UNIVERSITAS INDONESIA

ALGORITMA PELABELAN HARMONIS PADA GRAF LINTASAN, LINGKARAN, DAN LOBSTER TERATUR

SKRIPSI

WIDIYANI SUCIATI
0706261991

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

UNIVERSITAS INDONESIA

ALGORITMA PELABELAN HARMONIS PADA GRAF LINTASAN, LINGKARAN, DAN LOBSTER TERATUR

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains

WIDIYANI SUCIATI
0706261991

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

HALAMAN PERNYATAAN ORISINALITAS

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

Nama	: Widiyani Suciati
NPM	: 0706261991
Tanda Tangan	: 
Tanggal	: 21 Desember 2010

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Widiyani Suciati
NPM	: 0706261991
Program Studi	: Sarjana Matematika
Judul Skripsi	: Algoritma Pelabelan Harmonis pada Graf Lintasan, Lingkaran, dan Lobster Teratur

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

DEWAN PENGUJI

Pembimbing	: Dra. Denny R. Silaban, M.Kom	( )
Pembimbing	: Dra. Siti Aminah, M.Kom	( )
Penguji I	: Dra. Denny R. Silaban, M.Kom	( )
Penguji II	: Drs. Suryadi MT, M.T	( )
Penguji III	: Arie Wibowo, S.Si, M.Si	( )

Ditetapkan di	: Depok
Tanggal	: 21 Desember 2010

KATA PENGANTAR/UCAPAN TERIMA KASIH

Alhamdulillah, Puji syukur penulis panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Sains Jurusan Matematika pada Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Indonesia. Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi penulis untuk menyelesaikan skripsi ini. Oleh karena itu, penulis mengucapkan terima kasih 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) Dra. Ida Fithriani, M.Si selaku pembimbing akademik penulis selama menjalani masa kuliah.
(3) Dr. Kiki Ariyanti S, Bevina D. Handari, PhD, Dr. Alhadi Bustamam, M.Kom, dan Dr.rer.nat Hendri Murfi, M.Kom, yang telah hadir dan memberikan saran serta masukan bagi penulis pada SIG 1, SIG 2, dan kolokium.
(4) Seluruh staf pengajar di Matematika UI atas ilmu pengetahuan yang telah diberikan.
(5) Seluruh karyawan di departemen Matematika UI atas bantuan yang telah diberikan (khususnya : mba rusmi, mba santi, dan pak saliman).
(6) Orang tua, kakak, adik, dan ferdy yang selalu memberikan doa, semangat, dan dukungan bagi penulis.
(7) Dita, stefi, ka dhita, ka rian yang telah berjuang bersama selama penyusunan skripsi ini, serta terima kasih atas bantuan, semangat dan dukungannya.
(8) Ka yanu, ka milla, ka mella, ka tami, dan ka alfa, terima kasih atas bantuan, semangat dan dukungannya.

(9) Nora, farah, lois, winda, bapet, kiki, dan toto.Terima kasih atas semangat dan dukungannya.

(10) Seluruh teman-teman angkatan 2007 yang telah memberikan pengalaman perkuliahan yang tak terlupakan.

(11) Kepada semua teman-teman di Matematika UI angkatan 2005, 2006, 2008 (khusunya : ade dan qiqi) dan 2009 (khususnya : ai). Terima kasih atas dukungannya.

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 pengembangan ilmu.

Penulis

2010

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI

TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama	: Widiyani Suciati
NPM	: 0706261991
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 Harmonis pada Graf Lintasan, Lingkaran, dan Lobster Teratur.

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

(Widiyani Suciati)

ABSTRAK

Nama : Widiyani Suciati
Program Studi : Matematika
Judul : Algoritma Pelabelan Harmonis Pada Graf Lintasan, Lingkaran, dan Lobster Teratur

Misalkan *G* adalah graf dengan himpunan simpul tak-kosong *V* dan himpunan busur *E*, dimana |*V*(*G*)| dan |*E*(*G*)| masing-masing menyatakan banyak simpul dan busur pada *G*. Pelabelan harmonis dari graf G adalah suatu pemetaan $f: V(G) \rightarrow Z_m$ dengan menginduksi pelabelan pada himpunan busur $g: E(G) \rightarrow Z_m$ didefinisikan sebagai pemetaan $g(uv) \equiv f(u) + f(v)(mod\ m)$, untuk setiap busur $uv \in E(G)$. Jika G adalah graf pohon maka tepat satu label simpul berulang atau label simpul dapat dilabelkan dengan menggunakan $\{0,1,2, \ldots, m\}$. Dalam skripsi ini diberikan algoritma untuk menghasilkan semua pelabelan harmonis yang tidak isomorfik pada graf lintasan P_n, graf lingkaran C_n dan graf lobster teratur $L_{n,r,1}$ untuk nilai *n* dan *r* (untuk graf lobster teratur) yang diberikan. Algoritma-algoritma ini kemudian diimplementasikan dalam program. Diberikan juga simulasi banyak pelabelan harmonis yang mungkin dan tidak isomorfik sampai nilai *n* tertentu.

Kata Kunci : graf, pelabelan harmonis, graf lintasan, graf lingkaran, graf lobster teratur.
xiv+79 halaman; 52 gambar; 5 tabel
Daftar Pustaka : 10 (1980-2010)

ABSTRACT

Name : Widiyani Suciati
Program Study : Mathematics
Title : A Harmonious Labeling Algorithms on Path, Cycle, and Regular Lobster 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 respectively. A harmonious labeling of a connected graph G is an injection $f: V(G) \to Z_m$ if induced labeling on edge set $g: E(G) \to Z_m$ defined by $g(uv) \equiv f(u) + f(v)(mod\ m)$, for each edge $uv \in E(G)$. If G ia a tree, then either exactly one vertex label is allowed to repeat or the vertexes can be labeled by using $\{0,1,2, \ldots, m\}$. This *skripsi* explains harmonious labeling algorithms on path graph P_n, cycle graph C_n, and regular lobster graph $L_{n,r,1}$. The algorithms generate all non isomorphic labelings on those graphs for all possible values of n and values of r (for regular lobster graph). The implementation of the algorithms and simulation for certain values of n and values of r (for regular lobster graph) are also given in this *skripsi*.

Key Words : graph, harmonious labeling, path graph, cycle graph, regular lobster graph.
xiv+ 79 pages ; 52 pictures; 5 tables
Bibliography : 10 (1980-2010)

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang Masalah

Teori graf merupakan bagian dari matematika diskrit. Bentuk graf digunakan untuk menggambarkan atau menyatakan suatu persoalan agar lebih mudah dimengerti dan diselesaikan. Teori graf banyak berkaitan dengan misalnya teori grup, topologi, matriks, probabilitas, kombinatorik dan lain-lain.

Teori graf pertama kali diperkenalkan oleh seorang matematikawan Swiss yang bernama Leonhard Euler (1707-1783). Saat itu graf digunakan untuk menyelesaikan persoalan matematika rekreasi (*recreational mathematics*) yang berkaitan dengan jembatan Konigsberg. Di kota Konigsberg terdapat 7 jembatan yang menghubungkan 4 daratan. Warga kota Konigsberg ingin mengetahui apakah mungkin melewati semua jembatan tepat satu kali dan kembali ke tempat awal. Masalah tersebut telah dibuktikan tidak mungkin oleh Euler dengan menggunakan teori graf.

Suatu **graf** G didefinisikan sebagai pasangan himpunan (V, E), dimana $V=V(G)$ adalah himpunan tak kosong dari simpul-simpul dan $E= E(G)$ adalah himpunan busur-busur yang menghubungkan simpul-simpul pada V. Busur merupakan pasangan tak-terurut dari simpul-simpul pada V. Banyak anggota pada himpunan simpul V dinyatakan sebagai $n= |V|$. Sedangkan banyak anggota pada himpunan busur E dinyatakan sebagai $m= |E|$. Graf G dikatakan **graf berhingga** jika $|V|$ berhingga.

Terdapat macam-macam graf dengan ciri-ciri tertentu yang diberi nama khusus seperti graf lintasan, graf lingkaran, graf pohon, dan graf lobster. **Graf lintasan** (*path graph*), P_n, adalah graf dengan n simpul yang mempunyai busur-busur v_1v_2, v_2v_3, …, $v_{n-1}v_n$, dengan v_i menyatakan simpul ke-i dari lintasan. **Graf lingkaran** (*cycle graph*), C_n, adalah graf yang diperoleh dari graf lintasan dengan penambahan busur v_nv_1. **Graf pohon** (*tree graph*) adalah graf terhubung yang tidak mempunyai subgraf lingkaran. Graf lintasan termasuk graf pohon. **Graf**

lobster adalah salah satu jenis yang termasuk dalam graf pohon. Definisi tepat dari graf lobster akan diberikan pada Bab 2.

Dalam teori graf, **isomorfisma antara graf $G(V, E)$ dan $H(W, F)$** adalah suatu pemetaan bijektif antara himpunan simpul-simpul di G dengan himpunan simpul-simpul di H, ditulis $f: V \to W$, sedemikian sehingga dua simpul u dan v dari G bertetangga di G jika dan hanya jika $f(u)$ dan $f(v)$ juga bertetangga di H. Dua graf saling isomorfik, ditulis $G \cong H$, dibaca G isomorfik H, (Graph Isomorphism, 2010). Dengan perkataan lain, dua graf $G(V, E)$ dan $H(W, F)$ adalah isomorfik jika ada suatu pemetaan bijektif, $f: V \to W \ni \forall u, v \in V, uv \in E \leftrightarrow f(u)f(v) \in F$.

Pelabelan graf merupakan salah satu cabang dari teori graf. Pelabelan harmonis adalah salah satu pelabelan graf.

Misalkan *G*(*V*, *E*) graf sederhana berhingga dengan n simpul dan m busur, *V* dan *E* dinotasikan secara berurutan sebagai himpunan simpul dan himpunan busur dari graf G. **Pelabelan harmonis** dari graf *G* adalah suatu pemetaan $f: V \to Z_m$ dengan menginduksi pelabelan pada himpunan busur $g: E \to Z_m$, g didefinisikan sebagai pemetaan $g(uv) \equiv f(u) + f(v)(mod\ m)$, untuk setiap busur $uv \epsilon E$. Jika G bukan graf pohon maka f adalah pemetaan satu-satu. Jika G graf pohon maka tepat satu label simpul berulang atau label simpul dapat dilabelkan dengan menggunakan $\{0,1,2, \dots, m\}$. Graf yang memiliki pelabelan harmonis disebut **graf harmonis** (Hui-Chuan Lu & Hung-Lin Fu, 2007).

Misalkan terdapat graf $G(V, E)$ dan graf $H(W, F)$. Pelabelan simpul graf G dan H masing-masing merupakan pemetaan $l: V \to L$ dan pemetaan $l': W \to L$. Graf $G(V, E)$ dan $H(W, F)$ dikatakan **pelabelan isomorfik**, jika terdapat pemetaan bijektif $f: V \to W \ni \forall u, v \in V, uv \in E \leftrightarrow f(u)f(v) \in F \wedge \forall u \in V, l(u) = l'(f(u))$ (Graph Isomorphism, 2010).

Telah terdapat hasil-hasil mengenai eksistensi dari **pelabelan harmonis** untuk kelas-kelas graf tertentu yang telah dipublikasikan. Aldred dan McKay menggunakan komputer untuk menunjukkan semua pohon dengan $n \leq 26$ simpul adalah harmonis. G**raf lingkaran** , C_n, adalah harmonis jika dan hanya jika $n \equiv 1$ atau $3(mod\ 4)$ atau bilangan ganjil (Graham dan Sloane, 1980).

Baker dan Sawada (2008) telah membangun suatu algoritma pelabelan total simpul ajaib **(**PTSA) untuk graf lingkaran dan graf roda. Ananda (2010) telah membangun algoritma pelabelan total simpul ajaib (PTSA) pada graf lingkaran, matahari, dan kecebong. Utami (2010) telah membangun algoritma pelabelan total simpul ajaib (PTSA) pada graf *friendship*, kipas dan jahangir yang diperumum. Rachmawati (2010) telah membangun algoritma pelabelan total (a,d)-simpul antiajaib ((a,d)-PTSAA) pada graf lintasan dan lingkaran. Surgandini (2010) telah membangun algoritma pelabelan total busur ajaib (PTBA) pada graf lingkaran, kipas dan roda. Sedangkan pada skripsi ini akan dibangun algoritma pelabelan harmonis pada graf lintasan, graf lingkaran, dan graf lobster teratur dimana pelabelannya tidak isomorfik.

1.2 Permasalahan dan Ruang lingkup Penelitian

Bagaimanakah cara membangun algoritma pelabelan harmonis yang menghasilkan semua pelabelan yang berbeda (tidak isomorfik) pada graf lintasan, graf lingkaran, dan graf lobster teratur?

1.3 Jenis penelitian dan Metode Penelitian

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

1.4 Tujuan Penelitian

Tujuan dari penulisan skripsi ini adalah membangun algoritma pelabelan harmonis yang menghasilkan semua pelabelan yang berbeda (tidak isomorfik) pada graf lintasan, graf lingkaran, dan graf lobster teratur.

