UNIVERSITAS INDONESIA

ALGORITMA PELABELAN GRACEFUL PADA GRAF LINTASAN, GRAF MATAHARI DAN GRAF ULAR k-C_4

SKRIPSI

DHITA PUSPITASARI
0606067345

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

UNIVERSITAS INDONESIA

ALGORITMA PELABELAN GRACEFUL PADA GRAF LINTASAN, GRAF MATAHARI DAN GRAF ULAR k-C_4

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains

DHITA PUSPITASARI
0606067345

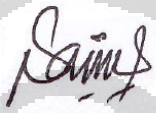
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI SARJANA MATEMATIKA
DEPOK
DESEMBER 2010

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	: Dhita Puspitasari
NPM	: 0606067345
Tanda Tangan	: 
Tanggal	: 21 Desember 2010

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh

Nama	: Dhita Puspitasari
NPM	: 0606067345
Program Studi	: Sarjana Matematika
Judul Skripsi	: Algoritma Pelabelan Graceful pada Graf Lintasan, Graf Matahari dan Graf Ular k-C_4

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi S1 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	: Dra. Denny R. Silaban, M.Kom	( )
Penguji	: Dra. Ida Fitrhriani, MSi	( )
Penguji	: Dhian Widya, M.Kom	( )

Ditetapkan di : Depok
Tanggal : 21 Desember 2010

KATA PENGANTAR

Alhamdulillah, puji syukur kepada Allah swt. atas semua rahmat dan karunia yang telah Dia berikan sehingga penulis dapat menyelesaikan tugas akhir ini. Penulis sadar bahwa penyelesaian tugas akhir ini tidak terlepas dari bantuan dan dukungan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin mengucapkan terima kasih kepada pihak-pihak yang telah berjasa dalam penulisan tugas akhir ini maupun selama penulis kuliah. Ucapan terima kasih terhatur kepada:

(1) Dra. Denny Riama Silaban, M.Kom selaku pembimbing I dan Dra. Siti Aminah, M.Kom selaku pembimbing II yang telah banyak meluangkan waktu dan pikiran serta memberikan masukan-masukan untuk penulis dalam menyelesaikan tugas akhir ini.
(2) Seluruh staf pengajar di Matematika UI atas ilmu pengetahuan yang telah kalian berikan.
(3) Seluruh karyawan di departemen Matematika UI atas bantuan yang telah diberikan.
(4) Orang tua dan keluarga penulis yang selalu memberikan doa, semangat, dan dukungan bagi penulis.
(5) Syukron Fahrurrozi berserta keluarganya yang telah menemani, membantu, dan memberikan semangat serta doa untuk penulis.
(6) Kiki, Annisa, Rachmanita dan Nadya yang selalu bersama-sama penulis selama menjalani kuliah dan juga diluar perkuliahan.
(7) Widi, Widita, Stefi, Tasya, Syafira, yang telah berjuang bersama penulis untuk dapat menyelesaikan skripsi.
(8) Alfa, Milla, Mella, dan Tami yang telah lulus terlebih dahulu dan memberikan inspirasi kepada penulis.
(9) Seluruh teman-teman angkatan 2006 yang telah memberikan pengalaman per-kuliahan yang tak terlupakan.
(10) Teman-teman angkatan 2004, 2005, 2007, dan 2008 yang membantu penulis saat menjalani perkuliahan dan saat menulis skripsi

Penulis juga ingin mengucapkan terima kasih kepada seluruh pihak yang tidak dapat disebutkan satu per satu, yang telah membantu dalam penyusunan skripsi ini. Akhir kata, penulis mohon maaf jika terdapat kesalahan atau kekurangan dalam skripsi ini. Penulis berharap semoga skripsi ini bermanfaat bagi pembaca.

Penulis

2010

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

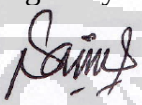
Nama	: Dhita Puspitasari
NPM	: 0606067345
Program Studi	: Sarjana Matematika
Departemen	: Matematika
Fakultas	: Matematika dan Ilmu Pengetahuan Alam
Jenis karya	: Skripsi

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 Graceful pada Graf lintasan, Graf Matahari dan Graf Ular k-C_4

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 : 21 Desember 2010
Yang menyatakan



(Dhita Puspitasari)

ABSTRAK

Nama : Dhita Puspitasari
Program Studi : Matematika
Judul : Algoritma Pelabelan Graceful Pada Graf Lintasan, Graf Matahari, dan Graf Ular k-C_4

Misalkan G adalah graf dengan himpunan simpul V dan himpunan busur E, dimana $|V(G)|$ dan $|E(G)|$ menyatakan banyaknya simpul dan busur pada G. Suatu pemetaan $f : V \rightarrow \{0, 1 , \ldots, |E|\}$ disebut pelabelan graceful jika f merupakan fungsi injektif yang menginduksi fungsi bijektif g, $g(uv) = |f(u) – f(v)|$, dimana uv merupakan sebuah busur yang mempunyai titik ujung simpul u dan v, $g : E \rightarrow \{1, 2 , \ldots, |E|\}$. Dalam skripsi ini diberikan algoritma untuk menghasilkan semua pelabelan graceful yang tidak isomorfik pada graf lintasan P_n, graf matahari $C_n \odot \bar{K}_1$ dan graf ular k-C_4 yang mungkin. Algoritma-algoritma ini kemudian diimplementasikan dalam program. Diberikan juga simulasi banyak pelabelan graceful mungkin sampai nilai n atau k tertentu.

Kata Kunci : Pelabelan graceful, graf lintasan, graf matahari, graf ular k-C_4.
xiii+58 halaman : 49 gambar; 4 tabel
Daftar Pustaka : 12 (1966-2010)

ABSTRACT

Name : Dhita Puspitasari
Study Program : Mathematics
Title : An Graceful Labeling Algorithms on Path, Sun, and k-C_4 Snake Graphs

Let G be a graph with vertex set V and edge set E, where $|V(G)|$ and $|E(G)|$ be the number of vertices and the number of edges of G. A injection $f: V \rightarrow \{0, 1, ..., |E|\}$ is called a graceful labeling if injective function f induce bijective function g, $g(uv) = |f(u)-f(v)|$, where uv is an edge which has end point vertices u and v, $g: E \rightarrow \{1, 2, ..., |E|\}$. This *skripsi* explains graceful labeling algorithms on path graph P_n, sun graph $C_n \odot \bar{K}_1$ and k-C_4 snake graph. The algorithms generate all non isomorphic labelings on those graphs. The implementation of the algorithms and simulation for certain values of n or k are also given in this *skripsi*.

Keywords : Graceful labeling, path graph, Sun graph, k-C_4 Snake graph.
xiii+58 pages : 49 pictures; 4 tables
Bibliography : 12 (1966-2010)

DAFTAR ISI

HALAMAN PERNYATAAN ORISINALITAS iii
HALAMAN PENGESAHAN iv
KATA PENGANTAR v
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI vii
ABSTRAK viii
ABSTRACT ix
DAFTAR ISI x
DAFTAR TABEL xi
DAFTAR GAMBAR xii
1. PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Perumusan Masalah dan Ruang Lingkup 3
1.3 Jenis Penelitian dan Metode yang Digunakan 3
1.4 Tujuan Penelitian 4
2. LANDASAN TEORI 5
2.1 Definisi dan Istilah dalam Teori Graf 5
2.2 Jenis-jenis Graf 8
2.3 Pelabelan Graf 10
2.4 Hasil yang Telah Diketahui 12
3. ALGORITMA PELABELAN GRACEFUL UNTUK GRAF LINTASAN, GRAF MATAHARI, DAN GRAF ULAR *k*-C_4 13
3.1 Algoritma Pelabelan Graceful untuk Graf Lintasan 14
3.2 Algoritma Pelabelan Graceful untuk Graf Matahari 23
3.3 Algoritma Pelabelan Graceful untuk Graf Ular *k*-C_4 34
4. IMPLEMENTASI DAN SIMULASI 45
4.1 Implementasi dan Simulasi Algoritma Pelabelan Graceful pada Graf Lintasan 45
4.2 Implementasi dan Simulasi Algoritma Pelabelan Graceful pada Graf Matahari 49
4.3 Implementasi dan Simulasi Algoritma Pelabelan Graceful pada Graf Ular *k*-C_4 53
5. KESIMPULAN 57
DAFTAR PUSTAKA 58

DAFTAR TABEL

Tabel 3.1 Pelabelan graceful yang tidak isomorfik pada graf $C_3 \odot K_1$ 34

Tabel 4.1 Banyaknya pelabelan graceful tidak isomorfik pada graf lintasan P_n untuk n = 2 s.d. n = 16..... 48
Tabel 4.2 Banyaknya pelabelan graceful tidak isomorfik pada graf matahari untuk n = 3 s.d. n = 8..... 52
Tabel 4.3 Banyaknya pelabelan graceful tidak isomorfik pada graf ular k-C_4 untuk n = 2 s.d. n = 4..... 56

DAFTAR GAMBAR

Gambar 2.1 Contoh graf *G* 6
Gambar 2.2 Graf *G* dan *G*' yang isomorfik 7
Gambar 2.3 Contoh graf P_5 8
Gambar 2.4 Contoh graf C_5 8
Gambar 2.5 Contoh graf $C_5 \odot K_1$ 9
Gambar 2.6 contoh graf 4-C_4 9
Gambar 2.7 Contoh pelabelan graceful pada graf dengan |E| = |V| - 1(a) dan graf dengan |E| > |V| - 1(b) 10
Gambar 2.8 Pelabelan isomorfik kasus rotasi 11
Gambar 2.9 Pelabelan isomorfik kasus refleksi 12

Gambar 3.1 Graf lintasan P_n 14
Gambar 3.2 Ilustrasi pembentukan pelabelan graceful pada graf P_n 15
Gambar 3.3 Pelabelan graceful pada graf P_5 yang isomorfik 17
Gambar 3.4 Proses pembentukan pelabelan graceful pada graf P_4 (1) 19
Gambar 3.5 Proses pembentukan pelabelan graceful pada graf P_4 (2) 19
Gambar 3.6 Proses pembentukan pelabelan graceful pada graf P_4 (3) 20
Gambar 3.7 Proses pembentukan pelabelan graceful pada graf P_4 (4) 21
Gambar 3.8 Proses pembentukan pelabelan graceful pada graf P_4 (5) 22
Gambar 3.9 Graf matahari $C_n \odot K_1$ 23
Gambar 3.10 Ilustrasi pembentukan pelabelan graceful pada graf $C_n \odot K_1$ 24
Gambar 3.11 Pelabelan graceful $C_4 \odot K_1$ yang isomorfik dengan kasus rotasi .. 26
Gambar 3.12 Pelabelan graceful $C_4 \odot K_1$ yang isomorfik dengan kasus refleksi 27
Gambar 3.13 Proses pembentukan pelabelan graceful pada graf $C_3 \odot K_1$ (1) 28
Gambar 3.14 Proses pembentukan pelabelan graceful pada graf $C_3 \odot K_1$ (2) 29
Gambar 3.15 Proses pembentukan pelabelan graceful pada graf $C_3 \odot K_1$ (3) 30
Gambar 3.16 Proses pembentukan pelabelan graceful pada graf $C_3 \odot K_1$ (4) 31
Gambar 3.17 Proses pembentukan pelabelan graceful pada graf $C_3 \odot K_1$ (5) 32
Gambar 3.18 Proses pembentukan pelabelan graceful pada graf $C_3 \odot K_1$ (6) 33
Gambar 3.19 Graf Ular 4-C_4 35
Gambar 3.20 Ilustrasi pembentukan pelabelan graceful pada graf ular *k*-C_4 36
Gambar 3.21 Pelabelan graceful ular *k*-C_4 yang isomorfik dengan kasus rotasi .. 38
Gambar 3.22 Pelabelan graceful ular *k*-C_4 yang isomorfik dengan kasus refleksi 38
Gambar 3.23 Proses pembentukan pelabelan graceful pada graf ular 2-C_4 (1) 40
Gambar 3.24 Proses pembentukan pelabelan graceful pada graf ular 2-C_4 (2) 41
Gambar 3.25 Proses pembentukan pelabelan graceful pada graf ular 2-C_4 (3) 43
Gambar 3.26 Proses pembentukan pelabelan graceful pada graf ular 2-C_4 (4) 44

Gambar 4.1 Tampilan awal *command window* pada program pelabelan graceful graf lintasan 46
Gambar 4.2 Tampilan *command window* pada program pelabelan graceful pada graf P_6 46

Gambar 4.3 Isi dari berkas keluaran 6p.txt 47
Gambar 4.4 Representasi visual peleblan graceful pada graf P_6 47
Gambar 4.5 Grafik pertambahan banyaknya pelabelan graceful tidak isomorfik pada graf lintasan P_n untuk n = 2 s.d. n = 16 48
Gambar 4.6 Tampilan awal *command window* pada program pelabelan graceful pada graf matahari..... 50
Gambar 4.7 Tampilan *command window* pada program pelabelan graceful pada graf matahari dengan n = 3 50
Gambar 4.8 Isi dari berkas keluaran 3s.txt..... 51
Gambar 4.9 Representasi visual pelabelan graceful pada graf matahari, n = 3 52
Gambar 4.10 Grafik pertambahan banyaknya pelabelan graceful tidak isomorfik pada graf matahari untuk n = 3 s.d. n = 8..... 53
Gambar 4.11 Tampilan awal *command window* pada program pelabelan graceful pada graf ular k-C_4 54
Gambar 4.12 Tampilan *command window* pada program pelabelan graceful pada graf ular 2-C_4..... 54
Gambar 4.13 Isi dari berkas keluaran 2s.txt..... 55
Gambar 4.14 Grafik pertambahan banyaknya pelabelan graceful tidak isomorfik pada graf ular k-C_4 untuk k = 2 s.d. k = 4..... 56

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Suatu graf merupakan gambaran abstrak dari sekumpulan obyek dimana pasangan obyek-obyek tersebut dihubungkan oleh suatu penghubung. Dalam graf sekumpulan obyek-obyek disebut simpul, sedangkan penghubung disebut busur. Graf $G = (V, E)$ adalah gabungan himpunan tidak kosong simpul, $V(G)$, dan himpunan busur, $E(G)$, sedemikian sehingga titik ujung semua busur di $E(G)$ merupakan anggota $V(G)$. Sedangkan banyaknya anggota himpunan simpul dan anggota himpunan busur masing-masing dinotasikan dengan $|V|$ dan $|E|$.

Kelas graf yang akan dibahas dalam skripsi ini adalah graf lintasan, matahari, dan ular k-C_4. Graf *lintasan*, P_n, adalah graf yang memiliki n simpul dan memiliki himpunan busur v_1v_2, v_2v_3, ..., $v_{n-1}v_n$, apabila ditambahkan busur v_1v_n maka graf tersebut menjadi graf *lingkaran*, C_n. Graf *matahari*, $C_n \odot \overline{K}_1$ adalah graf yang diperoleh dari graf lingkaran C_n dengan menambahkan satu simpul berderajat satu di setiap simpul graf C_n. Graf *ular* k-C_4 adalah graf yang diperoleh dari graf lintasan P_{2k+1}, yang memiliki simpul v_1, v_2, ..., v_{2k+1}, dengan menghubungkan simpul v_i dan v_{i+2} ke suatu simpul baru w_i, $i = 1, 3, ..., 2k-1$. Graf ular k-C_4 dapat pula dipandang sebagai rangkaian k graf C_4.

Pelabelan suatu graf adalah pemberian label (biasanya merupakan bilangan asli) pada simpul-simpul, atau busur-busur, atau keduanya pada suatu graf. Dalam skripsi ini akan dibahas tentang pelabelan graceful, pengertian dari *pelabelan graceful* pada graf dengan $|E|$ busur adalah memberikan label pada setiap simpul dengan bilangan 0, 1, 2, ..., $|E|$, yang menginduksi adanya pelabelan busur dengan bilangan 1, 2, 3, ..., $|E|$, dimana label setiap busur merupakan selisih positif dari label kedua simpul yang dihubungkannya. Graf yang dapat dilabelkan dengan pelabelan graceful disebut dengan *graf graceful*.

Pelabelan graceful merupakan pemetaan injektif. Untuk graf dengan $|E|=|V|-1$, bilangan 0, 1, 2, ..., $|E|$ akan dipakai seluruhnya untuk melabelkan

simpul pada graf tersebut. Sedangkan untuk graf yang memiliki busur lebih banyak dari $|V|$-1, ada sebagian dari bilangan 0, 1, 2, ..., $|E|$ tersebut yang tidak dipakai untuk melabelkan simpul pada graf.

Berdasarkan Golomb (1972) nama pelabelan *graceful* diberikan oleh Solomon W. Golomb, namun sebenarnya pelabelan graceful pertama kali diperkenalkan oleh Rosa (1967) dalam tulisannya tentang pelabelan graf dengan nama *β-valuation*, istilah pelabelan graceful tidak pernah dipakai sampai Golomb mempelajari pelabelan tersebut beberapa tahun kemudian.

Menurut Bloom & Golomb (1977) aplikasi dari pelabelan graceful cukup luas. Dalam teori coding pelabelan graceful berguna untuk mendesain kode *radar-type* yang bagus, himpunan kode yang selaras dan kode yang konvolusional dengan autokorelasi yang optimal. Selain itu dapat juga diterapkan untuk menentukan ambiguitas di X-ray kristalografi analisis, merancang *addressing* dalam jaringan komunikasi, dan menentukan susunan sirkuit yang optimal.

Selama 30 tahun terakhir ini diperkirakan sekitar 200 tulisan mengenai pelabelan graceful telah diterbitkan. Sebagian besar tulisan yang telah diterbitkan membahas pelabelan graceful secara teori, dan hanya fokus pada kelas graf tertentu. Tulisan-tulisan tersebut seringkali memberikan sejumlah argumen dengan menyediakan formula tertentu untuk melabelkan secara graceful suatu kelas graf tertentu, ataupun membuktikan bahwa suatu kelas graf tertentu tidak mungkin dapat dilabelkan secara graceful. Sejumlah kelas graf yang telah dinyatakan graceful adalah graf lintasan P_n dan graf lingkaran C_n hanya untuk $n \equiv 0$ atau 3 (mod 4) oleh Rosa (1967), graf komplit K_n hanya untuk $n \leq 4$ oleh Golomb (1972), dan seluruh graf roda W_n dan matahari $C_n \odot \overline{K}_1$ oleh Frucht (1979). Dan masih banyak lagi penelitian tentang pelabelan graceful untuk kelas-kelas graf lainnya yang dapat dilihat pada Gallian (2009).

Dalam salah satu tulisannya, Baker & Sawada (2008) telah membangun suatu algoritma iteratif yang berfungsi untuk menghasilkan semua pelabelan total simpul ajaib dan pelabelan total busur ajaib yang tidak isomorfik. Namun algoritma tersebut hanya dapat digunakan untuk kelas graf tertentu saja, dan dalam tulisan Baker & Sawada (2008) hanya membuat algoritma untuk kelas graf lingkaran C_n dan kelas graf roda W_n. Algoritma tersebut mampu menghasilkan

pelabelan total simpul ajaib dan pelabelan total busur ajaib yang tidak isomorfik untuk setiap nilai konstanta ajaib *k* yang mungkin untuk setiap ukuran kelas graf tersebut.

Berpedoman pada tulisan Baker & Sawada (2008) tersebut, Ananda (2010) menulis tentang algoritma pelabelan total simpul ajaib untuk graf lingkaran, matahari, dan kecebong, Surgandini (2010) menulis tentang algoritma pelabelan total busur ajaib untuk graf lingkaran, kipas, dan roda, Utami (2010) menulis tentang algoritma pelabelan total simpul ajaib untuk graf friendship, kipas, dan jahangir yang diperumum, dan Rachmawati (2010) menulis tentang algoritma pelabelan total (*a*, *d*)-simpul antiajaib pada graf lintasan dan graf lingkaran. Sedangkan dalam skripsi ini akan dibangun algoritma-algoritma iteratif yang mampu menghasilkan seluruh pelabelan graceful untuk masing-masing kelas graf lintasan, matahari, dan ular k-C_4.

1.2 Perumusan Masalah dan Ruang Lingkup

Masalah yang akan dibahas adalah bagaimana membuat algoritma yang mampu menghasilkan seluruh pelabelan graceful pada graf dengan $|E|$ busur yang diinginkan. Dan disyaratkan pada pelabelan graceful yang dihasilkan oleh algoritma tersebut tidak ada pelabelan yang isomorfik. Algoritma pelabelan graceful dibangun hanya untuk kelas graf lintasan, matahari, dan ular k-C_4.

1.3 Jenis Penelitian dan Metode yang Digunakan

Jenis penelitian yang digunakan adalah studi literatur dan metode yang digunakan adalah pengembangan algoritma dan simulasi.

1.4 Tujuan Penelitian

Dalam skripsi ini akan dibangun algoritma untuk menemukan seluruh pelabelan graceful pada graf lintasan, matahari dan ular k-C_4. Selain itu akan ditinjau pula apakah label-label yang dihasilkan oleh algoritma tersebut tidak ada yang isomorfik.

BAB 2
LANDASAN TEORI

Pada bab ini akan diberikan beberapa definisi dan konsep dasar mengenai teori graf dan pelabelan graf yang akan digunakan pada bab selanjutnya. Pada Subbab 2.1 akan dibahas beberapa definisi dan istilah dalam teori graf yang akan digunakan dalam skripsi ini. Definisi dan istilah pada Subbab 2.1 diambil dari Rosen (2003).

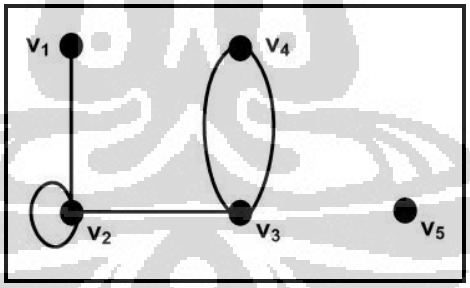
2.1 Definisi dan Istilah dalam Teori Graf

Suatu **graf** G merupakan gabungan himpunan tidak kosong **simpul**, $V(G)$, dan himpunan **busur**, $E(G)$, sedemikian sehingga titik ujung semua busur di $E(G)$ merupakan anggota $V(G)$, dinotasikan $G = (V, E)$. Untuk penyederhanaan himpunan simpul dan busur masing-masing dinotasikan dengan V dan E, sedangkan banyaknya simpul dan busur pada suatu graf masing-masing dinotasikan dengan $|V|$ dan $|E|$. Suatu graf dikatakan **graf berhingga** jika banyaknya simpul, berhingga. Himpunan busur boleh kosong. Apabila himpunan busur kosong artinya setiap simpul tidak terhubung dengan simpul lain, graf tersebut disebut **graf kosong**.

Suatu graf G disebut **graf berarah** (*directed graf*) bila anggota dari himpunan busur graf tersebut adalah **busur berarah** (*arc*), yaitu pasangan terurut dari himpunan simpul V. Apabila himpunan busur dari graf G tersebut bukan merupakan busur berarah maka graf G disebut **graf tak berarah** (*undirecteed graf*). Dalam skripsi ini graf yang akan dibahas merupakan graf tak berarah, maka definisi dan istilah yang akan diberikan selanjutnya berhubungan dengan graf tak berarah.

Dua simpul (boleh simpul yang sama) yang dihubungkan oleh busur dikatakan sebagai **titik ujung** (*endpoint*) dari busur tersebut. Busur yang mempunyai titik ujung sama dinamakan **gelung** (*loop*). Apabila sebuah pasangan simpul dihubungkan oleh lebih dari satu busur, maka busur-busur tersebut dinamakan **busur berganda** (*multiple edge*). Graf yang tidak mengandung gelung dan busur berganda dinamakan **graf sederhana** (*simple graf*).

