.  for each available label i do
2.      λ(s1):= i
3.      avail [i]:= false
4.      for each available label j do
5.        λ(rnm):= j
6.        avail [j]:= false
7.        for each available label i> λ(rnm)
          do
8.          λ(r1):=i
9.          avail[i]:=false
10.         λ(v1):= k -λ(s1) -λ(r1)- λ(rnm)
11.         if 0 < λ(v1) ≤ 2nm + n + 1 and
            avail [λ(v1)] then
12.           avail [λ(v1)]:= false
13.           jahangir2(2)
14.           avail [λ(v1)]:= true
            end if
15.         avail[i]:=true
          end for
16.       avail [j]:= true
        end for
17.     avail [i]:= true
      end for
```

Algoritma 4(ii).
function jahangir2(*t*)

```
1.  if t = nm
2.    jahangir4(t)
3.  else
4.    for each available label i do
5.      λ(rt):= i
6.      avail [i]:= false
7.      λ(vt):= k - λ(rt) - λ(rt-1)
8.      if 0 < λ(vt) ≤ 2nm + n + 1 and
        avail [λ(vt)] then
9.        avail [λ(vt)]:= false
10.       jahangir3(t+1)
11.       avail [λ(vt)]:= true
        end if
12.     avail [i]:= true
      end for
    end if
```

Langkah pelabelan yang dilakukan untuk simpul $v_{m+1},\ldots,v_{(n-2)m+1}$ sebenarnya serupa dengan langkah yang dilakukan pada tahap pelabelan v_1. Bedanya, untuk tahap ini hanya akan diawali dengan dua pelabelan busur. Untuk memperoleh label dari simpul v_t dengan $t = m + 1, \ldots, (n-2)m + 1$, pertama akan dila-bel s_t dengan $\lambda(s_t)$. Lalu, busur r_t dengan $\lambda(r_t)$. Setelah kedua busur tersebut dila-bel, maka label v_t bisa diperoleh dari $\lambda(v_t) = \mathrm{k} - \lambda(s_t) - \lambda(r_t) - \lambda(r_{t-1})$. Jika $\lambda(v_t)$ yang diperoleh sesuai dengan label yang tersedia, maka pelabelan dilanjutkan ke tahap selanjutnya. Jika sebaliknya, maka langkah yang dilakukan sama seperti tahap pelabelan v_1. Bedanya, jika $\lambda(v_t)$ tidak diperoleh setelah dilakukan penggantian $\lambda(r_t)$ atau $\lambda(s_t)$, maka akan dilakukan *backtracking* ke tahap sebelumnya. Label

dari simpul dan busur terakhir yang dilabel pada tahap t - 1 akan tersedia kembali. Kemudian, label dari busur tersebut akan diganti dengan label lain yang tersedia dan proses pelabelan berlanjut seperti sebelumnya.



Gambar 3. 31 Proses pelabelan graf jahangir yang diperumum $J_{n,m}$

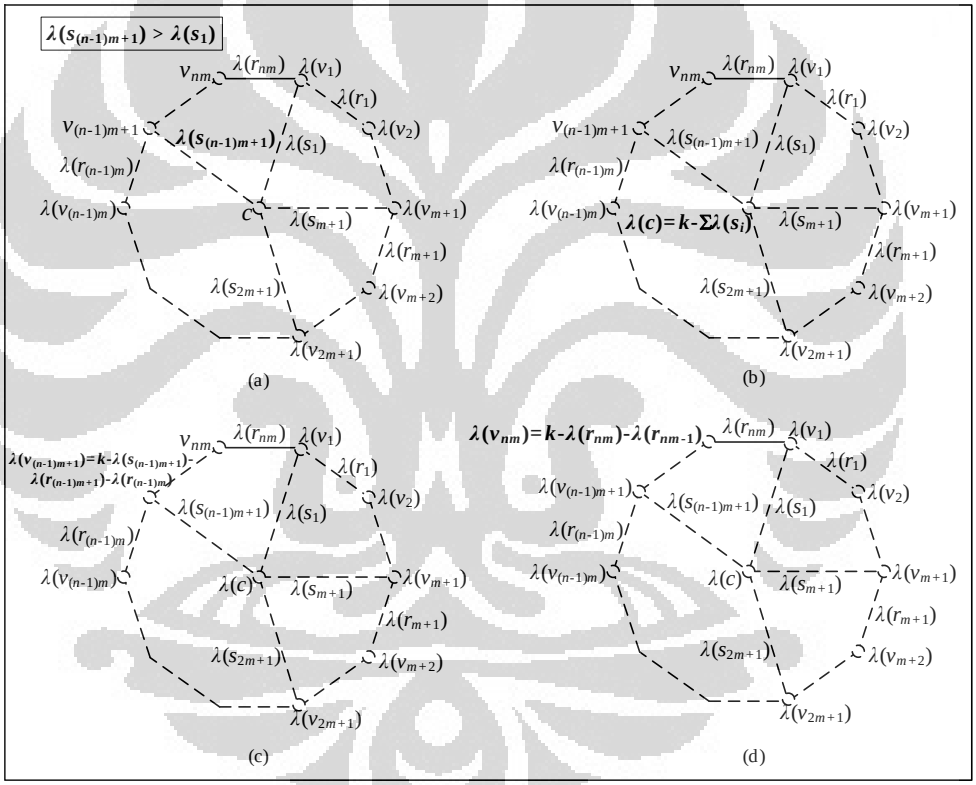
Algoritma 4(iii).
function jahangir3(*t*)
1. **if** $t = nm - m + 1$
2. **jahangir4(*t*)**
3. **elseif** (t mod m) $\neq 1$
4. **jahangir2(*t*)**
5. **else**
6. **for** each available label $i > \lambda(s_1)$ **do**
7. $\lambda(s_t) := i$
8. *avail* [i]:= **false**
9. **for** each available label j **do**
10. $\lambda(r_t) := j$
11. *avail* [j]:= **false**
12. $\lambda(v_t) := k - \lambda(s_t) - \lambda(r_t) - \lambda(r_{t-1})$
13. **if** $0 < \lambda(v_t) \leq 2nm + n + 1$ **and** *avail* [$\lambda(v_t)$] **then**
14. *avail* [$\lambda(v_t)$]:= **false**
15. **jahangir2(*t*+1)**
16. *avail* [$\lambda(v_t)$]:= **true**
end if
17. *avail* [j]:=**true**
end for
18. *avail* [j]:= **true**
end for
end if

Tahap pelabelan v_t dengan $t = (n - 1)m + 1$ sebenarnya akan diawali dengan pelabelan dua busur sama seperti simpul lain pada bagian rim yang terhubung dengan simpul c. Bedanya, setelah s_t dilabel dengan $\lambda(s_t)$ (Gambar 3.33a), proses pelabelan dilanjutkan dengan menentukan label untuk simpul c terlebih

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

dahulu, yaitu $\lambda(c) = k - \Sigma\lambda(s_i)$ (Gambar 3.33b). Setelah itu, barulah r_t dilabel dengan $\lambda(r_t)$ dan v_t diperoleh dari $\lambda(v_t) = k - \lambda(s_t) - \lambda(r_t) - \lambda(r_{t-1})$ (Gambar 3.33c).

Untuk tahap pelabelan v_{nm}, karena busur r_{nm} telah dilabel pada tahap pertama untuk menentukan label v_1, maka pada tahap pelabelan v_{nm} tidak akan diawali dengan pelabelan busur. Label untuk v_{nm} diperoleh dari $\lambda(v_{nm}) = k - \lambda(r_{nm}) - \lambda(r_{nm-1})$ (Gambar 3.33d). Jika $\lambda(v_{nm})$ yang diperoleh sesuai dengan label yang tersedia, maka satu PTSA selesai dibentuk. PTSA ini akan dicetak oleh fungsi print. Jika sebaliknya, maka dilakukan *backtracking*. Langkah ini juga dilakukan untuk mencari PTSA lain yang mungkin setelah satu PTSA berhasil dibentuk.



Gambar 3. 32 Proses pelabelan graf jahangir yang diperumum $J_{n,m}$

Dengan demikian algoritma PTSA untuk graf $J_{n,m}$ dapat dibentuk dari empat fungsi. Fungsi jahangir1 (Algoritma 4(i)) adalah untuk melabel simpul v_1. Fungsi jahangir2 (Algoritma 4(ii)) berguna untuk melabel simpul pada bagian rim yang tidak terhubung dengan simpul c, kecuali v_{nm}. Sedangkan fungsi jahangir3 (Algoritma 4(iii)) untuk melabel simpul pada bagian rim yang terhubung dengan simpul c, kecuali simpul terakhir yang terhubung dengan simpul c yaitu $v_{(n-1)m+1}$.

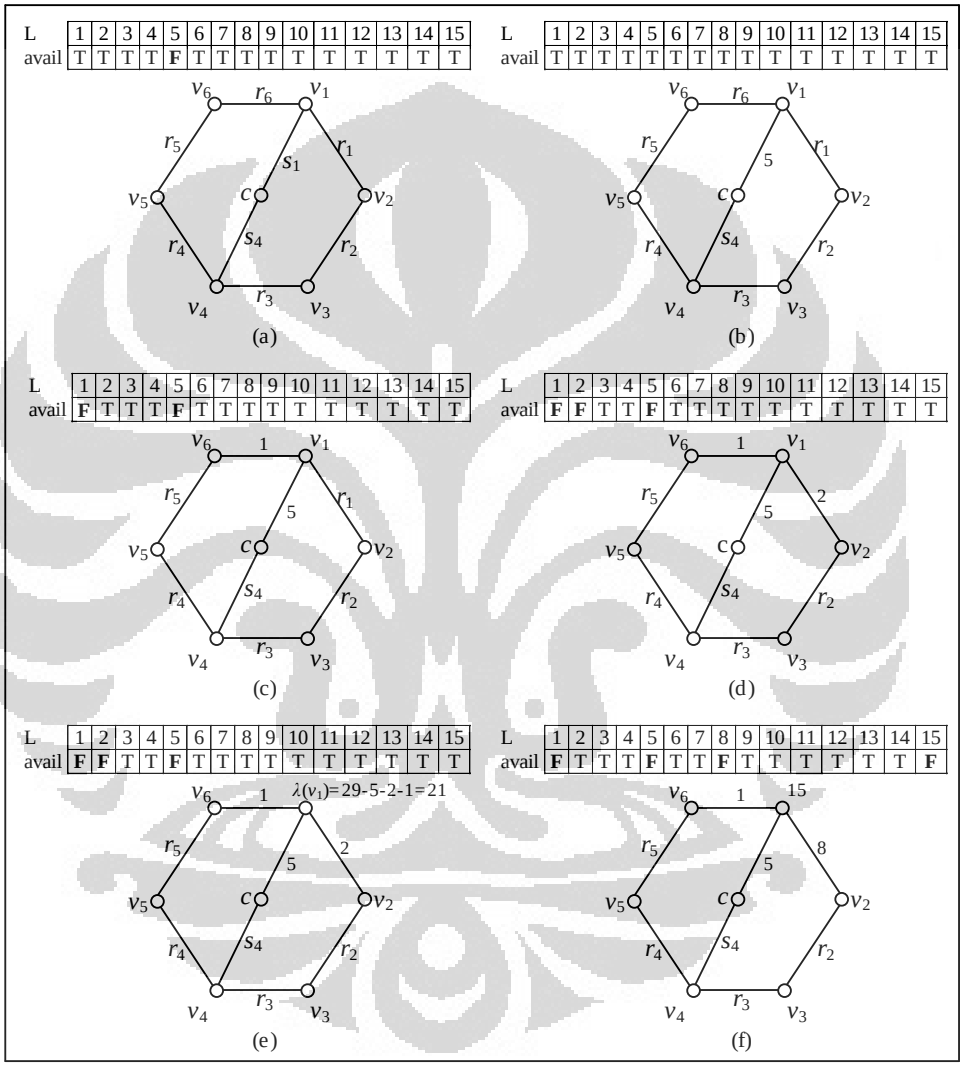
Terakhir, fungsi finalizejahangir (Algoritma 4(iv)) berguna untuk melabel simpul $v_{(n-1)m+1}$, c, dan v_{nm}. Syarat yang diberikan untuk mencegah terbentuknya PTSA yang isomorfis, yaitu $\lambda(r_n) > \lambda(r_1)$ ditambahkan pada fungsi jahangir1. Sementara syarat $\lambda(s_t) > \lambda(s_1)$ untuk $t = 2,\ldots,nm$ ditambahkan pada fungsi jahangir2 dan finalizejahangir.