BAB 2
LANDASAN TEORI

Pada bab ini akan diberikan beberapa definisi dan konsep dasar dari teori graf, serta akan dijelaskan beberapa jenis pelabelan graf yang akan digunakan pada bab selanjutnya.

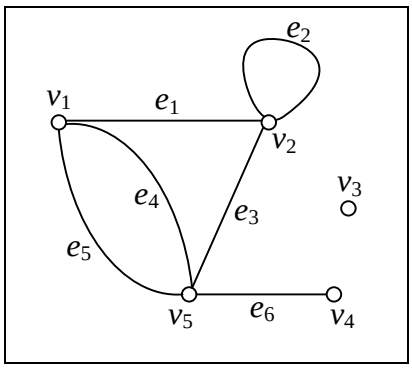
2.1 Definisi dan Istilah dalam Teori Graf

Suatu graf $G(V, E)$ terdiri atas suatu himpunan **simpul** yang tak-kosong dan berhingga, dinotasikan dengan $V(G)$ atau V dan suatu himpunan **busur** yang berhingga, dinotasikan dengan $E(G)$ atau E. Suatu busur menghubungkan dua simpul (boleh sama) pada graf, simpul tersebut disebut **titik ujung** (*endpoint*) dari busur. Banyak simpul dinotasikan dengan $|V(G)|$ atau $|V|$ atau n simpul. Sedangkan banyak busur dinotasikan dengan $|E(G)|$ atau $|E|$ atau m busur.

Apabila banyak simpul berhingga, maka graf G disebut **graf berhingga**. Jika himpunan busur pada graf G adalah himpunan kosong maka graf G disebut **graf kosong**. Suatu graf G disebut **graf berarah** apabila graf G terdiri dari himpunan simpul V dan himpunan busur E, dimana setiap busur adalah pasangan terurut dari simpul. Jika setiap busur pada graf *G* bukan merupakan pasangan terurut maka graf *G* disebut **graf tak berarah**.

Suatu **jalan** (*walk*) adalah sederetan $v_0, e_1, v_1, e_2, \ldots, e_k, v_k$ dari simpul dan busur sedemikian sehingga $e_i = v_{i-1}v_i$ untuk $1 \leq i \leq k$. **Jalur** (*trail*) adalah suatu jalan dengan busur yang tidak berulang. **Lintasan** (*path*) adalah jalur yang deretan simpul-simpulnya tidak pernah berulang. **Jalan**-*u*,*v* atau **jalur**-*u*,*v* memiliki simpul pertama *u* dan simpul terakhir *v*, *u* dan *v* adalah titik-titik ujung (*endpoints*). Panjang dari suatu jalan (jalur) adalah banyak busur pada jalan (jalur). Jalan (jalur ataupun lintasan) disebut **tertutup** (*closed*) jika titik ujungnya sama. Graf G disebut **graf terhubung** (*connected graph*) jika graf G memiliki lintasan- uv, *u* dan *v* terhubung di G (West, 2001).

Dalam bentuk gambar, simpul-simpul pada graf direpresentasikan sebagai lingkaran kecil dan busur-busur pada graf direpresentasikan sebagai garis yang menghubungkan simpul-simpul pada graf. **Simpul** biasanya ditulis sebagai v_1, v_2, … , v_n. Sedangkan **busur** dapat ditulis dengan menyatakan kedua simpul akhirnya v_iv_j, dimana $v_iv_j \in E$ dan $i, j \in \{1, \dots, n\}$, atau dengan menulisnya sebagai e_1, e_2, … , e_m. Contoh graf G dengan himpunan simpul $V = \{v_1, v_2, v_3, v_4, v_5\}$ dan himpunan busur $E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$ ditunjukkan pada Gambar 2.1.



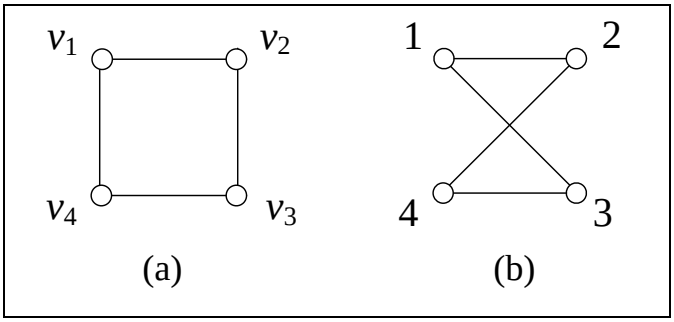
Gambar 2.1 Contoh graf G dengan $|V| = 5$ dan $|E| = 6$

Derajat (*degree*) dari suatu simpul pada graf G adalah banyak busur yang hadir pada simpul tersebut (kecuali untuk gelung pada simpul, derajatnya adalah dua). Derajat dari suatu simpul v dinotasikan dengan $deg\ (v)$. Simpul dengan derajat nol ($d(v) = 0$) disebut **simpul terpencil** (*isolated vertex*) dan simpul dengan derajat satu ($d(v) = 1$) disebut **simpul terminal** (*terminal vertex*) atau dapat juga disebut **simpul daun** (*leaves vertex*). Pada Gambar 2.1, graf G memiliki $d(v_1) = 3$, $d(v_2) = d(v_5) = 4$, $d(v_3) = 0$, $d(v_4) = 1$, sehingga v_3 disebut simpul terpencil dan v_4 disebut simpul terminal.

Pada graf G, dua simpul dikatakan **bertetangga** (*adjacent*) apabila terdapat satu atau lebih busur yang menghubungkan kedua simpul tersebut. Suatu busur dikatakan **hadir** (*incident*) pada suatu simpul apabila simpul tersebut merupakan salah satu titik ujung dari busur tersebut. Busur yang memiliki titik ujung yang sama disebut **gelung** (*loop*). Dua busur disebut **busur ganda** (*multiple edges*) apabila kedua busur ini memiliki pasangan titik ujung yang sama. Suatu graf yang tidak memiliki gelung dan busur ganda disebut **graf sederhana** (*simple*

graph). Pada Gambar 2.1, v_1 dan v_2 bertetangga, e_1 hadir pada simpul v_1 dan v_2, busur e_2 disebut gelung dan busur e_4, e_5 disebut busur ganda.

Isomorfisma antara graf $G(V, E)$ dan $H(W, F)$ adalah suatu pemetaan bijektif antara himpunan simpul-simpul di G dengan himpunan simpul-simpul di H, ditulis $f: V \to W$, sedemikian sehingga dua simpul u dan v dari G bertetangga di G jika dan hanya jika $f(u)$ dan $f(v)$ juga bertetangga di H. Dua graf saling isomorfik, ditulis $G \cong H$, dibaca G isomorfik H, (Graph Isomorphism, 2010). Dengan perkataan lain, dua graf $G(V, E)$ dan $H(W, F)$ adalah isomorfik jika ada suatu pemetaan bijektif, $f: V \to W \ni \forall u, v \in V, uv \in E \leftrightarrow f(u)f(v) \in F$.



Gambar 2.2 Contoh graf yang isomorfik

Pada Gambar 2.2 diberikan dua graf terhubung yaitu, graf $G(V, E)$ pada Gambar 2.2a dan graf $H(W, F)$ pada Gambar 2.2b. Kedua graf tersebut sama-sama memiliki $|V| = 4$ dan $|E| = 4$. Untuk membuktikan $G \cong H$, pertama diberikan nama pada kedua graf tersebut, yaitu v_1, v_2, v_3, v_4 untuk setiap simpul pada graf G dan 1,2,3,4 untuk setiap simpul pada graf H. Kemudian ditentukan pemetaan bijektif $f: V \to W$, yaitu $f(v_1) = 1, f(v_2) = 2, f(v_3) = 4, f(v_4) = 3$ dan diperiksa apakah pemetaan bijektif tersebut mempertahankan hubungan kebertetanggaan (*adjacency relation*). Ternyata pemetaan bijektif tersebut mempertahankan hubungan kebertetanggaan (*adjacency relation*). Jadi G isomorfik H.

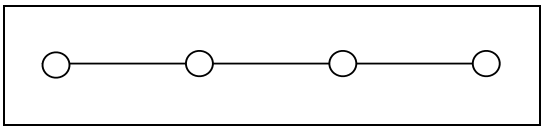
Pada subbab selanjutnya akan diberikan definisi dari beberapa jenis graf yang akan digunakan dalam skripsi ini.

2.2 Jenis-jenis Graf

Semua graf yang digunakan dalam skripsi ini merupakan graf berhingga, sederhana dan tak berarah.

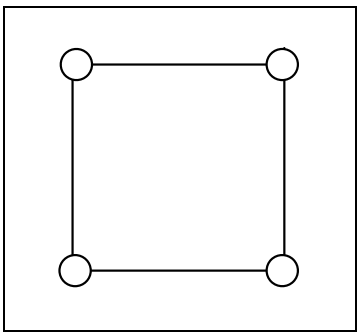
Graf pohon (*tree graph*) T_n adalah graf terhubung dengan n simpul yang tidak mempunyai subgraf lingkaran. Pada graf pohon, $n = m + 1$.

Graf lintasan (*path graph*) P_n adalah graf dengan n simpul ($n \geq 2$) dengan busur v_1v_2, v_2v_3, …, $v_{n-1}v_n$ atau dapat juga dinotasikan dalam $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_n$ atau untuk mempersingkat dapat ditulis e_1, e_2, ... ,e_n. Semua simpul berderajat 2 kecuali untuk simpul awal dan simpul akhir berderajat satu. Simpul v_1 disebut simpul awal dan simpul v_n adalah simpul akhir. Dalam graf lintasan berlaku $|V(P_n)| = |E(P_n)| + 1$. Suatu graf dikatakan terhubung jika untuk setiap pasang simpul u dan v terdapat lintasan dari u ke v. Graf lintasan termasuk kedalam graf pohon. Contoh graf lintasan diberikan pada Gambar 2.3.



Gambar 2.3 Graf lintasan P_4

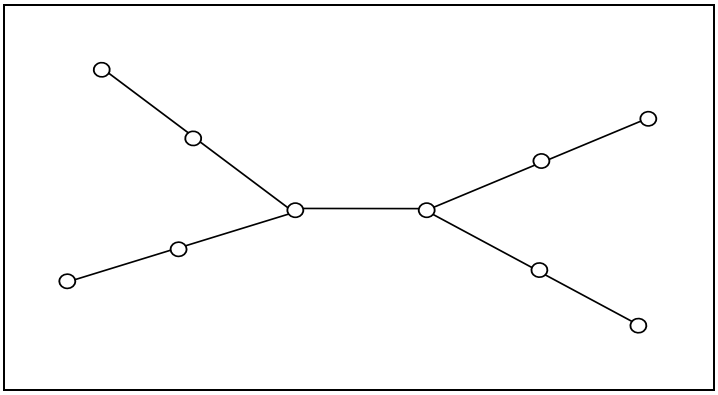
Graf lingkaran (c*ycle graph*), C_n ($n \geq 3$) adalah graf yang diperoleh dari graf lintasan P_n yang diberi tambahan busur antara simpul awal dan simpul akhir (busur v_nv_1). Semua simpul pada graf memilliki derajat dua. Dalam graf lingkaran berlaku $|V(C_n)| = |E(C_n)|$. Contoh graf lingkaran diberikan pada Gambar 2.4.



Gambar 2.4 Graf lingkaran C_4

Graf caterpillar $P_n \odot K_1$ adalah graf yang apabila seluruh simpul daunnya dihapus akan menghasilkan graf lintasan (Galian, 2009). Graf caterpillar juga dapat dibangun dari graf lintasan P_n dengan menambahkan sejumlah simpul daun pada setiap simpul dari graf lintasan. Jika banyak simpul daun yang ditambahkan sama (sebut sebanyak r) untuk setiap simpul dari graf lintasan, maka disebut graf caterpillar teratur. Lintasan P_n disebut backbone dari caterpillar.

Graf lobster adalah graf yang apabila seluruh simpul daunnya dihapus akan menghasilkan graf cartepillar (Galian, 2009). Graf lobster dapat juga dibangun dengan menambahkan sejumlah simpul daun pada setiap simpul daun dari graf caterpillar. Jika graf caterpillar teratur dan banyak simpul daun yang ditambahkan sama (sebut sebanyak p) untuk setiap simpul daun dari graf caterpillar teratur akan diperoleh graf lobster teratur. Sehingga graf lobster teratur dapat dinotasikan dengan $L_{n,r,p}$. Graf lobster yang dibahas di sini adalah graf lobster teratur $L_{n,r,1}$. $|V(L_{n,r,1})| = 2nr+n$ dan $|E(L_{n,r,1})| = (2nr+n)-1$. Contoh graf lobster $L_{2,2,1}$ diberikan pada Gambar 2.5.