Dua simpul yang dihubungkan oleh busur dikatakan **bertetangga** (*adjacent*). Simpul dan busur yang saling terhubung dikatakan **saling hadir** (*incident*). Dua busur dikatakan bertetangga apabila dua busur tersebut saling hadir pada minimal satu simpul yang sama. Banyaknya busur yang saling hadir pada suatu simpul dinamakan **derajat** (*degree*) simpul tersebut dan dinotasikan dengan $d(v)$. **Simpul terisolasi** (*isolated vertex*) adalah sebutan untuk simpul yang memiliki derajat 0. Sedangkan **simpul akhir** atau **simpul terminal** (*terminal vertex*) adalah simpul yang memiliki derajat 1. Jika dalam graf G terdapat gelung pada suatu simpul, maka gelung tersebut saling hadir dua kali dalam menghitung derajat simpul tersebut. Derajat terkecil dari suatu graf G dinotasikan dengan $\delta = \delta(G) = \min_{v \in V} d(v)$ dan derajat terbesar dari suatu graf G dinotasikan dengan $\Delta = \Delta(G) = \text{maks}_{v \in V} d(v)$.

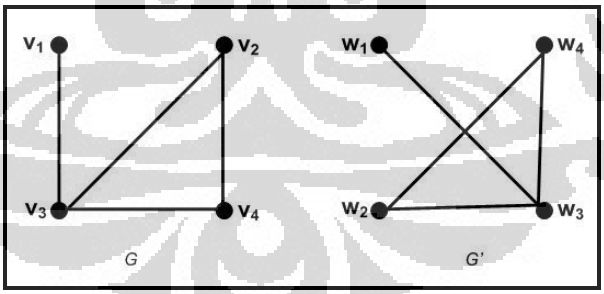


Gambar 2.1 Contoh graf G

Pada Gambar 2.1 diberikan contoh graf G yang direpresentasikan dalam bentuk gambar. Dapat dilihat pada Gambar 2.1, graf G memiliki himpunan simpul $V = \{v_1, v_2, v_3, v_4, v_5\}$ dan himpunan busur $E = \{e_1, e_2, e_3, e_4, e_5\} = \{v_1v_2, v_2v_2, v_2v_3, v_3v_4, v_3v_4\}$, sehingga banyaknya simpul dan busur adalah $|V| = 5$ dan $|E| = 5$. Hal lain yang dapat dilihat dari Gambar 2.1 adalah v_1 merupakan simpul akhir, karena selain e_1 tidak ada lagi busur yang terhubung dengan v_1, sehingga $d(v_1) = 1$.

Simpul v_5 merupakan simpul terisolasi, karena tidak ada busur yang mempunyai titik ujung di v_5, sehingga $d(v_5) = 0$. Selain itu dapat terlihat pula bahwa e_2 merupakan gelung karena titik ujung dari e_2 adalah satu simpul yang sama yaitu v_2, sedangkan e_4 dan e_5 merupakan busur berganda karena menghubungkan pasangan simpul yang sama yaitu v_3 dan v_4. Derajat dari setiap simpul di graf G adalah $d(v_1) = 1$, $d(v_2) = 4$, $d(v_3) = 3$, $d(v_4) = 2$, $d(v_5) = 0$.

Dua graf G dan G' dikatakan **isomorfik** apabila terdapat pemetaan bijektif h dari $V(G)$ ke $V(G')$ dengan sifat bahwa simpul v_1 dan v_2 bertetangga di G jika dan hanya jika simpul $h(v_1)$ dan $h(v_2)$ bertetangga di G'. Sifat tersebut berlaku untuk setiap pasangan simpul pada graf G. Sebagai akibat dari pemetaan bijektif tersebut maka dua graf yang isomorfik memiliki jumlah simpul dan busur yang sama. Demikian pula dengan derajat simpul-simpul yang saling berkorespondensi juga harus sama, yaitu $d(v)$ di G harus sama dengan $d(h(v))$ di G'. Pada Gambar 2.2 diberikan graf G dan G' yang isomorfik dengan pemetaan h dari V ke V' dimana $h(v_1) = w_1$, $h(v_2) = w_2$, $h(v_3) = w_3$, $h(v_4) = w_4$. Dapat dilihat bahwa sifat ketetanggaan dalam pemetaan h terpenuhi yaitu $h(v_1v_3) = w_1w_3$, $h(v_2v_3) = w_2w_3$, $h(v_2v_4) = w_2w_4$, $h(v_3v_4) = w_3w_4$. Dan terlihat pula bahwa simpul yang saling berkorespondensi berderajat sama, $d(v_1) = d(w_1) = 1$, $d(v_2) = d(w_2) = 2$, $d(v_3) = d(w_3) = 3$, $d(v_4) = d(w_4) = 2$.

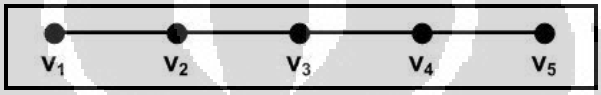


Gambar 2.2 Graf G dan G' yang isomorfik

Dalam skripsi ini graf yang akan dibahas merupakan graf berhingga, sederhana, terhubung, dan tak berarah. Pada subbab berikutnya akan diberikan definisi mengenai kelas-kelas graf yang akan dibahas dalam skripsi ini.

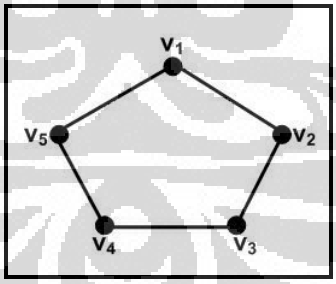
2.2 Jenis-jenis Graf

Graf lintasan (*path graph*), P_n, merupakan graf dengan *n* simpul yang memiliki himpunan busur $E(P_n)= \{v_1v_2, v_2v_3, v_3v_4, ..., v_{n-1}v_n\}$. Simpul v_1 disebut **simpul awal**, sedangkan v_n disebut **simpul akhir**. Simpul awal dan akhir mempunyai derajat 1, sedangkan simpul lainnya mempunyai derajat 2. Pada graf lintasan P_n banyaknya simpul dan busur masing-masing adalah $|V|=n$ dan $|E|=n-1$. Suatu graf dikatakan **terhubung** jika untuk setiap pasang simpul *u* dan *v* terdapat lintasan dari *u* ke *v*. Pada Gambar 2.3 diberikan contoh graf P_5.



Gambar 2.3 Contoh graf P_5

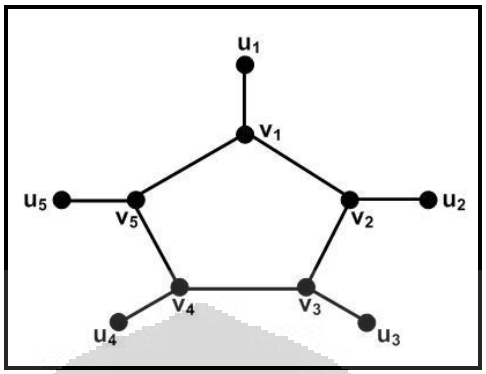
Graf lingkaran (*cycle graph*), C_n, adalah graf yang diperoleh dari graf lintasan P_n dengan penambahan busur yang menghubungkan simpul awal dan simpul akhir, yaitu busur v_1v_n. Pada graf lingkaran C_n banyaknya simpul dan busur masing-masing adalah $|V| = |E| = n$. Pada Gambar 2.4 diberikan contoh graf C_5, yang diperoleh dari graf P_5 dengan menambahkan busur v_1v_5.



Gambar 2.4 Contoh graf C_5

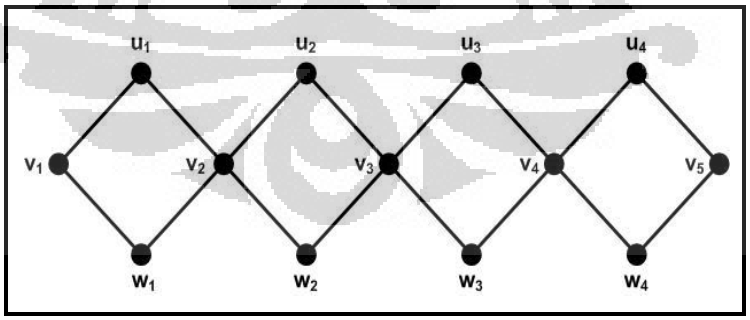
Graf matahari (*sun graph*), $C_n \odot \overline{K}_1$, adalah graf yang diperoleh dari graf lingkaran C_n dengan penambahan busur berderajat satu pada setiap simpul di graf C_n. Simpul-simpul pada graf lingkaran disebut **simpul dalam** (*inner vertex*), yang dinotasikan dengan v_i. Sedangkan simpul-simpul selain simpul dalam disebut **simpul luar** (*outer vertex*), yang dinotasikan dengan u_i. Pada graf

matahari $C_n \odot \overline{K}_1$ banyaknya simpul dan busur masing-masing adalah $|V| = |E| = 2n$. Pada Gambar 2.5 diberikan contoh graf $C_5 \odot \overline{K}_1$.



Gambar 2.5 Contoh graf $C_5 \odot \overline{K}_1$

Graf ular *k*-C_4 (*k*-C_4 *snake graph*) adalah graf yang diperoleh dari graf lintasan P_{2k+1}, yang memiliki simpul v_1, v_2, ..., v_{2k+1}, dengan menghubungkan simpul v_i dan v_{i+2} ke suatu simpul baru w_i, i = 1, 3, ..., 2k-1. Graf ular k-C_4 dapat pula dipandang sebagai rangkaian k graf C_4, nilai $k > 1$. Simpul-simpul yang sejajar dengan simpul penghubung graf-graf C_4 dinotasikan dengan v_i, i = 1, 2, ..., k+1. Sedangkan simpul-simpul di atas simpul penghubung dinotasikan dengan u_i, i = 1, 2, ..., k. Dan simpul-simpul di bawah simpul penghubung dinotasikan dengan w_i, i = 1, 2, ..., k. Pada graf ular k-C_4 banyaknya simpul dan busur masing-masing adalah $|V| = 3k + 1$ dan $|E| = 4k$. Pada Gambar 2.6 diberikan contoh graf 4-C_4.



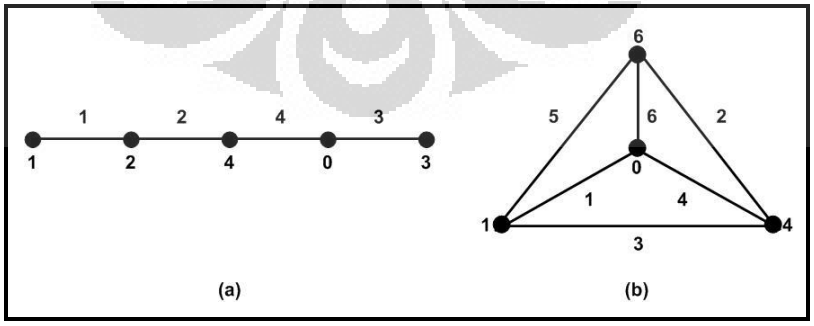
Gambar 2.6 contoh graf 4-C_4

Pada Subbab selanjutnya akan diberikan definisi pelabelan graf yang akan digunakan dalam skripsi ini.

2.3 Pelabelan Graf

Pelabelan pada graf merupakan pemetaan bijektif λ yang memetakan himpunan simpul atau himpunan busur atau himpunan simpul dan busur, ke subhimpunan bilangan (biasanya bilangan asli). Bilangan hasil pemetaan disebut **label**. Pada skripsi ini yang akan dibahas adalah pelabelan graceful.

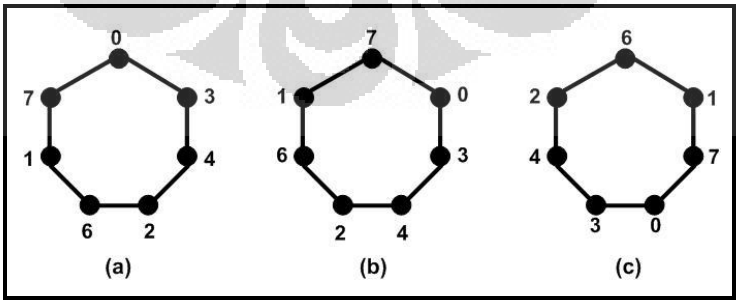
Konsep pelabelan graceful pertama kali diperkenalkan oleh Rosa (1967) dalam tulisannya tentang pelabelan graf. Tetapi nama pelabelan graceful diberikan oleh Golomb (1972). Berdasarkan Golomb (1972) **pelabelan graceful** adalah fungsi injektif f dari $V(G)$ ke himpunan $\{0, 1, 2, ..., |E|\}$ yang menginduksi pelabelan busur g dengan $g(uv) = |f(u) - f(v)|$, dimana uv merupakan busur yang mempunyai titik ujung simpul u dan v, g merupakan fungsi bijektif dari $E(G)$ ke himpunan $\{1, 2, 3, ..., |E|\}$. Graf yang dapat dilabel dengan pelabelan graceful disebut dengan **graf graceful**. Pada Gambar 2.7 diberikan contoh pelabelan graceful pada graf dengan $|E| = |V| - 1$(a) dan graf dengan $|E| > |V| - 1$(b). Dapat dilihat pada Gambar 2.7 (a) himpunan $\{0, 1, 2, ..., |E|\}$ terpakai seluruhnya untuk melabelkan simpul pada graf. Sedangkan pada Gambar 2.7 (b), himpunan $\{0, 1, 2, ..., |E|\}$ hanya terpakai sebagian saja untuk melabelkan simpul pada graf. Secara umum bila graf G memiliki jumlah busur $|E| = |V| - 1$, maka label terpakai seluruhnya, bila graf G memiliki jumlah busur $|E| > |V|-1$, maka label tidak seluruhnya terpakai.



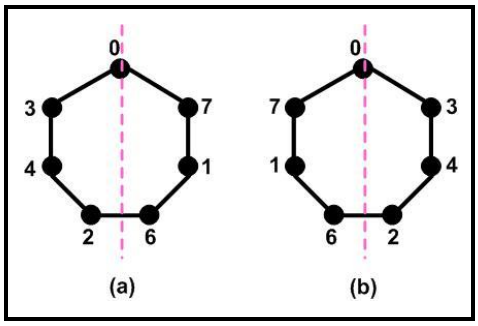
Gambar 2.7 Contoh pelabelan graceful pada graf dengan |E| = |V| - 1(a) dan graf dengan |E| > |V| - 1(b)

Ketika mempelajari pelabelan graceful pada graf *G*, yang dipertimbangkan hanyalah graf sederhana tanpa terdapat gelung ataupun busur berganda. Apabila terdapat gelung pada graf *G*, maka pada label yang dihasilkan akan terdapat label busur yang bernilai 0. Hal ini tidak sesuai dengan sifat pelabelan graceful, yaitu label busur yang dihasilkan merupakan bilangan asli yang berbeda antara 1, 2, ...|*E*|. Label 0 tidak termasuk pada label busur yang seharusnya dihasilkan pada pelabelan graceful. Sedangkan bila terdapat busur berganda dari pasangan simpul tertentu akan menghasilkan dua label busur yang sama, maka sifat bahwa setiap label busur yang dihasilkan harus berbeda tidak terpenuhi.

Jika dua graf $G = (V, E)$ dan $H = (W, F)$ tersebut merupakan graf berlabel yang memiliki pelabelan simpul $l : V \rightarrow L$, $l' : W \rightarrow L$, maka graf *G* dan *H* 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, 2010). Jika diberikan suatu graf berlabel *G*, suatu pelabelan isomorfik pada graf *G* dapat ditunjukkan sebagai hasil rotasi atau refleksi dari graf *G* atau labelnya. Dengan kata lain jika ditemukan satu pelabelan pada graf *G*, maka semua pelabelan lain yang isomorfik dengan pelabelan tersebut dapat ditemukan. Pada Gambar 2.8 diberikan contoh pelabelan isomorfik kasus rotasi, pada gambar tersebut terlihat bahwa pelabelan graf (b) dan (c) merupakan hasil rotasi dari pelabelan graf (a). Pada Gambar 2.9 diberikan contoh pelabelan isomorfik kasus refleksi, pada gambar tersebut terlihat bahwa pelabelan graf (b) merupakan hasil refleksi dari pelabelan graf (a).



Gambar 2.8 Pelabelan isomorfik kasus rotasi



Gambar 2.9 Pelabelan isomorfik kasus refleksi

2.4 Hasil yang Telah Diketahui

Sejumlah kelas graf yang telah dinyatakan graceful adalah graf lintasan P_n dan graf lingkaran C_n hanya untuk $n \equiv 0$ atau 3 (mod 4) oleh Rosa (1967), graf komplit K_n hanya untuk $n \leq 4$ oleh Golomb (1972), dan seluruh graf roda W_n dan graf matahari $C_n \odot \overline{K}_1$ oleh Frucht (1979). Dan masih banyak lagi penelitian tentang pelabelan graceful untuk kelas-kelas graf lainnya yang dapat dilihat pada Gallian (2009).

Pendekatan lain yang dapat dilakukan untuk memperoleh semua pelabelan graceful yang mungkin dari suatu graf adalah menggunakan algoritma. Dalam makalah Baker & Sawada (2008) telah dibahas algoritma untuk membangun PTSA yang tidak isomorfik pada graf lingkaran. Pada bab selanjutnya, akan dibahas tentang pembangunan algoritma pelabelan graceful untuk masing-masing graf lintasan, matahari, dan ular k-C_4.

BAB 3
ALGORITMA PELABELAN GRACEFUL UNTUK GRAF LINTASAN, GRAF MATAHARI, DAN GRAF ULAR k-C4

Pada bab ini akan dibahas mengenai pembuatan algoritma pelabelan graceful. Ide yang dipakai dalam pembuatan algoritma ini adalah, algoritma iteratif yang dibuat oleh Baker dan Sawada, yaitu algoritma pelabelan total simpul ajaib untuk graf lingkaran dan roda. Ide algoritma iteratif tersebut akan dikembangkan untuk membuat algoritma pelabelan graceful untuk kelas graf lintasan, matahari dan ular k-C_4.

Pada Subbab 2.3 sebelumnya telah dibahas tentang definisi pelabelan graceful. Pelabelan graceful adalah fungsi injektif f dari $V(G)$ ke anggota himpunan $\{0, 1, 2, ..., |E|\}$ yang menginduksi pelabelan busur g dengan $g(uv) = |f(u) – f(v)|$, dimana uv merupakan busur yang mempunyai titik ujung simpul u dan v, g merupakan fungsi bijektif dari $E(G)$ ke anggota himpunan $\{1, 2, 3, ..., |E|\}$. Dari definisi tersebut terlihat bahwa pelabelan graceful adalah pelabelan simpul yang sekaligus juga menghasilkan pelabelan busur. Karena pada pelabelan graceful label simpul telah menginduksi label busur, maka algoritma yang akan dibuat disini hanya akan menghasilkan label-label simpul.

Masukan untuk algoritma pelabelan graceful adalah ukuran dari graf yang akan dilabel, yaitu n untuk graf lintasan dan matahari, dan k untuk graf ular k-C_4. Dari masukan algoritma, dapat diketahui jumlah busur pada graf yang akan dilabel. Dengan mengetahui jumlah busur pada graf maka akan diperoleh himpunan bilangan yang tersedia untuk melabelkan simpul dan busur. Algoritma pelabelan graceful dibagi menjadi beberapa fungsi. Fungsi-fungsi tersebut dapat dikategorikan menjadi 2, yaitu fungsi inisial dan fungsi perluasan. Fungsi inisial melakukan pelabelan pada simpul inisial, yaitu simpul yang pertama kali dilabelkan saat melabel suatu graf. Sedangkan fungsi perluasan melakukan

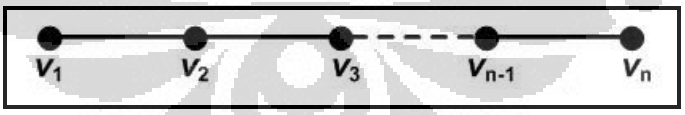
pelabelan pada simpul selain simpul inisial. Fungsi perluasan dibentuk berdasarkan pengulangan proses yang terjadi saat melabelkan suatu graf.

Agar mendapatkan pelabelan graceful yang berbeda, maka pelabelan isomorfik harus dihindari. Untuk menghindari pelabelan isomorfik tersebut, diberikan syarat tertentu pada algoritma sesuai dengan struktur kelas graf yang akan dilabel. Pemberian syarat pada algoritma yang menjamin pelabelan yang dihasilkan tidak isomorfik merupakan cara yang lebih efisien dibandingkan membangun seluruh pelabelan graceful yang mungkin, kemudian memeriksa apakah pelabelan yang dihasilkan ada isomorfik atau tidak.

Pada Subbab 3.1, 3.2, dan 3.3 akan diberikan penjelasan lengkap tentang algoritma pelabelan graceful untuk kelas graf lintasan, matahari dan ular k-C_4.

3.1 Algoritma Pelabelan Graceful untuk Graf Lintasan

Graf lintasan P_n adalah graf dengan n simpul yang memiliki himpunan busur $E(P_n)$= {v_1v_2, v_2v_3, v_3v_4, ..., $v_{n-1}v_n$}. Pada graf lintasan P_n jumlah simpulnya adalah n sedangkan jumlah busurnya adalah n-1. Gambar 3.1 merupakan contoh graf lintasan P_n. Yang menjadi masukan (*input*) dalam algoritma pelabelan graceful untuk graf lintasan ini adalah n. Himpunan label yang tersedia untuk melabelkan simpul adalah {0, 1, 2, ..., n - 1}, dan himpunan label yang tersedia untuk melabelkan busur adalah {1, 2, 3, ..., n - 1}.

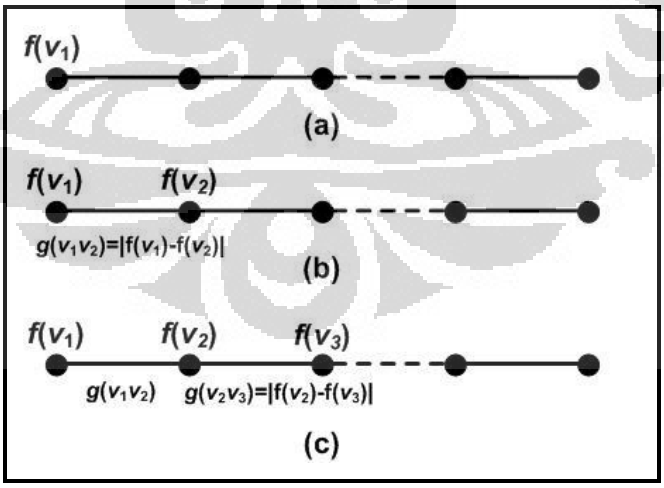


Gambar 3.1 Graf lintasan P_n

Algoritma pelabelan graceful untuk graf lintasan dibagi menjadi 2 fungsi yaitu fungsi inisialisasi (***initializePath***) dan fungsi perluasan (***extendPath***). Fungsi inisialisasi melabelkan simpul inisial v_1 dengan himpunan label simpul yang tersedia, sebut dengan $f(v_1)$. Kemudian label $f(v_1)$ ditandai agar tidak digunakan lagi pada pelabelan simpul selanjutnya. Setiap label yang telah dipakai ditandai agar semua label hanya terpakai satu kali. Kemudian fungsi inisialisasi memanggil

fungsi perluasan dengan parameter $t = 2$. Fungsi perluasan dibuat berdasarkan kesamaan proses pelabelan simpul-simpul selanjutnya pada graf lintasan. Proses tersebut adalah melabelkan simpul dengan label yang tersedia. Kemudian menentukan label busur yang menghubungkan simpul tersebut dengan simpul sebelumnya yang telah dilabelkan.