Algoritma 4(iv).
function finalizejahangir(*t*)

```
1.  if t = nm then
2.    if nm = 4 then
3.      λ(v_nm):= k- λ(r_nm)-λ(r_nm-1)
4.      if λ(c) < λ(v_nm) ≤ 2nm + n + 1 and
        avail [λ(v_nm)] then
5.        print
        end if
6.    else
7.      λ(v_nm):= k - λ(r_nm) - λ(r_nm-1)
8.      if 0 < λ(v_nm) ≤ 2nm + n + 1 and
        avail [λ(v_nm)] then
9.        print
        end if
      end if
10. else
11.   for each available label [i >λ(s_1)] do
12.     λ(s_t):= i
13.     avail [i]:= false
14.     if nm = 4
15.       λ(c):=k-∑_{i=1}^{n} s_i
16.       if [λ(v_1) < λ(c)] ≤ 2nm + n + 1
          and avail [λ(c)] then
17.         avail [λ(c)]:=false
18.         for each available label j do
19.           λ(r_t):= j
20.           avail [j]:= false
21.           λ(v_t):= k - λ(s_t) - λ(r_t) - λ(r_t-1)
22.           if 0 < λ(v_t) ≤ 2nm + n + 1
              and avail [λ(v_t)] then
23.             avail [λ(v_t)]:= false
24.             jahangir2(t+1)
25.             avail [λ(v_t)]:=true
              end if
26.           avail [j]:=true
            end for
27.         avail [λ(c)]:= true
          end if
28. else
29.   λ(c):=k - ∑_{i=1}^{n} s_i
30.   if 0 < λ(c) ≤ 2nm + n + 1 and avail
      [λ(c)] then
31.     avail [λ(c)]:=false
32.     for each available label j do
33.       λ(r_t):= j
34.       avail [j]:= false
35.       λ(v_t):= k - λ(s_t) - λ(r_t) - λ(r_t-1)
36.       if 0 < λ(v_t) ≤ 2nm + n + 1 and
          avail [λ(v_t)] then
37.         avail [λ(v_t)]:= false
38.         jahangir2(t+1)
39.         avail [λ(v_t)]:=true
          end if
40.       avail [j]:=true
        end for
41.     avail [λ(c)]:= true
    end if
  end if
42. avail [i]:=true
 end for
end if
```

Selanjutnya akan diberikan contoh pembentukan PTSA graf $J_{2,3}$ dengan $k = 29$. Label yang tersedia untuk membentuk PTSA graf ini adalah $\{1,2,\ldots,15\}$ (Gambar 3.34a). Misalkan jari-jari s_1 dilabel dengan 5 (Gambar 3.34b). Lalu,

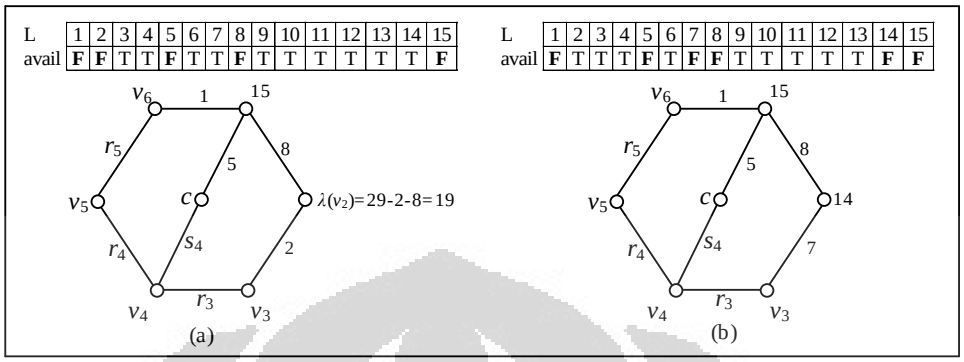
busur r_6 dilabel dengan 1 (Gambar 3.34c). Label yang mungkin untuk r_1 adalah label yang lebih besar dari r_6 (Algoritma 4(i) baris ke-7). Seperti telah dijelaskan sebelumnya, hal ini berguna untuk mencegah kasus refleksi pada PTSA graf $J_{n,m}$. Maka, r_1 akan dilabel dengan 2 (Gambar 3.34d), sehingga label untuk v_1 bisa diperoleh dari $\lambda(v_1) = 29 - 5 - 2 - 1 = 21$ (Gambar 3.34e).



Gambar 3. 33 Contoh penerapan algoritma PTSA graf $J_{n,m}$ dengan $n = 2$, $m = 3$, dan $k = 29$ (Tahap pelabelan simpul v_1)

Karena label 21 tidak tersedia, maka label r_1 akan diganti hingga diperoleh $\lambda(v_1)$ yang mungkin. Label r_1 yang telah digunakan akan tersedia kembali (Algoritma 4(i) baris ke-15). Busur r_1 akan dilabel dengan 8 sehingga diperoleh $\lambda(v_1) = 29 - 5 - 8 - 1 = 15$ (Gambar 3.34f). Proses pelabelan akan berlanjut

dengan memanggil fungsi jahangir2 dengan parameter $t = 2$ (Algoritma 4(i) baris ke-13).

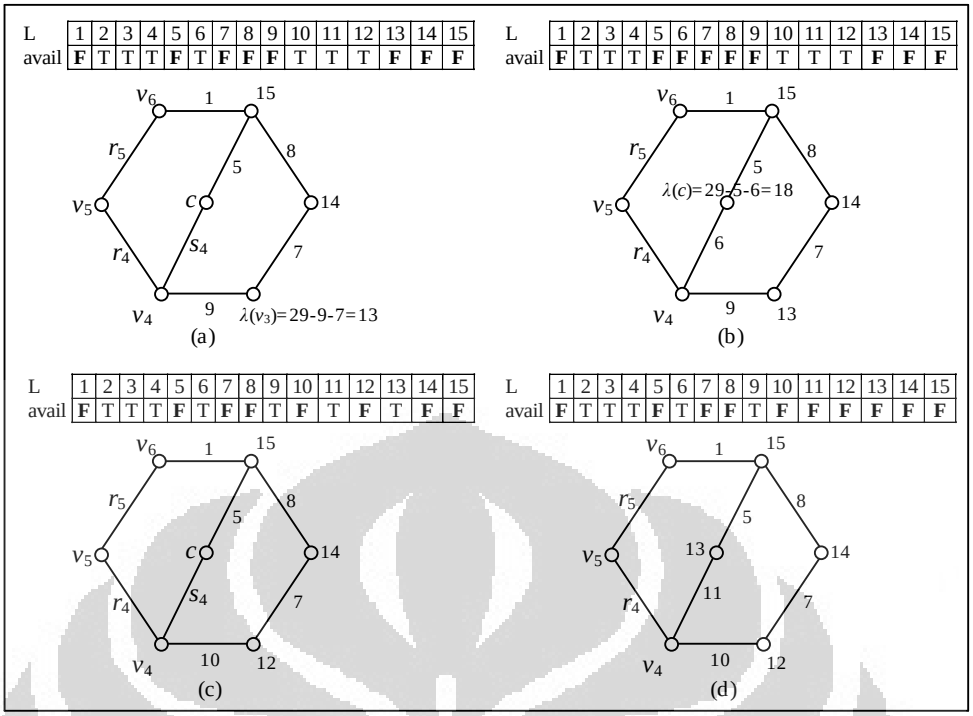


Gambar 3. 34 Contoh penerapan algoritma PTSA graf $J_{n,m}$ dengan $n = 2$, $m = 3$, dan $k = 29$ (Tahap pelabelan simpul v_2)

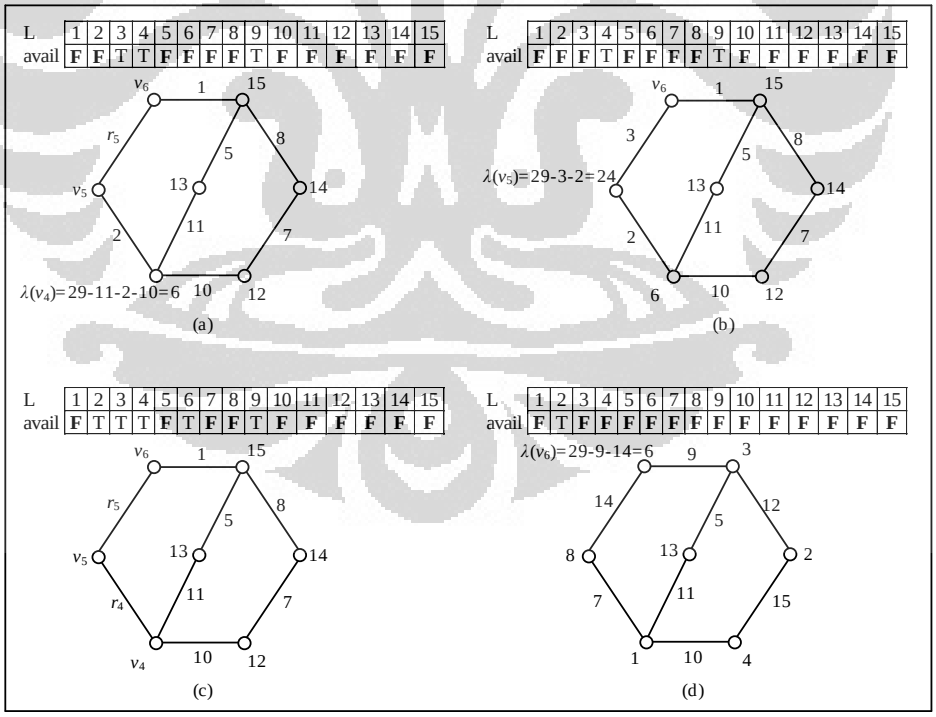
Pada tahap ini hanya akan dilabel satu busur, yaitu r_2. Busur r_2 akan dilabel dengan 2 sehinggga diperoleh $\lambda(v_2) = 29 - 2 - 8 = 19$ (Gambar 3.35a). Karena label yang diperoleh v_2 tidak tersedia, sama seperti pada tahap pelabelan v_1, label dari r_2 akan diganti hingga diperoleh $\lambda(v_2)$ yang mungkin. Label r_2 yang telah digunakan akan tersedia kembali (Algoritma 4(ii) baris ke-12). Busur r_2 akan dilabel dengan 7 sehingga label yang diperoleh untuk v_2 adalah $\lambda(v_2) = 29 - 7 - 8 = 14$ (Gambar 3.35b). Tahap pelabelan dilanjutkan dengan memanggil fungsi jahangir3 dengan parameter $t = 3$ (Algoritma 4(ii) baris ke-10).