Gambar 2.5 Graf lobster teratur $L_{2,2,1}$

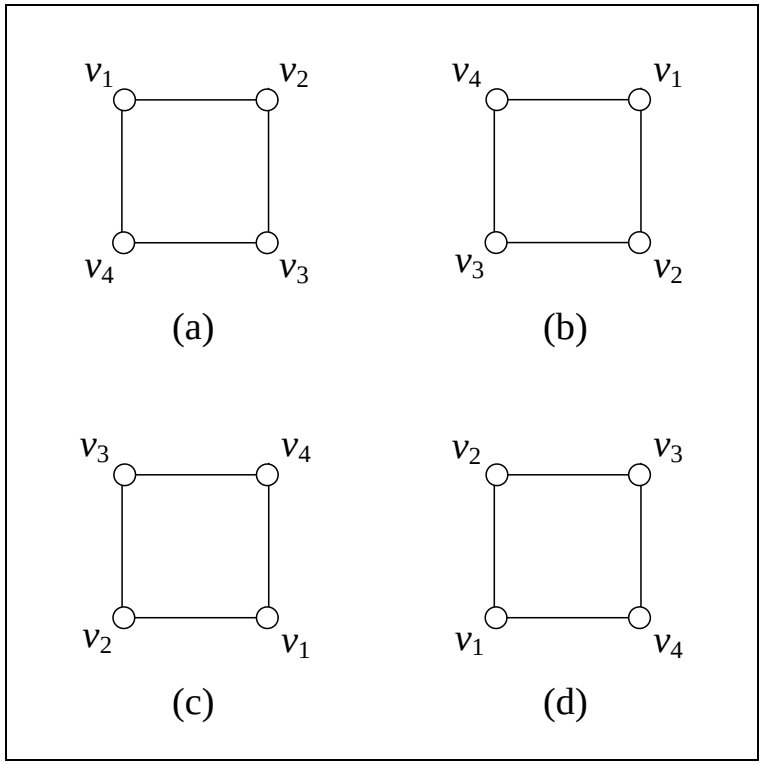
Pada subbab selanjutnya akan diberikan definisi-definisi dan istilah dalam pelabelan graf.

2.3 Pelabelan Graf

Misalkan G graf sederhana berhingga dengan n simpul dan m busur, V dan E dinotasikan secara berurutan sebagai himpunan simpul dan himpunan busur

dari graf G. **Pelabelan harmonis** dari graf G adalah suatu pemetaan $f: V \rightarrow Z_m$ dengan menginduksi pelabelan pada himpunan busur $g: E \rightarrow Z_m$, dimana g didefinisikan sebagai pemetaan $g(uv) \equiv (f(u) + f(v)) mod\ m$, untuk setiap busur $uv \epsilon E$. Jika G bukan graf pohon maka f adalah pemetaan satu-satu. Jika G adalah graf pohon maka tepat satu label simpul berulang atau label simpul dapat dilabelkan dengan menggunakan $\{0,1,2, \ldots, m\}$. Graf yang memiliki pelabelan harmonis disebut **graf harmonis** (Hui-Chuan Lu & Hung-Lin Fu, 2007).

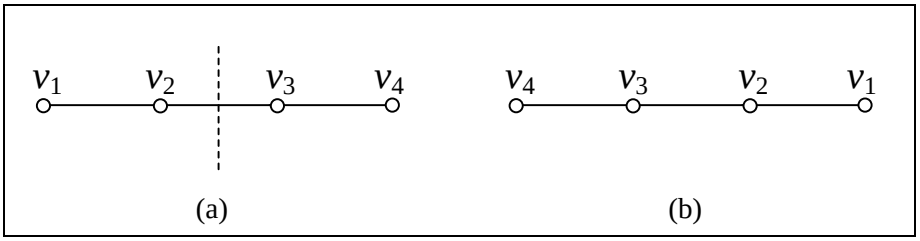
Untuk mencari banyak pelabelan yang berbeda perlu diberikan definisi **pelabelan yang isomorfik**. Misalkan $G(V, E)$ dan $H(W, F)$ adalah graf dengan palabelan simpul $l : V \rightarrow L$, $l' : W \rightarrow L$ dimana L adalah himpunan label. Graf G dan H adalah graf berlabel yang isomorfik jika terdapat fungsi bijektif $f: V \rightarrow W$ $\ni\ \forall\, u, v \in V\,, uv \in E \leftrightarrow\ f(u)f(v) \in F\ \wedge\ \forall\, u \in V, l(u) = l'(f(u))$ (Graph Isomorphism, 2010).



Gambar 2.6 Contoh kasus rotasi pada graf lingkaran C_4

Jika diobservasi, untuk memperoleh pelabelan yang isomorfik terhadap pelabelan l adalah hasil dari rotasi atau refleksi. Kasus dimana suatu pelabelan diperoleh dengan melakukan rotasi yang ada disebut **kasus rotasi** (*rotational symmetry*). Pada Gambar 2.6 diberikan kasus rotasi dimana pelabelan pada graf (b), (c) dan (d) diperoleh dengan merotasi pada graf (a). Saat graf (d) dirotasikan lagi akan mendapatkan graf semula atau graf (a).

Kasus dimana suatu pelabelan diperoleh dengan melakukan refleksi terhadap pelabelan yang ada disebut **kasus refleksi** (*reflective symmetry*). Pada Gambar 2.7 diberikan contoh kasus refleksi pada graf lintasan P_4 dimana garis putus-putus merupakan sumbu refleksi. Pelabelan pada graf (b) diperoleh dengan merefleksi label pada graf (a). Jika graf (b) direfleksikan kembali dengan garis refleksinya busur v_3v_2 akan diperoleh graf awal yaitu graf (a). Untuk P_n, dimana n bernilai bilangan ganjil, sumbu simetrinya berada pada simpul ke-($(n+1)/2$). Sedangkan, jika P_n, dimana n bernilai bilangan genap, sumbu simetrinya berada pada busur ke-($n/2$).



Gambar 2.7 Kasus refleksi pada graf P_4

Seperti yang diperlihatkan pada Gambar 2.6 dan Gambar 2.7, pelabelan dari hasil rotasi atau refleksi adalah sama dengan pelabelan awal.

2.4 Hasil yang Telah Diketahui

Terdapat hasil-hasil penelitian yang telah dipublikasikan mengenai eksistensi dari Pelabelan harmonis. Menurut Galian (2009), Aldred dan McKay menggunakan komputer untuk menunjukkan semua pohon dengan $n \leq 26$ simpul adalah harmonis. G**raf lingkaran**, C_n, adalah harmonis jika dan hanya jika

$n \equiv 1$ atau $3(mod\ 4)$ (Graham dan Sloane, 1980). Yang dituliskan disini hanya sebagian atau bukan seluruh hasil yang dituliskan Galian.

BAB 3
ALGORITMA PELABELAN HARMONIS PADA GRAF LINTASAN, GRAF LINGKARAN DAN GRAF LOBSTER TERATUR

Dalam bab ini akan dibahas mengenai algoritma pelabelan harmonis pada graf lintasan, graf lingkaran, dan graf lobster teratur. Telah disebutkan pada bab sebelumnya bahwa Baker dan Sawada (2008) telah membangun suatu algoritma pelabelan total simpul ajaib **(**PTSA) untuk graf lingkaran dan graf roda. Ananda (2010) telah membangun algoritma pelabelan total simpul ajaib (PTSA) pada graf lingkaran, matahari, dan kecebong. Utami (2010) telah membangun algoritma pelabelan total simpul ajaib (PTSA) pada graf *friendship*, kipas dan jahangir yang diperumum. Rachmawati (2010) telah membangun algoritma pelabelan total (*a*,*d*)-simpul antiajaib ((*a*,*d*)-PTSAA) pada graf lintasan dan lingkaran. Surgandini (2010) telah membangun algoritma pelabelan total busur ajaib (PTBA) pada graf lingkaran, kipas dan roda. Dengan menggunakan ide dari algoritma-algoritma tersebut, dibangun algoritma pelabelan harmonis pada graf lintasan, graf lingkaran, dan graf lobster teratur.

Secara umum, ide dari algoritma pelabelan harmonis untuk graf lintasan, graf lingkaran, dan graf lobster teratur adalah melabel semua simpul dan busur dengan label yang mungkin secara iteratif. Algoritma pelabelan harmonis akan menghasilkan semua pelabelan yang berbeda (tidak isomorfik) untuk graf terkait. Untuk menjamin pelabelan yang dihasilkan (jika ada) tidak isomorfik, diberikan syarat tertentu dalam algoritma tersebut. Syarat yang diberikan tergantung dari kasus isomorfik yang mungkin terjadi pada setiap kelas graf. Selain mereduksi banyak pelabelan yang dicari, syarat ini juga membantu mengurangi komputasi yang dilakukan. Jika di observasi, pelabelan isomorfik pada suatu graf merupakan hasil refleksi atau rotasi dari label graf tersebut seperti yang telah dijelaskan pada Subbab 2.3.

Masukan (*input*) dari algoritma pelabelan harmonis adalah nilai dari *n* (banyak simpul pada graf). Keluarannya (*output*) adalah label untuk semua simpul dan busur dari graf yang berkaitan yang memenuhi ketentuan pelabelan harmonis dan pelabelannya tidak isomorfik. Hasil pemetaan $f(u)$ dan $g(uv)$ masing-

masing menyatakan label dari simpul *u* dan busur *uv*. Label busur akan diberikan sesuai dengan label yang tersedia. Sementara label simpul $f(v_t)$ ditentukan, yaitu $f(v_t) = \big(mk + g(e_{t-1})\big) - f(v_{t-1})$, dimana *m*=|*E*|. Label simpul pada graf lintasan dan graf lobster teratur menggunakan $\{0,1,2, \dots, m\}$. Label simpul pada graf lingkaran mengguna-kan {0, 1, ..., *m*-1}. Label busur pada ketiga graf tersebut menggunakan bilangan modulo *m*, yaitu {0, 1, ..., *m*-1}. Label simpul dan label busur masing-masing di-nyatakan sebagai *array*.

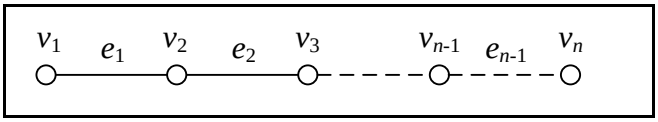
Jika label simpul yang diperoleh tidak memenuhi kondisi $0 \leq f(v_t) \leq n - 1$ $(t = 1,2, \dots, |V|)$ dan tidak tersedia pada *array* label simpul, maka label busur diganti dengan label busur lainnya. Jika label simpul tetap tidak diperoleh, maka dilakukan *backtracking* untuk mengganti label simpul yang dilabel sebelumnya. Hal ini dilakukan hingga diperoleh label simpul yang memenuhi kondisi $0 \leq f(v_t) \leq n - 1$ $(t = 1,2, \dots, |V|)$ dan tersedia pada *array* label simpul. Jika setelah dilakukan *backtracking* tetap tidak diperoleh label simpul yang sesuai dengan label yang tersedia, maka graf tersebut untuk nilai *n* yang diberikan tidak memiliki pelabelan harmonis. Tetapi, jika diperoleh label simpul yang memenuhi kondisi $0 \leq f(v_t) \leq n - 1$ dan tersedia, maka satu pelabelan harmonis diperoleh dan label untuk setiap elemen graf dicetak. Kemudian proses akan berlanjut secara *backtracking* ke simpul sebelumnya untuk memperoleh pelabelan harmonis yang lain.

Pembahasan lengkap mengenai algoritma pelabelan harmonis untuk graf lintasan, graf lingkaran, dan graf lobster teratur masing-masing diberikan pada Subbab 3.1, 3.2, dan 3.3.

3.1 Algoritma Pelabelan Harmonis pada Graf Lintasan P_n

Graf lintasan P_n yaitu graf dengan *n* simpul ($n \geq 2$) dengan busur v_1v_2, v_2v_3, …, $v_{n-1}v_n$. Untuk mempersingkat,busur dinotasikan dengan, e_1, e_2, ... ,e_{n-1}. Semua simpul berderajat 2 kecuali untuk simpul awal dan simpul akhir berderajat satu. Simpul v_1 disebut simpul awal dan simpul v_n adalah simpul akhir. Dalam graf lintasan, banyak simpul dan banyak busur masing-masing adalah *n* dan *n*-1. Pada

Gambar 3.1 diberikan graf lintasan dengan *n* simpul beserta penamaan simpul dan busurnya.