Pada fungsi perluasan digunakan parameter *t* untuk merujuk pada simpul yang akan dilabel. Misalkan v_t dilabelkan dengan $f(v_t)$, label $f(v_t)$ ditandai agar tidak terpakai lagi. Setelah melabelkan v_t dapat ditentukan label busur $v_{t-1}v_t$, yaitu $g(v_{t-1}\ v_t)=|f(v_{t-1})-f(v_t)|$. Label yang diperoleh dengan cara ini disebut ***determined label***. Apabila *determined label* $g(v_{t-1}v_t)$ tersedia maka $g(v_{t-1}v_t)$ ditandai agar tidak terpakai lagi, dan fungsi perluasan akan memanggil dirinya sendiri dengan parameter $t+1$. Namun bila *determined label* $g(v_{t-1}v_t)$ tidak tersedia, karena telah terpakai untuk melabelkan busur sebelumnya, maka dilakukan *backtracking*. ***Backtracking*** adalah langkah menelusuri kembali jalur yang sudah dilalui secara mundur. Dalam algoritma ini *backtracking* dilakukan dengan kembali ke v_{t-1} untuk mengubah label simpul tersebut. Pada proses *backtracking* untuk suatu elemen graf, status label yang digunakan sebelumnya untuk melabel elemen graf tersebut harus diubah menjadi tersedia kembali. Pada fungsi perluasan terjadi pemanggilan rekursif sampai parameter $t = n$.



Gambar 3.2 Ilustrasi pembentukan pelabelan graceful pada graf P_n

Gambar 3.2 merupakan ilustrasi pembentukan pelabelan graceful untuk graf lintasan P_n. Pada Gambar 3.2 (a) merupakan proses inisialisasi, yaitu

melabelkan simpul inisial v_1 dengan label yang tersedia, sebut dengan $f(v_1)$. Setelah itu fungsi inisialisasi memanggil fungsi perluasan dengan parameter $t = 2$, sehingga selanjutnya simpul v_2 dilabelkan dengan label yang tersedia. Dapat dilihat pada Gambar 3.2 (b), setelah melabelkan simpul v_2 dapat ditentukan label busur v_1v_2 yaitu $g(v_1v_2)=|f(v_1)-f(v_2)|$, label busur v_1v_2 merupakan *determined label*. Dalam ilustrasi ini diasumsikan seluruh *determined label* tersedia. Kemudian fungsi perluasan memanggil dirinya sendiri dengan parameter $t + 1$. Selanjutnya dapat dilihat pada Gambar 3.2 (c), setelah melabelkan simpul v_3 dengan $f(v_3)$ dapat ditentukan label busur v_2v_3 yaitu $g(v_2v_3)=|f(v_2)-f(v_3)|$, label busur v_2v_3 merupakan *determined label*. Karena diasumsikan label $g(v_2v_3)$ ada, maka fungsi perluasan akan memanggil dirinya sendiri dengan parameter $t + 1$ dan seterusnya.

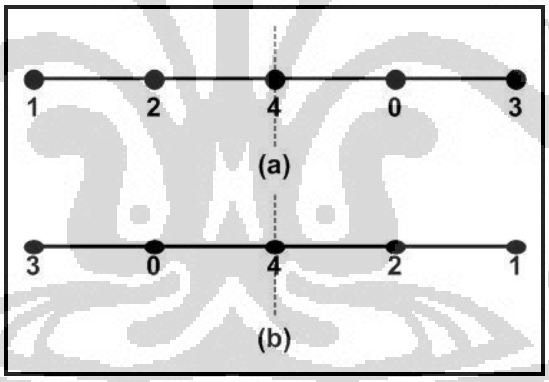
Algoritma 1.a
Fungsi initializePath()
for each available label *i* **do**
    $f(v_1):=i$
    availf [*i*]:=**false**
    extendPath(2)
    availf [*i*]:=**true**

Algoritma 1.b Fungsi extendPath(*t*)
if $t = n$ **then**
  for each available label *i* **do**
  $f(v_t):=i$
  $g(v_t\ v_{t-1})=|\ f(v_t) - f(v_{t-1})|$
  if *availg* [$g(v_t\ v_{t-1})$] **then**
    print()
  end if
else
  for each available label i **do**
    $g(v_t\ v_{t-1})=|i - f(v_{t-1})|$
    if *availg* [$g(v_t\ v_{t-1})$] **then**
      $f(v_t):=i$
      availf [*i*]:=**false**
      availg [$g(v_t\ v_{t-1})$]:=**false**
      extendPath(*t*+1)
      availg [$g(v_t\ v_{t-1})$]:=**true**
      availf [*i*]:=**true**
    end if
end if

Proses rekursif pada fungsi perluasan akan terus berlangsung sampai nilai $t = n$. Saat $t = n$ yang akan dilabelkan adalah v_n. Setelah mendapatkan label $f(v_n)$ dapat ditentukan label $g(v_{n-1}\ v_t)=|f(v_{n-1})-f(v_n)|$. Apabila *determined label* $v_{n-1}v_n$ tersedia, maka diperoleh suatu pelabelan graceful untuk graf P_n, dan labelnya akan dicetak. Setelah menemukan suatu pelabelan graceful, proses algoritma pelabelan graceful tidak langsung berhenti. Namun komputasi akan tetap dilanjutkan secara *backtracking* untuk menemukan seluruh pelabelan graceful untuk graf P_n.

Algoritma pelabelan graceful untuk graf lintasan diberikan oleh Algoritma 1. Algoritma 1.a merupakan fungsi inisialisasi, sedangkan Algortima 1.b merupakan fungsi perluasan. Pada algoritma-algoritma ini $f(x)$ adalah label yang digunakan untuk melabel simpul x, sedangkan $g(y)$ adalah label yang digunakan untuk melabelkan busur y. *Availf* dan *availg* dalam algoritma ini merupakan *array* yang digunakan untuk memberikan status pada label. Apabila ***availf* [*f*(*x*)]** bernilai ***true*** artinya label $f(x)$ tersedia, tapi bila bernilai ***false*** artinya label tersebut telah terpakai, begitu pula dengan $g(y)$. Bisa dilihat bahwa pada algoritma 1.b terjadi pemanggilan fungsi secara rekursif.

Pada Bab 2 dituliskan bahwa pelabelan isomorfik dapat terjadi karena kasus rotasi atau refleksi. Jika dilihat dari struktur graf lintasan, pelabelan isomorfik yang terdapat pada graf lintasan merupakan kasus refleksi. Dengan memberikan syarat tambahan pada algoritma 1 agar pelabelan isomorfik tidak terjadi, maka algoritma 1 akan lebih efisien. Pada Gambar 3.3 diberikan pelabelan graceful pada graf P_5 yang isomorfik hasil refleksi.



Gambar 3.3 Pelabelan graceful pada graf P_5 yang isomorfik

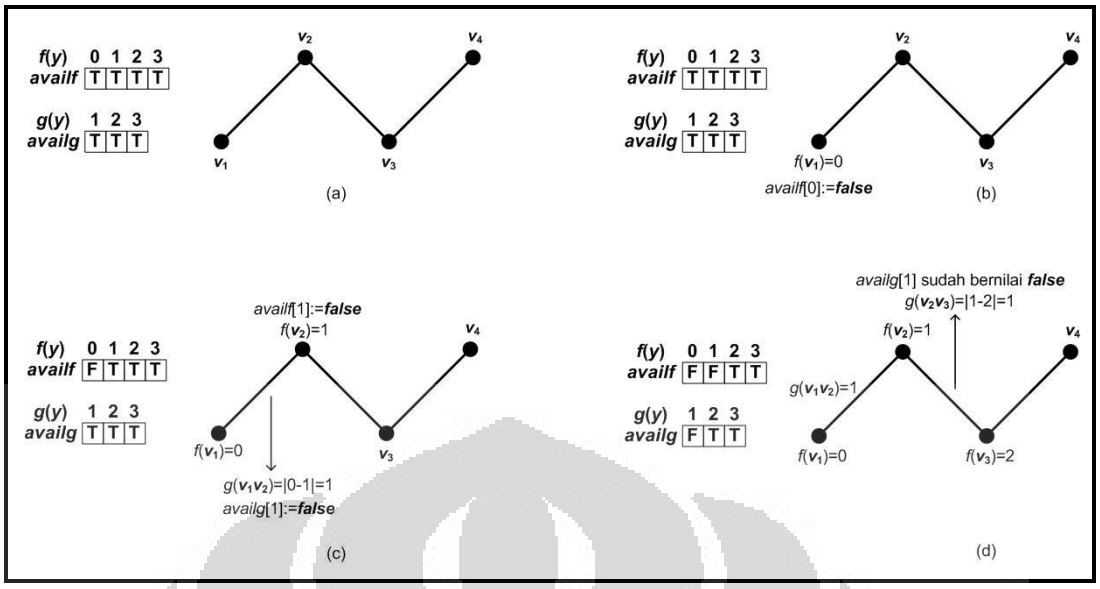
Dari Gambar 3.3 dapat dilihat bahwa, syarat yang harus ditambahkan untuk menghindari pelabelan isomorfik pada kasus ini adalah $f(v_1)<f(v_n)$ atau $f(v_1)>f(v_n)$. Pada algoritma ini syarat yang dipilih adalah $f(v_1)<f(v_n)$. Sehingga pada fungsi perluasan akan ditambahkan syarat $f(v_1)<f(v_n)$, pada saat melabelkan simpul v_n. Himpunan label yang tersedia untuk melabelkan simpul adalah {0, 1, 2, ..., n - 1}. Label $n - 1$ adalah label terbesar, maka label $n - 1$ tidak mungkin digunakan untuk melabelkan v_1. Sehingga untuk mengurangi komputasi, pada fungsi inisialisasi ditambahkan syarat bahwa label maksimum untuk v_1 adalah n-2.

```
1.  Algoritma 1.a
    Fungsi initializePath()
2.  for each available label i
    where 0 ≤ i ≤ n-2 do
3.      f(v1):=i
4.      availf[i]:=false
5.      extendPath(2)
6.      availf[i]:=true
```

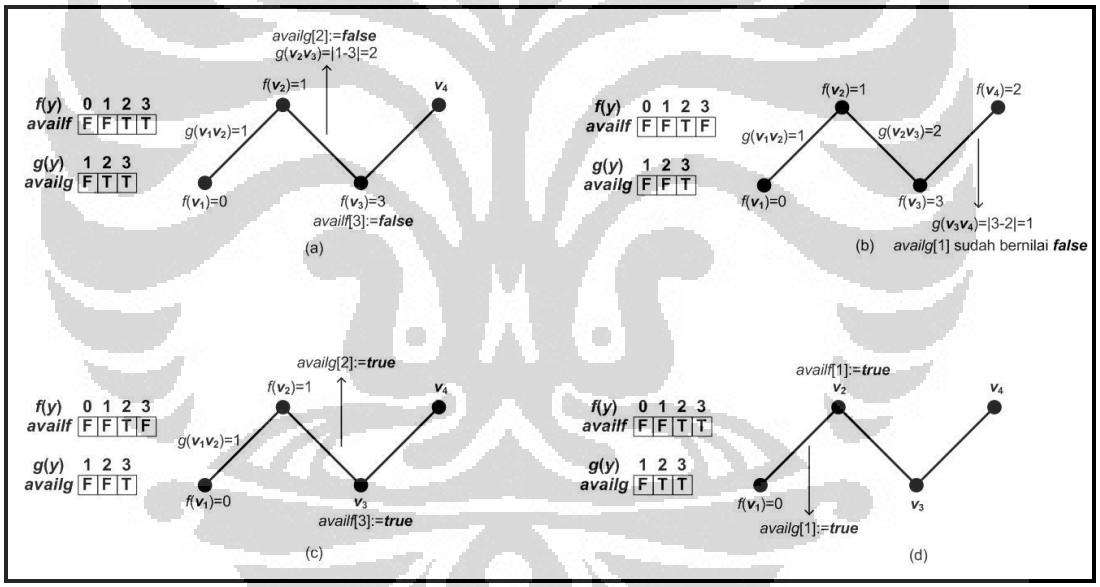
```
1.  Algoritma 1.b Fungsi extendPath(t)
2.  If t = n then
3.      for each available label i do
4.      f(vt):=i
5.      g(vt vt-1)=| f(vt) - f(vt-1)|
6.      if f(v1) < f(vn) and availg [g(vt vt-1)] then
7.          print()
        end if
8.  else
9.      for each available label i do
10.     g(vt vt-1)=| i - f(vt-1)|
11.     If availg [g(vt vt-1)] then
12.         f(vt):=i
13.         availf [i]:=false
14.         availg [g(vt vt-1)]:=false
15.         extendPath(t+1)
16.         availg [g(vt vt-1)]:=true
17.         availf [i]:=true
        end if
    end if
```

Sebagai contoh penggunaan algoritma pelabelan graceful pada graf lintasan, akan dilakukan pelabelan untuk graf P_4. Karena $n = 4$, maka himpunan bilangan yang digunakan untuk melabelkan simpul adalah {0, 1, 2, 3}, dan yang digunakan untuk melabelkan busur adalah {1, 2, 3}. Maka untuk $f(x)$ disediakan *availf* = [T, T, T, T], dan untuk $g(y)$ *availg* = [T, T, T] (Gambar 3.4 (a)).

Menggunakan fungsi inisial yang diberikan pada algoritma 1.a, yang pertama kali dilabelkan adalah v_1. Label yang tersedia adalah {0, 1, 2, 3}. Misalkan $f(v_1) = 0$, maka nyatakan *availf*[0] = *false*, karena 0 sudah terpakai (Gambar 3.4 (b)). Setelah fungsi inisial melabelkan simpul yang pertama, fungsi inisial memanggil fungsi perluasan dengan $t = 2$. Sehingga selanjutnya v_2 dilabelkan dengan label yang tersedia, yaitu {1, 2, 3}. Dilabelkan $f(v_2) = 1$. Setelah v_1 dan v_2 dilabelkan dapat ditentukan $g(v_1v_2) = |0\text{-}1| = 1$, karena label busur $g(v_1v_2) = 1$ masih tersedia, maka dapat digunakan dan nyatakan *availf*[1] = *false* dan *availg*[1] = *false* (Gambar 3.4 (c)). Setelah itu fungsi perluasan memanggil dirinya sendiri dengan $t = t + 1 = 3$. Selanjutnya dilabelkan v_3 dengan label yang tersedia, misalkan $f(v_3) = 2$. Lalu dapat ditentukan $g(v_2v_3) = |1\text{-}2| = 1$, karena label busur tersebut sudah terpakai, maka digunakan label lain yang tersedia untuk melabelkan v, yaitu $f(v_3) = 3$ (Gambar 3.4 (d)).



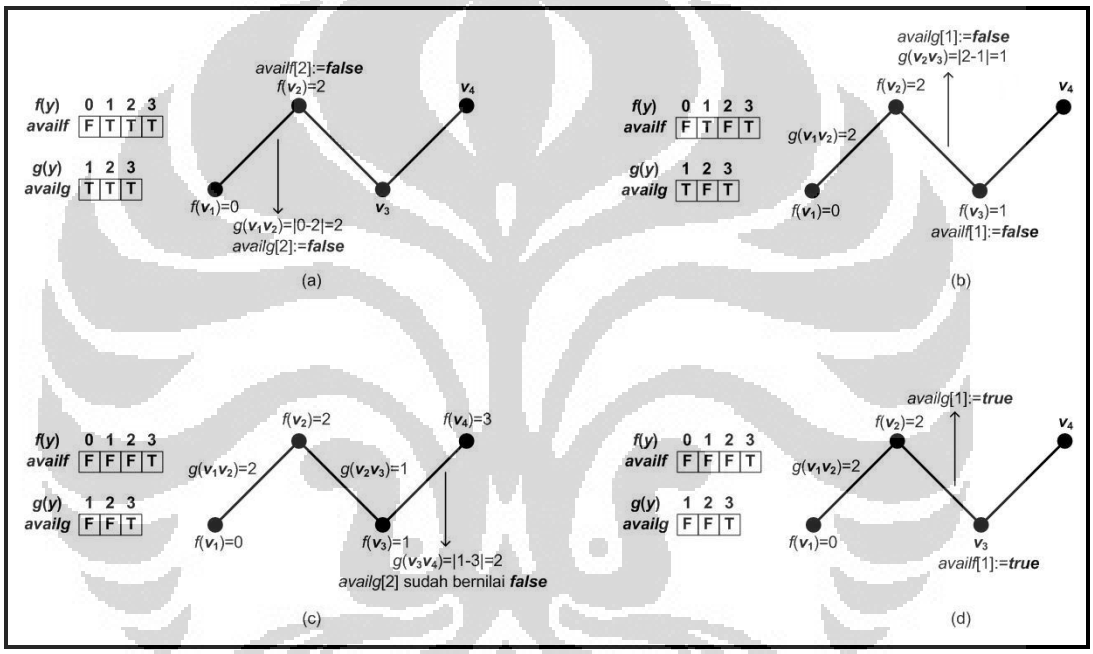
Gambar 3.4 Proses pembentukan pelabelan graceful pada graf P_4 (1)



Gambar 3.5 Proses pembentukan pelabelan graceful pada graf P_4 (2)

Kemudian dapat ditentukan $g(v_2v_3) = |1\text{-}3| = 2$, karena label $g(v_2v_3) = 2$ tersedia, maka dapat digunakan dan nyatakan *availf*[3] = *false* dan *availg*[2] = *false* (Gambar 3.5 (a)). Fungsi perluasan memanggil dirinya sendiri dengan parameter $t = t + 1 = 4$. Selanjutnya v_4 dilabelkan dengan label yang tersedia, yaitu $f(v_4) = 2$, kemudian dapat ditentukan $g(v_3v_4) = |3\text{-}2| = 1$, ternyata label busur tersebut telah terpakai (Gambar 3.5 (b)). Karena tidak ada lagi label lain yang

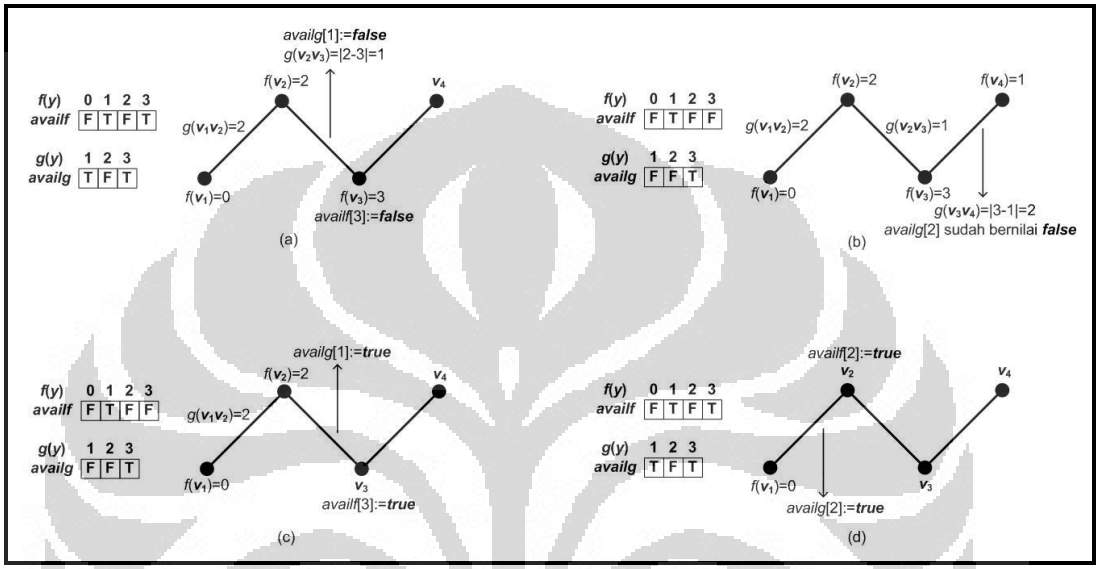
tersedia untuk melabelkan v_4, maka dilakukan *backtracking* pada v_3. Parameter *t* pada fungsi perluasan kembali menjadi *t* = 3 dalam proses *backtracking*. Label yang digunakan untuk melabel v_3 dan busur v_2v_3 statusnya dikembalikan menjadi tersedia, maka *availf*[3] = *true* dan *availg*[2] = *true* (Gambar 3.5 (c)). Karena tidak ada lagi label lain yang tersedia untuk melabelkan v_3, maka dilakukan *backtracking* pada v_2. Parameter *t* pada fungsi perluasan kembali menjadi *t* = 2 dalam proses *backtracking*. Label yang digunakan untuk melabel v_2 dan busur v_1v_2 statusnya dikembalikan menjadi tersedia, maka *availf*[1] = *true* dan *availg*[1] = *true* (Gambar 3.5 (d)).



Gambar 3.6 Proses pembentukan pelabelan graceful pada graf P_4 (3)

Digunakan label lain yang tersedia untuk melabel v_2, yaitu $f(v_2) = 2$, kemudian dapat ditentukan $g(v_1v_2) = |0-2| = 2$, karena label $g(v_1v_2) = 2$ tersedia, maka dapat digunakan dan nyatakan *availf*[2] = *false* dan *availg*[2] = *false* (Gambar 3.6 (a)). Fungsi perluasan memanggil dirinya sendiri dengan parameter *t* = *t* +1 = 3. Selanjutnya v_3 dilabelkan dengan label yang tersedia, yaitu $f(v_3) = 1$, kemudian dapat ditentukan $g(v_2v_3) = |2-1| = 1$, karena label $g(v_2v_3) = 1$ tersedia, maka dapat digunakan dan nyatakan *availf*[1] = *false* dan *availg*[1] = *false* (Gambar 3.6 (b)). Fungsi perluasan memanggil dirinya sendiri dengan parameter *t* = *t* +1 = 4. Selanjutnya v_4 dilabelkan dengan label yang tersedia, yaitu $f(v_4) = 3$,

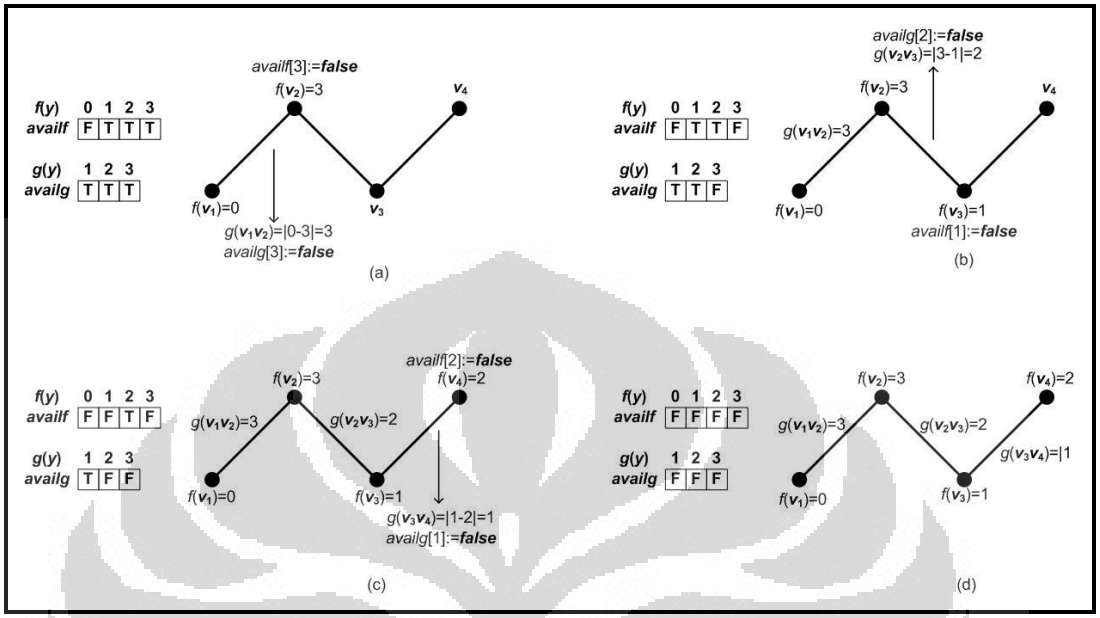
kemudian dapat ditentukan $g(v_3v_4) = |1\text{-}3| = 2$, ternyata label busur tersebut telah terpakai (Gambar 3.6 (c)). Karena tidak ada lagi label lain yang tersedia untuk melabelkan v_4, maka dilakukan *backtracking* pada v_3. Parameter *t* pada fungsi perluasan kembali menjadi $t = 3$ dalam proses *backtracking*. Label yang digunakan untuk melabel v_3 dan busur v_2v_3 statusnya dikembalikan menjadi tersedia, maka *availf*[1] = *true* dan *availg*[1] = *true* (Gambar 3.6 (d)).