Pada fungsi jahangir3 akan diperiksa nilai t (Algoritma 4(iii) baris ke-1 dan 3). Karena nilai t tidak sama dengan 6 dan nilai (t mod 3) $\neq 1$, artinya tahap $t = 3$ merupakan tahap pelabelan simpul pada bagian rim yang tidak terhubung dengan simpul tengah. Maka, fungsi jahangir2 akan dipanggil kembali. Tahap pelabelan simpul v_3 sama seperti simpul v_2. Gambar 3.36a menunjukkan tahap pelabelan simpul ini.

Proses pelabelan dilanjutkan dengan memanggil fungsi jahangir3 dengan $t = 4$. Namun, karena v_4 merupakan simpul terakhir pada bagian rim yang terhubung dengan simpul tengah, maka fungsi jahangir4 akan dipanggil (Algoritma 4(iii) baris ke-2). Pertama, jari-jari s_4 akan dilabel dengan label yang lebih besar dari label s_1 (Algoritma 4(iv) baris ke-6). Hal ini berguna untuk mencegah kasus rotasi. Maka, s_4 akan dilabel dengan 6 (Gambar 3.36b). Berbeda dengan tahap



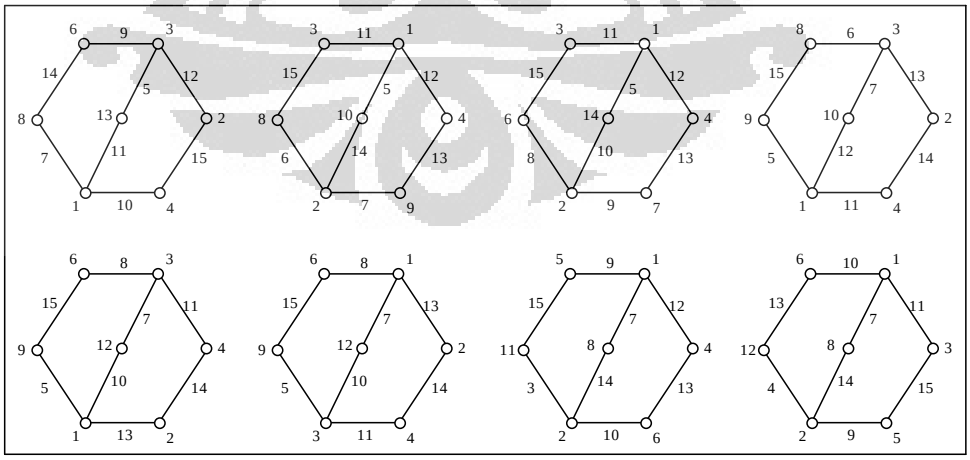
Gambar 3. 35 Contoh penerapan algoritma PTSA graf $J_{n,m}$ dengan n = 2, m = 3, dan k = 29 (Tahap pelabelan simpul v_3 dan c)



Gambar 3. 36 Contoh penerapan algoritma PTSA graf $J_{n,m}$ dengan n = 2, m = 3, dan k = 29 (Tahap pelabelan simpul v_4, v_5, dan v_6)

pelabelan simpul v_1, setelah s_4 dilabel, akan terlebih dahulu ditentukan label untuk simpul c, kemudian r_4 dilabel. Label simpul c diperoleh dari $\lambda(c) = 29 - 6 - 5 = 18$. Karena 18 tidak tersedia, maka label s_4 akan diganti sampai diperoleh $\lambda(c)$ yang mungkin. Label s_4 yang telah digunakan sebelumnya tersedia kembali (Gambar 3.36c). Jari-jari s_4 akan dilabel dengan 11, sehingga diperoleh $\lambda(c) = 29 - 11 - 5 = 13$ (Gambar 3.36d). Setelah label simpul c diperoleh, barulah busur r_4 dilabel. Busur r_4 akan dilabel dengan 2, sehingga diperoleh label untuk v_4 sebesar $\lambda(v_4) = 29 - 11 - 2 - 10 = 6$ (Gambar 3.37a). Karena label ini tersedia, proses pelabelan dilanjutkan dengan memanggil fungsi jahangir2 dengan $t = 5$.

Tahap pelabelan simpul v_5 sama dengan simpul v_2 dan v_3 (Gambar 3.37b). Namun, semua label yang tersedia untuk r_5, {2, 4, 9}, ketiganya tidak menghasilkan $\lambda(v_5)$ yang mungkin. Karena itu, *backtracking* dilakukan. Label v_4 dan r_4 yang telah digunakan akan tersedia kembali (Gambar 3.37c). Mengganti label r_4 dengan label lain yang tersedia juga akan berakhir dengan kegagalan dalam memperoleh label v_5. *Backtracking* akan kembali dilakukan dan proses pelabelan akan dilakukan seperti telah dijelaskan sebelumnya. PTSA yang akan terbentuk bisa terlihat pada Gambar 3.37d. Setelah satu PTSA terbentuk, maka *backtracking* akan dilakukan untuk memperoleh PTSA lain yang tidak isomorfik untuk graf $J_{2,3}$ dengan $k = 29$. PTSA yang terbentuk untuk graf $J_{2,3}$ dengan $k = 29$ ada seba-nyak 8 (delapan) PTSA. Kedelapan PTSA tersebut bisa dilihat pada Gambar 3.38.



Gambar 3. 37 PTSA tidak isomorfik untuk graf $J_{2,3}$ dengan $k = 29$

BAB 4
IMPLEMENTASI DAN SIMULASI

Pada bab sebelumnya telah dibahas mengenai algoritma PTSA untuk graf *friendship*, kipas, dan jahangir yang diperumum. Dalam bab ini, akan dibahas mengenai implementasi dari algoritma-algoritma tersebut yang kemudian disimulasikan untuk mendapatkan semua PTSA yang tidak isomorfik untuk beberapa nilai *n* untuk graf *friendship* dan kipas, serta beberapa nilai *n* dan *m* untuk graf jahangir yang diperumum.

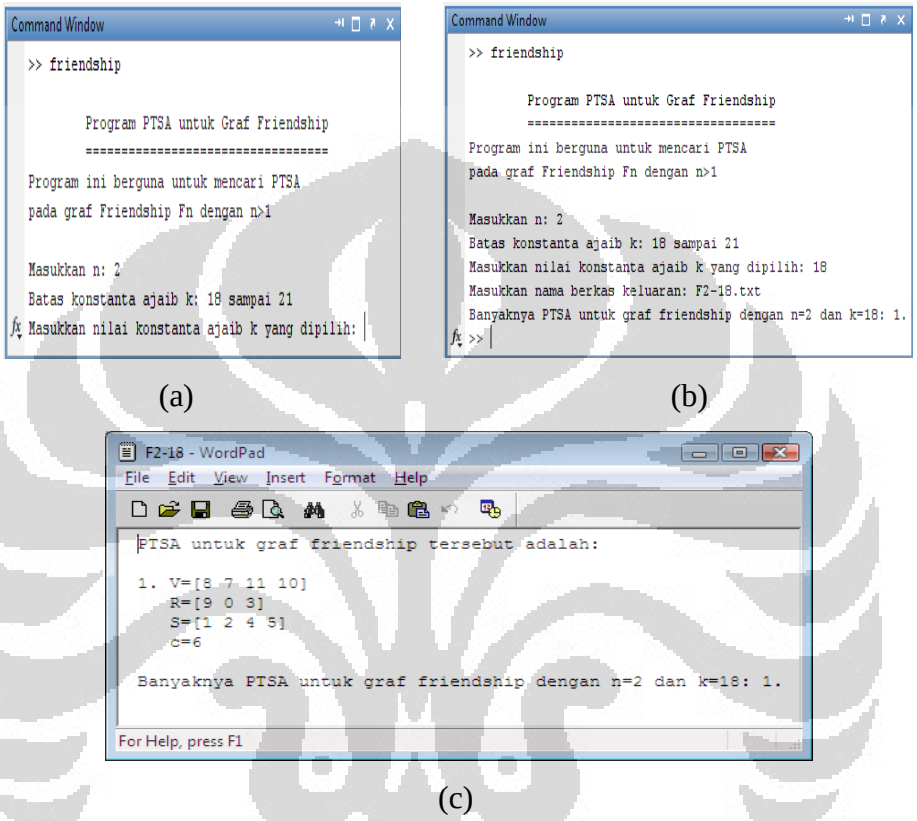
Ketiga algoritma PTSA yang telah dibentuk pada bab sebelumnya, diimplementasikan dengan menggunakan program MATLAB yang kemudian dijalankan pada *PC* dengan *processor Pentium(R) Dual-Core* 2.00 GHz 2.00 GHz dan *RAM* 3035 MB. Cara penggunaan, interpretasi keluaran (*output*), serta simulasi program dari algoritma PTSA untuk graf *friendship*, kipas, dan jahangir yang diperumum masing-masing akan dijelaskan pada Subbab 4.1, 4.2, dan 4.3.