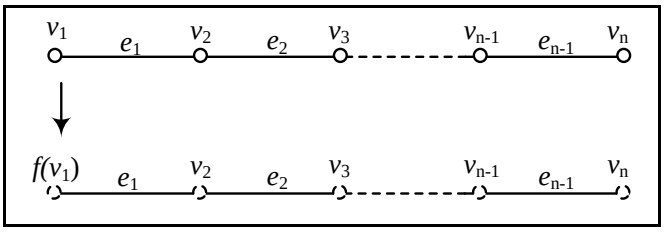


Gambar 3.1 Graf lintasan dengan *n* simpul

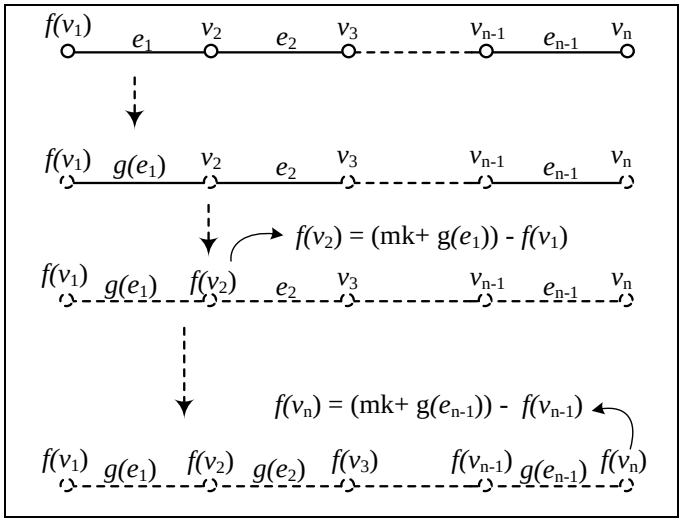
Algoritma pelabelan harmonis untuk graf ini bekerja secara iteratif. Label simpul pada graf lintasan menggunakan $\{0,1,2,\dots,m\}$ dan label busur menggunakan bilangan modulo *m*, yaitu {0, 1, ..., *m*-1} dengan syarat label busur adalah jumlah (mod *m*) dari label dua simpul yang dihubungkan. Label-label tersebut digunakan tepat 1 kali. Label simpul dan label busur masing-masing dinyatakan sebagai *array*. *Array* label simpul dan *array* label busur masing-masing berukuran *n* dan berukuran *n*-1, dimana *array* tersebut berisi status dari label-label tersebut bernilai *true* atau *false*. Status dari label-label mula-mula bernilai *true* yang menandakan label tersedia.

Masukan dari algoritma ini adalah banyak simpul *n*. Label simpul $f(v_1)$ dan label busur $g(e_t)$ akan diberikan sesuai dengan label yang tersedia. Sedangkan label simpul $f(v_t)$ ditentukan, yaitu $f(v_t) = \big(mk + g(e_{t-1})\big) - f(v_{t-1})$, dimana $t = 2,\dots,n$ dan $m = |E|$, serta nilai k dapat bernilai $k = 0$ dan $k = 1$. Kondisi yang harus dipenuhi adalah $0 \le f(v_t) \le n-1$, dimana $t = 1,2,\dots,n$. Karena label simpul ditentukan dari $\big(mk + g(e_{t-1})\big) - f(v_{t-1})$ maka label simpul $f(v_t)$ disebut *determined label*. Jika label simpul $f(v_t)$ dan label busur $g(e_t)$ tersedia maka akan digunakan. Setelah label simpul $f(v_t)$ dan label busur $g(e_t)$ terpakai masing-masing akan dinyatakan *avail* $[f(v_t)]$:= *false* dan *avail* $[g(e_t)]$:= *false*.

Langkah pertama pada algoritma ini adalah melabel simpul v_1, yang merupakan tahap inisialisasi (Gambar 3.2). Setelah tahap inisialisasi dilakukan, masuk ke tahap perluasan (Gambar 3.3). Langkah pertama yang dilakukan pada tahap perluasan adalah melabel busur e_1, kemudian menentukan label simpul v_2. Setelah label v_2 diperoleh, melabel busur e_2. Kemudian menentukan label simpul v_3 dan seterusnya hingga melabel busur e_{t-1} dan simpul v_t.



Gambar 3.2 Tahap inisialisasi algoritma pelabelan harmonis graf P_n



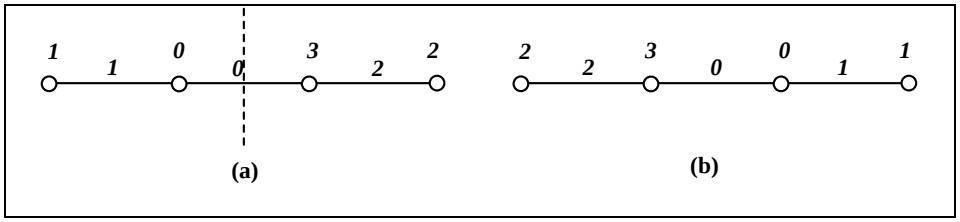
Gambar 3.3 Tahap perluasan algoritma pelabelan harmonis graf P_n

Karena algoritma ini terdiri dari dua tahap, yaitu tahap inisialisasi dan tahap perluasan, maka algoritma ini terdiri dari dua fungsi yaitu fungsi initialize-Path dan extendPath(*t*). Tahap inisialisasi berada dalam fungsi initializePath. Sedangkan Tahap perluasan berada dalam fungsi extendPath(*t*) dengan parameter *t* = 2, 3, ..., *n*.

Pelabelan simpul v_1 berada dalam fungsi initializePath. Kemudian memanggil fungsi extendPath(*t*) dengan parameter *t* = 2. Fungsi extendPath(*t*) untuk *t* = 2 dimulai dengan melabel busur e_1. Kemudian menentukan label simpul v_2, yaitu $f(v_2) = \big(mk + g(e_1)\big) - f(v_1)$. Jika label simpul v_2 memenuhi kondisi $0 \leq f(v_t) \leq n-1$ dan tersedia, maka digunakan dan dinyatakan *avail* $[f(v_2)]$:= *false* dan *avail* $[g(e_1)]$:= *false*. Selanjutnya secara rekursif memanggil fungsi extendPath(*t*) dengan parameter *t* = *t* + 1 = 3 untuk melabel busur e_2 dan menentukan label simpul v_3, dan seterusnya hingga memanggil fungsi extendPath(*t*)

dengan parameter *t* = *n*. Saat *t* = *n*, busur e_{n-1} diberi label kemudian menentukan label simpul v_n, dimana label busur $g(e_{n-1})$ dan label simpul $f(v_n)$ menggunakan label terakhir yang masih tersedia. Jika label simpul $f(v_n)$ tersedia, maka satu pelabelan harmonis berhasil dibentuk. Kemudian elemen pelabelan harmonis yang terbentuk pada graf lintasan P_n di cetak dengan fungsi print().

Setelah melabel busur e_{t-1}, ditentukan label simpul $f(v_t)$. Jika label simpul $f(v_t)$ tidak tersedia, maka diganti label busur $g(e_{t-1})$ hingga diperoleh label simpul $f(v_t)$. Jika tidak ada label busur $g(e_{t-1})$ yang memenuhi untuk diperoleh label simpul $f(v_t)$, maka dilakukan *backtracking* untuk mengubah label simpul yang dilabel sebelumnya, yaitu $f(v_{t-1})$ dan mengubah label busur sebelumnya, yaitu $g(e_{t-2})$. Label simpul dan label busur yang digunakan sebelumnya menjadi *available* (tersedia kembali), yaitu masing-masing ditandai dengan *avail* $[f(v_{t-1})]$:= *true* dan *avail* $[g(e_{t-2})]$:= *true*. Langkah ini dilakukan seterusnya hingga diperoleh label simpul $f(v_t)$ yang sesuai dengan label yang tersedia. Jika label simpul $f(v_t)$ tetap tidak tersedia, berarti tidak terdapat pelabelan harmonis yang berbeda pada graf lintasan dengan nilai *n* yang diberikan. Algoritma ini akan berlanjut untuk mencari pelabelan harmonis lain dengan melakukan *backtracking* ke simpul v_{n-1} dan seterusnya sesuai dengan urutan langkah-langkah sebelumnya. Algoritma ini akan berhenti jika telah diperoleh semua pelabelan harmonis yang tidak isomorfik dengan nilai *n* yang diberikan.



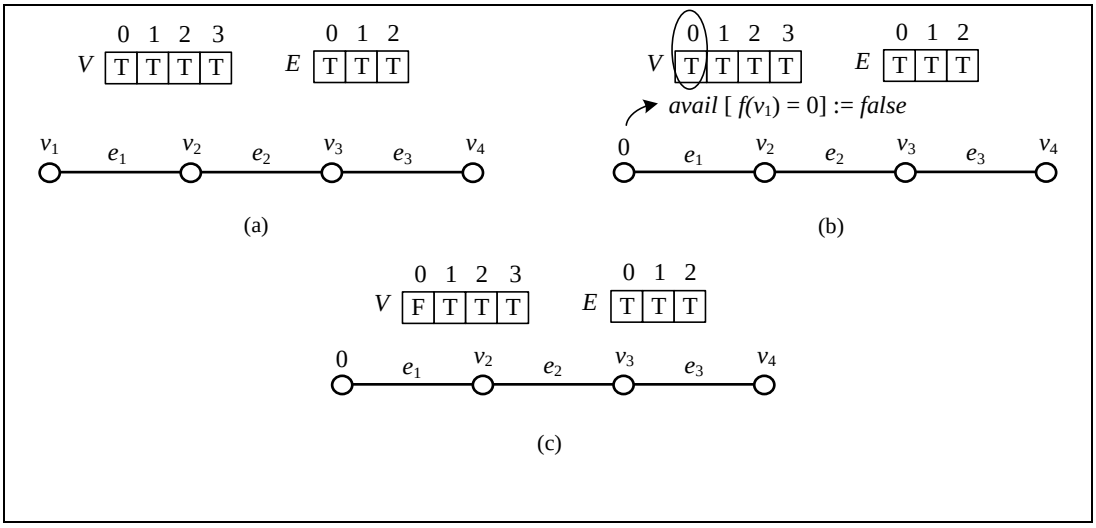
Gambar 3.4 Kasus refleksi pada graf lintasan

```
Algoritma 1.1
function initializePath
1       for  each available label i do
2               f(v1) := i
3               avail [i] := false
4               extendPath(2)
5               avail [i]:= true
        end for
```

```
Algoritma 1.2
function extendPath (t)
1       if t = n then
2               for each available label j do
3                       g(e_{t-1}) := j
4                       for k=0 or k=1 do
5                               f(v_t) = [mk + g(e_{t-1})] − f(v_{t-1})
6                               if 0 ≤ f(v_t) ≤ n − 1 and avail [f(v_t)] and f(v_t) > f(v_1) then
7                               print()
                                end if
                        end for
                end for
8       else
9               for each available label j do
10                      g(e_{t-1}) := j
11                      for k=0 or k=1 do
12                              f(v_t) = [mk + g(e_{t-1})] − f(v_{t-1})
13                              if 0 ≤ f(v_t) ≤ n − 1 and avail [f(v_t)] then
14                                      avail [f(v_t)] := false
15                                      avail [j] := false
16                                      extendPath (t+1)
17                                      avail [f(v_t)] := true
18                                      avail [j] := true
                                end if
                        end for
                end for
        end if
```

Algoritma ini bertujuan untuk menghasilkan semua pelabelan yang berbeda (tidak isomorfik) pada graf lintasan dengan nilai *n* tertentu. Pelabelan harmonis yang berbeda dapat diperoleh dengan menghindari pelabelan yang isomorfik. Pelabelan harmonis yang isomorfik pada graf lintasan dapat terjadi karena kasus refleksi terhadap pelabelan yang ada. Pada Gambar 3.4 diberikan contoh kasus refleksi untuk P_4. Kasus refleksi pada graf lintasan dapat dihindari dengan pemberian syarat, label simpul pertama $f(v_1)$ lebih kecil atau lebih besar dari label simpul terakhir $f(v_n)$. Disini diambil $f(v_1)$ lebih kecil dari $f(v_n)$, diberikan pada Algoritma 1.2 baris 2. Label terbesar yang dapat digunakan untuk melabel simpul v_1 adalah $n-2$, dikarenakan label simpul $f(v_1)$ lebih kecil dari label simpul $f(v_n)$ sehingga harus disisakan satu label yang lebih besar dari label simpul $f(v_1)$ yaitu $n-1$, yang akan digunakan untuk melabel simpul v_n. Algoritma diberikan pada Algoritma 1.1 dan Algoritma 1.2.

Selanjutnya, akan diberikan contoh penggunaan algoritma ini untuk membangun pelabelan harmonis pada P_4. Banyak simpul dan busur pada P_4 adalah $n = 4$ dan $m = 3$ maka label simpul yang tersedia adalah $V = \{0,1,2,3\}$ dan label busur yang mungkin tersedia adalah $E = \{0,1,2\}$.