Gambar 3.7 Proses pembentukan pelabelan graceful pada graf P_4 (4)

Digunakan label lain yang tersedia untuk melabel v_3, yaitu $f(v_3) = 3$, kemudian dapat ditentukan $g(v_2v_3) = |2\text{-}3| = 1$, karena label $g(v_2v_3) = 1$ tersedia, maka dapat digunakan dan nyatakan *availf*[3] = *false* dan *availg*[1] = *false* (Gambar 3.7 (a)). Fungsi perluasan memanggil dirinya sendiri dengan parameter $t = t + 1 = 4$. Selanjutnya v_4 dilabelkan dengan label yang tersedia, yaitu $f(v_4) = 1$, kemudian dapat ditentukan $g(v_3v_4) = |3\text{-}1| = 2$, ternyata label busur tersebut telah terpakai (Gambar 3.7 (b)). Karena tidak ada lagi label lain yang tersedia untuk melabelkan v_4, maka dilakukan *backtracking* pada v_3. Parameter *t* pada fungsi perluasan kembali menjadi $t = 3$ dalam proses *backtracking*. Label yang digunakan untuk melabel v_3 dan busur v_2v_3 statusnya dikembalikan menjadi tersedia, maka *availf*[3] = *true* dan *availg*[1] = *true* (Gambar 3.7 (c)). Karena tidak ada lagi label lain yang tersedia untuk melabelkan v_3, maka dilakukan *backtracking* pada v_2. Parameter *t* pada fungsi perluasan kembali menjadi $t = 2$

dalam proses *backtracking*. Label yang digunakan untuk melabel v_2 dan busur v_1v_2 statusnya dikembalikan menjadi tersedia, maka *availf*[2] = *true* dan *availg*[2] = *true* (Gambar 3.7 (d)).



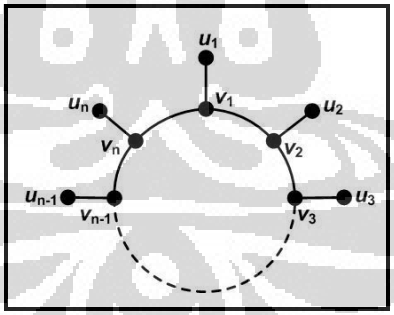
Gambar 3.8 Proses pembentukan pelabelan graceful pada graf P_4 (5)

Digunakan label lain yang tersedia untuk melabel v_2, yaitu $f(v_2) = 3$, kemudian dapat ditentukan $g(v_1v_2) = |0\text{-}3| = 3$, karena label $g(v_1v_2) = 3$ tersedia, maka dapat digunakan dan nyatakan *availf*[3] = *false* dan *availg*[3] = *false* (Gambar 3.8 (a)). Fungsi perluasan memanggil dirinya sendiri dengan parameter $t = t + 1 = 3$. Selanjutnya v_3 dilabelkan dengan label yang tersedia, yaitu $f(v_3) = 1$, kemudian dapat ditentukan $g(v_2v_3) = |3\text{-}1| = 2$, karena label $g(v_2v_3) = 2$ tersedia, maka dapat digunakan dan nyatakan *availf*[1] = *false* dan *availg*[2] = *false* (Gambar 3.8 (b)). Fungsi perluasan memanggil dirinya sendiri dengan parameter $t = t + 1 = 4$. Selanjutnya v_4 dilabelkan dengan label yang tersedia, yaitu $f(v_4) = 2$, kemudian dapat ditentukan $g(v_3v_4) = |1\text{-}2| = 1$, karena label $g(v_3v_4) = 1$ tersedia, maka dapat digunakan dan nyatakan *availf*[2] = *false* dan *availg*[1] = *false* (Gambar 3.8(c)). Karena nilai $t = n$, dan v_4 serta busur v_3v_4 telah dilabelkan maka telah diperoleh suatu pelabelan graceful. Pelabelan graceful yang diperoleh memiliki label-label sebagai berikut, $f(v_1) = 0$, $f(v_2) = 3$, $f(v_3) = 1$, dan $f(v_4) = 2$ (Gambar 3.8(d)).

Proses komputasi pada algoritma terus berlanjut secara *backtracking* untuk menemukan pelabelan graceful lain yang tidak isomorfik pada graf P_4. Jumlah pelabelan graceful yang tidak isomorfik pada graf P_4 ada 2. Pelabelan graceful yang lain untuk graf P_4 adalah, $f(v_1) = 1$, $f(v_2) = 2$, $f(v_3) = 0$, dan $f(v_4) = 3$.

3.2 Algoritma Pelabelan Graceful untuk Graf Matahari

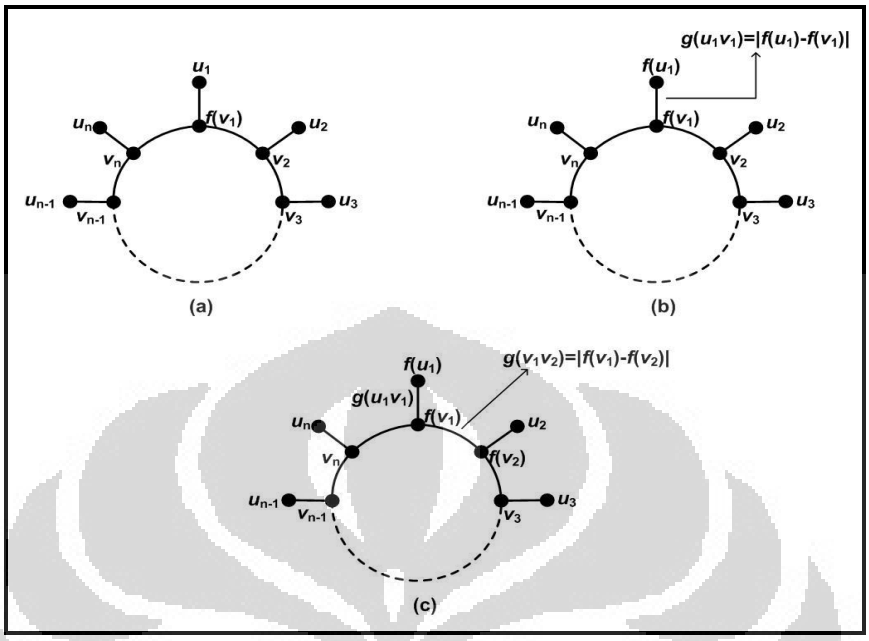
Graf matahari $C_n \odot \overline{K}_1$, diperoleh dari graf C_n dengan penambahan satu simpul berderajat 1 di setiap simpul pada graf C_n. Pada graf matahari, simpul yang membentuk graf C_n dinotasikan dengan v_1, v_2, ... v_n, dan disebut sebagai simpul dalam. Sedangkan simpul yang dihubungkan ke masing-masing simpul pada graf C_n dinotasikan dengan u_1, u_2, ... u_n, indeksnya disesuaikan dengan indeks simpul pada graf C_n yang terhubung dengan simpul tersebut. Gambar 3.9 diberikan gambar graf matahari $C_n \odot \overline{K}_1$. Yang menjadi masukan (*input*) dalam algoritma pelabelan graceful untuk graf matahari ini adalah *n*. Himpunan label yang tersedia untuk melabelkan simpul adalah {0, 1, 2, ..., 2*n*}, dan himpunan label yang tersedia untuk melabelkan busur adalah {1, 2, 3, ..., 2*n*}.



Gambar 3.9 Graf matahari $C_n \odot \overline{K}_1$

Algoritma pelabelan graceful untuk graf matahari ini dibagi menjadi 2 fungsi yaitu fungsi inisial (***initializeSun***) dan fungsi perluasan (***extendSun***). Fungsi inisial melabelkan simpul inisial v_1 dengan himpunan label simpul yang tersedia, sebut dengan $f(v_1)$. Setiap label yang telah digunakan harus ditandai agar tidak terpakai lagi pada saat pelabelan selanjutnya, maka label $f(v_1)$ ditandai karena telah digunakan. Fungsi inisial akan memanggil fungsi perluasan dengan

parameter $t = 1$. Fungsi perluasan menggunakan parameter t untuk merujuk pada simpul yang akan dilabel.



Gambar 3.10 Ilustrasi pembentukan pelabelan graceful pada graf $C_n \odot \overline{K}_1$

Fungsi perluasan dengan parameter t akan melabelkan u_t, dengan label yang tersedia, sebut dengan $f(u_t)$. Lalu dapat ditentukan label busur v_tu_t yaitu $g(v_tu_t) = |f(v_t) - f(u_t)|$, label $g(v_tu_t)$ merupakan *determined label*. Jika label $g(v_tu_t)$ tersedia maka fungsi perluasan akan melabelkan v_{t+1}, namun jika label $g(v_tu_t)$ tidak tersedia karena sudah terpakai maka dilakukan *backtracking* ke simpul v_t. Simpul v_{t+1} dilabelkan dengan label yang tersedia, sebut dengan $f(v_{t+1})$. Kemudian dapat ditentukan $g(v_tv_{t+1}) = |f(v_t) - f(v_{t+1})|$. Jika label $g(v_tv_{t+1})$ tersedia maka fungsi perluasan akan memanggil dirinya sendiri dengan parameter $t = t + 1$, namun jika label $g(v_tv_{t+1})$ tidak tersedia karena sudah terpakai maka dilakukan *backtracking* ke simpul u_t. Pada fungsi perluasan akan terjadi pemanggilan fungsi secara rekursif sampai $t = n$. Saat parameter $t = n$, setelah melabelkan u_n dengan label yang tersedia, sebut dengan $f(u_n)$, dapat ditentukan label busur $g(v_nu_n) = |f(v_n) - f(u_n)|$ dan $g(v_1v_n) = |f(v_1) - f(v_n)|$, label $g(v_nu_n)$ dan $g(v_1v_n)$ merupakan *determined label*. Jika label busur $g(v_nu_n)$ dan $g(v_1v_n)$ tersedia, berarti telah ditemukan suatu pelabelan graceful untuk graf matahari $C_n \odot \overline{K}_1$. Gambar 3.10 merupakan ilustrasi pelabelan graceful pada graf matahari $C_n \odot \overline{K}_1$. Setelah mendapatkan

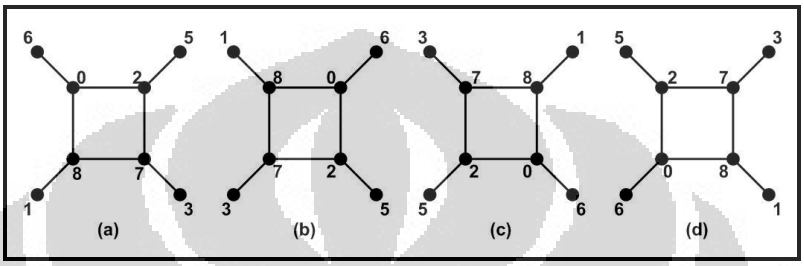
suatu pelabelan graceful untuk graf matahari $C_n \odot \overline{K}_1$, komputasi tetap dilanjutkan secara *backtracking* untuk mendapatkan pelabelan graceful lainnya.

Algoritma 2.a
Fungsi initializeSun()
for each available label *i* **do**
    $f(v_1):=i$
    availf [*i*]:=**false**
    extendSun(1)
    availf [*i*]:=**true**

Algoritma 2.b Fungsi extendSun(*t*)
if $t = n$ **then**
    for each available label *e* **do**
        $f(u_t):=e$
        $g(u_tv_t)=|f(u_t) - f(v_t)|$
        $g(v_tv_1)=|f(v_t) - f(v_1)|$
        if *availg* [$g(u_tv_t)$] and *availg* [$g(v_tv_1)$] and $g(u_tv_t) \neq g(v_tv_1)$ **then**
            print()
        end if
else
    for each available label *e* **do**
        $g(u_tv_t)=|e- f(v_t)|$
        if *availg* [$g(u_tv_t)$] **then**
        $f(u_t):=e$
        availf [*e*]:=**false**
        availg [$g(u_tv_t)$]:=**false**
        for each available *i* **do**
        $g(v_tv_{t+1})=|f(v_t) - i|$
        if *availg*[$g(v_tv_{t+1})$] **then**
        $f(v_{t+1}):=i$
        availf [*i*]:=**false**
        availg [$g(v_tv_{t+1})$]:=**false**
        extendSun(t+1)
        availg [$g(v_tv_{t+1})$]:=**true**
        availf [*i*]:=**true**
        end if
        availg [$g(u_tv_t)$]:=**true**
        availf [*e*]:=**true**
        end if
end if

Algoritma pelabelan graceful untuk graf matahari diberikan oleh Algoritma 2. Algoritma 2.a merupakan fungsi inisialisasi, sedangkan Algoritma 2.b merupakan fungsi perluasan. Pada algoritma-algoritma ini $f(x)$ adalah label yang digunakan untuk melabel simpul *x*, sedangkan $g(y)$ adalah label yang digunakan untuk melabelkan busur *y*. *Availf* dan *availg* dalam algoritma ini merupakan *array* yang digunakan untuk memberikan status pada label. Apabila ***availf* [*f*(*x*)]** bernilai ***true*** artinya label $f(x)$ tersedia, tapi bila bernilai ***false*** artinya label tersebut telah terpakai, begitu pula dengan $g(y)$. Bisa dilihat bahwa pada Algoritma 2.b terjadi pemanggilan fungsi secara rekursif.

Pada Bab 2 dituliskan bahwa pelabelan isomorfik dapat terjadi karena kasus rotasi atau refleksi. Jika dilihat dari struktur graf matahari, pelabelan isomorfik yang terdapat pada graf matahari merupakan kasus rotasi dan refleksi. Dengan memberikan syarat tambahan pada Algoritma 2 agar pelabelan isomorfik tidak terjadi, maka Algoritma 2 akan lebih efisien. Pada Gambar 3.11 diberikan pelabelan graceful yang isomorfik untuk graf $C_4 \odot \overline{K}_1$ yang merupakan kasus rotasi.

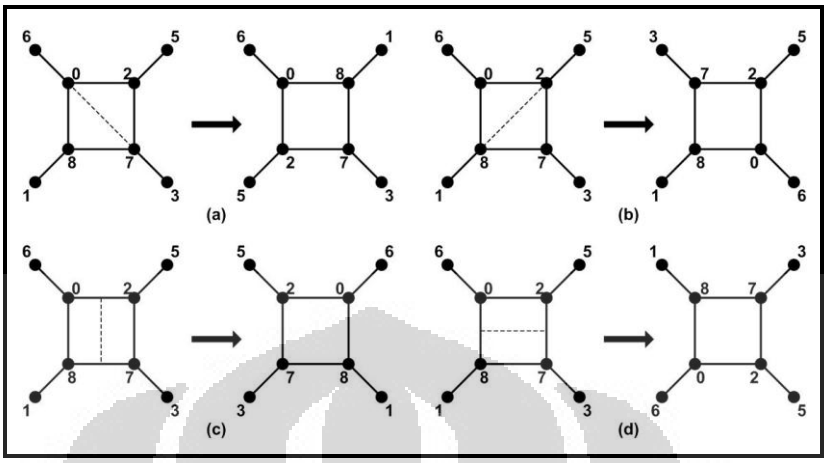


Gambar 3.11 Pelabelan graceful $C_4 \odot \overline{K}_1$ yang isomorfik dengan kasus rotasi

Pada Gambar 3.11 dapat dilihat bahwa cara untuk menghindari pelabelan isomorfik dengan kasus rotasi adalah, salah satu simpul diantara v_1, v_2, ..., v_n harus memiliki label lebih kecil dari label simpul-simpul dalam lainnya. Dalam algoritma ini dipilih v_1 sebagai simpul dalam dengan label minimum, di antara simpul-simpul dalam lainnya. Syarat ini diberikan pada fungsi perluasan saat melabelkan simpul-simpul dalam selain v_1. Dengan pemberian syarat tersebut, pelabelan graceful pada Gambar 3.11 (b), (c), dan (d) tidak mungkin terjadi. Selain itu karena syarat tersebut, label yang mungkin untuk melabel v_1 adalah {0, 1, 2, ..., n+1}. Maka agar algoritma lebih efisien, pada fungsi inisial diberikan syarat bahwa label maksimum yang dipakai untuk melabelkan v_1 adalah n+1.

Pada Gambar 3.12 diberikan pelabelan graceful yang isomorfik untuk graf $C_4 \odot \overline{K}_1$ yang merupakan kasus refleksi, dengan sumbu refleksi dinyatakan dengan garis putus-putus. Karena sebelumnya telah diberikan syarat bahwa label minimum diantara label-label simpul dalam harus diberikan di v_1, maka refleksi yang mungkin terjadi hanya refleksi dengan sumbu yang melewati v_1. Sehingga label minimum tetap dilabelkan pada v_1. Kasus refleksi dengan sumbu yang tidak melewati v_1, seperti pada Gambar 3.12 (b), (c), dan (d) tidak mungkin terjadi. Syarat yang harus diberikan untuk menghindari pelabelan isomorfik dengan kasus

refleksi seperti Gambar 3.12 (a) adalah, label v_n harus lebih besar dari v_2. Syarat tersebut diberikan pada fungsi perluasan, saat melabelkan simpul v_n.

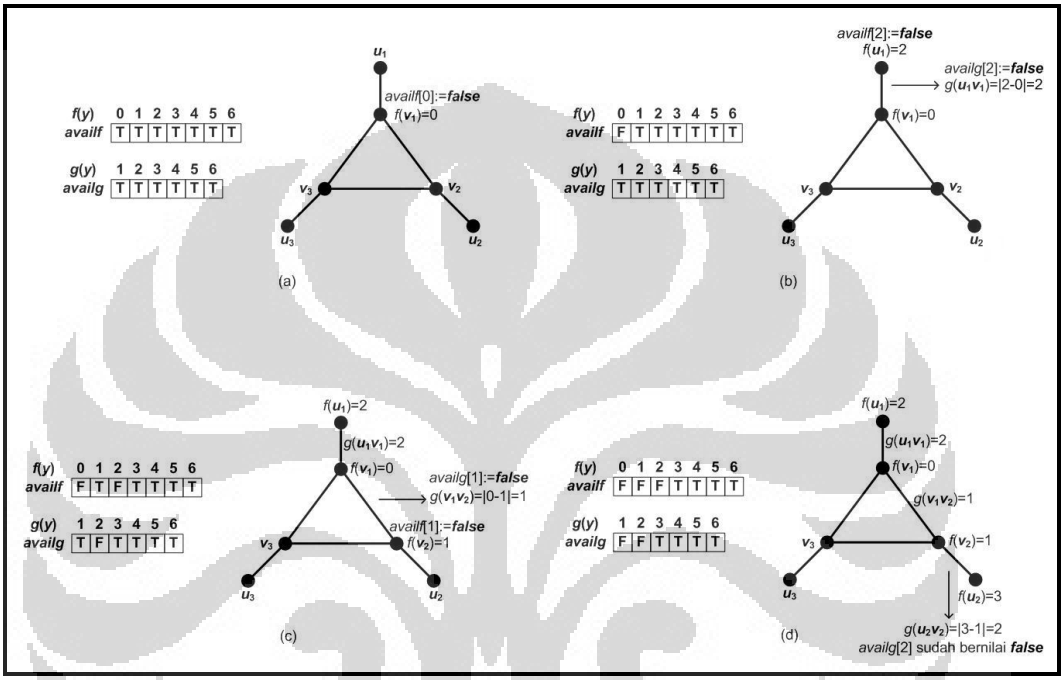


Gambar 3.12 Pelabelan graceful $C_4 \odot \overline{K}_1$ yang isomorfik dengan kasus refleksi

```
1.  Algoritma 2.a
    Fungsi initializeSun()
2.  for each available label i
    where 0≤i≤n+1 do
3.      f(v1):=i
4.      availf [i]:=false
5.      extendSun(1)
6.      availf [i]:=true
```

```
1.  Algoritma 2.b Fungsi extendSun(t)
2.  if t = n then
3.      for each available label e do
4.          f(ut):=e
5.          g(utvt)=|f(ut) - f(vt)|
6.          g(vtv1)=|f(vt) -  f(v1)|
7.          if availg [g(utvt)] and availg [g(vtv1)]  and
    g(utvt) ≠ g(vtv1) and f(vt)>f(v1) and f(vt)>f(v2) then
8.              print()
            end if
9.  else
10.     for each available label e do
11.         g(utvt)=| e - f(vt)|
12.         if availg [g(utvt)] then
13.             f(ut):=e
14.             availf [e]:=false
15.             availg [g(utvt)]:=false
16.             for each available i do
17.                 g(vtvt+1)=|f(vt) - i|
18.                 if availg[g(vtvt+1)] and f(vt+1)>f(v1) then
19.                     f(vt+1):=i
20.                     availf [i]:=false
21.                     availg [g(vtvt+1)]:=false
22.                     extendSun(t+1)
23.                     availg [g(vtvt+1)]:=true
24.                     availf [i]:=true
                    end if
25.             availg [g(utvt)]:=true
26.             availf [e]:=true
            end if
    end if
```

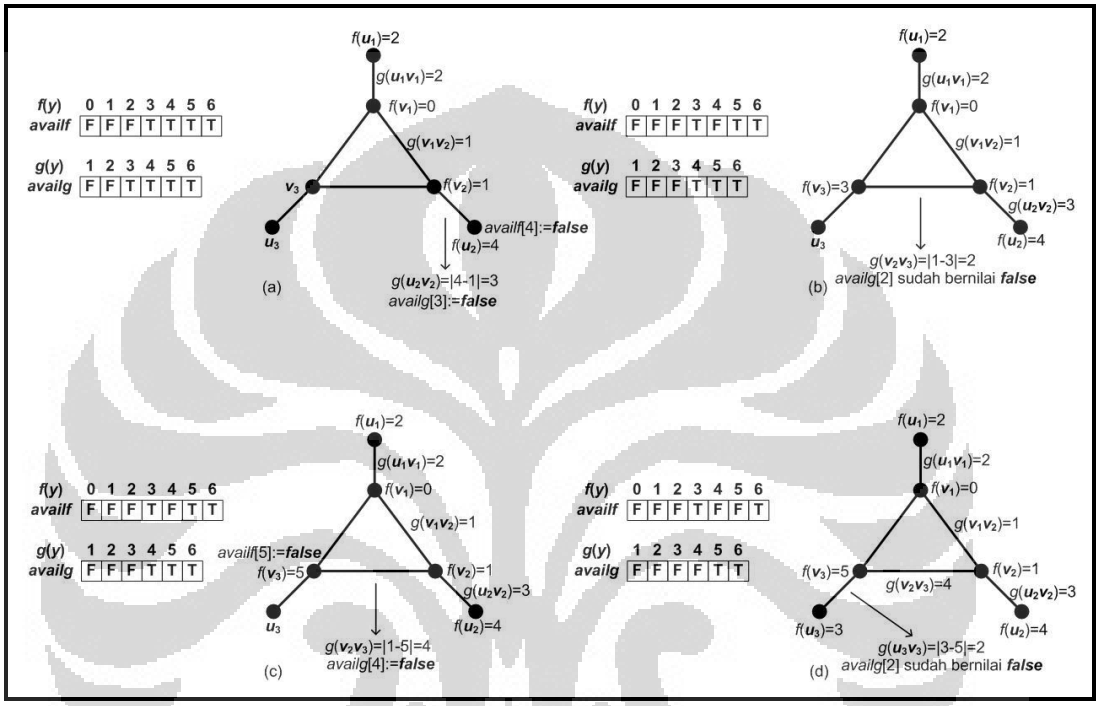
Sebagai contoh penggunaan algoritma pelabelan graceful pada graf lintasan, akan dilakukan pelabelan untuk graf $C_3 \odot \overline{K}_1$. Karena n = 3, maka himpunan bilangan yang digunakan untuk melabelkan simpul adalah {0, 1, 2, 3, 4, 5, 6}, dan yang digunakan untuk melabelkan busur adalah {1, 2, 3, 4, 5, 6}. Maka untuk $f(x)$ disediakan *availf* = [T, T, T, T, T, T, T], dan untuk $g(y)$ *availg* = [T, T, T, T, T, T] (Gambar 3.13 (a)).