4.1 Implementasi dan Simulasi Algoritma PTSA Graf *Friendship*

Seperti telah dijelaskan pada Bab 3, algoritma PTSA untuk graf *friendship* dibentuk dari tiga fungsi. Dalam implementasinya, program untuk mecari PTSA graf *friendship* juga terdiri dari tiga fungsi utama, yaitu friendship1, friendship2, dan finalizefriendship. Secara keseluruhan ketiga fungsi ini akan dijalankan oleh fungsi friendship. Fungsi ini akan meminta masukan (*input*) berupa nilai *n* dimana nilai *n* ini menyatakan banyaknya segitiga pembentuk graf *friendship*, nilai konstanta ajaib *k*, dan nama berkas keluaran (*file output*) dengan tipe txt atau doc. Setelah nilai *n* dimasukkan, akan muncul batas konstanta ajaib *k* yang mungkin untuk membangun PTSA dari graf tersebut. Lalu, pengguna harus memberikan nilai *k* yang dipilih yang sesuai dengan batasan yang diberikan dan nama berkas keluaran. Setelah ketiga masukkan diberikan, maka dipanggil fungsi friendship1 yang akan melakukan tahap pelabelan v_1. Fungsi friendship1 kemudian akan

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

memanggil fungsi friendship2, dan sebaliknya secara bergantian, untuk melakukan tahap pelabelan v_2 sampai v_{2n-1}. Ketika parameter t mencapai $2n$, maka fungsi finalizefriendship dipanggil untuk menyelesaikan tahap akhir pelabelan. Untuk lebih jelas, program ini dapat dilihat pada Lampiran 1 (*CD*).

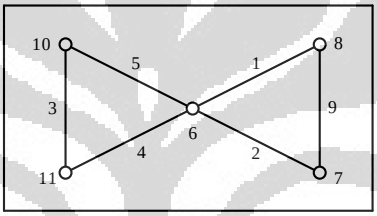


Gambar 4. 1 (a) Tampilan awal pada *Command Window* (b) Tampilan keluaran pada *Command Window* (c) Tampilan keluaran pada berkas

Cara penggunaan dan menginterpretasikan keluaran dari program tersebut adalah sebagai berikut. Pemanggilan program dilakukan dengan mengetik "friendship" pada *Command Window*. Kemudian pengguna harus memasukkan nilai *n* dari graf *friendship* yang ingin dicari PTSAnya. Misalkan nilai *n* yang diberikan adalah 2. Tampilan yang muncul pada *Command Window* bisa dilihat pada Gambar 4.1a. Pada *Command Window* akan tampil batasan nilai *k* yang dapat dipilih untuk membentuk PTSA pada graf F_2. Setelah itu, pengguna harus memasukkan nilai konstanta ajaib yang dipilih serta nama berkas keluaran. Misalkan nilai *k* yang dipilih adalah 18 dan nama berkas keluaran yang dimasukkan adalah F2-18.txt. Keluaran yang ditampilkan pada *Command Window* bisa dilihat

pada Gambar 4.1b. Dari *Command Window* tersebut bisa diketahui bahwa PTSA yang terbentuk untuk graf F_2 dengan $k = 18$ ada sebanyak satu PTSA. PTSA yang terbentuk bisa dilihat pada berkas keluaran yang sebelumnya telah dipilih. Gambar 4.1c menunjukkan isi dari berkas F2-18.txt.