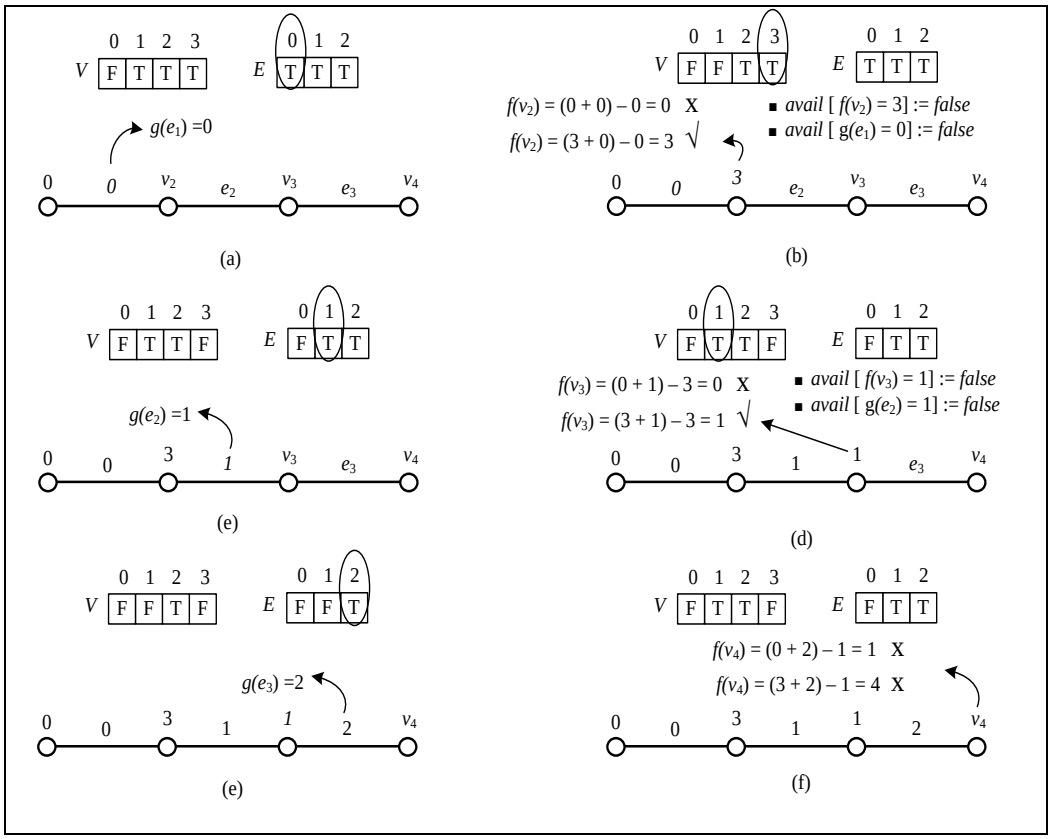
Gambar 3.5 Tahap inisialisasi algoritma pelabelan harmonis P_4

Algoritma ini dimulai dari fungsi initializePath untuk melabel simpul v_1. Misalkan $f(v_1) = 0$, karena $0 \leq n - 2 = 2$, maka avail $[f(v_1) = 0] \coloneqq false$, yang menandakan bahwa label 0 sudah digunakan dan tidak tersedia lagi (Gambar 3.5b). Label simpul yang tersedia menjadi *V*={1,2,3} dan label busur yang tersedia tetap (Gambar 3.5c).

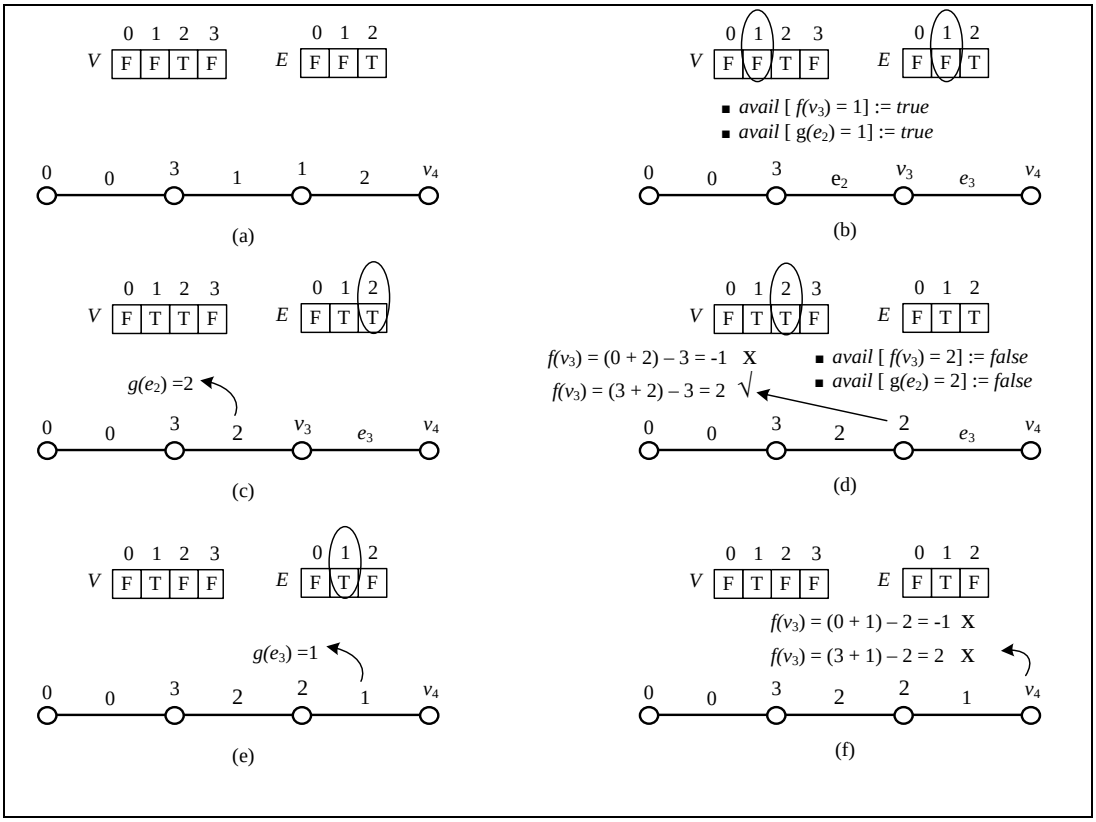
Kemudian memanggil fungsi extendPath(*t*) dengan parameter *t* = 2 untuk melabel busur e_1 dan simpul v_2. Misalkan $g(e_1) = 0$ (Gambar 3.6a). Dengan mengambil $k = 0$, label simpul $f(v_2) = (mk+ g(e_1)) - f(v_1) = (0 + 0) - 0 = 0$, label 0 tidak tersedia. Sehingga diambil $k = 1$, $f(v_2) = (mk+ g(e_1)) - f(v_1) = (3 + 0) - 0 = 3$. Karena label $3 \geq 0$, $3 \leq n - 1 = 3$, dan tersedia, maka label digunakan dan dinyatakan avail $[f(v_2) = 3] \coloneqq false$ dan avail $[g(e_1) = 0] \coloneqq false$ (Gambar 3.6b). Sehingga label simpul yang tersedia menjadi $V = \{1,2\}$ dan label busur yang tersedia menjadi $E = \{1,2\}$.

Kemudian secara rekursif memanggil fungsi extendPath(*t*) untuk *t* = 3, yaitu melabel busur e_2 dan simpul v_3. Misalkan $g(e_2) = 1$ (Gambar 3.6c). Setelah itu, ditentukan label simpul v_3. Untuk $k = 0$, label simpul $f(v_3) = (mk+ g(e_2)) - f(v_2) = (0 + 1) - 3 = -2$. Karena $-2 \leq 0$, maka -2 tidak bisa untuk melabel simpul v_3. Sehingga diambil $k = 1$, $f(v_3) = (mk+ g(e_2)) - f(v_2) = (3 + 1) - 3 = 1$. Karena $1 \geq 0$, $1 \leq n - 1 = 3$, dan tersedia, maka $f(v_3) = 1$ dan dinyatakan avail$[f(v_3) = 1] \coloneqq false$ dan avail $[g(e_2) = 1] \coloneqq false$ (Gambar 3.6d). Sehingga label simpul yang tersedia menjadi $V = \{2\}$ dan label busur yang tersedia menjadi

$E = \{2\}$. Secara rekursif, dipanggil fungsi extendPath(t) untuk $t = 4 = n$ untuk melabel busur e_3 dan simpul v_4, yang merupakan langkah terakhir dari fungsi extendPath(t). Karena label busur yang tersedia adalah $E = \{2\}$, maka $g(e_3) = 2$ (Gambar 3.6e). Dengan mengambil $k = 0$, label simpul $f(v_4) = (mk + g(e_3)) - f(v_3) = (0 + 2) - 1 = 1$, label 1 tidak tersedia. Sehingga, diambil $k = 1$, $f(v_4) = (mk + g(e_3)) - f(v_3) = (3 + 2) - 1 = 4$. Karena $4 \geq n - 1 = 3$, maka 4 tidak bisa untuk melabel simpul v_4 (Gambar 3.6f). Sehingga perlu dilakukan *backtracking* ke tahap sebelumnya yaitu $t = 3$ dengan membuat label simpul v_3 dan label busur e_2 tersedia kembali, yaitu avail$[f(v_3) = 1] :=$ *true* dan avail $[g(e_2) = 1] :=$ *true* (Gambar 3.7b). Sehingga label simpul yang tersedia menjadi $V = \{1,2\}$ dan label busur yang tersedia menjadi $E = \{1,2\}$.



Gambar 3.6 Tahap perluasan algoritma pelabelan harmonis untuk $t = 2$ dan $t = 3$



Gambar 3.7 *Backtracking* ke simpul v_3

Misalkan $g(e_2) = 2$ (Gambar 3.7c). Dengan mengambil $k = 0$, $f(v_3) = (mk + g(e_2)) - f(v_2) = (0 + 2) - 3 = -1$. Karena $-1 \leq 0$, maka -1 tidak bisa untuk melabel simpul v_3. Sehingga diambil $k = 1$, $f(v_3) = (mk + g(e_2)) - f(v_2) = (3 + 2) - 3 = 2$. Karena $2 \geq 0$ dan $2 \leq n - 1 = 3$, serta tersedia, maka label 2 digunakan dan dinyatakan $avail\,[f(v_3) = 2] := false$ dan avail $[g(e_2) = 2] := false$ (Gambar 3.7d). Label simpul yang tersedia menjadi $V = \{1\}$ dan label busur yang tersedia menjadi $E = \{1\}$.

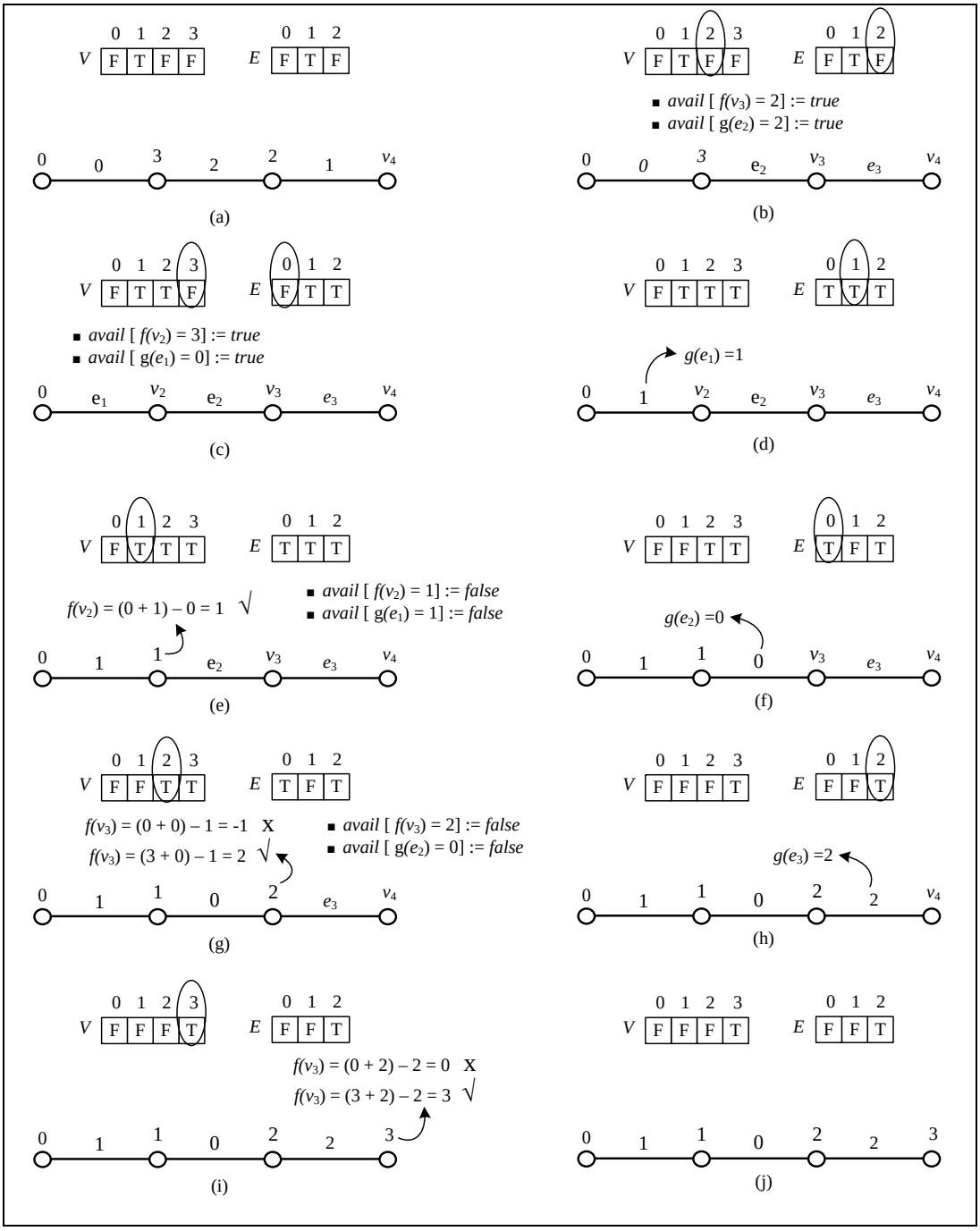
Kemudian memanggil fungsi extendPath(t) untuk $t = 4 = n$, yaitu melabel busur e_3 dan simpul v_4. Karena label busur yang masih tersisa adalah 1, maka label busur $g(e_3) = 1$ (Gambar 3.7e). Dengan mengambil $k = 0$, label simpul $f(v_4) = (mk + g(e_3)) - f(v_3) = (0 + 1) - 2 = -1$. Karena $-1 \leq 0$, maka -1 tidak bisa untuk melabel simpul v_4. Sehingga diambil $k = 1$, $f(v_4) = (mk + g(e_3)) - f(v_3) = (3 + 1) - 2 = 2$. Tetapi label 2 tidak tersedia (Gambar 3.7f). Sehingga perlu dilakukan *backtracking* ke tahap sebelumnya yaitu $t = 3$. Label simpul v_3 dan label busur e_2 dibuat tersedia kembali, yaitu avail$[f(v_3) = 2] := true$ dan