Gambar 3.13 Proses pembentukan pelabelan graceful pada graf $C_3 \odot \overline{K}_1$ (1)

Menggunakan fungsi inisial yang diberikan pada algoritma 2.a, yang pertama kali dilabelkan adalah v_1. Simpul v_1 dilabelkan dengan label yang tersedia, misalkan $f(v_1) = 0$, maka *availf*[0] = *false* (Gambar 3.13 (a)). Setelah fungsi inisial melabelkan simpul yang pertama, maka fungsi inisial memanggil fungsi perluasan dengan t = 1. Sehingga selanjutnya u_1 dilabelkan dengan label yang tersedia. Menurut komputasi yang sesungguhnya u_1 harus dilabel dengan label $f(u_1) = 1$ terlebih dahulu, tapi untuk mempersingkat contoh dimisalkan label $f(u_1) = 2$. Setelah itu dapat ditentukan $g(u_1v_1) = |2- 0| = 2$, karena label busur $g(u_1v_1) = 2$ tersedia, maka dapat digunakan dan nyatakan *availf*[2] = *false* dan *availg*[2] = *false* (Gambar 3.13 (b)). Kemudian v_2 dilabelkan dengan label yang tersedia, yaitu $f(v_2) = 1$. Lalu dapat ditentukan $g(v_1v_2) = | 0-1| = 1$, karena label

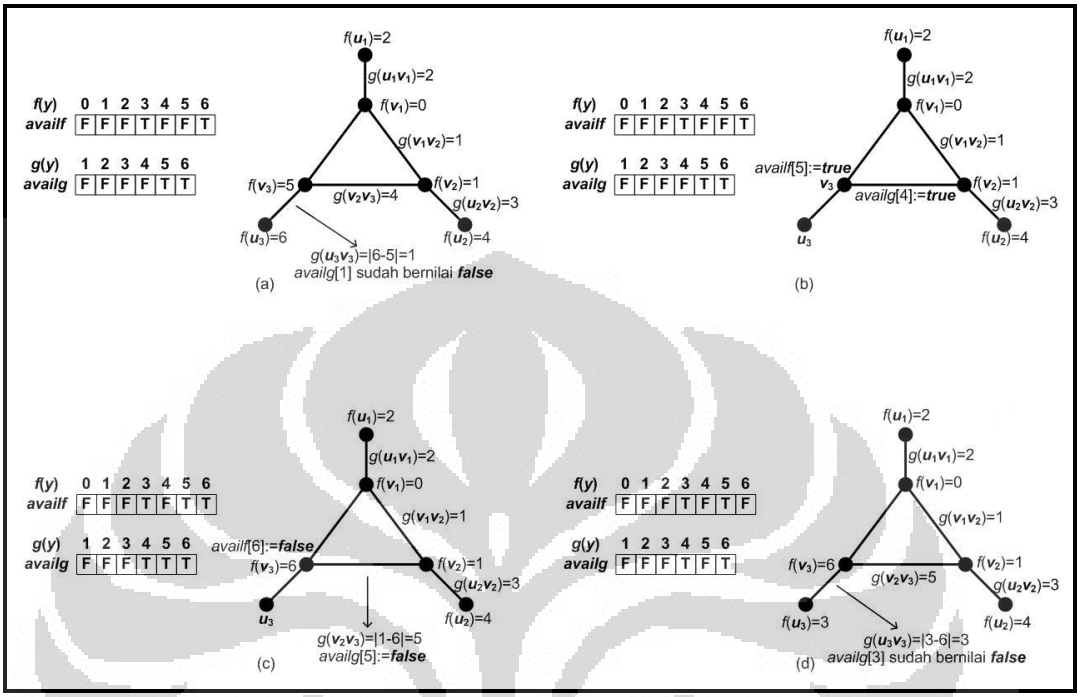
busur $g(v_1v_2) = 1$ tersedia, maka dapat digunakan dan nyatakan *availf*[1] = *false* dan *availg*[1] = *false* (Gambar 3.13 (c)). Fungsi perluasan memanggil dirinya sendiri dengan parameter $t = t + 1 = 2$, sehingga selanjutnya u_2 dilabelkan dengan label yang tersedia. Misalkan u_2 dilabelkan dengan $f(u_2)=3$. Setelah itu dapat ditentukan label busur $g(u_2v_2) = |3\text{-}1| = 2$, ternyata label busur telah terpakai, maka dipakai label lain yang tersedia untuk melabelkan u_2.



Gambar 3.14 Proses pembentukan pelabelan graceful pada graf $C_3 \odot \overline{K}_1$ (2)

Simpul u_2 dilabelkan dengan label lain yang tersedia, yaitu $f(u_2) = 4$. Setelah itu dapat ditentukan $g(u_2v_2) = |4\text{-}1| = 3$, karena label busur $g(u_2v_2) = 3$ tersedia, maka dapat digunakan dan nyatakan *availf*[4] = *false* dan *availg*[3] = *false* (Gambar 3.14 (a)). Kemudian v_3 dilabelkan dengan label yang tersedia, yaitu $f(v_3) = 3$. Setelah itu ditentukan label busur $g(v_2v_3) = |1\text{-}3|=2$, ternyata label telah terpakai, maka dipakai label lain yang tersedia untuk melabelkan v_3 (Gambar 3.14 (b)). Simpul v_3 dilabelkan dengan label lain yang tersedia, yaitu $f(v_3)=5$. Setelah itu dapat ditentukan $g(v_2v_3) = |1\text{-}5|=4$, karena label busur $g(v_2v_3)=4$ tersedia, maka dapat digunakan dan nyatakan *availf*[5] = *false* dan *availg*[4]=*false* (Gambar 3.14 (c)). Fungsi perluasan memanggil dirinya sendiri dengan parameter $t = t +1 = 3$, sehingga selanjutnya u_3 dilabelkan dengan label yang tersedia. Misalkan u_3

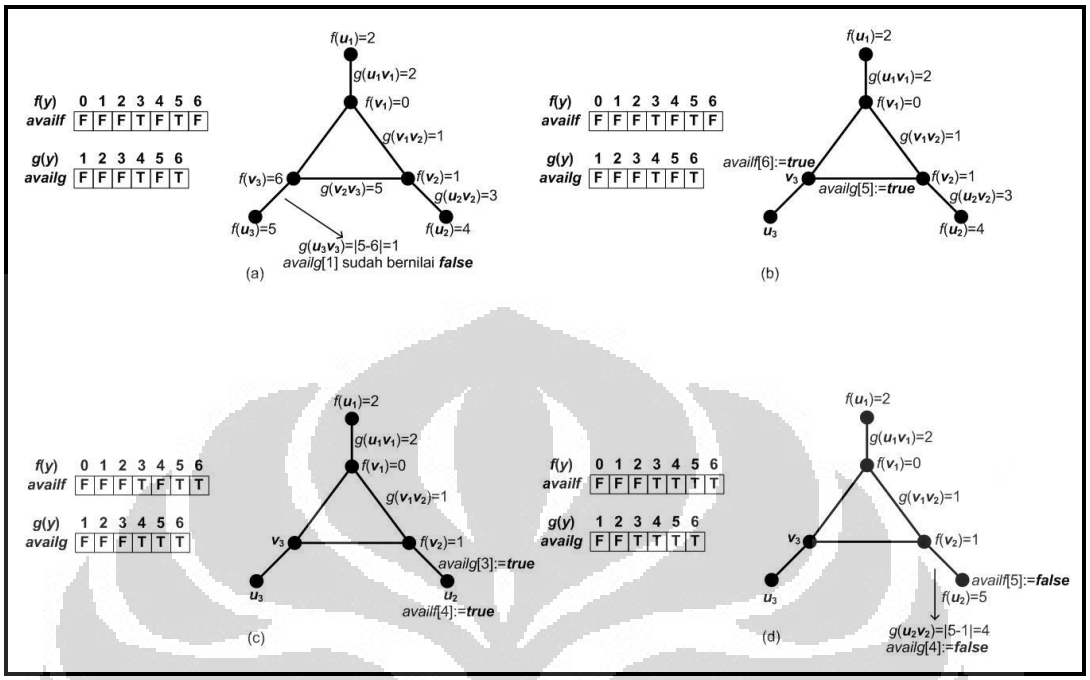
dilabelkan dengan $f(u_3)$=3. Setelah itu dapat ditentukan label busur $g(u_3v_3)$ = |3-5| = 2, ternyata label busur telah terpakai, maka dipakai label lain yang tersedia untuk melabelkan u_3 (Gambar 3.14 (d)).



Gambar 3.15 Proses pembentukan pelabelan graceful pada graf $C_3 \odot \overline{K}_1$ (3)

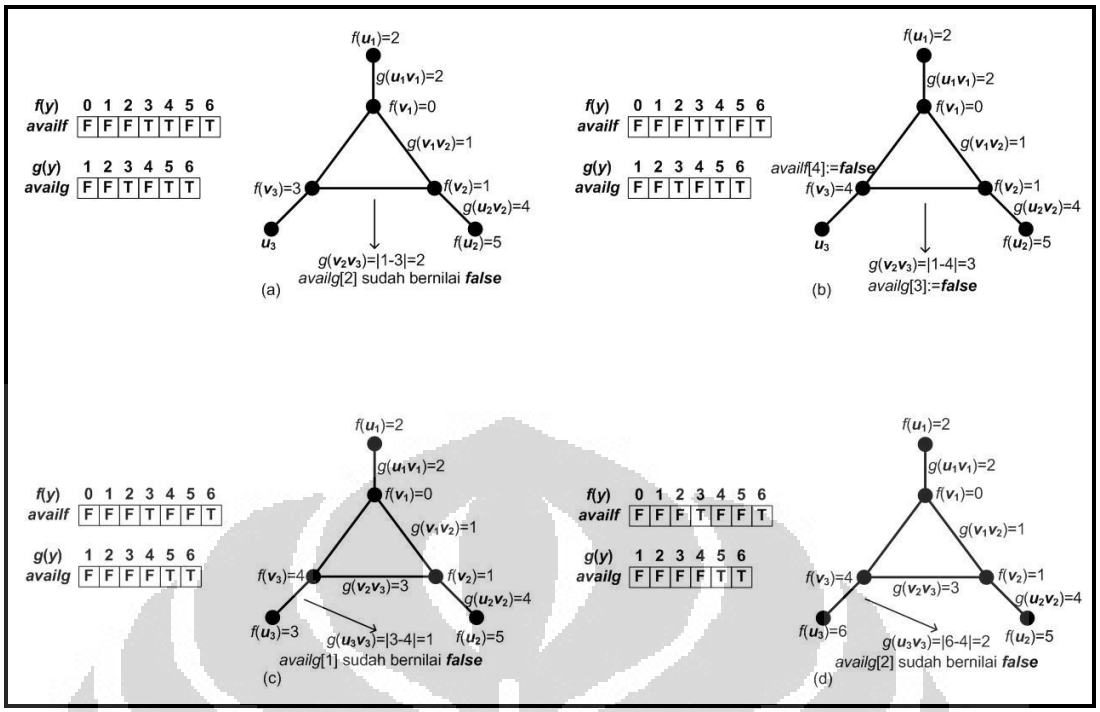
Simpul u_3 dilabelkan dengan label lain yang tersedia, yaitu $f(u_3)$=6. Setelah itu dapat ditentukan label busur $g(u_3v_3)$ = |6-5| =1, ternyata label busur telah terpakai, karena tidak ada label lain yang tersedia untuk melabelkan u_3 maka dilakukan *backtracking* pada v_3 (Gambar 3.15 (a)). Parameter *t* pada fungsi perluasan kembali menjadi *t* = 2 dalam proses *backtracking*. Label yang digunakan untuk melabel v_3 dan busur v_2v_3 statusnya dikembalikan menjadi tersedia, maka nyatakan *availf*[5] = *true* dan *availg*[4] = *true* (Gambar 3.15 (b)). Simpul v_3 dilabelkan dengan label lain yang tersedia, yaitu $f(v_3)$ = 6. Setelah itu dihitung $g(v_2v_3)$ = |1-6|=5, karena label busur $g(v_2v_3)$=5 tersedia, maka dapat digunakan dan nyatakan *availf*[6] = *false* dan *availg*[5]=*false* (Gambar 3.15 (c)). Fungsi perluasan memanggil dirinya sendiri dengan parameter *t* = *t* +1 = 3, sehingga selanjutnya u_3 dilabelkan dengan label yang tersedia. Misalkan u_3 dilabelkan dengan $f(u_3)$ = 3. Setelah itu dapat ditentukan label busur $g(u_3v_3)$ yaitu

$g(u_3v_3)$=|3-6|=3, ternyata label busur telah terpakai, maka dipakai label lain yang tersedia untuk melabelkan u_3 (Gambar 3.15 (d)).



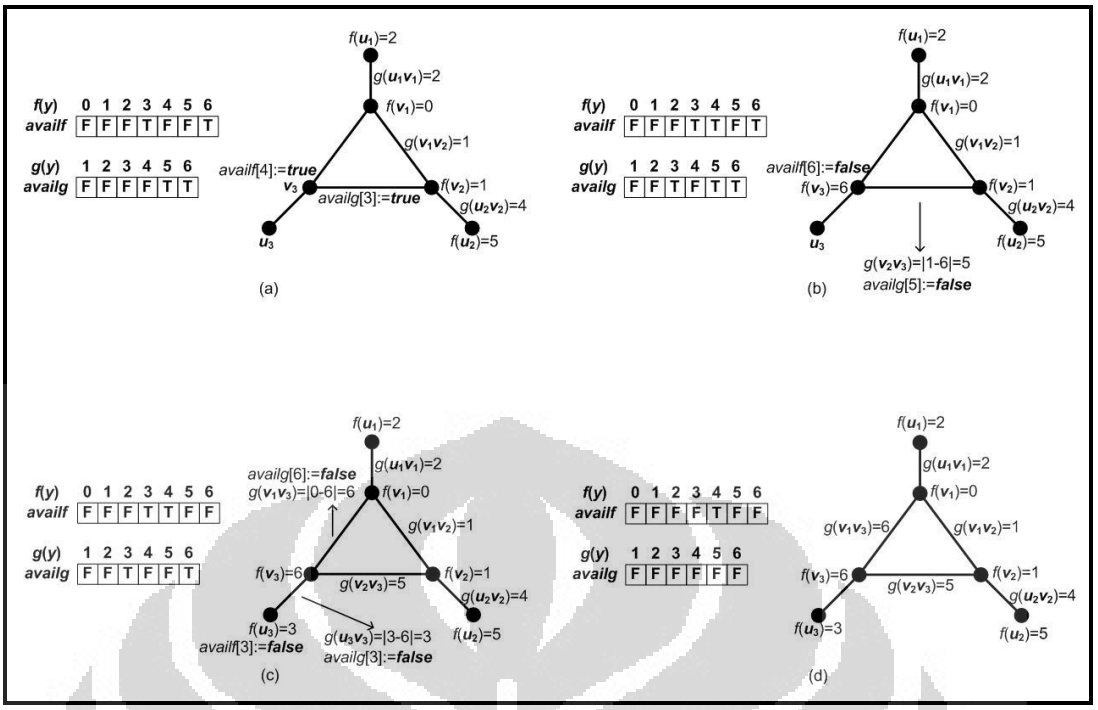
Gambar 3.16 Proses pembentukan pelabelan graceful pada graf $C_3 \odot \overline{K}_1$ (4)

Simpul u_3 dilabelkan dengan label lain yang tersedia, yaitu $f(u_3)$ = 5. Setelah itu dapat ditentukan label busur $g(u_3v_3)$ = |5-6|=1, ternyata label busur telah terpakai, karena tidak ada label lain yang tersedia untuk melabelkan u_3 maka dilakukan *backtracking* pada v_3 (Gambar 3.16 (a)). Parameter *t* pada fungsi perluasan kembali menjadi *t* = 2 dalam proses *backtracking*. Label yang digunakan untuk melabel v_3 dan busur v_2v_3 statusnya dikembalikan menjadi tersedia, maka nyatakan *availf*[6] = *true* dan *availg*[5] = *true* (Gambar 3.16 (b)). Karena tidak ada label lain yang tersedia untuk melabelkan v_3 maka dilakukan *backtracking* pada u_2. Label yang digunakan untuk melabel u_2 dan busur u_2v_2 statusnya dikembalikan menjadi tersedia, maka nyatakan *availf*[4] = *true* dan *availg*[3] = *true* (Gambar 3.16 (c)). Simpul u_2 dilabelkan dengan label lain yang tersedia, yaitu $f(u_2)$=5. Setelah itu dapat ditentukan $g(u_2v_2)$ = |5-1|=4, karena label busur $g(u_2v_2)$=4 tersedia, maka dapat digunakan dan nyatakan *availf*[5] = *false* dan *availg*[4] = *false* (Gambar 3.16 (d)).



Gambar 3.17 Proses pembentukan pelabelan graceful pada graf $C_3 \odot \overline{K}_1$ (5)

Simpul v_3 dilabelkan dengan label yang tersedia, yaitu $f(v_3)$=3. Setelah itu dapat ditentukan label busur $g(v_2v_3)$ = |1-3|=2, ternyata label busur telah terpakai, maka dipakai label lain yang tersedia untuk melabelkan v_3 (Gambar 3.17 (a)). Simpul v_3 dilabelkan dengan label lain yang tersedia, yaitu $f(v_3)$=4. Setelah itu dapat ditentukan $g(v_2v_3)$ = |1-4|=3, karena label busur $g(v_2v_3)$=3 tersedia, maka dapat digunakan dan nyatakan *availf*[4] = *false* dan *availg*[3] = *false* (Gambar 3.17 (b)). Fungsi perluasan memanggil dirinya sendiri dengan parameter $t = t$ +1= 3, sehingga selanjutnya u_3 dilabelkan dengan label yang tersedia. Misalkan u_3 dilabelkan dengan $f(u_3)$ = 3. Setelah itu dapat ditentukan label busur $g(u_3v_3)$ yaitu $g(u_3v_3)$ = |3-4|=1, ternyata label busur telah terpakai, maka dipakai label lain yang tersedia untuk melabelkan u_3 (Gambar 3.17 (c)). Simpul u_3 dilabelkan dengan label lain yang tersedia, yaitu $f(u_3)$ = 6. Setelah itu dapat ditentukan label busur $g(u_3v_3)$ = |6-4|=2, ternyata label busur telah terpakai, karena tidak ada label lain yang tersedia untuk melabelkan u_3 maka dilakukan *backtracking* pada v_3 (Gambar 3.17 (d)).



Gambar 3.18 Proses pembentukan pelabelan graceful pada graf $C_3 \odot \overline{K}_1$ (6)

Parameter *t* pada fungsi perluasan kembali menjadi *t* = 2 dalam proses *backtracking*. Label yang digunakan untuk melabel v_3 dan busur v_2v_3 statusnya dikembalikan menjadi tersedia, maka nyatakan *availf*[4] = *true* dan *availg*[3] = *true* (Gambar 3.18 (a)). Simpul v_3 dilabelkan dengan label lain yang tersedia, yaitu $f(v_3) = 6$. Setelah itu dapat ditentukan $g(v_2v_3) = |1\text{-}6| = 5$, karena label busur $g(v_2v_3) = 5$ tersedia, maka dapat digunakan dan nyatakan *availf*[6] = *false* dan *availg*[5] = *false* (Gambar 3.18 (b)). Fungsi perluasan memanggil dirinya sendiri dengan parameter *t* = *t*+1= 3, sehingga selanjutnya u_3 dilabelkan dengan label yang tersedia. Misalkan u_3 dilabelkan dengan $f(u_3) = 3$. Setelah itu dapat ditentukan $g(u_3v_3) = |3\text{-}6| = 3$, karena label busur $g(u_3v_3) = 3$ tersedia, maka dapat digunakan dan nyatakan *availf*[3] = *false* dan *availg*[3] = *false*. Kemudian dapat ditentukan pula $g(v_1v_3) = |0\text{-}6| = 6$, karena label busur $g(v_1v_3) = 6$ tersedia, maka dapat digunakan dan nyatakan *availg*[6] = *false* (Gambar 3.18 (c)). Maka terbentuklah pelabelan graceful dengan label simpul sebagai berikut, $f(v_1) = 0$, $f(v_2) = 1$, $f(v_3) = 6$, $f(u_1) = 2$, $f(u_2) = 5$, dan $f(u_3) = 3$ (Gambar 3.18 (d)).

Proses komputasi pada algoritma terus berlanjut secara *backtracking* untuk menemukan pelabelan graceful lain yang tidak isomorfik pada graf

$C_3 \odot \bar{K}_1$. Jumlah pelabelan graceful yang tidak isomorfik pada graf $C_3 \odot \bar{K}_1$ adalah 18. Tabel 3.1 merupakan hasil pelabelan graceful yang tidak isomorfik pada graf $C_3 \odot \bar{K}_1$.

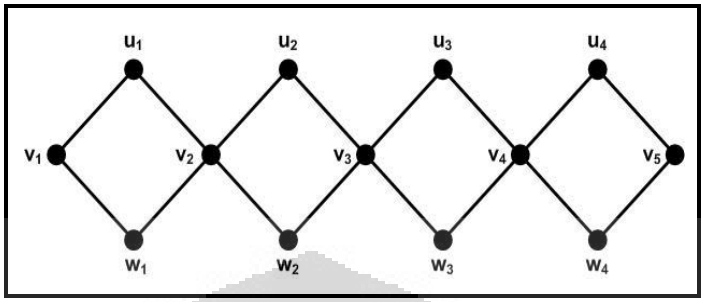
Tabel 3.1 Pelabelan graceful yang tidak isomorfik pada graf $C_3 \odot \bar{K}_1$

	i	1	2	3		*i*	1	2	3
1	*v*	0	1	6	10	*v*	0	3	5
	u	2	5	3		*u*	6	2	1
2	*v*	0	5	6	11	*v*	0	3	5
	u	2	1	3		*u*	6	4	1
3	*v*	0	1	6	12	*v*	0	4	5
	u	3	5	4		*u*	6	1	3
4	*v*	0	4	6	13	*v*	1	2	6
	u	3	5	1		*u*	3	5	0
5	*v*	0	5	6	14	*v*	1	5	6
	u	3	1	4		*u*	3	2	0
6	*v*	0	2	6	15	*v*	1	5	6
	u	5	1	3		*u*	4	3	0
7	*v*	0	1	5	16	*v*	1	3	6
	u	6	3	2		*u*	5	2	0
8	*v*	0	1	5	17	*v*	1	3	6
	u	6	4	3		*u*	5	4	0
9	*v*	0	2	5	18	*v*	1	4	6
	u	6	3	1		*u*	5	3	0

3.3 Algoritma Pelabelan Graceful untuk Graf Ular *k*-C_4

Pada Bab 2 telah diberikan definisi graf ular *k*-C_4. Graf ular *k*-C_4 dapat dipandang sebagai rangkaian *k* graf lingkaran C_4, dengan menghubungkan simpul-simpulnya. Simpul-simpul yang merangkai graf-graf C_4 tersebut dinotasikan dengan v_i, $i = 1, 2, ..., k + 1$. Sedangkan simpul-simpul di atas simpul penghubung dinotasikan dengan u_i, $i = 1, 2, ..., k$. Dan simpul-simpul di bawah simpul penghubung dinotasikan dengan w_i, $i = 1, 2, ..., k$. Pada graf ular *k*-C_4 banyaknya simpul dan busur masing-masing adalah $|V| = 3k + 1$ dan $|E| = 4k$. Pada Gambar 3.19 diberikan contoh graf ular 4-C_4. Yang menjadi masukan dalam algoritma

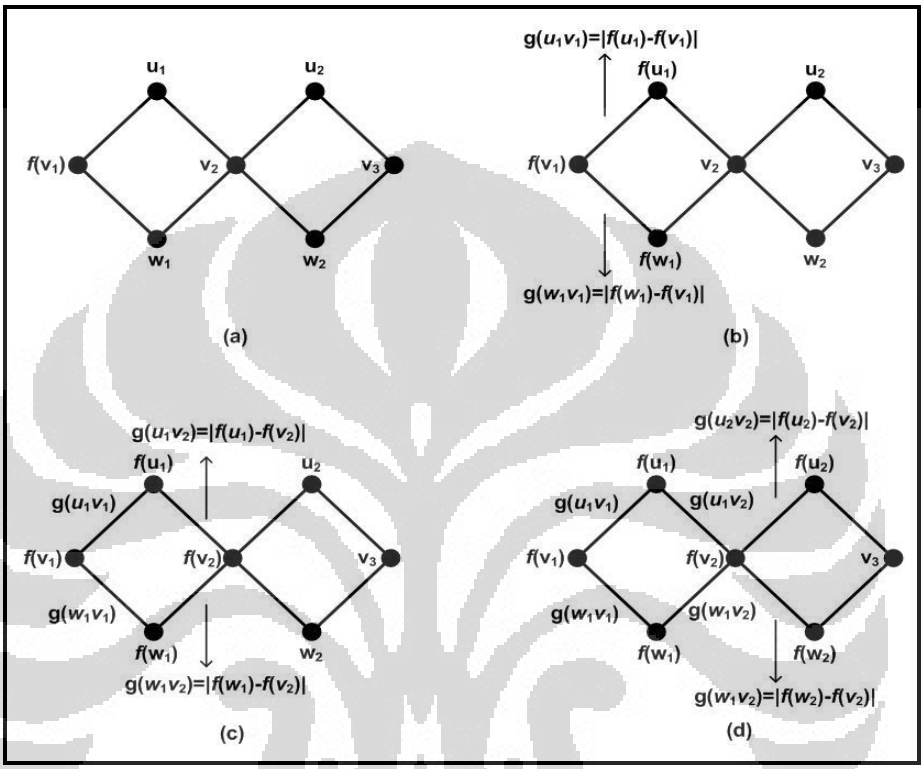
pelabelan graceful untuk graf ular ini adalah *k*. Himpunan label yang tersedia untuk melabelkan simpul adalah {0, 1, 2, ..., 4*k*}, dan himpunan label yang tersedia untuk melabelkan busur adalah {1, 2, 3, ..., 4*k*}.