Pada berkas keluaran, V = [8 7 11 10] menunjukkan label dari $v_1,\ldots,v_4$ secara berurutan, R = [9 0 3] menunjukkan label dari r_1=9 dan r_3=3. S menunjukkan label dari jari-jari $s_1,\ldots,s_4$ secara berurutan, dan c = 6 menunjukkan label dari simpul tengah c. Jika digambarkan, PTSA yang terbentuk akan tampak seperti pada Gambar 4.2. PTSA yang diperoleh ini sama dengan PTSA yang diperoleh pada contoh pembentukan PTSA di Subbab 3.1.



Gambar 4. 2 PTSA graf F_2 dengan k = 18

Hasil simulasi yang dilakukan untuk program PTSA dari graf *friendship* bisa dilihat pada Tabel 4.1. Hasil simulasi dari program ini sebenarnya adalah semua PTSA yang tidak isomorfik yang terbentuk untuk graf *friendship*. Namun, pada tabel hanya disajikan banyak PTSA yang tebentuk. Baris pertama pada tabel menunjukkan jenis graf dengan nilai n tertentu. Kolom k menunjukkan nilai konstanta ajaib yang terkait dengan graf F_n. Cara untuk mendapatkan nilai dari konstanta ajaib k ini telah dibahas pada Bab 3. Kolom #PTSA menunjukkan banyak PTSA yang terbentuk untuk jenis graf dan nilai k yang terkait. Baris terakhir dari setiap tabel menyatakan banyaknya semua PTSA yang mungkin dan tidak isomorfik untuk graf terkait dengan n tertentu.

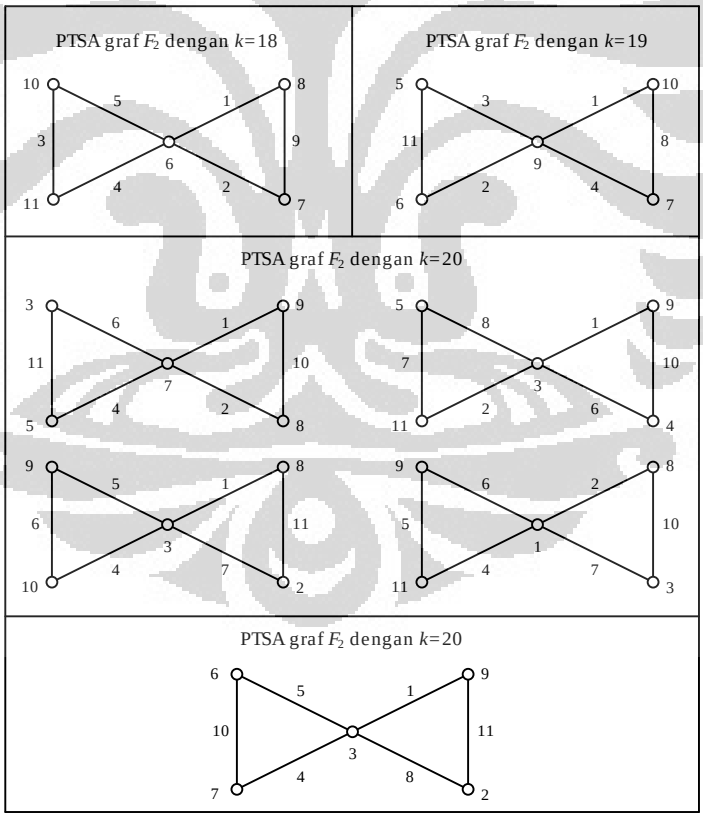
Seperti telah disebutkan pada Bab 2 dan ditunjukkan melalui perhitungan batas konstanta ajaib k pada Subbab 3.1, Graf *friendship* F_n tidak memiliki PTSA untuk $n > 3$. Jadi, banyak PTSA pada Tabel 4.1 merupakan semua PTSA yang mungkin dan tidak isomorfik untuk graf *friendship*. Sehingga hanya terdapat 52

graf *friendship* yang memiliki PTSA. Graf tersebut dan pelabelannya diberikan pada Gambar 4.3 – 4.6.

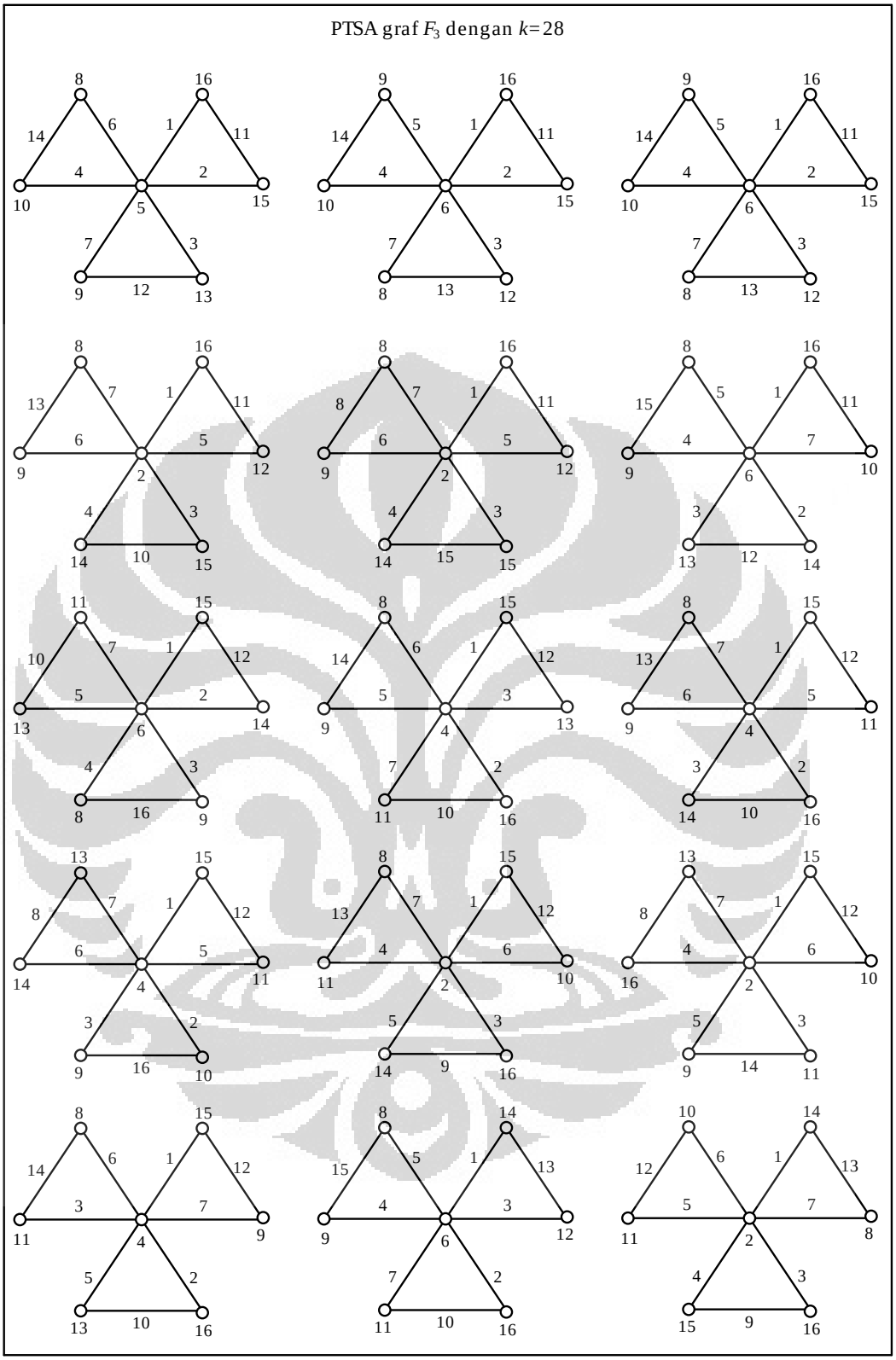
Tabel 4. 1 Banyaknya PTSA tidak isomorfik untuk graf *friendship* F_n dengan $n = 2, 3$

$$\left(\frac{17n^2+9n+1}{2n+1} \le k \le \frac{23n^2+12n+1}{2n+1}\right) \cap \left((2n+1)(n+1) \le k \le \frac{17n+9}{2}\right)$$

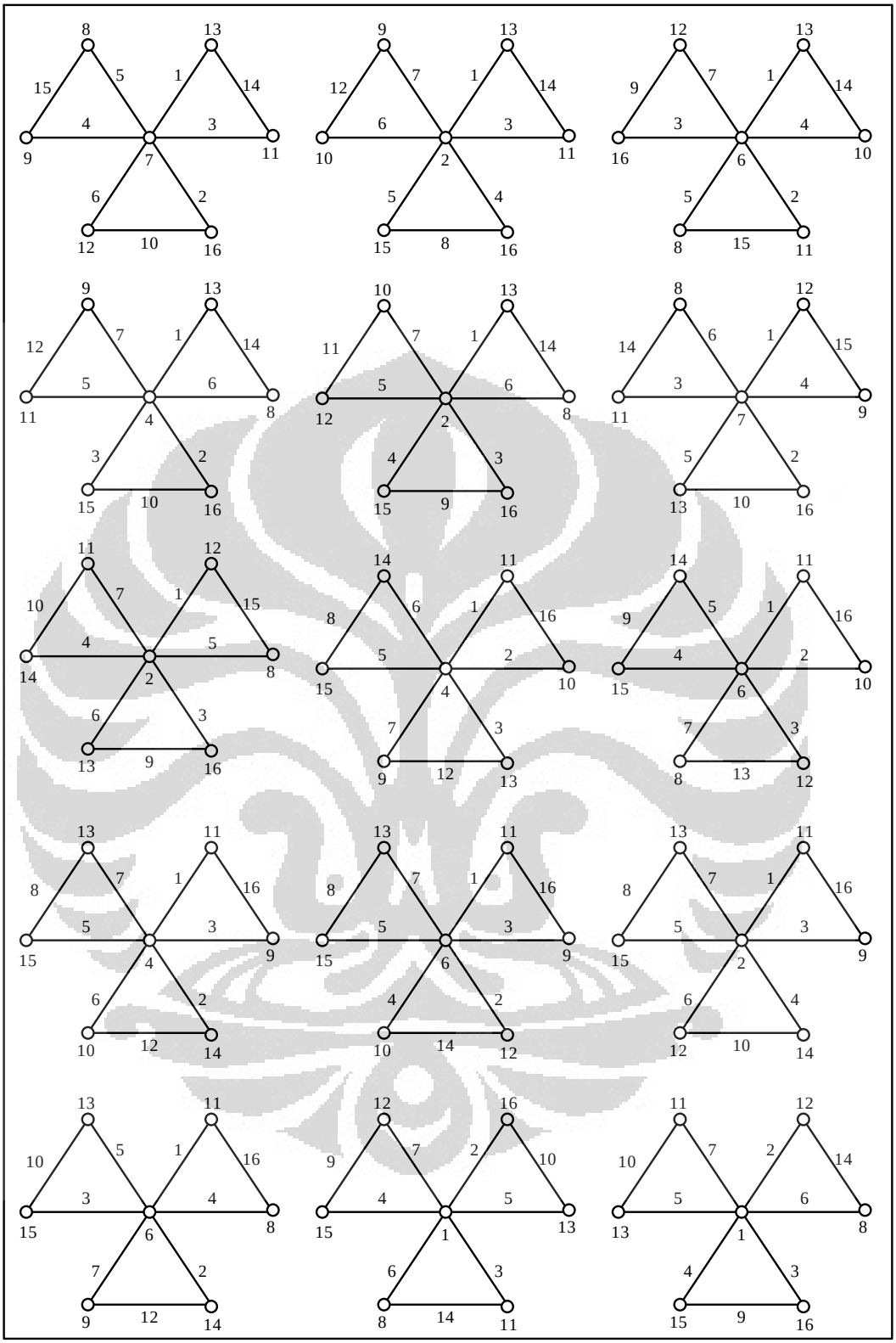
F_2 ($\|V\|$=5)		F_3 ($\|V\|$=7)	
k	#PTSA	k	#PTSA
18	1	28	30
19	1	29	9
20	4	30	6
21	1		
Total	7	Total	45



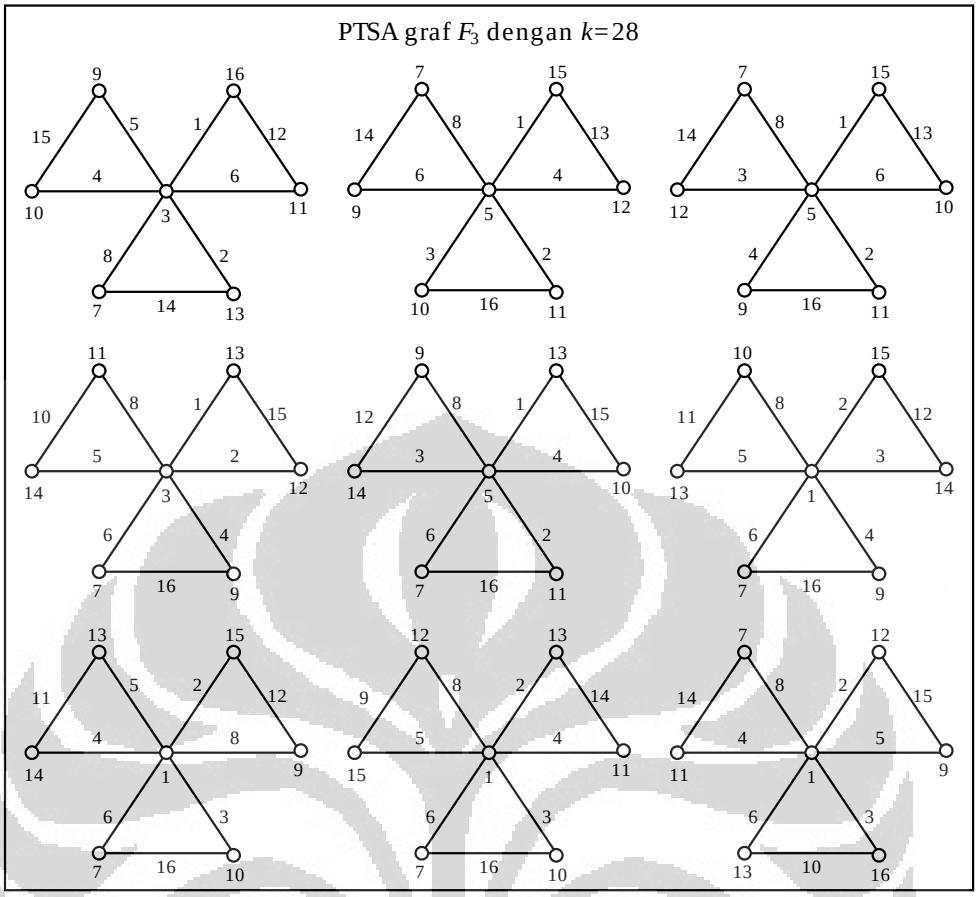
Gambar 4. 3 Seluruh PTSA tidak isomorfik pada graf F_2



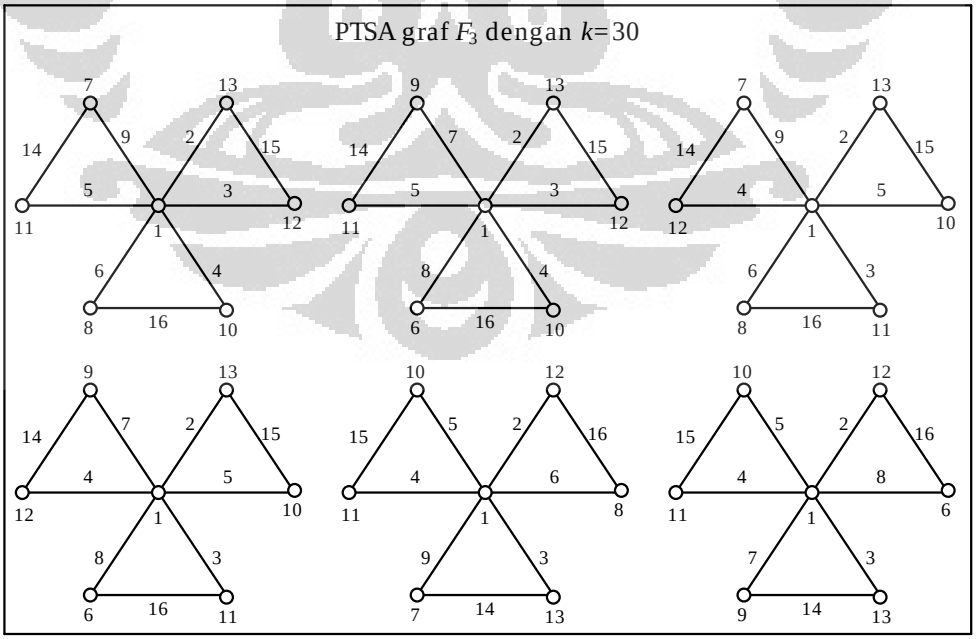
Gambar 4. 4 Seluruh PTSA tidak isomorfik untuk graf F_3 dengan $k = 28$



Gambar 4.4 (Sambungan)



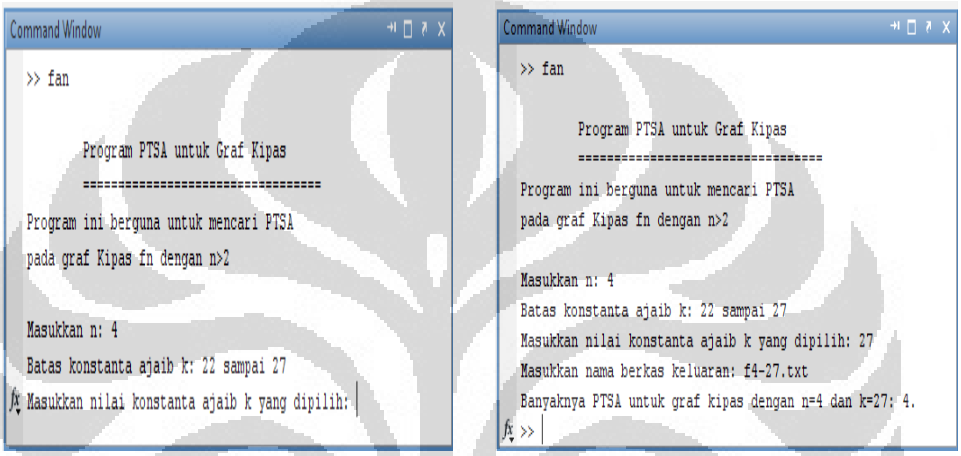
Gambar 4. 5 Seluruh PTSA tidak isomorfik untuk graf F_3 dengan $k = 29$



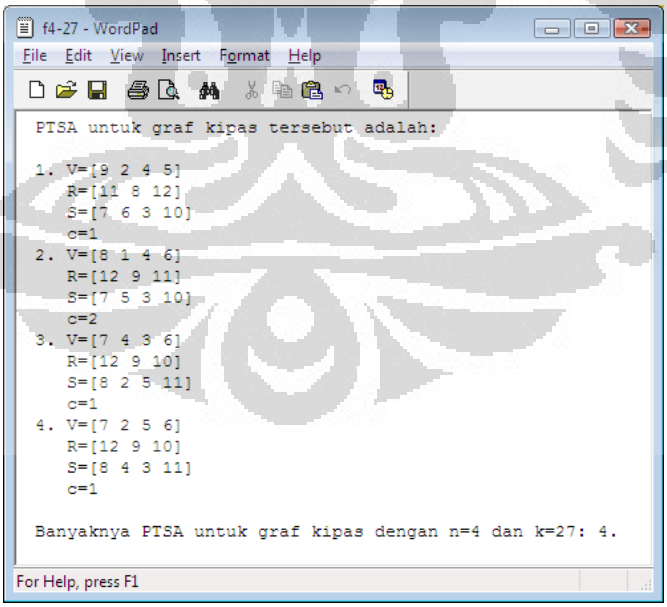
Gambar 4. 6 Seluruh PTSA tidak isomorfik untuk graf F_3 dengan $\boldsymbol{k = 30}$

4.2 Implementasi dan Simulasi Algoritma PTSA Graf Kipas

Algoritma PTSA graf kipas yang telah dibentuk pada Subbab 3.2 diimplementasikan dalam bentuk program yang terdiri dari dua fungsi utama, yaitu fungsi fan1 dan finalizefan. Keseluruhan program akan dijalankan oleh fungsi fan. Fungsi fan akan meminta masukkan dari pengguna berupa nilai *n*, konstanta ajaib *k*, dan nama berkas keluaran.