avail $[g(e_2) = 2] \coloneqq true$ (Gambar 3.8b). Sehingga label simpul yang tersedia menjadi $V = \{1,2\}$ dan label busur yang tersedia menjadi $E = \{1,2\}$. Karena semua kemungkinan label busur yang tersedia dan kemungkinan nilai *k* di *t* = 3 telah di-coba, maka perlu dilakukan *backtracking* kembali ke tahap sebelumnya yaitu *t* = 2. Label simpul v_2 dan label busur e_1 dibuat tersedia kembali, yaitu avail$[f(v_2) = 3] \coloneqq true$ dan avail $[g(e_1) = 0]$:= *true* (Gambar 3.8c). Sehingga label simpul yang tersedia menjadi $V = \{1,2,3\}$ dan label busur yang tersedia menjadi $E = \{0,1,2\}$.

Kemudian melabel kembali busur e_1, misalkan $g(e_1) = 1$ (Gambar 3.8d). Dengan mengambil $k = 0, f(v_2) = (mk + g(e_1)) - f(v_1) = (0 + 1) - 0 = 1$. Karena $1 \geq 0$ dan $1 \leq n - 1 = 3$, dan tersedia, maka label 1 digunakan dan dinyatakan $avail\ [f(v_2) = 1] \coloneqq false$ dan avail $[g(e_1) = 1] \coloneqq false$ (Gambar 3.8e). Sehingga label simpul yang tersedia menjadi $V = \{2,3\}$ dan label busur yang tersedia menjadi $E = \{0,2\}$. Label simpul $f(v_2)$ telah diperoleh maka tidak perlu lagi menghitung label simpul $f(v_2)$ untuk $k = 1$.

Kemudian memanggil fungsi extendPath(*t*) untuk *t* = 3, yaitu melabel busur e_2 dan simpul v_3. Misalkan $g(e_2) = 0$ (Gambar 3.8f). Dengan mengambil $k = 0$, label simpul $f(v_3) = (mk + g(e_2)) - f(v_2) = (0 + 0) - 1 = -1$. Karena $-1 \leq 0$, maka -1 tidak bisa untuk melabel simpul v_3. Sehingga diambil $k = 1$, $f(v_3) = (mk + g(e_2)) - f(v_2) = (3 + 0) - 1 = 2$. Karena $2 \geq 0$ dan $2 \leq n - 1 = 3$, dan tersedia, maka label 2 digunakan dan dinyatakan avail$[f(v_3) = 2] \coloneqq false$ dan avail $[g(e_2) = 0] \coloneqq false$ (Gambar 3.8g). Sehingga label simpul yang tersedia menjadi $V = \{3\}$ dan label busur yang tersedia menjadi $E = \{2\}$.

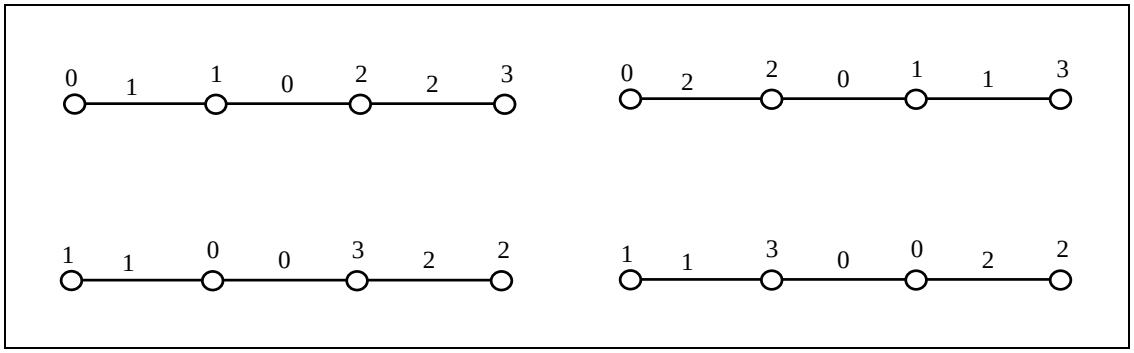
Secara rekursif, dipanggil fungsi extendPath(*t*) untuk *t* = 4 = *n* untuk melabel busur e_3 dan simpul v_4. Karena label busur yang masih tersisa di *array* label busur adalah 2, maka $g(e_3) = 2$ (Gambar 3.8h). Dengan mengambil $k = 0$, $f(v_4) = (mk + g(e_3)) - f(v_3) = (0 + 2) - 2 = 0$, tetapi label 0 tidak tersedia. Sehingga diambil $k = 1$, $f(v_4) = (mk + g(e_3)) - f(v_3) = (3 + 2) - 2 = 3$. Karena label simpul $3 \geq 0, 3 \leq n - 1 = 3$, dan memenuhi syarat $f(v_4) > f(v_1)$, serta tersedia, maka terbentuklah satu pelabelan harmonis pada graf lintasan P_4 dimana V={0, 1, 2, 3} dan E = {1, 0, 2} (Gambar 3.8i). Kemudian memanggil fungsi print() untuk mencetak elemen pelabelan harmonis tersebut.



Gambar 3.8 *Backtracking* ke simpul v_3 dan *backtracking* ke simpul v_2

Untuk memperoleh pelabelan harmonis lainnya pada graf lintasan P_4, maka dilakukan *backtracking* ke tahap sebelumnya sesuai dengan urutan langkah-langkah sebelumnya. Algoritma ini akan berhenti hingga diperoleh semua

pelabelan harmonis yang mungkin dan berbeda pada graf lintasan P_4. Terdapat 4 pelabelan harmonis yang berbeda pada graf lintasan P_4 (Gambar 3. 9).

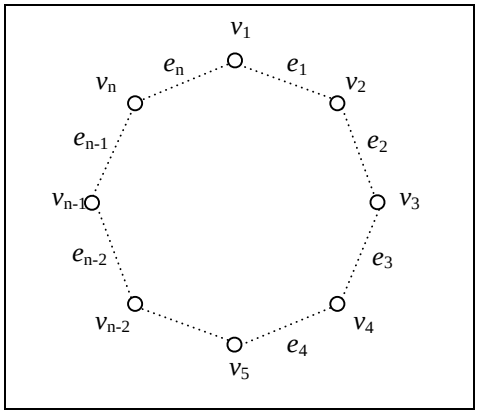


Gambar 3. 9 Semua pelabelan harmonis yang mungkin dan berbeda pada graf lintasan P_4

Pada bagian selanjutnya akan dijelaskan mengenai pembentukan algoritma pelabelan harmonis untuk graf lingkaran C_n.

3.2 Algoritma Pelabelan Harmonis pada Graf Lingkaran C_n

Pada pembahasan Subbab 2.2 telah diberikan definisi dari graf lingkaran, C_n ($n \geq 3$), yaitu graf yang diperoleh dari graf lintasan P_n yang diberi tambahan busur antara simpul awal dan simpul akhir (busur $v_n v_1$). Semua simpul pada graf memilliki derajat dua. Dalam graf lingkaran, banyak simpul dan banyak busur adalah sama ($|V(C_n)| = |E(C_n)| = n = m$). Untuk mempermudah pembuatan algoritma pelabelan harmonis pada graf lingkaran C_n, penamaan simpul dan busur graf lingkaran dengan n simpul diberikan pada Gambar 3.10.



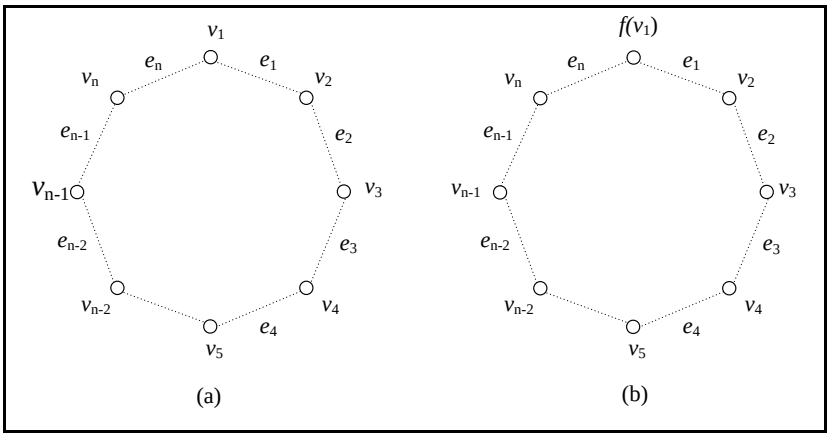
Gambar 3.10 Graf lingkaran dengan *n* simpul

Algoritma pelabelan harmonis untuk graf ini bekerja secara iteratif. Label simpul dan label busur pada graf lingkaran menggunakan bilangan modulo *m*, yaitu $\{0, 1, 2, \dots, m-1\}$. Pelabelan harmonis pada graf ini merupakan pemetaan bijektif, sehingga label-label tersebut hanya dapat digunakan tepat 1 kali. Label simpul dan label busur masing-masing dinyatakan sebagai *array*. *Array* label simpul dan *array* label busur berukuran *n*, dimana *array* tersebut berisi status dari label-label tersebut bernilai *true* atau *false*. Status dari label-label mula-mula bernilai *true* yang menandakan label tersedia.

Masukan dari algoritma ini sama seperti graf lintasan yaitu *n* (banyak simpul). Label simpul $f(v_1)$ dan busur $g(e_t)$ akan diberikan sesuai dengan label yang tersedia. Sedangkan label simpul $f(v_t)$ ditentukan, yaitu $f(v_t) = (mk + g(e_{t-1})) - f(v_{t-1})$, dimana $t = 2, \dots, n$ dan $m = |E|$, serta nilai k dapat bernilai $k = 0$ dan $k = 1$. Kondisi yang harus dipenuhi adalah $0 \leq f(v_t) \leq n-1$, dimana $t = 1,2, \dots, n$. Karena label simpul $f(v_t)$ ditentukan dari $\left(mk + g(e_{t-1})\right) - f(v_{t-1})$, maka label simpul $f(v_t)$ disebut *determined label*. Jika label simpul $f(v_t)$ dan label busur $g(e_t)$ tersedia masing-masing akan dinyatakan *avail* $[f(v_t)]$:= *true* dan *avail* $[g(e_t)]$:= *true*. Setelah label simpul $f(v_t)$ dan label busur $g(e_t)$ terpakai masing-masing akan dinyatakan *avail* $[f(v_t)]$:= *false* dan *avail* $[g(e_t)]$:= *false*.

Langkah pertama pada algoritma ini adalah melabel simpul v_1, yang merupakan tahap inisialisasi (Gambar 3.11). Setelah tahap inisialisasi dilakukan, masuk ke tahap perluasan (Gambar 3.12). Langkah pertama yang dilakukan pada tahap perluasan adalah melabel busur e_1. Kemudian menentukan label simpul v_2.

Setelah label v_2 diperoleh, melabel busur e_2. Kemudian menentukan label simpul v_3 dan seterusnya hingga melabel busur e_{t-1}, simpul v_t, dan busur e_t.