Gambar 3.19 Graf Ular 4-C_4

Algoritma pelabelan graceful untuk graf ular ini dibagi menjadi 3 fungsi. Yang pertama adalah fungsi inisial (***initializeSnake***), fungsi ini melabelkan simpul inisial v_1 dengan himpunan label simpul yang tersedia, sebut dengan $f(v_1)$. Setiap label yang telah digunakan harus ditandai agar tidak terpakai lagi pada saat pelabelan selanjutnya, maka label $f(v_1)$ ditandai karena telah digunakan. Kemudian fungsi inisialisasi memanggil fungsi perluasan 1 dengan parameter t = 1. Yang kedua adalah fungsi perluasan 1 (***extendSnake1***), menggunakan parameter t untuk merujuk pada simpul yang akan dilabel. Fungsi perluasan 1 melabelkan simpul u_t dengan himpunan label simpul yang tersedia, sebut dengan $f(u_t)$. Setelah melabelkan simpul u_t dapat ditentukan label busur $g(u_tv_t)=|f(u_t)-f(v_t)|$, label busur $g(u_tv_t)$ merupakan *determined label*. Bila label busur $g(u_tv_t)$ tersedia berlanjut ke tahap berikutnya, namun bila tidak tersedia dilakukan *backtracking*. Tahap berikutnya melabelkan simpul w_t dengan himpunan label simpul yang tersedia, sebut dengan $f(w_t)$. Setelah melabelkan simpul w_t dapat ditentukan label busur $g(w_tv_t)=|f(w_t)-f(v_t)|$, label busur $g(w_tv_t)$ merupakan *determined label*. Bila label busur $g(w_tv_t)$ tersedia fungsi perluasan 1 memanggil fungsi perluasan 2 dengan parameter $s = t$ +1, namun bila tidak tersedia dilakukan *backtracking*. Fungsi selanjutnya adalah fungsi perluasan 2 (***extendSnake2***), menggunakan parameter s untuk merujuk pada simpul yang akan dilabel. Fungsi perluasan 2 melabelkan simpul v_s dengan himpunan label simpul yang tersedia, sebut dengan $f(v_s)$. Setelah melabelkan simpul v_s dapat ditentukan label busur $g(u_{s-1}v_s)=|f(u_{s-1})-$

$f(v_s)|$ dan $g(w_{s-1}v_s)=|f(w_{s-1})-f(v_s)|$, label busur $g(u_{s-1}v_s)$ dan $g(w_{s-1}v_s)$ merupakan *determined label*. Bila label busur $g(u_{s-1}v_s)$ dan $g(w_{s-1}v_s)$ tersedia, fungsi perluasan 2 memanggil fungsi perluasan 1 dengan parameter $t = s$, namun bila tidak tersedia dilakukan *backtracking*. Gambar 3.20 merupakan ilustrasi pelabelan graceful pada graf ular k-C_4.



Gambar 3.20 Ilustrasi pembentukan pelabelan graceful pada graf ular k-C_4

Ketika fungsi perluasan 2 memiliki parameter $s = k + 1$, kemudian v_{k+1} telah dilabelkan dan label busur $g(u_kv_{k+1})$ dan $g(w_kv_{k+1})$ yang ditentukan tersedia, maka terbentuk suatu pelabelan graceful untuk graf ular k-C_4. Setelah mendapatkan suatu pelabelan graceful untuk graf ular k-C_4, komputasi tetap dilanjutkan secara *backtracking* untuk mendapatkan pelabelan graceful lainnya. Sama seperti algoritma sebelumnya, setelah suatu label terpakai untuk melabelkan maka label tersebut ditandai agar tidak terpakai lagi, sehingga setiap label hanya terpakai satu kali. Setiap melakukan *backtracking* label simpul yang sebelumnya terpakai statusnya harus diubah menjadi tersedia kembali.

Algoritma 3.a
Fungsi initializeSnake()
for each available label *i* **do**
  $f(v_1):=i$
  availf [*i*]:=**false**
  extendSnake1(1)
  availf [*i*]:=**true**

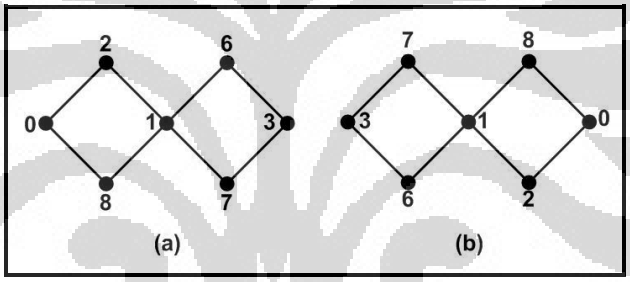
Algoritma 3.b Fungsi extendSnake1 (*t*)
for each available label *e* **do**
  $g(u_tv_t)=|e- f(v_t)|$
  if *availg* [$g(u_tv_t)$] **then**
    $f(u_t):=e$
    availf [*e*]:=**false**
    availg [$g(u_tv_t)$]:=**false**
    for each available *i* **do**
      $g(w_tv_t)=|i - f(v_t)|$
      if *availg* [$g(w_tv_t)$] and **then**
        $f(w_t):=i$
        availf [*i*]:=**false**
        availg [$g(w_tv_t)$]:=**false**
        extendSnake2(t+1)
        availg [$g(w_tv_t)$]:=**true**
        availf [*i*]:=**true**
      end if
      availg [$g(u_tv_t)$]:=**true**
  availf [*e*]:=**true**
  end if

Algoritma 3.c Fungsi extendSnake2 (*s*)
for each available label *i* **do**
  $g(u_{t-1}v_t)=|f(u_t) - i|$
  $g(w_{t-1}v_t)=|f(u_t) - i|$
if *availg* [$g(u_{t-1}v_t)$] and *availg* [$g(w_{t-1}v_t)$] **then**
    $f(v_t):=i$
    availf[*i*]:=**false**
    availg [$g(u_{t-1}v_t)$]:=**false**
    availg [$g(w_{t-1}v_t)$]:=**false**
    if $s = k+1$ **then**
      print()
    else
      extendsnake1(s)
    end if
    availg [$g(u_{t-1}v_t)$]:=**true**
    availg [$g(w_{t-1}v_t)$]:=**true**
    availf [*i*]:=**true**
  end if

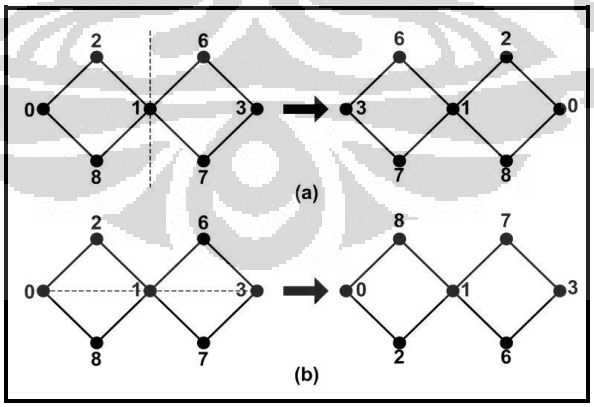
Algoritma pelabelan graceful untuk graf ular k-C_4 diberikan oleh Algoritma 3 yang dibagi menjadi 3 fungsi, yaitu 3.a, 3.b, dan 3.c. Algoritma 3.a merupakan fungsi inisialisasi, sedangkan Algortima 3.b merupakan fungsi perluasan 1 dan Algortima 3.c merupakan fungsi perluasan 2. Pada algoritma-algoritma ini $f(x)$ adalah label yang digunakan untuk melabel simpul x, sedangkan

$g(y)$ adalah label yang digunakan untuk melabelkan busur y. *Availf* dan *availg* dalam algoritma ini merupakan *array* yang digunakan untuk memberikan status pada label. Apabila ***availf* [*f*(*x*)]** bernilai ***true*** artinya label $f(x)$ tersedia, tapi bila bernilai ***false*** artinya label tersebut telah terpakai, begitu pula dengan $g(y)$.

Pada Bab 2 dituliskan bahwa pelabelan isomorfik bisa merupakan kasus rotasi atau refleksi. Jika dilihat dari struktur graf ular, pelabelan isomorfik yang terdapat pada graf ular merupakan kasus rotasi dan refleksi. Dengan memberikan syarat tambahan pada Algoritma 3 agar pelabelan isomorfik tidak terjadi, maka Algoritma 3 akan lebih efisien. Pada Gambar 3.21 diberikan pelabelan graceful yang isomorfik untuk graf ular k-C_4 yang merupakan kasus rotasi. Untuk menghindari pelabelan seperti pada Gambar 3.21, syarat yang harus ditambahkan untuk menghindari pelabelan isomorfik pada kasus ini adalah $f(v_1)<f(v_{k+1})$. Syarat diberikan pada fungsi perluasan 2 saat melabelkan v_{k+1}.



Gambar 3.21 Pelabelan graceful ular k-C_4 yang isomorfik dengan kasus rotasi



Gambar 3.22 Pelabelan graceful ular k-C_4 yang isomorfik dengan kasus refleksi

Pada Gambar 3.22 diberikan pelabelan graceful yang isomorfik untuk graf ular k-C_4 yang merupakan kasus refleksi. Untuk menghindari pelabelan

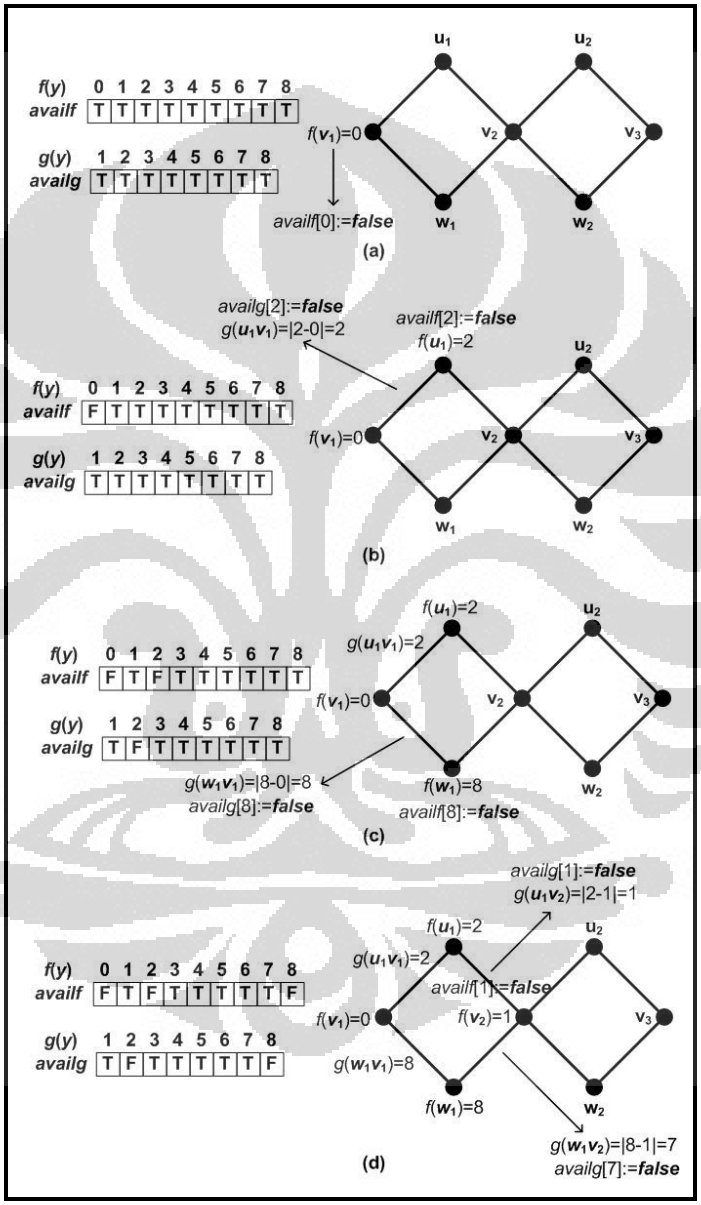
seperti pada Gambar 3.22, syarat yang harus ditambahkan untuk menghindari pelabelan isomorfik pada kasus (b) adalah $f(u_i)<f(w_i)$. $i = 1, 2, ..., k$. Sedangkan kasus yang pertama telah terhindarkan berkat syarat sebelumnya yang diberikan untuk mencegah terjadinya kasus rotasi.

```
1. Algoritma 3.a
   Fungsi initializeSnake()
2. for each available label i where
     0≤i≤4k-1 do
3.   f(v1):=i
4.   availf [i]:=false
5.   extendSnake1(1)
6.   availf [i]:=true
```

```
1.  Algoritma 3.b Fungsi extendSnake1 (t)
2.  for each available label e do
3.    g(utvt)=|e - f(vt)|
4.    if availg [g(utvt)] then
5.      f(ut):=e
6.      availf [e]:=false
7.      availg [g(utvt)]:=false
8.    for each available i do
9.      g(wtvt)=|i- f(vt)|
10.     if availg [g(wtvt)] and f(wt)>f(ut) then
11.       f(wt):=i
12.       availf [i]:=false
13.       availg [g(wtvt)]:=false
14.       extendSnake2(t+1)
15.       availg [g(wtvt)]:=true
16.       availf [i]:=true
          end if
17.     availg [g(utvt)]:=true
18.     availf [e]:=true
        end if
```

```
1.  Algoritma 3.c Fungsi extendSnake2 (s)
2.  for each available label i do
3.    g(us-1vs)=|f(us-1) - i|
4.    g(ws-1vs)=|f(ws-1) - i|
5.  if availg [g(us-1vs)] and availg [g(ws-1vs)] then
6.      f(vs):=i
7.      availf[i]:=false
8.      availg [g(us-1vs)]:=false
9.      availg [g(ws-1vs)]:=false
10.     if s = k+1 then
11.       if f(v1)<f(vs) then
12.         print()
13.       end if
14.     else
15.       extendsnake1(s)
        end if
16.   availg [g(us-1vs)]:=true
17.   availg [g(ws-1vs)]:=true
18.   availf [i]:=true
end if
```

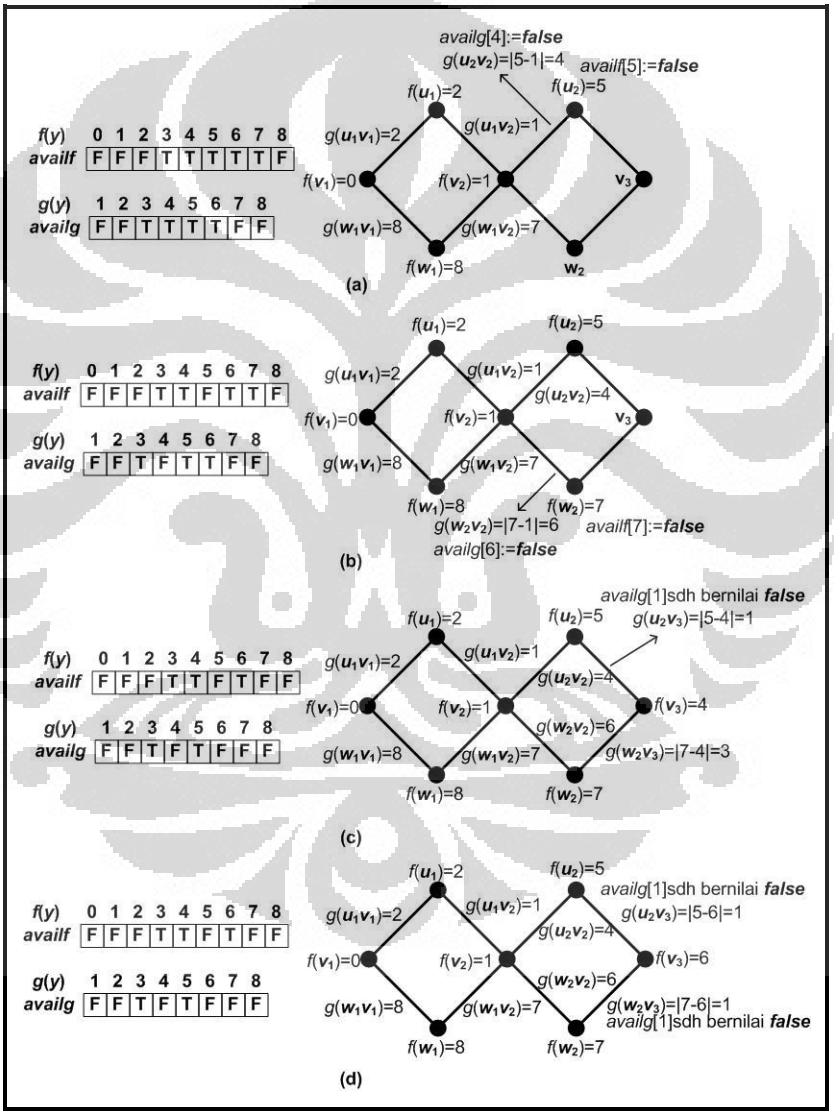
Sebagai contoh penggunaan algoritma pelabelan graceful pada graf ular *k*-C_4, akan dilakukan pelabelan untuk graf ular 2-C_4. Karena $k = 2$, maka himpunan bilangan yang digunakan untuk melabelkan simpul adalah {0, 1, 2, 3, 4, 5, 6, 7, 8}, dan yang digunakan untuk melabelkan busur adalah {1, 2, 3, 4, 5, 6, 7, 8}. Maka untuk $f(x)$ disediakan *availf* = [T, T, T, T, T, T, T, T, T], dan untuk $g(y)$ *availg* = [T, T, T, T, T, T, T, T] (Gambar 3.23 (a)).



Gambar 3.23 Proses pembentukan pelabelan graceful pada graf ular 2-C_4 (1)

Menggunakan fungsi inisial yang diberikan pada algoritma 3.a, yang pertama kali dilabelkan adalah v_1. Simpul v_1 dilabelakan dengan label yang

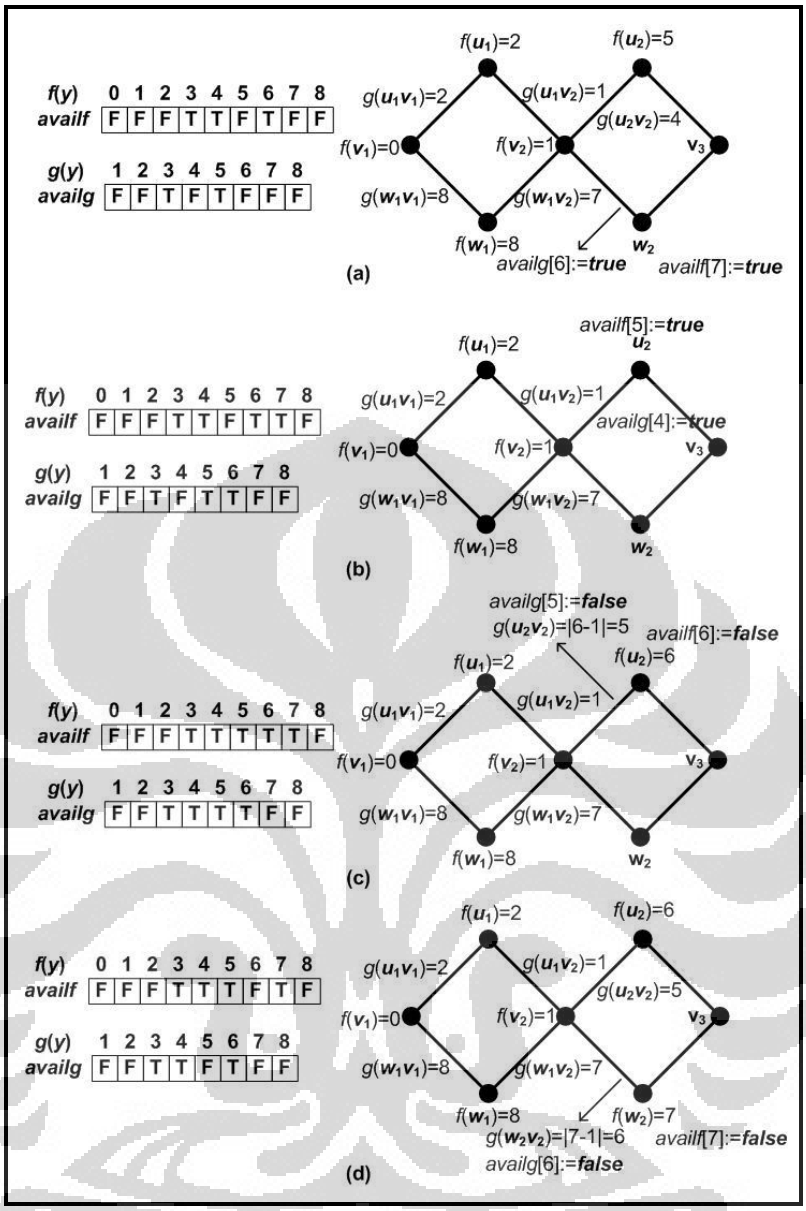
tersedia, misalkan $f(v_1) = 0$, maka *availf*[0] = *false* (Gambar 3.23 (a)). Setelah melabelkan v_1 sungsi inisial akan memanggil fungsi perluasan 1 dengan parameter $t = 1$. Sehingga selanjutnya u_1 dilabelkan dengan label yang tersedia. Menurut komputasi yang sesungguhnya u_1 harus dilabel dengan label $f(u_1) = 1$ terlebih dahulu, tapi untuk mempersingkat contoh dimisalkan label $f(u_1) = 2$. Setelah itu dapat ditentukan $g(u_1v_1) = |0-2|=2$, karena label busur $g(u_1v_1)=2$ tersedia, maka dapat digunakan dan nyatakan *availf*[2] = *false* dan *availg*[2] = *false* (Gambar 3.23 (b)).