(a) (b)

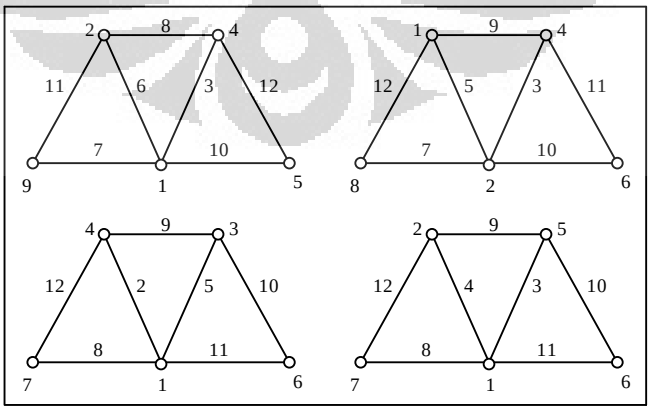


(c)

Gambar 4. 7 (a) Tampilan awal pada *Command Window* (b) Tampilan keluaran pada *Command Window* (c) Tampilan keluaran pada berkas

Berbeda dengan graf *friendship*, nilai *n* pada graf kipas meyatakan banyaknya simpul pada bagian rim. Setelah ketiga masukkan ini dibe-rikan, fungsi fan akan memanggil fungsi fan1 untuk melakukan tahap pelabelan v_1 sampai v_{n-1}. Jadi, saat pertama kali dipanggil, fungsi ini akan melakukan tahap pe-labelan v_1. Setelah itu, fungsi fan akan memanggil dirinya sendiri untuk melaku-kan tahap pelabelan simpul selanjutnya. Ketikan parameter *t* mencapai *n*, maka fungsi finalizefan akan dipanggil untuk melakukan tahap pelabelan simpul v_n dan *c*. Program ini selengkapnya bisa dilihat pada Lampiran 2 (*CD*).

Pada dasarnya, cara penggunaan dan interpretasi keluaran untuk program ini sama dengan program untuk graf *friendship* sebelumnya. Pemanggilan program dilakukan dengan mengetikkan "fan" pada *Command Window*. Kemudian pengguna harus memasukkan nilai *n* yang menunjukkan banyaknya simpul pada bagian rim graf kipas, misalnya dipilih $\boldsymbol{n = 4}$. Setelah nilai *n* yang dipilih dimasukkan, maka pada *Command Window* akan terlihat batas nilai *k* yang mungkin untuk membentuk PTSA graf f_n tersebut. Untuk $\boldsymbol{n = 4}$, batas nilai *k* yang mungkin adalah $\boldsymbol{22 \leq k \leq 27}$ (Gambar 4.7a). Selanjutnya, pengguna harus memasukkan nilai *k* dan nama berkas keluaran, misalkan dipilih $\boldsymbol{k = 27}$ dan nama berkas keluaran f4-27.txt. Setelah ketiga masukkan ini diberikan, maka akan diperoleh PTSA yang tidak isomorfik untuk graf f_4 dengan $\boldsymbol{k = 27}$ sebanyak empat PTSA (Gambar 4.7b). Keempat PTSA ini dapat dilihat pada berkas keluaran f4-27.txt (Gambar 4.7c).



Gambar 4. 8 PTSA tidak isomorfik untuk graf kipas f_4

Interpretasi keluaran dari program ini sama dengan program sebelumnya. *V* dan *R* menunjukan label untuk simpul dan busur pada bagian rim secara berurutan, *S* menunjukkan label untuk jari-jari secara berurutan, dan *c* menunjukkan label dari simpul tengah *c*. Jika digambarkan, keempat PTSA yang terbentuk akan tampak seperti Gambar 4.8. PTSA yang diperoleh ini sesuai dengan PTSA yang dihasilkan pada contoh pembentukan PTSA pada Subbab 3.2.

Tabel 4. 2 Banyaknya PTSA tidak isomorfik untuk graf kipas f_n ($\mathbf{3 \le n \le 7}$)

$\left(\frac{13n^2+n}{2(n+1)} \le k \le \frac{17n^2+3n-2}{2(n+1)}\right) \cap \left(\frac{(n+1)(n+2)}{2} \le k \le 7n - \frac{2}{n}\right)$

f_3		f_4		f_5		f_6		f_7	
k	#PTSA	*k*	#PTSA	*k*	#PTSA	*k*	#PTSA	*k*	#PTSA
15	0	22	25	28	105	34	53	41	2414
16	2	23	59	29	475	35	2073	42	45486
17	3	24	93	30	773	36	9652	43	146377
18	0	25	62	31	1283	37	17946	44	257179
19	1	26	30	32	914	38	17489	45	250746
20	0	27	4	33	473	39	13012	46	185637
				34	86	40	6018	47	67139
						41	674	48	5972
Total	6	Total	273	Total	4109	Total	66917	Total	960950

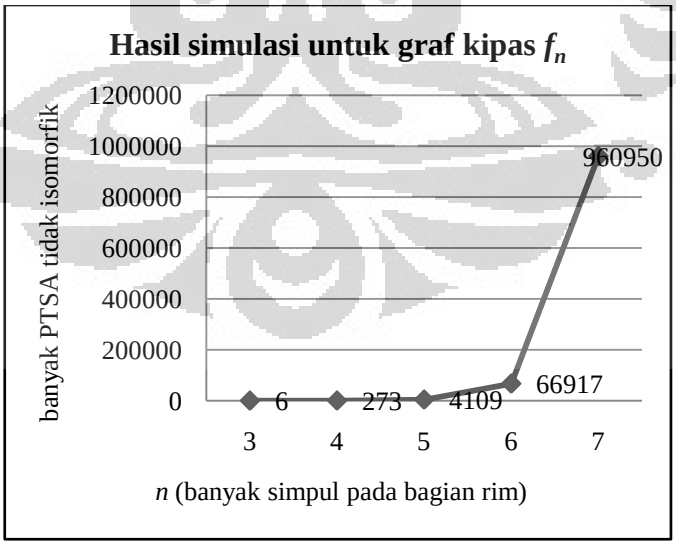
Tidak seperti pada graf *friendship*, simulasi yang dilakukan untuk program PTSA graf kipas belum mencapai seluruh kemungkinan nilai *n*. Seperti telah disebutkan pada Bab 2, graf kipas memiliki PTSA untuk $n \le 10$. Simulasi yang dilakukan untuk program ini telah mencapai seluruh kemungkinan nilai *k* untuk graf f_n dengan $3 \le n \le 7$ dan beberapa nilai *k* untuk graf f_8, tetapi karena keterbatasan waktu simulasi belum dilakukan untuk $n = 9$ dan 10. Hasil simulasi program ini adalah seluruh PTSA yang tidak isomorfik. Namun, karena PTSA yang terbentuk sangat banyak, maka yang akan ditampilkan disini hanyalah banyak PTSA yang terbentuk. Tabel 4.2 menunjukkan banyak PTSA tidak isomorfik yang terbentuk untuk graf f_n dengan $3 \le n \le 7$.

Sementara untuk f_8, dimana terdapat sembilan nilai *k* yang mungkin ($47 \le k \le 55$) untuk membentuk PTSA, simulasi baru dilakukan untuk *k* = 47,48, 49,50, dan 55. Banyaknya PTSA yang terbentuk untuk f_8 dengan *k* tersebut masing-

masing adalah 0, 65240, 734014, 2258821, dan 26620 PTSA. Jadi, total sementara PTSA yang terbentuk untuk f_8 adalah 3084695.

Jika diperhatikan, banyaknya PTSA yang terbentuk bertambah secara signifikan seiring dengan meningkatnya nilai *n*. Dengan kata lain, semakin banyak simpul yang membentuk graf tersebut semakin banyak pula PTSA yang terbentuk untuk graf tersebut. Bertambahnya elemen pembentuk graf berarti banyak label pembentuk PTSA juga akan bertambah. Hal ini menyebabkan banyaknya kemungkinan PTSA untuk graf ini juga bertambah, dan berdasarkan hasil simulasi, PTSA yang terbentuk memang meningkat secara signifikan. Fenomena ini bisa terlihat jelas pada grafik yang menunjukkan hubungan nilai *n* dengan PTSA yang terbentuk (Gambar 4.9).

Pada grafik Gambar 4.9, hubungan antara nilai *n* dari graf f_n dengan PTSA yang terbentuk hanya ditunjukkan untuk $3 \leq n \leq 7$. Untuk $n = 8$ tidak diser-takan, karena total PTSA yang diperoleh masih total sementara. Walaupun demikian, total sementara ini mengindikasikan bahwa banyak PTSA untuk graf f_n dengan $n = 8$ meningkat secara signifikan dibandingkan untuk $n = 7$. Total PTSA graf f_8 yang terbentuk untuk lima nilai k dari sembilan yang mungkin mencapai tiga kali lipatnya total PTSA yang terbentuk untuk f_7.