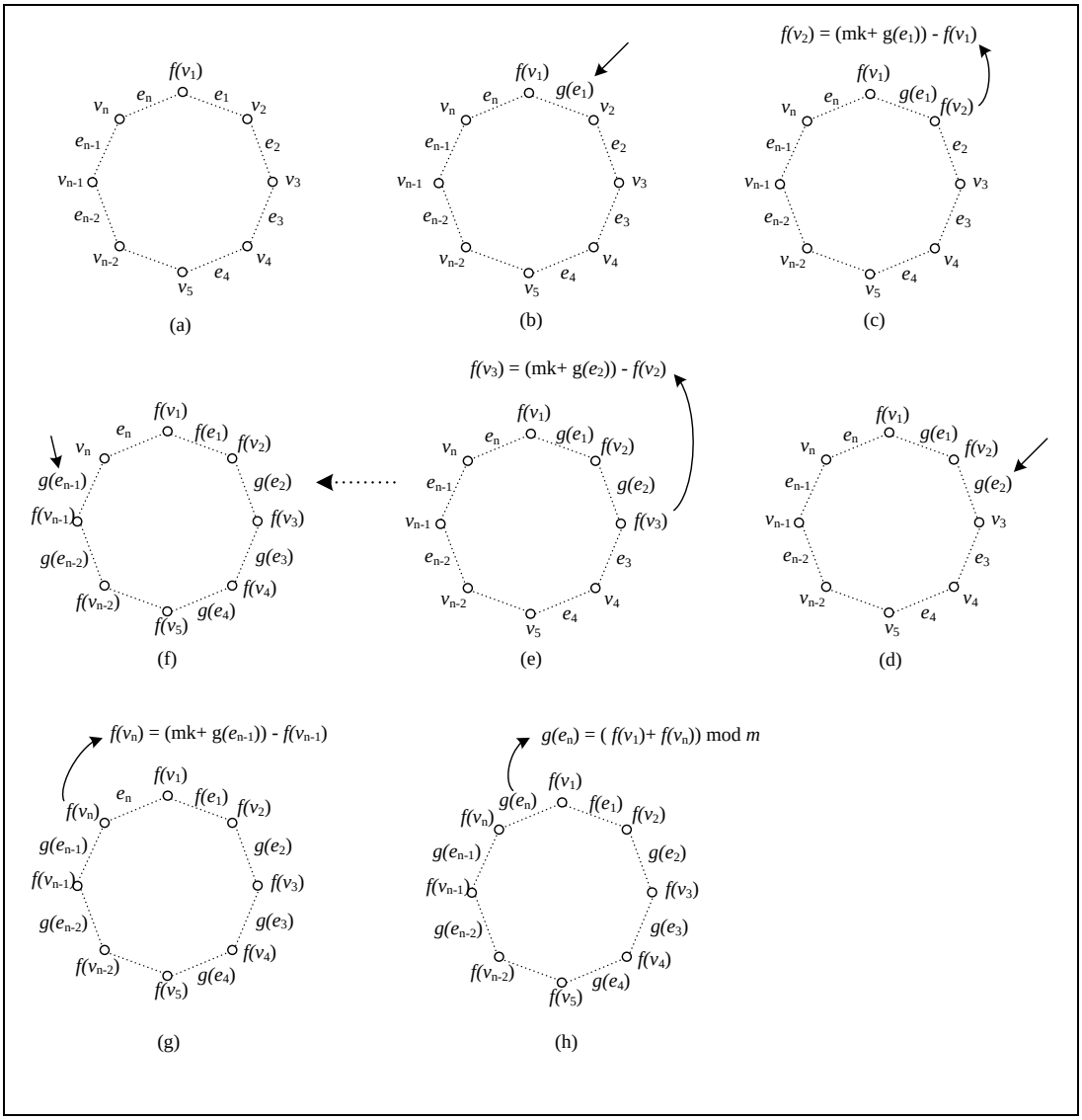


Gambar 3.11 Tahap inisialisasi algoritma pelabelan harmonis C_n

Karena algoritma ini terdiri dari dua tahap, yaitu tahap inisialisasi dan tahap perluasan, maka algoritma ini terdiri dari dua fungsi yaitu fungsi initialize-Cycle dan extendCycle(*t*). Tahap inisialisasi berada dalam fungsi initializePath. Sedangkan Tahap perluasan berada dalam fungsi extendCycle(*t*) dengan parameter *t* = 2, 3, ..., *n*.

Pelabelan simpul v_1 berada dalam fungsi initializeCycle. Kemudian memanggil fungsi extendCycle(*t*) dengan parameter *t* = 2. Fungsi extendCycle(*t*) ini dengan parameter *t* = 2 dimulai dengan melabel busur e_1. Kemudian melabel simpul v_2, yaitu $f(v_2) = \big(mk + g(e_1)\big) - f(v_1)$. Jika label simpul v_2 memenuhi kondisi $0 \leq f(v_t) \leq n-1$ dan tersedia maka digunakan dan dinyatakan *avail* $[f(v_2)]$:= *false* dan *avail* $[g(e_1)]$:= *false*. Selanjutnya secara rekursif memanggil fungsi extendCycle(*t*) dengan parameter *t* = *t* + 1 = 3 untuk melabel busur e_2 dan menentukan label simpul v_3, dan seterusnya hingga memanggil fungsi extend-Cycle(*t*) dengan parameter *t* = *n*. Saat *t* = *n*, busur e_{n-1} diberi label, kemudian menentukan label simpul v_n. Setelah melabel simpul v_n, maka label busur $g(e_{n-1})$ sudah tidak tersedia dan dinyatakan *avail* $[g(e_{n-1})]$:= *false*. Kemudian ditentukan busur e_n, yaitu $g(e_n) = \big(f(v_n) + f(v_1)\big) \, mod \, m$. Label busur $g(e_n)$ menggunakan label terakhir yang masih tersedia. Jika label busur $g(e_n)$ tersedia,

maka satu pelabelan harmonis berhasil dibentuk. Kemudian elemen pelabelan harmonis yang terbentuk pada graf lingkaran C_n di cetak dengan fungsi print().



Gambar 3.12 Tahap perluasan algoritma pelabelan harmonis C_n

Setelah melabel busur e_{t-1}, ditentukan label simpul $f(v_t)$. Jika label simpul $f(v_t)$ tidak tersedia maka ganti label busur $g(e_{t-1})$ hingga memperoleh label simpul $f(v_t)$. Jika tidak ada label busur $g(e_{t-1})$ yang memenuhi sehingga memperoleh label simpul $f(v_t)$, maka dilakukan *backtracking* untuk mengubah label simpul yang dilabel sebelumnya, yaitu mengubah label simpul $f(v_{t-1})$ dan label busur $g(e_{t-2})$. Label simpul dan label busur yang digunakan sebelumnya menjadi *available* (tersedia kembali) pada *array* label simpul dan *array* label

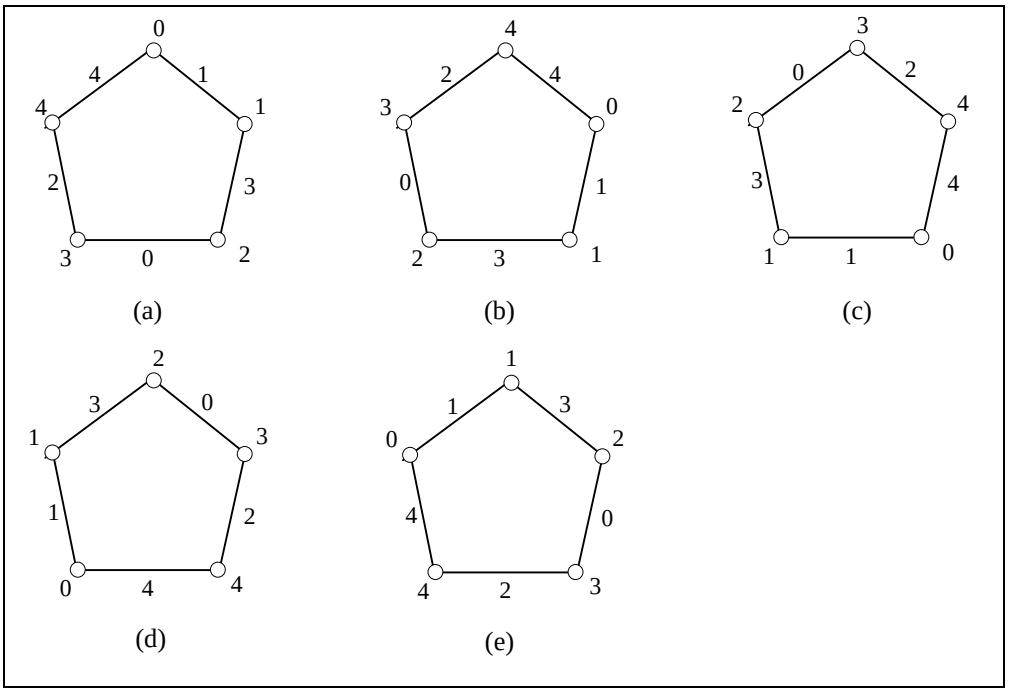
busur,yaitu masing-masing ditandai dengan *avail* $[f(v_{t-1})]$:= *true* dan *avail* $[g(e_{t-2})]$:= *true*. Langkah ini dilakukan hingga diperoleh label simpul $f(v_t)$ yang sesuai dengan label yang tersedia. Jika label simpul $f(v_t)$ tetap tidak tersedia, berarti tidak terdapat pelabelan harmonis yang berbeda pada graf lingkaran dengan nilai *n* yang diberikan. Algoritma ini akan berlanjut untuk mencari pelabelan harmonis lainnya dengan melakukan *backtracking* ke simpul v_{n-1} dan seterusnya sesuai dengan urutan langkah-langkah sebelumnya. Algoritma ini akan berhenti jika telah diperoleh semua pelabelan harmonis yang tidak isomorfik dengan nilai *n* yang diberikan.

```
Algoritma 2.1
function initialCycle()
1       f(v_1) := 0
2       avail [0] := false
3       extendCycle(2)
```

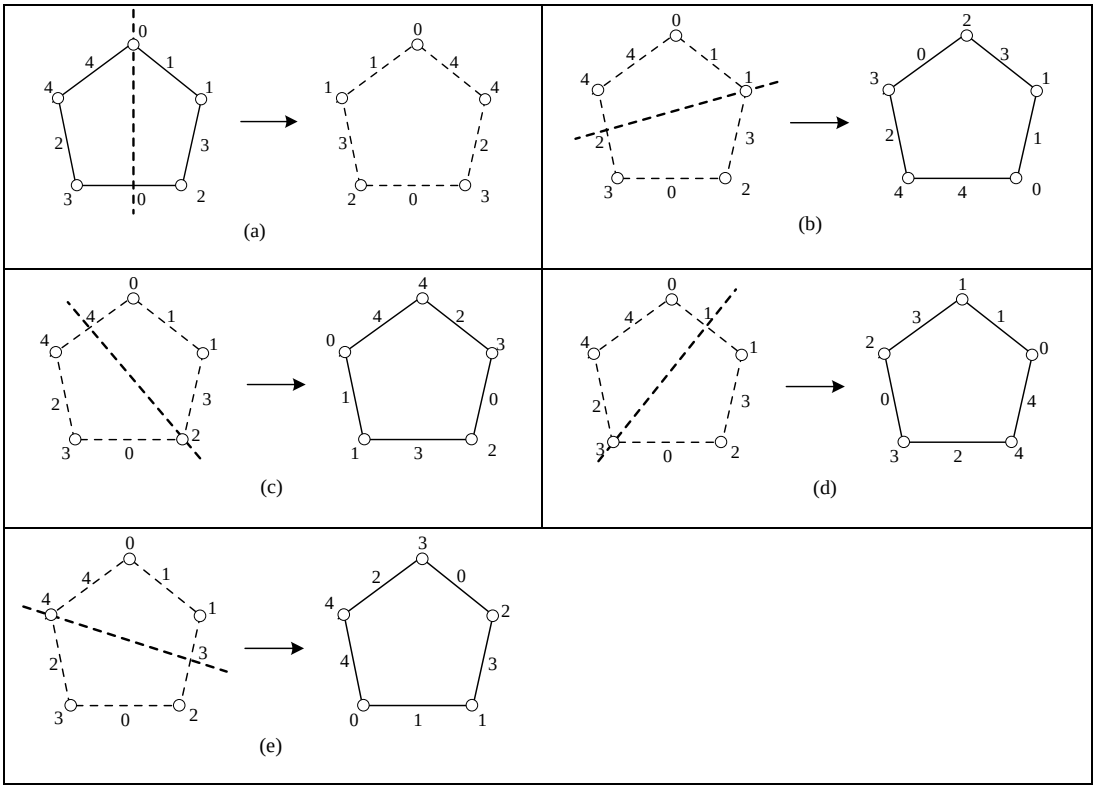
```
Algoritma 2.2
function extendCycle(t)
1       if t = n then
2           for each available label j do
3               g(e_{t-1}) := j
4               for k=1 or k=2 do
5                   f(v_t) = [mk + g(e_{t-1})] - f(v_{t-1})
6                   if 0 ≤ f(v_t) ≤ n - 1 and avail [f(v_t)] and f(v_t) > f(v_1) then
7                       avail [j] := false
8                       g(e_t) := [f(v_1) + f(v_t)] mod m
9                       if avail [g(e_t)] and g(e_t) > g(e_1) then
10                          print()
                        end if
11                      avail [j] := true
                    end if
                end for
            end for
12      else
13          for each available label j do
14              g(e_{t-1}) := j
15              for k=1 or k=2 do
16                  f(v_t) = [mk + g(e_{t-1})] - f(v_{t-1})
17                  if 0 ≤ f(v_t) ≤ n - 1 and avail [f(v_t)] and f(v_t) > f(v_1) then
18                      avail [f(v_t)] := false
19                      avail [j] := false
20                      extendCycle (t+1)
21                      avail [f(v_t)] := true
22                      avail [j] := true
                    end if

                end for
            end for
        end if
```

Kemudian saat $t = n$, jika label busur $g(e_n)$ tidak tersedia, maka dilakukan *backtracking*, yaitu mengubah label simpul $f(v_n)$ dan label busur $g(e_{n-1})$. Label simpul $f(v_n)$ dan label busur $g(e_{n-1})$ menjadi *available* (tersedia kembali), yaitu masing-masing dinyatakan dengan *avail* $[f(v_n)]$:= *true* dan *avail* $[g(e_{n-1})]$:= *true*. Kemudian busur e_{n-1} dan simpul v_n diberi label kembali. Jika tidak tersedia label simpul yang memenuhinya, maka dilakukan *backtracking* ke label simpul sebelumnya, yaitu simpul v_{n-1}. Langkah ini dilakukan hingga diperoleh label busur $g(e_n)$ yang sesuai dengan label yang tersedia. Jika label busur $g(e_n)$ tetap tidak tersedia, berarti tidak terdapat pelabelan harmonis yang berbeda pada graf lingkaran dengan nilai *n* yang diberikan. Algoritma ini terangkum pada Algoritma 2.1 dan Algoritma 2.2.