Gambar 3.24 Proses pembentukan pelabelan graceful pada graf ular 2-C_4 (2)

Kemudian w_1 dilabelkan dengan label yang tersedia. Menurut komputasi yang sesungguhnya w_1 harus dilabel dengan label $f(w_1) = 1$ terlebih dahulu, tapi untuk mempersingkat contoh dimisalkan label $f(w_1) = 8$. Setelah itu dapat ditentukan $g(w_1v_1) = |0\text{-}8|=8$, karena label busur $g(w_1v_1)=8$ tersedia, maka dapat digunakan dan nyatakan *availf*[8] = *false* dan *availg*[8] = *false* (Gambar 3.23 (c)). Fungsi perluasan 1 memanggil fungsi perluasan 2 dengan parameter $s = t + 1 = 2$. Selanjutnya simpul v_2 dilabelkan dengan label yang tersedia, yaitu $f(v_2) = 1$. Kemudian dapat ditentukan $g(u_1v_2) = |2\text{-}1|=1$ dan $g(w_1v_2) = |8\text{-}1|=7$. Karena label busur $g(u_1v_2) =1$ dan $g(w_1v_2) = 7$ tersedia, maka dapat digunakan dan nyatakan *availf*[1] = *false*, *availg*[1] = *false* dan *availg*[7] = *false* (Gambar 3.23 (d)).

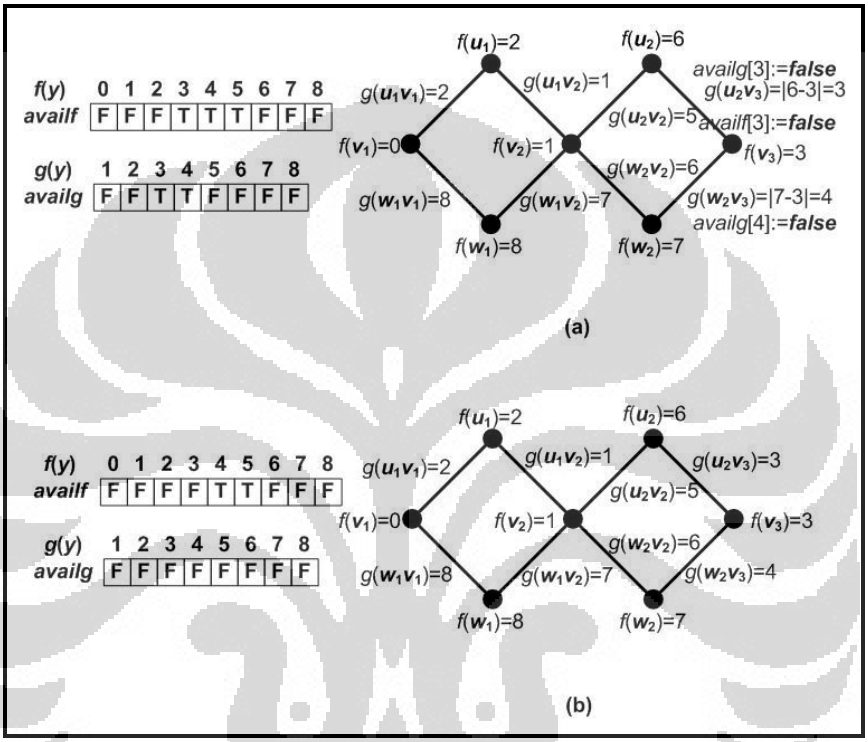
Fungsi perluasan 2 memanggil fungsi perluasan 1 dengan parameter $t = s = 2$. Sehingga selanjutnya u_2 dilabelkan dengan label yang tersedia. Menurut komputasi yang sesungguhnya u_2 harus dilabel dengan label $f(u_2) = 3$ terlebih dahulu, tapi untuk mempersingkat contoh dimisalkan label $f(u_2) = 5$. Setelah itu dapat ditentukan $g(u_2v_2) = |5\text{-}1|=4$, karena label busur $g(u_2v_2)=4$ tersedia, maka dapat digunakan dan nyatakan *availf*[5] = *false* dan *availg*[4] = *false* (Gambar 3.24 (a)). Kemudian w_2 dilabelkan dengan label yang tersedia. Menurut komputasi yang sesungguhnya w_2 harus dilabel dengan label $f(w_2) = 6$ terlebih dahulu, tapi untuk mempersingkat contoh dimisalkan label $f(w_2) = 7$. Setelah itu dapat ditentukan $g(w_2v_2) = |7\text{-}1|=6$, karena label busur $g(w_2v_2)=6$ tersedia, maka dapat digunakan dan nyatakan *availf*[7] = *false* dan *availg*[6]=*false* (Gambar 3.24 (b)). Fungsi perluasan 1 memanggil fungsi perluasan 2 dengan parameter $s = t + 1 = 3$. Selanjutnya simpul v_3 dilabelkan dengan label yang tersedia, yaitu $f(v_3) = 4$. Kemudian dapat ditentukan $g(u_2v_3) = |5\text{-}4|=1$ dan $g(w_2v_3) = |7\text{-}4|=3$ (Gambar 3.24 (c)). Karena label busur $g(u_2v_3)$ tidak tersedia maka v_3 dilabelkan dengan label lain yang tersedia, yaitu $f(v_3) = 4$. Kemudian dapat ditentukan $g(u_2v_3) = |5\text{-}6|=1$ dan $g(w_2v_3) = |7\text{-}6|=1$. Karena label busur $g(u_2v_3)$ dan $g(w_2v_3)$ tidak tersedia dan tidak ada label lain yang tersedia untuk melabelkan v_3, maka dilakukan *backtracking* pada simpul w_2 (Gambar 3.24 (d)).



Gambar 3.25 Proses pembentukan pelabelan graceful pada graf ular 2-C_4 (3)

Label yang digunakan untuk melabel w_2 dan busur v_2w_2 statusnya dikembalikan menjadi tersedia, maka nyatakan *availf*[7] = *true* dan *availg*[6] = *true* (Gambar 3.25 (a)). Karena tidak ada label lain yang tersedia untuk melabelkan w_2, maka dilakukan *backtraking* pada simpul u_2. Label yang digunakan untuk melabel u_2 dan busur v_2u_2 statusnya dikembalikan menjadi tersedia, maka nyatakan *availf*[5] = *true* dan *availg*[4] = *true* (Gambar 3.25 (b)). Selanjutnya u_2 dilabelkan dengan label yang tersedia, yaitu $f(u_2) = 6$. Setelah itu dapat ditentukan $g(u_2v_2) = |6\text{-}1|=5$, karena label busur $g(u_2v_2)=5$ tersedia, maka

dapat digunakan dan nyatakan *availf*[6] = *false* dan *availg*[5]=*false* (Gambar 3.25 (c)). Kemudian w_2 dilabelkan dengan label yang tersedia, yaitu $f(w_2) = 7$. Setelah itu dapat ditentukan $g(w_2v_2) = |7\text{-}1|=6$, karena label busur $g(w_2v_2)=6$ tersedia, maka dapat digunakan dan nyatakan *availf*[7] = *false* dan *availg*[6]=*false* (Gambar 3.24 (d)). Fungsi perluasan 1 memanggil fungsi perluasan 2 dengan parameter $s = t + 1 = 3$.



Gambar 3.26 Proses pembentukan pelabelan graceful pada graf ular 2-C_4 (4)

Selanjutnya simpul v_3 dilabelkan dengan label yang tersedia, yaitu $f(v_3) = 3$. Kemudian dapat ditentukan $g(u_2v_3) = |6\text{-}3|=3$ dan $g(w_2v_3) = |\ 7\text{-}3|=4$. Karena label busur $g(u_2v_3)$ dan $g(w_2v_3)$ tersedia, maka dapat digunakan dan nyatakan *availf*[3] = *false*, *availg*[3] = *false* dan *availg*[4] = *false* (Gambar 3.26 (a)). Karena seluruh simpul dan busur telah dilabel, maka terbentuklah suatu pelabelan graceful untuk graf ular 2-C_4. Label-label simpulnya adalah sebagai berikut, $f(u_1) = 2$, $f(u_2) = 6$, $f(v_1) = 0$, $f(v_2) = 1$, $f(v_3) = 3$, $f(w_1) = 8$, dan $f(w_2) = 7$. Pelabelan graceful lainnya untuk graf ular 2-C_4 dapat diperoleh dengan proses *backtracking*. Jumlah pelabelan graceful yang tidak isomorfik untuk graf ular 2-C_4 adalah 70.

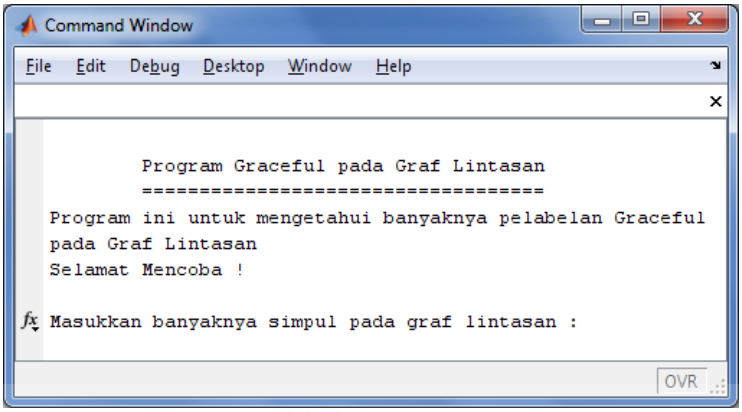
BAB 4
IMPLEMENTASI DAN SIMULASI

Pada Bab 3 telah dijelaskan mengenai pembentukan algoritma pelabelan graceful pada graf lintasan, matahari dan ular kC_4. Pada bab ini akan diberikan implementasi dari algoritma-algoritma tersebut menggunakan MATLAB beserta simulasinya. Program dijalankan pada *NoteBook* dengan prosesor *Intel Pentium Dual-Core* T3200 2 GHz, memori 1 GB, dan sistem operasi *Windows7 Professional*. Keluaran dari program adalah pelabelan graceful yang tidak isomorfik pada graf yang diberikan dan banyaknya pelabelan yang tidak isomorfik tersebut. Secara teoritis, program yang dibuat dapat menghasilkan semua pelabelan yang tidak isomorfik untuk sembarang *n*. Namun simulasi hanya dijalankan sampai *n* atau *k* tertentu saja.

4.1 Implementasi dan Simulasi Algoritma Pelabelan Graceful pada Graf Lintasan

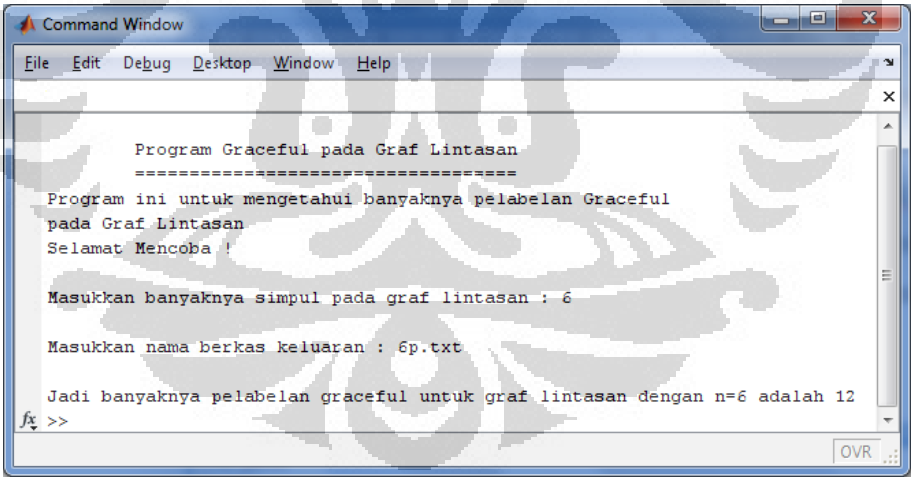
Pada algoritma pelabelan graceful yang diberikan pada Subbab 3.1, yang pertama dilakukan adalah tahap inisialisasi. Tahap inisialisasi tersebut dijalankan oleh fungsi *initializePath* yg diimplementasikan dalam program dengan nama "path". Sedangkan untuk fungsi *extendPath* diberi nama "extendpath". Listing dari program dapat dilihat di Lampiran 1 (CD).

Pemanggilan program dilakukan dengan cara mengetik "path" pada *Command Window*. Setelah memanggil program, tampilan yang muncul pada *command window* adalah seperti pada Gambar 4.1. Pengguna diminta untuk memasukkan banyaknya simpul ($n \geq 1$) pada graf lintasan yang ingin dilabelkan. Nilai *n* pada graf lintasan menunjukan banyaknya simpul pada graf tersebut.



Gambar 4.1 Tampilan awal *command window* pada program pelabelan graceful graf lintasan

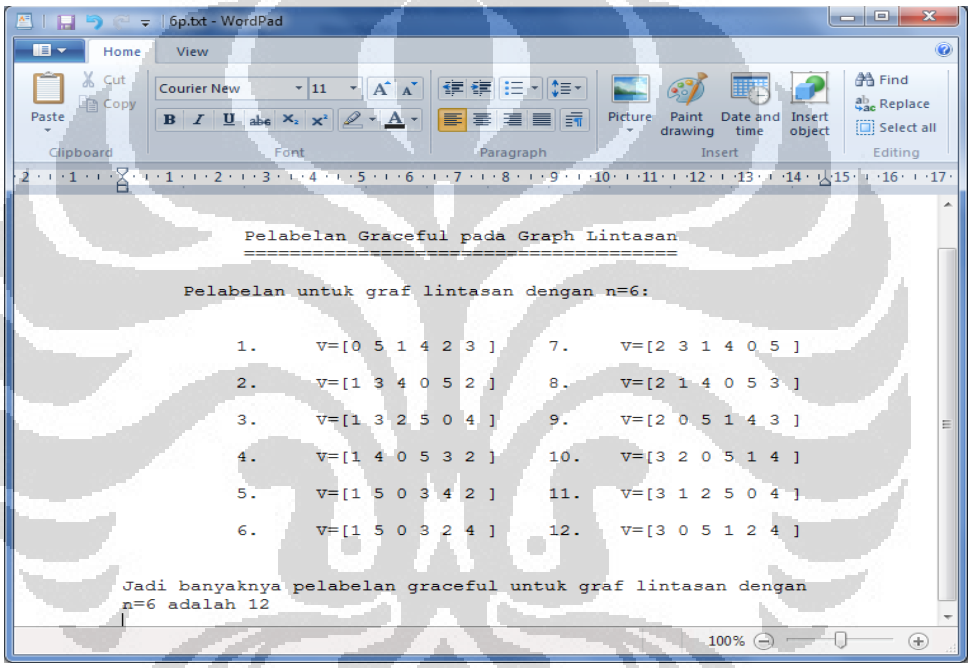
Selanjutnya program meminta pengguna untuk memasukkan nama berkas keluaran yang akan menyimpan semua pelabelan yang dihasilkan. Program akan berhenti apabila telah dihasilkan semua pelabelan graceful yang tidak isomorfik, pada graf lintasan dengan nilai *n* yang dimasukkan oleh pengguna. Program juga mengeluarkan banyaknya pelabelan graceful yang dihasilkan dan mencetak semua pelabelan yang diperoleh dalam berkas keluaran.



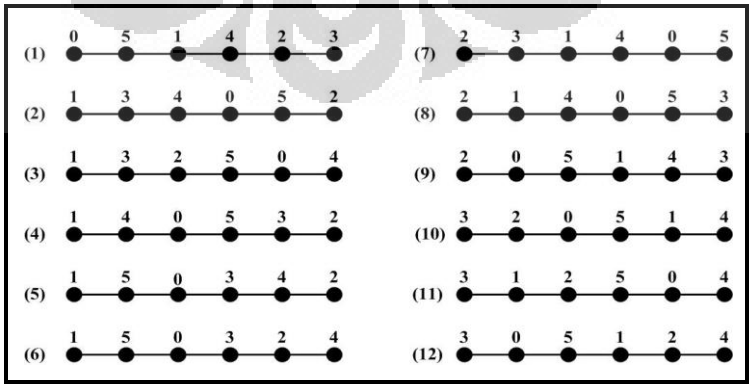
Gambar 4.2 Tampilan *command window* pada program pelabelan graceful pada graf P_6

Pada Gambar 4.2 diberikan contoh pemanggilan program pelabelan graceful pada graf lintasan, untuk mencari banyaknya pelabelan graceful yang tidak isomorfik pada graf P_6. Dalam hal ini masukan yang diberikan oleh

pengguna adalah $n = 6$. Kemudian program meminta pengguna untuk memberikan masukan nama dari berkas keluaran yaitu 6p.txt. Dari Gambar 4.2 terlihat bahwa banyaknya pelabelan graceful tidak isomorfik pada graf P_6 adalah 12. Pada keluaran dari pelabelan graceful pada graf P_6, yang dikeluarkan hanyalah label-label simpul saja, karena label-label busur akan diinduksi oleh label-label simpul. Label-label simpul direpresentasikan oleh *V*. Isi dari berkas keluaran 6p.txt diberikan pada Gambar 4.3. Untuk nilai *n* yang kecil bisa dilihat bahwa label-label yang dihasilkan tidak isomorfik. Agar lebih jelas representasi visual dari label-label tersebut diberikan oleh Gambar 4.4.



Gambar 4.3 Isi dari berkas keluaran 6p.txt

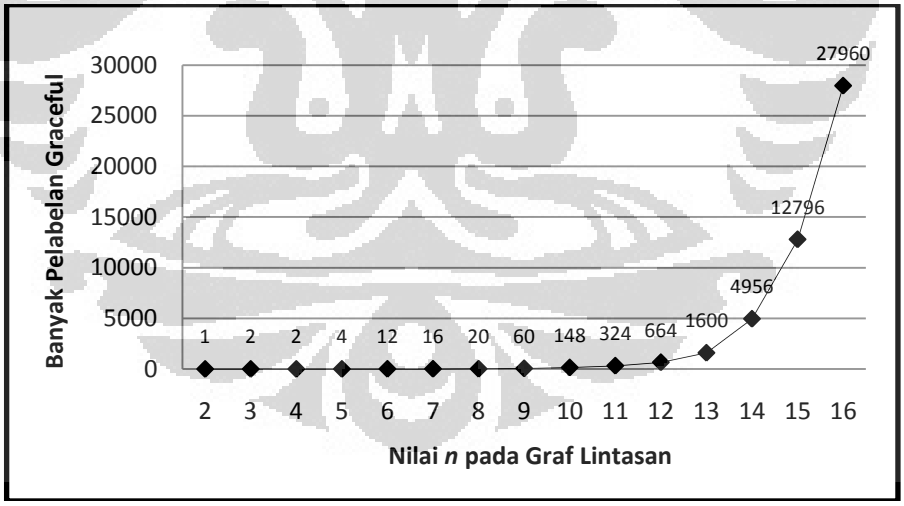


Gambar 4.4 Representasi visual peleblan graceful pada graf P_6

Simulasi dari program pelabelan graceful pada graf lintasan P_n dijalankan untuk mengetahui berapa banyak pelabelan graceful tidak isomorfik yang mungkin untuk setiap $n > 1$. Simulasi dijalankan sampai $n = 16$. Hasil simulasi diberikan pada Tabel 4.1 yang menunjukkan banyaknya pelabelan graceful tidak isomorfik pada graf lintasan P_n.

Tabel 4.1 Banyaknya pelabelan graceful tidak isomorfik pada graf lintasan P_n untuk $n = 2$ s.d. $n = 16$

n	Banyaknya Pelabelan Graceful	n	Banyaknya Pelabelan Graceful
2	1	10	148
3	2	11	324
4	2	12	664
5	4	13	1600
6	12	14	4956
7	16	15	12796
8	20	16	27960
9	60		



Gambar 4.5 Grafik pertambahan banyaknya pelabelan graceful tidak isomorfik pada graf lintasan P_n untuk $n = 2$ s.d. $n = 16$

Pada Gambar 4.5 diberikan grafik yang menunjukkan pertambahan banyaknya pelabelan graceful tidak isomorfik yang terjadi pada graf lintasan P_n sesuai dengan bertambahnya nilai n. Dari gambar tersebut terlihat bahwa dengan

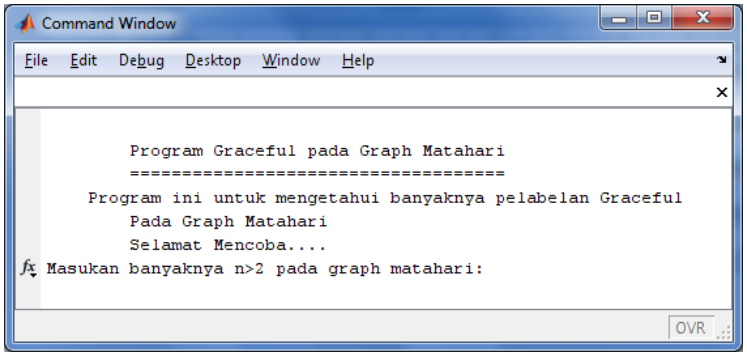
bertambahnya nilai *n*, banyaknya pelabelan graceful tidak isomorfik pada graf lintasan P_n yang dihasilkan juga semakin bertambah, kecuali untuk $n = 3$ dan $n = 4$ banyaknya pelabelan graceful tidak isomorfik adalah sama, yaitu 2. Kecenderungan pertambahan banyaknya pelabelan graceful tidak isomorfik yang diperoleh seperti grafik eksponensial sedemikian sehingga jika untuk *n* yang lebih besar disimulasikan maka waktu komputasi yang dibutuhkan akan lebih lama. Hal ini disebabkan karena setiap penambahan satu nilai *n* maka akan ada penambahan satu busur dan simpul yang harus dilabel.

Pada subbab selanjutnya akan diberikan implementasi dan simulasi algoritma pelabelan graceful pada graf matahari.

4.2 Implementasi dan Simulasi Algoritma Pelabelan Graceful pada Graf Matahari

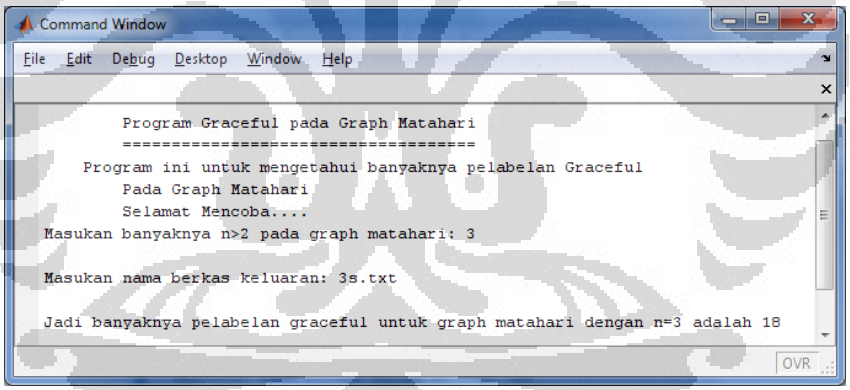
Pada algoritma pelabelan graceful yang diberikan pada Subbab 3.2, yang pertama dilakukan adalah tahap inisialisasi. Tahap inisialisasi tersebut dijalankan oleh fungsi *initializeSun* yg diimplementasikan dalam program dengan nama "sun". Sedangkan untuk fungsi *extendSun* diberi nama "extendsun". Listing dari program dapat dilihat di Lampiran 2 (CD).

Pemanggilan program dilakukan dengan cara mengetik "sun" pada *Command Window*. Setelah memanggil program, tampilan yang muncul pada *command window* adalah seperti pada Gambar 4.6. Pengguna diminta untuk memasukkan nilai *n* ($n \geq 2$) pada graf matahari yang ingin dilabelkan. Nilai *n* pada graf matahari menyatakan banyaknya simpul yang membentuk graf lingkaran C_n dari graf matahari tersebut.