Gambar 4. 9 Grafik hasil simulasi untuk graf kipas f_n dengan $\mathbf{3 \leq n \leq 7}$

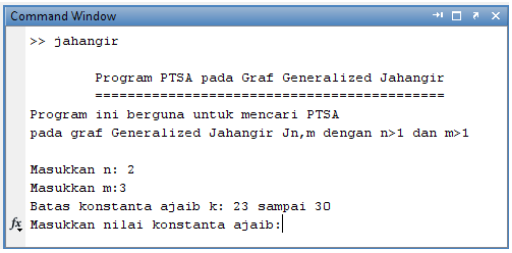
Hal lain yang menarik untuk diperhatikan adalah total PTSA yang terbentuk untuk setiap *k*. Pada Tabel 4.2 terlihat bahwa total PTSA meningkat seiring dengan meningkatnya nilai *k*, lalu menurun ketika telah mencapai separuh dari kemungkinan nilai *k* yang ada. Contohnya untuk graf f_7. Nilai k yang mungkin untuk membentuk PTSA graf ini ada sebanyak delapan, yaitu $41 \leq k \leq 48$. Total PTSA yang terbentuk untuk graf ini mengalami peningkatan ketika $k = 41$ sampai $k = 44$, setelah itu total PTSA yang terbentuk menurun. Berdasarkan perilaku ini, maka bisa diperkirakan total PTSA yang akan terbentuk untuk f_8 adalah sekitar enam juta PTSA.

Total PTSA yang terbentuk untuk graf kipas f_n sejauh ini tentu cukup mengejutkan, karena jauh berbeda dengan total PTSA yang terbentuk untuk graf *friendship* F_n yang hanya mencapai angka 52. Hal ini terjadi karena struktur dari kedua graf tersebut yang memang berbeda. Dengan banyak simpul yang sama, graf kipas memiliki PTSA yang lebih banyak dibandingkan dengan graf *friendship*. Graf *friendship* F_2 memiliki 7 PTSA, sedangkan graf kipas f_4 memiliki 273 PTSA, dimana kedua graf ini memiliki $|V| = 5$. Begitu juga kedua graf tersebut dengan $|V| = 7$, graf F_3 memiliki 45 PTSA, sedangkan f_6 memiliki 66917 PTSA.

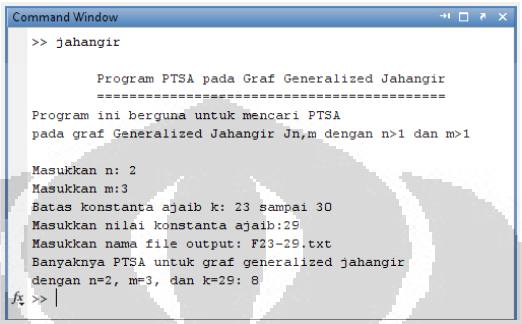
4.3 Implementasi dan Simulasi Algoritma PTSA Graf Jahangir yang Diperumum

Sama seperti dua algoritma sebelumnya yang diimplementasikan dalam bentuk program, algoritma PTSA graf jahangir yang diperumum juga dimplementasikan dalam bentuk program. Program ini terdiri dari empat fungsi utama, yaitu jahangir1, jahangir2, jahangir3, dan finalizejahangir. Keseluruhan program akan dijalankan oleh fungsi jahangir. Fungsi ini akan meminta masukkan dari pengguna berupa nilai *n*, *m*, *k*, dan nama berkas keluaran. Setelah keempat masukan tersebut diberikan, fungsi jahangir1 akan dipanggil. Fungsi jahangir1 akan melakukan tahap pelabelan simpul v_1. Setelah itu, fungsi jahangir2 dipanggil untuk melakukan tahap pelabelan v_2. Jika $m > 2$, maka fungsi jahangir2 akan memanggil dirinya sendiri untuk melakukan tahap pelabelan $v_3,\ldots,v_m$. Selanjut-

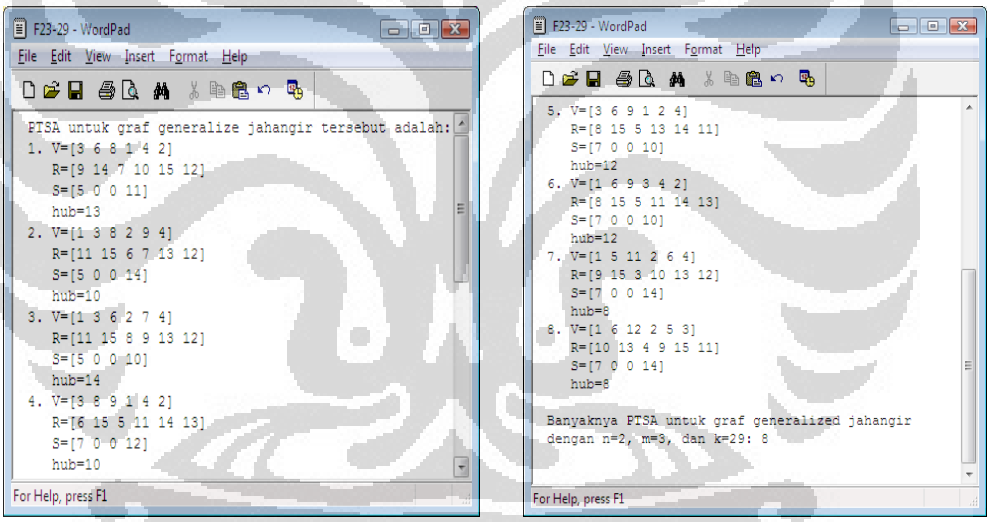
Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010



(a)



(b)



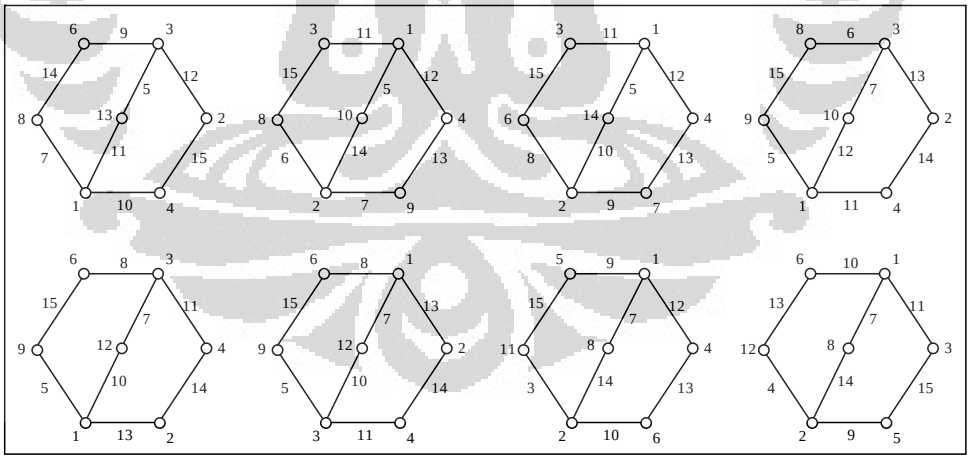
(c)

Gambar 4. 10 (a) Tampilan awal pada *Command Window* (b) Tampilan keluaran pada *Command Window* (c) Tampilan keluaran pada berkas

nya, fungsi jahangir3 akan dipanggil untuk melakukan tahap pelabelan simpul v_{m+1}. Lalu, fungsi jahangir2 akan kembali dipanggil untuk melakukan tahap pelabelan $v_{m+2},\ldots,v_{2m}$. Fungsi jahangir3 kembali dipanggil untuk melakukan tahap pelabelan v_{2m+1}. Begitu seterusnya, hingga ketika $t = (n-1)m + 1$, fungsi yang dipanggil adalah finalizejahangir. Setelah itu, dilanjutkan dengan memanggil

fungsi jahangir2, lalu finalizejahangir untuk melabel v_{nm}. Untuk lebih jelasnya, program ini bisa dilihat pada Lampiran 3 (*CD*).

Serupa dengan kedua program sebelumnya, program ini dipanggil dengan mengetik nama fungsi pertamanya "jahangir" pada *Command Window*. Selanjutnya pengguna perlu memberikan masukkan berupa nilai *n* yang menyatakan banyaknya simpul pada bagian rim yang terhubung dengan simpul *c*, dan *m* yang menunjukkan jarak antar simpul pada bagian rim yang terhubung dengan simpul *c*. Misalnya masukkan yang diberikan adalah $n = 2$ dan $m = 3$. Batas konstanta *k* yang mungkin untuk membentuk PTSA pada graf $J_{2,3}$ ini akan tampil pada *Command Window* setelah dua masukkan tersebut diberikan (Gambar 4.10a). Setelah itu, pengguna harus memasukkan nilai *k* yang dipilih dan nama berkas keluaran. Misalnya dipilih *k* = 29 dan nama berkas keluaran J23-29.txt. Gambar 4.10b menunjukkan tampilan keluaran pada *Command Window*. Sementara Gambar 4.10c menunjukkan tampilan isi dari berkas J23-29.txt. Cara menginterpretasikan keluaran dari program ini tidak berbeda dengan dua program sebelumnya. Jika digambarkan, delapan PTSA yang terbentuk tersebut akan tampak seperti pada Gambar 4.11.



Gambar 4. 11 PTSA tidak isomorfik untuk graf $J_{2,3}$ dengan *k* = 29

Pada Bab 2 telah disebutkan bahwa graf jahangir yang diperumum $J_{n,t+1}$ memiliki PTSA untuk $n = 3$ dan $t = 1$ ($J_{3,2}$) dan tidak memiliki PTSA untuk $n \geq 7t + 11$ dan $t \geq 1$. Namun, karena keterbatasan waktu, simulasi yang

Tabel 4. 3 Hasil simulasi program PTSA untuk graf $J_{n,m}$ dengan $\boldsymbol{n = 2}$ dan $\boldsymbol{2 \le m \le 5}$

$$\frac{5n^2m^2+6n^2m+7nm+2n^2+4n+2}{2(nm+1)} \le k \le \frac{7n^2m^2+8n^2m+9nm+2n^2+6n+2}{2(nm+1)}$$

$J_{2,2}$ (|V|=5)		$J_{2,3}$ (|V|=7)		$J_{2,4}$ (|V|=9)		$J_{2,5}$ (|V|=11)	
k	#PTSA	k	#PTSA	k	#PTSA	k	#PTSA
18	1	23	38	28	449	33	10534
19	2	24	117	29	1088	34	32183
20	1	25	168	30	2712	35	66427
21	0	26	134	31	3851	36	133018
22	0	27	197	32	4721	37	160668
23	0	28	54	33	3302	38	192611
		29	8	34	2180	39	146073
		30	0	35	585	40	103740
				36	109	41	45416
				37	0	42	14150
						43	1454
						44	0
Total	4	Total	716	Total	18997	Total	906274

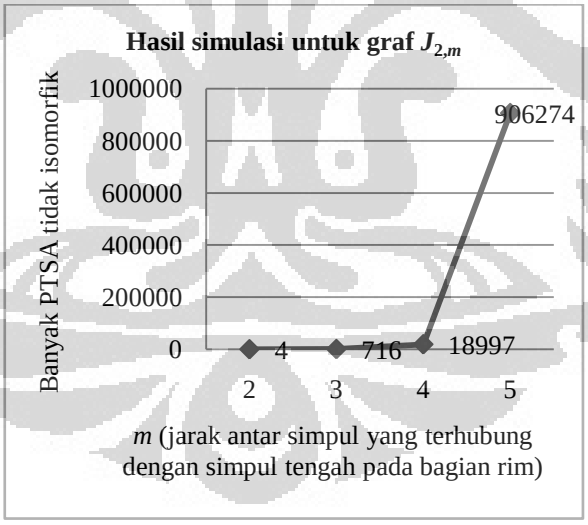
Tabel 4. 4 Hasil simulasi program PTSA untuk graf $J_{n,m}$ dengan $\boldsymbol{n = 3, 4}$ dan $\boldsymbol{m = 2}$ serta $\boldsymbol{n = 3}$ dan $\boldsymbol{m = 3}$

$$\frac{5n^2m^2+6n^2m+7nm+2n^2+4n+2}{2(nm+1)} \le k \le \frac{7n^2m^2+8n^2m+9nm+2n^2+6n+2}{2(nm+1)}$$

$J_{3,2}$ (|V|=7)		$J_{3,3}$ (|V|=10)		$J_{4,2}$ (|V|=9)	
k	#PTSA	k	#PTSA	k	#PTSA
26	29	34	11149	35	3603
27	150	35	37930	36	17175
28	268	36	91194	37	40329
29	465	37	148630	38	97047
30	706	38	198855	39	117608
31	464	39	231573	40	137797
32	228	40	177240	41	101904
33	0	41	116888	42	59248
34	0	42	45109	43	24728
		43	9543	44	2978
		44	403	45	0
		45	0	46	0
Total	2310	Total	1068514	Total	602417

dilakukan untuk mencari PTSA untuk graf $J_{n,m}$ belum mencapai keseluruhan nilai n dan m yang mungkin. Simulasi yang dilakukan baru mencapai $n = 2$ dan $2 \leq m \leq 5$, $n = 3, 4$ dan $m = 2$, serta $n = 3$ dan $m = 3$. Sesuai dengan hasil yang telah diketahui bahwa graf $J_{3,2}$ memiliki PTSA, hasil simulasi menunjukkan hal serupa, dimana PTSA tidak isomorfik yang terbentuk untuk graf ini ada se-banyak 2310 PTSA. Karena PTSA yang dihasilkan cukup banyak sehingga tidak mungkin digambarkan satu persatu, maka disini hanya akan ditampilkan banyak PTSA yang terbentuk untuk setiap nilai *n* dan *m* yang telah disimulasikan. Hasil ini bisa dilihat pada Tabel 4.3 dan 4.4.

Hal yang menarik untuk dicermati, yang berbeda dengan hasil simulasi kedua graf sebelumnya, adalah bahwa untuk setiap nilai *n* dan *m* yang disimulasikan, banyak PTSA yang terbentuk untuk nilai konstanta ajaib *k* terbesar selalu 0 (tidak terdapat PTSA). Berarti, ada kemungkinan batas atas konstanta ajaib *k* untuk graf jahangir yang diperumum ini masih bisa diperketat. Hal ini tentu perlu dibuktikan secara analitik.

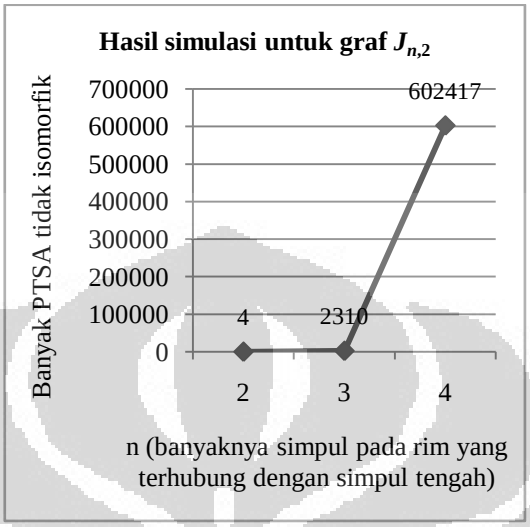


Gambar 4. 12 Grafik hasil simulasi untuk graf $J_{2,m}$ dengan $\mathbf{2 \leq m \leq 5}$

Jika dilihat dari banyaknya PTSA yang terbentuk, hasil simulasi untuk graf jahangir yang diperumum ini menunjukkan perilaku yang sama dengan kedua graf sebelumnya. Banyaknya PTSA yang terbentuk meningkat secara signifikan

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

seiring dengan bertambahnya simpul yang membentuk graf tersebut. Untuk lebih jelasnya, bisa dilihat pada grafik Gambar 4.12-4.13.



Gambar 4. 13 Grafik hasil simulasi untuk graf $J_{n,2}$ dengan $\mathbf{2 \leq n \leq 4}$

Grafik yang disajikan pada pada Gambar 4.12 menunjukkan hasil simulasi untuk graf $J_{2,m}$ dengan $2 \leq m \leq 5$. Sementara grafik pada Gambar 4.13 menunjukkan hasil simulasi untuk graf $J_{n,2}$ dengan $2 \leq n \leq 4$. Hasil simulasi untuk graf $J_{3,m}$ tidak ditampilkan dalam bentuk grafik. Namun, peningkatan banyaknya PTSA yang terbentuk bisa terlihat jelas pada Tabel 4.4. PTSA yang terbentuk untuk graf $J_{3,2}$ adalah sebanyak 2310 PTSA, sedangkan untuk graf $J_{3,3}$ ada sebanyak 1068514 PTSA.

Serupa dengan kedua graf sebelumnya, PTSA graf $J_{n,m}$ yang terbentuk untuk setiap nilai konstanta ajaib *k* yang mungkin meningkat seiring dengan bertambahnya nilai *k*, hingga ketika mencapai separuh dari banyaknya nilai *k* yang mungkin, PTSA yang terbentuk mengalami penurunan. Contohnya pada graf $J_{3,3}$, banyak nilai *k* yang mungkin untuk membentuk PTSA graf ini adalah 12, yaitu $34 \leq k \leq 45$. PTSA yang terbentuk mengalami peningkatan saat $34 \leq k \leq 39$, lalu menurun ketika $40 \leq k \leq 45$.

BAB 5
KESIMPULAN

Dalam skripsi ini telah dibangun algoritma-algoritma PTSA untuk kelas graf yang berkaitan dengan graf roda, yaitu graf friendship F_n, graf kipas f_n, dan graf jahangir yang diperumum $J_{n,m}$. Seluruh PTSA yang tidak isomorfik dari masing-masing kelas graf bisa diperoleh dengan menggunakan algoritma-algoritma tersebut untuk sembarang n dan m dan nilai k yang diberikan.

Ketiga algoritma tersebut diimplementasikan dalam bentuk program yang kemudian disimulasikan. Pada dasarnya, program secara umum berlaku untuk sembarang nilai n dan m. Namun, karena keterbatasan waktu simulasi dilakukan hanya untuk nilai n dan m tertentu untuk graf kipas dan jahangir yang diperumum. Sementara untuk graf *friendship*, simulasi telah dilakukan untuk semua nilai n yang mungkin. Hasil yang diperoleh disajikan pada Tabel 5.1.

Tabel 5. 1 Hasil yang yang diketahui dan hasil yang diperoleh

Hasil yang telah diketahui	Hasil yang diperoleh
Graf *friendship* F_n tidak memiliki PTSA untuk $n > 3$ (MacDougall *et al*., 2002)	- Algoritma PTSA graf *friendship* - Implementasi algoritma dalam bentuk program - Hasil simulasi program berupa seluruh PTSA yang tidak isomorfik
Graf kipas f_n tidak memiliki PTSA untuk $n > 10$ (MacDougall *et al*., 2002)	- Algoritma PTSA graf kipas - Implementasi algoritma dalam bentuk program - Hasil simulasi program berupa PTSA yang tidak isomorfik untuk setiap nilai k yang mungkin dari graf fn dengan $3 \leq n \leq 7$ dan $k = 47,48,49,50,55$ dari graf f_n dengan $n = 8$
Graf jahangir yang diperumum $J_{n,t+1}$ memiliki PTSA untuk $n = 3$ dan $t = 1$ dan tidak memiliki PTSA untuk $n \geq 7t + 11$ dan $t \geq 1$ (Gallian, 2009)	- Algoritma PTSA graf $J_{n,m}$ - Implementasi dalam bentuk program - Hasil simulasi program berupa PTSA tidak isomorfik untuk setiap nilai k yang mungkin dari graf $J_{n,m}$ dengan $n = 2$ dan $2 \leq m \leq 5$, $n = 3,4$ dan $m = 2$, dan $n = 3$ dan $m = 3$

Algoritma pelabelan..., Budi Utami, FMIPA UI, 2010

Berdasarkan hasil simulasi yang diperoleh bisa disimpulkan hal berikut:

- Banyaknya PTSA yang terbentuk untuk ketiga graf tersebut meningkat secara signifikan seiring dengan bertambahnya nilai n (dan m untuk graf jahangir yang diperumum).
- PTSA yang terbentuk untuk setiap nilai k yang mungkin dari ketiga graf tersebut dengan nilai n (dan m untuk graf jahangir yang diperumum) meningkat seiring dengan meningkatnya nilai k, lalu menurun ketika mencapai separuh dari banyaknya nilai k yang mungkin.

DAFTAR PUSTAKA

Baker, A., & Sawada, J. (2008). Magic Labelings on Cycles and Wheels. *COCOA LNCS* , 361-373.

Gallian, J. A. (2009). A Dynamic Survey of Graph Labeling. *The Electronic Journal of Combinatorics 5* , #DS6.

Graph Isomorphism. (n.d.). Retrieved Februari 22, 2010, from http://www.cs.uu.nl/docs/vakken/an/

Hartsfield, N., & Ringel, G. (2003). *Pearls in Graph Theory*. New York: Dover Publications.

MacDougall, J. A., Miller, M., & Wallis, W. (2002). Vertex-Magic Total Labelings of Wheels and Related Graphs. *Util. Math. 62* , 175-183.

MacDougall, J. A., Miller, M., Slamin, & Wallis, W. (2002). Vertex-Magic Total Labelings of Graphs. *Util. Math 61* , 3-21.