Gambar 4.6 Tampilan awal *command window* pada program pelabelan graceful pada graf matahari

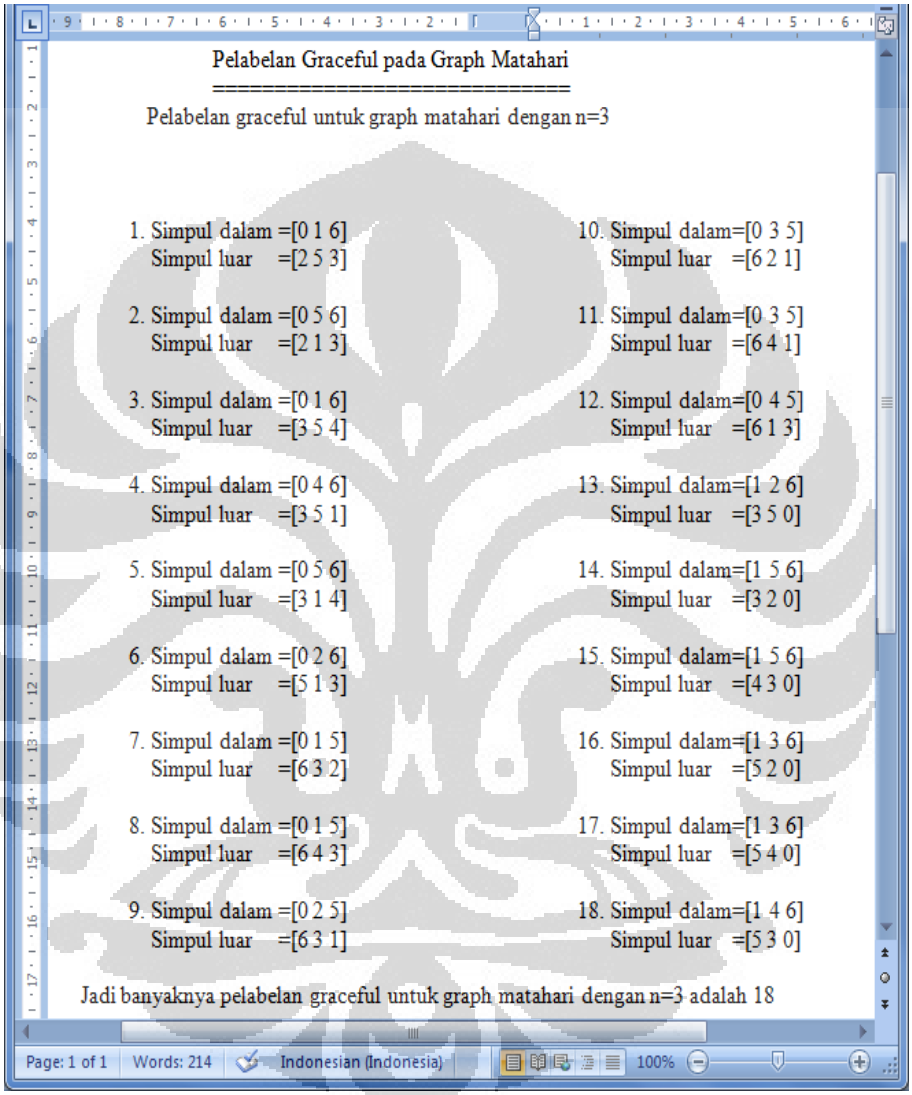
Selanjutnya program meminta pengguna untuk memasukkan nama berkas keluaran yang akan menyimpan semua pelabelan yang dihasilkan. Program akan berhenti apabila telah dihasilkan semua pelabelan graceful yang tidak isomorfik, pada graf matahari dengan nilai *n* yang dimasukkan oleh pengguna. Program juga mengeluarkan banyaknya pelabelan graceful yang dihasilkan dan mencetak semua pelabelan yang diperoleh dalam berkas keluaran.



Gambar 4.7 Tampilan *command window* pada program pelabelan graceful pada graf matahari dengan $n = 3$

Pada Gambar 4.7 diberikan contoh pemanggilan program pelabelan graceful pada graf matahari, untuk mencari banyaknya pelabelan graceful yang tidak isomorfik pada graf matahari dengan $n = 3$. Dalam hal ini masukan yang diberikan oleh pengguna adalah $n = 3$. Kemudian program meminta pengguna untuk memberikan masukan nama dari berkas keluaran yaitu 3s.txt. Dari Gambar 4.7 terlihat bahwa banyaknya pelabelan graceful tidak isomorfik pada graf

matahari dengan n = 3 adalah 18. Pada keluaran dari pelabelan graceful pada graf matahari dengan n = 3, dihasilkan label-label simpul dalam dan simpul luar. Isi dari berkas keluaran 3s.txt diberikan pada Gambar 4.8. Untuk nilai n yang kecil bisa dilihat bahwa label-label yang dihasilkan tidak isomorfik.Agar lebih jelas representasi visual dari label-label tersebut diberikan oleh Gambar 4.9.

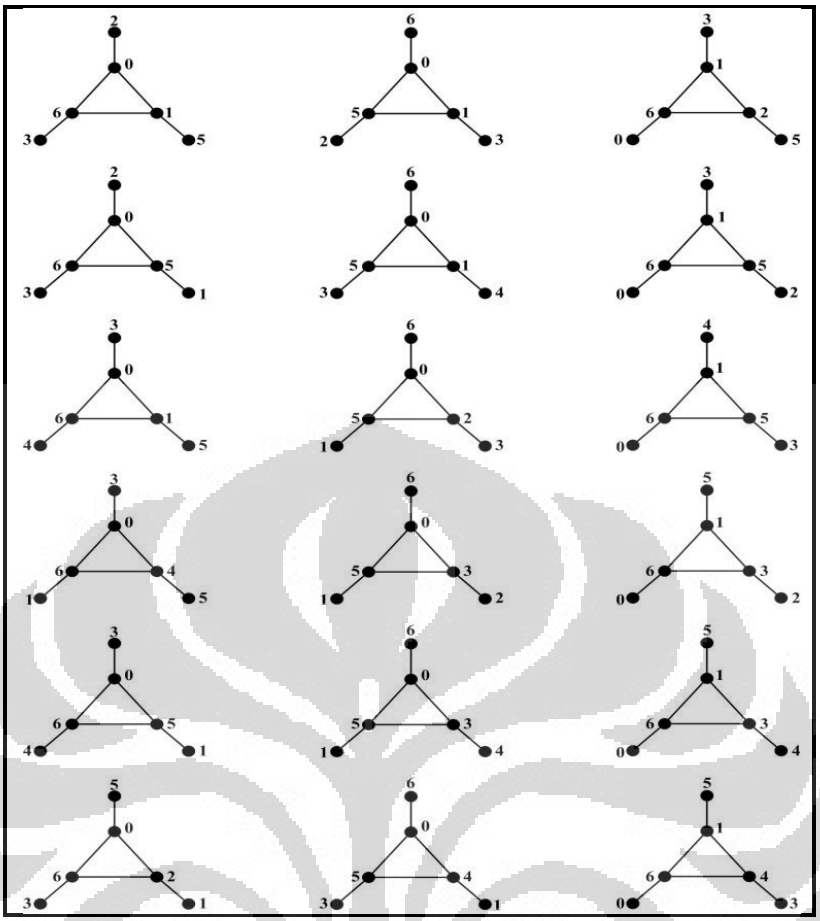

Pelabelan Graceful pada Graph Matahari
==========
Pelabelan graceful untuk graph matahari dengan n=3

1. Simpul dalam =[0 1 6]
   Simpul luar =[2 5 3]
2. Simpul dalam =[0 5 6]
   Simpul luar =[2 1 3]
3. Simpul dalam =[0 1 6]
   Simpul luar =[3 5 4]
4. Simpul dalam =[0 4 6]
   Simpul luar =[3 5 1]
5. Simpul dalam =[0 5 6]
   Simpul luar =[3 1 4]
6. Simpul dalam =[0 2 6]
   Simpul luar =[5 1 3]
7. Simpul dalam =[0 1 5]
   Simpul luar =[6 3 2]
8. Simpul dalam =[0 1 5]
   Simpul luar =[6 4 3]
9. Simpul dalam =[0 2 5]
   Simpul luar =[6 3 1]
10. Simpul dalam=[0 3 5]
    Simpul luar =[6 2 1]
11. Simpul dalam=[0 3 5]
    Simpul luar =[6 4 1]
12. Simpul dalam=[0 4 5]
    Simpul luar =[6 1 3]
13. Simpul dalam=[1 2 6]
    Simpul luar =[3 5 0]
14. Simpul dalam=[1 5 6]
    Simpul luar =[3 2 0]
15. Simpul dalam=[1 5 6]
    Simpul luar =[4 3 0]
16. Simpul dalam=[1 3 6]
    Simpul luar =[5 2 0]
17. Simpul dalam=[1 3 6]
    Simpul luar =[5 4 0]
18. Simpul dalam=[1 4 6]
    Simpul luar =[5 3 0]

Jadi banyaknya pelabelan graceful untuk graph matahari dengan n=3 adalah 18

Page: 1 of 1 Words: 214 Indonesian (Indonesia) 100%

Gambar 4.8 Isi dari berkas keluaran 3s.txt

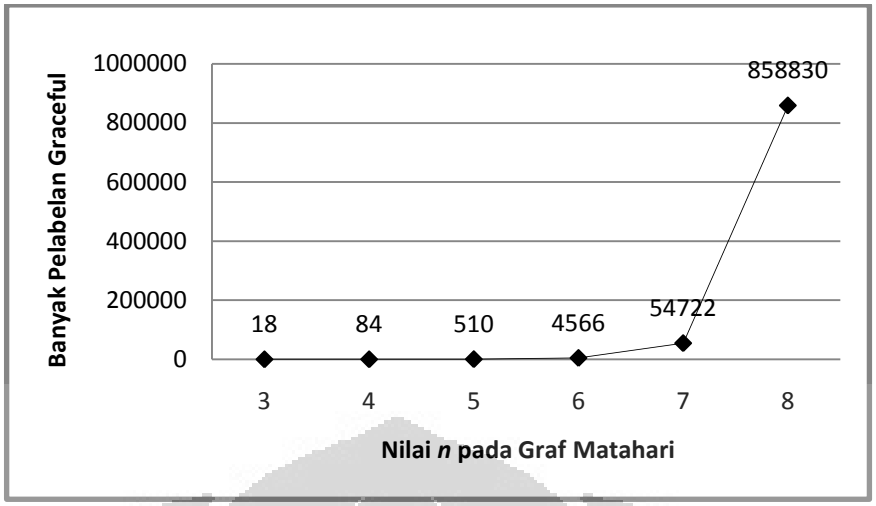


Gambar 4.9 Representasi visual pelabelan graceful pada graf matahari, $n = 3$

Simulasi dari program pelabelan graceful pada graf matahari dijalankan untuk mengetahui berapa banyak pelabelan graceful tidak isomorfik yang mungkin untuk setiap $n > 2$. Program ini bisa disimulasikan untuk sembarang $n > 2$, dalam skripsi ini simulasi hanya dijalankan sampai $n = 8$. Hasil simulasi diberikan pada Tabel 4.2 yang menunjukkan banyaknya pelabelan graceful tidak isomorfik pada graf matahari.

Tabel 4.2 Banyaknya pelabelan graceful tidak isomorfik pada graf matahari untuk $n = 3$ s.d. $n = 8$

n	Banyaknya Pelabelan Graceful	n	Banyaknya Pelabelan Graceful
3	18	6	4566
4	84	7	54722
5	510	8	858830



Gambar 4.10 Grafik pertambahan banyaknya pelabelan graceful tidak isomorfik pada graf matahari untuk $n = 3$ s.d. $n = 8$

Pada Gambar 4.10 diberikan grafik yang menunjukkan pertambahan banyaknya pelabelan graceful tidak isomorfik yang terjadi pada graf matahari sesuai dengan bertambahnya nilai *n*. Dari gambar tersebut terlihat bahwa dengan bertambahnya nilai *n*, banyaknya pelabelan graceful tidak isomorfik pada graf matahari yang dihasilkan juga semakin bertambah. Kecenderungan pertambahan banyaknya pelabelan graceful tidak isomorfik yang diperoleh seperti grafik eksponensial sedemikian sehingga jika untuk *n* yang lebih besar disimulasikan maka waktu komputasi yang dibutuhkan akan lebih lama. Hal ini disebabkan karena setiap penambahan satu nilai *n* maka akan ada penambahan dua busur dan simpul yang harus dilabel.

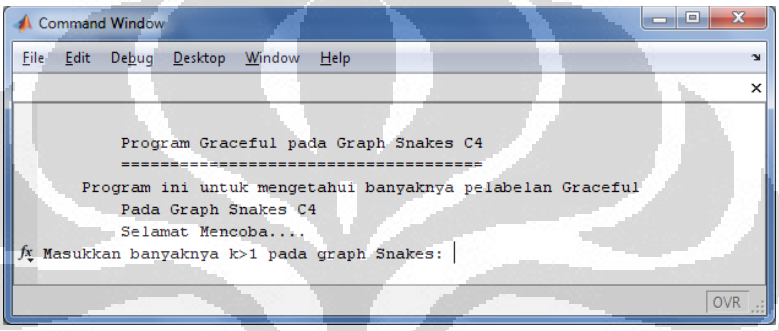
Pada subbab selanjutnya akan diberikan implementasi dan simulasi algoritma pelabelan graceful pada graf ular k-C_4.

4.3 Implementasi dan Simulasi Algoritma Pelabelan Graceful pada Graf Ular k-C_4

Pada algoritma pelabelan graceful yang diberikan pada Subbab 3.3, yang pertama dilakukan adalah tahap inisialisasi. Tahap inisialisasi tersebut dijalankan oleh fungsi *initializeSnake* yg diimplementasikan dalam program dengan nama

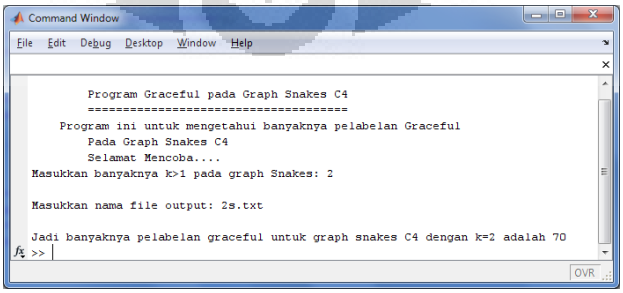
"snake". Sedangkan untuk fungsi *extendSnake1* dan *extendSnake2* diberi nama "extendsnake1" dan "extendsnake2". Listing dari program dapat dilihat di Lampiran 3 (CD).

Pemanggilan program dilakukan dengan cara mengetik "snake" pada *Command Window*. Setelah memanggil program, tampilan yang muncul pada *command window* adalah seperti pada Gambar 4.11. Pengguna diminta untuk memasukkan banyaknya graf C_4 ($k > 1$) pada graf ular k-C_4 yang ingin dilabelkan. Nilai k pada graf ular k-C_4 menyatakan banyaknya graf C_4 yang membentuk graf ular k-C_4.



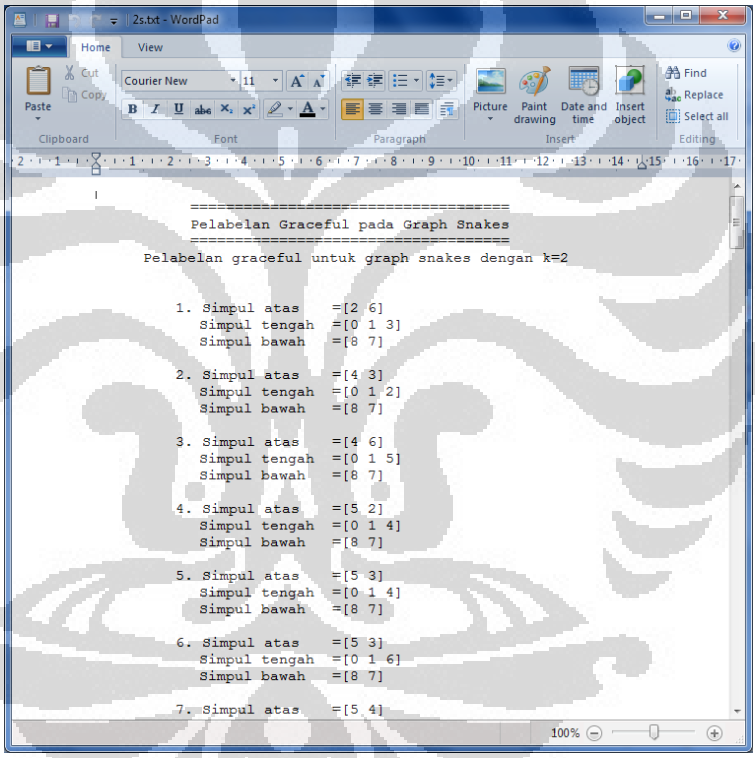
Gambar 4.11 Tampilan awal *command window* pada program pelabelan graceful pada graf ular k-C_4

Selanjutnya program meminta pengguna untuk memasukkan nama berkas keluaran yang akan menyimpan semua pelabelan yang dihasilkan. Program akan berhenti apabila telah dihasilkan semua pelabelan graceful yang tidak isomorfik, pada graf ular k-C_4 dengan nilai k yang dimasukkan oleh pengguna. Program juga mengeluarkan banyaknya pelabelan graceful yang dihasilkan dan mencetak semua pelabelan yang diperoleh dalam berkas keluaran.



Gambar 4.12 Tampilan *command window* pada program pelabelan graceful pada graf ular 2-C_4

Pada Gambar 4.12 diberikan contoh pemanggilan program pelabelan graceful pada graf ular k-C_4, untuk mencari banyaknya pelabelan graceful yang tidak isomorfik pada graf ular 2-C_4. Dalam hal ini masukan yang diberikan oleh pengguna adalah $k = 2$. Kemudian program meminta pengguna untuk memberikan masukan nama dari berkas keluaran yaitu 2s.txt. Dari Gambar 4.12 terlihat bahwa banyaknya pelabelan graceful tidak isomorfik pada graf ular 2-C_4 adalah 70. Pada keluaran dari pelabelan graceful pada graf ular dengan $k = 3$, dihasilkan label-label simpul atas, tengah dan bawah. Isi dari berkas keluaran 2s.txt diberikan pada Gambar 4.13.

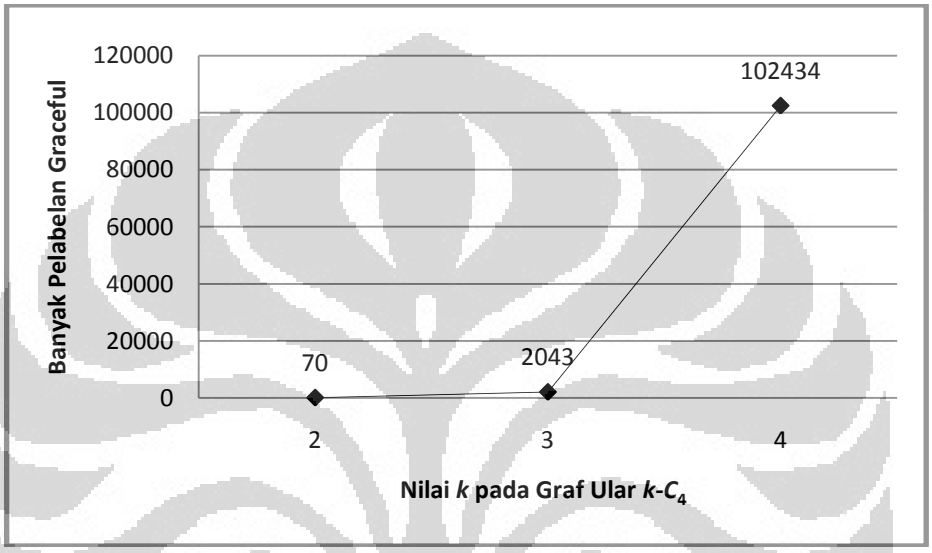


Gambar 4.13 Isi dari berkas keluaran 2s.txt

Simulasi dari program pelabelan graceful pada graf ular k-C_4 dijalankan untuk mengetahui berapa banyak pelabelan graceful tidak isomorfik yang mungkin untuk setiap $k > 1$. Program ini bisa disimulasikan untuk sembarang $k > 1$, dalam skripsi ini simulasi hanya dijalankan sampai $k = 4$. Hasil simulasi diberikan pada Tabel 4.3 yang menunjukkan banyaknya pelabelan graceful tidak isomorfik pada graf ular k-C_4.

Tabel 4.3 Banyaknya pelabelan graceful tidak isomorfik pada graf ular k-C_4 untuk $n = 2$ s.d. $n = 4$

n	Banyaknya Pelabelan Graceful
2	70
3	2054
4	102434



Gambar 4.14 Grafik pertambahan banyaknya pelabelan graceful tidak isomorfik pada graf ular k-C_4 untuk $k = 2$ s.d. $k = 4$

Pada Gambar 4.13 diberikan grafik yang menunjukkan pertambahan banyaknya pelabelan graceful tidak isomorfik yang terjadi pada graf ular k-C_4 sesuai dengan bertambahnya nilai k. Dari gambar tersebut terlihat bahwa dengan bertambahnya nilai k, banyaknya pelabelan graceful tidak isomorfik pada graf ular k-C_4 yang dihasilkan juga semakin bertambah. Kecenderungan pertambahan banyaknya pelabelan graceful tidak isomorfik yang diperoleh seperti grafik eksponensial sedemikian sehingga jika untuk k yang lebih besar disimulasikan maka waktu komputasi yang dibutuhkan akan lebih lama.

BAB 5
KESIMPULAN

Dalam skripsi ini telah dibangun algoritma-algoritma pelabelan graceful untuk masing-masing graf lintasan, matahari dan ular k-C_4. Dengan menggunakan algoritma pelabelan graceful pada masing-masing graf lintasan, matahari dan ular dapat diperoleh semua pelabelan graceful yang tidak isomorfik pada masing-masing graf untuk setiap n atau k yang diberikan. Algoritma-algoritma tersebut telah diimplementasikan dalam bentuk program. Secara teoritis, program-program ini dapat menghasilkan semua pelabelan graceful tidak isomorfik pada graf terkait untuk sembarang nilai n atau k.

Berdasarkan hasil-hasil simulasi dapat disimpulkan bahwa, semakin bertambah nilai n (nilai k pada graf ular k-C_4), maka banyaknya pelabelan graceful tidak isomorfik yang dihasilkan pada graf yang dibahas semakin bertambah secara eksponensial. Dapat juga di simpulkan bahwa graf ular k-C_4 memiliki pelabelan graceful untuk k = 2, 3, 4.

DAFTAR PUSTAKA

Ananda, A. I. (2010). *Algoritma Pelabelan Total simpul Ajaib pada Graf Lingkaran, Matahari, dan Kecebong.* Depok: Skripsi, Departemen Matematika, FMIPA, Universitas Indonesia.

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

Bloom, G. S., & Golomb, S. W. (1977). Applications of Numbered Undirected Graphs. *Proceedings of the Institute of Electrical and Electronics Engineers 65*, 562-570.

Frucht, R. (1979). Graceful Numbering of Wheels and Other Related Graph. *Annals New York Academy of Sciences 319*, 219-229.

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

Golomb, S. (1972). How to Number a Graph. *Graph Theory anf Computing*, 23-37.

Graph Isomorphism. (t.thn.). Dipetik Februari 22, 2010, dari http://www.cs.uu.nl/docs/vakken/an/

Rachmawati, M. (2010). *Algoritma Pelabelan Total (a,d) Simpul Antiajaib pada Graf Lintasan dan Graf Lingkaran.* Depok: Skripsi, Departemen Matematika, FMIPA, Universitas Indonesia.

Rosa, A. (1967). On Certain Valuation of the Vertices of a Graph. *Theory on Graph*, 349-355.

Rosen, K. H. (2003). *Discrete Mathematics and Its Applictions* (5 ed.). New York: McGraw-Hill.

Surgandini, A. (2010). *Algoritma Pelabelan Total Busur Ajaib pada Graf Lingkaran, Kipas, dan Roda.* Depok: Skripsi, Departemen Matematika, FMIPA, Universitas Indonesia.

Utami, B. (2010). *Algorima Pelabelan Total Simpul Ajaib pada Graf Friendship, Kipas, dan Jahangir yang Diperumum.* Depok: Skripsi, Departemen Matematika, FMIPA, Universitas Indonesia.