Algoritma ini bertujuan untuk membangun algoritma pelabelan harmonis yang menghasilkan semua pelabelan yang berbeda (tidak isomorfik) pada graf lingkaran dengan nilai *n* tertentu. Pelabelan harmonis yang berbeda dapat diperoleh dengan menghindari pelabelan harmonis yang isomorfik. Pelabelan harmonis yang isomorfik pada graf lingkaran dapat terjadi karena kasus refleksi dan kasus rotasi terhadap pelabelan yang ada. Contoh kasus rotasi dan refleksi untuk C_5 masing-masing diberikan pada Gambar 3.13 dan Gambar 3.14.



Gambar 3.13 Kasus rotasi pada graf lingkaran C_5



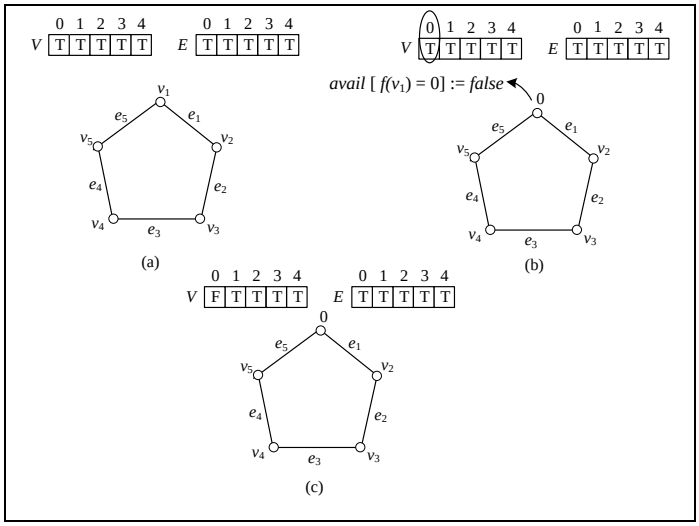
Gambar 3.14 Kasus refleksi pada graf lingkaran C_5

Kasus rotasi pada graf lingkaran dapat dihindari dengan pemberian syarat, yaitu label simpul pertama $f(v_1)$ lebih kecil dari label simpul lainnya $f(v_t)$. Syarat $f(v_1) < f(v_t)$ diberikan pada Algoritma 2.2 baris 6 dan 17. Karena label simpul yang tersedia adalah {0,1,...,*n*-1}, maka label simpul $f(v_1)$ hanya bisa dilabel dengan label 0. Sedangkan kasus refleksi pada graf lingkaran dapat dihindari dengan pemberian syarat, yaitu label busur pertama $g(e_1)$ lebih kecil dari label simpul $g(e_n)$, yang diberikan pada Algoritma 2.2 baris 9.

Selanjutnya, akan diberikan contoh penggunaan algoritma ini. Misalkan akan dibangun pelabelan harmonis pada C_5, berarti $n = m = 5$. Label simpul yang tersedia, $V = \{0,1,2,3,4\}$ dan label busur yang mungkin tersedia, $E = \{0,1,2,3,4\}$.

Algoritma ini dimulai dengan melabel simpul v_1 pada fungsi initialize-Cycle(t). Karena harus memenuhi $f(v_1) < f(v_t)$ maka label terkecil untuk melabel simpul v_1 adalah 0, sehingga $f(v_1) = 0$ dan dinyatakan avail $[f(v_1) = 0] \coloneqq false$, yang menandakan bahwa label 0 sudah digunakan dan tidak tersedia

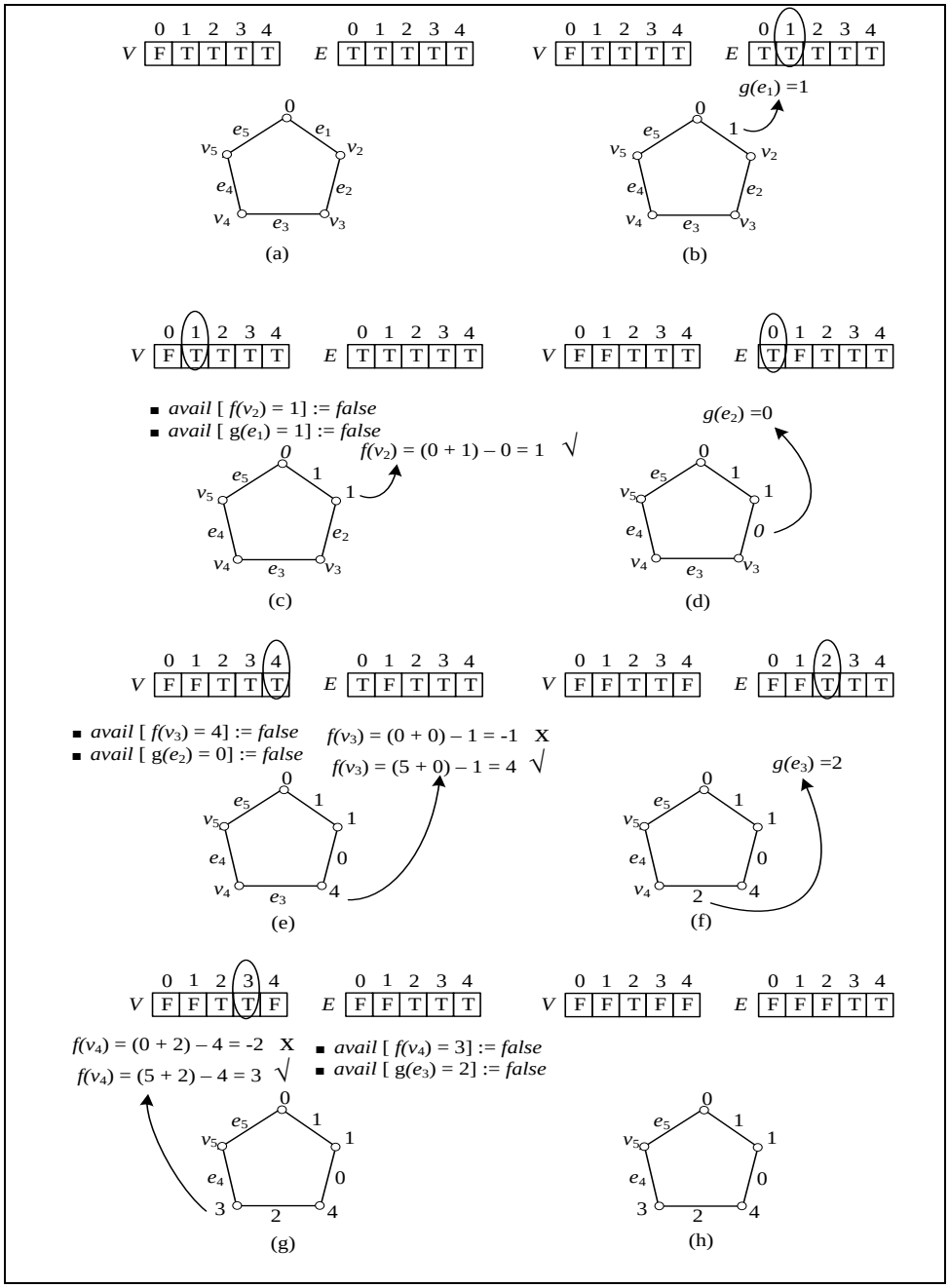
lagi. Label simpul yang tersedia menjadi *V*={1,2,3,4} dan label busur yang tersedia tetap (Gambar 3.15). Kemudian secara iteratif memanggil fungsi extendCycle(*t*) dengan parameter *t* = 2 untuk melabel busur e_1 dan simpul v_2. Misalkan label busur $g(e_1) = 1$ (Gambar 3.16b). Dengan mengambil $k = 0$, $f(v_2) = (mk+ g(e_1)) - f(v_1) = (0 + 1) - 0 = 1$. Karena $1 \geq 0$ dan $1 \leq n - 1 = 4$, dan tersedia, maka digunakan dan dinyatakan avail $[f(v_2) = 1] \coloneqq false$ dan avail $[g(e_1) = 1] \coloneqq false$ (Gambar 3.16c). Sehingga label simpul yang tersedia menjadi $V = \{2,3,4\}$ dan label busur yang tersedia menjadi $E = \{0,2,3,4\}$.



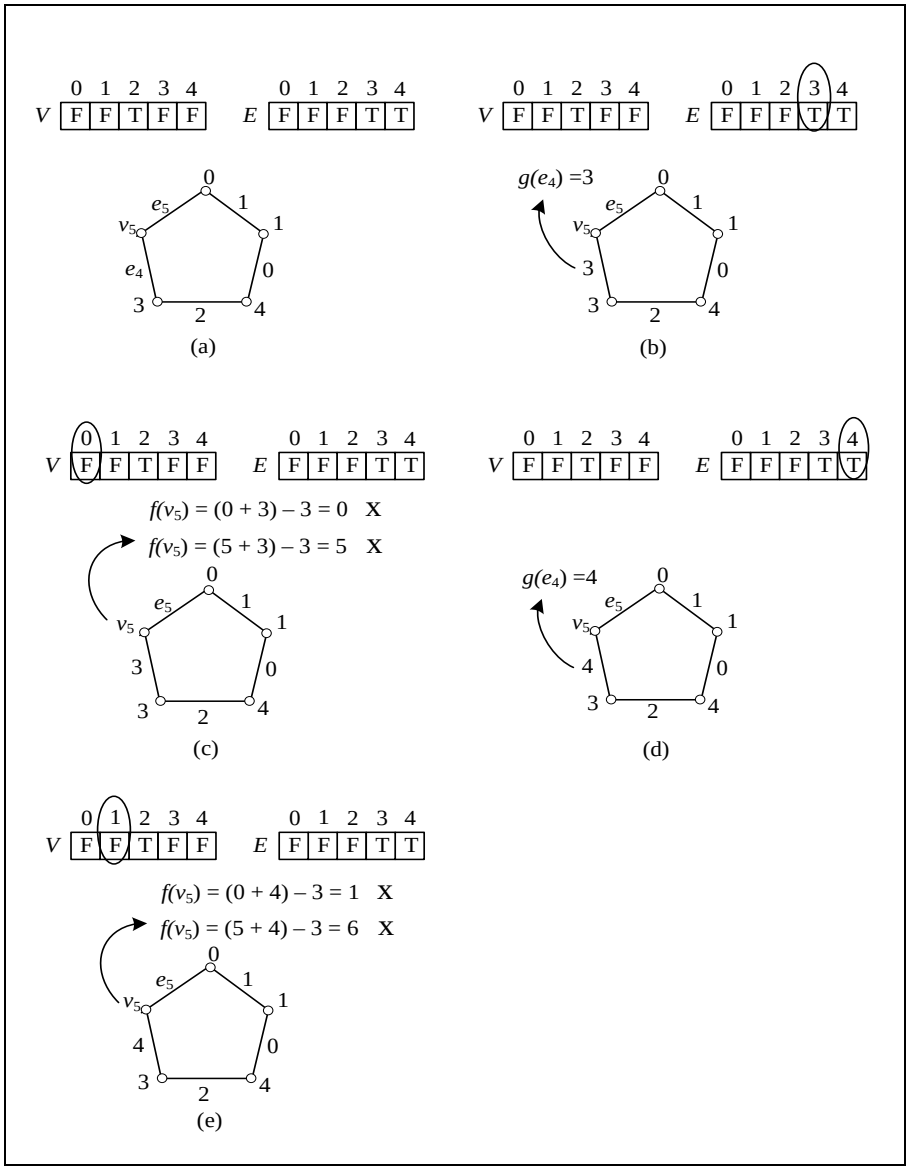
Gambar 3.15 Tahap inisialisasi algoritma pelabelan harmonis C_5

Secara rekursif, dipanggil fungsi extendPath(*t*) untuk *t* = *t* +1 = 3, untuk melabel busur e_2 dan simpul v_3. Misalkan $g(e_2) = 0$ (Gambar 3.16d). Dengan mengambil $k = 0$, $f(v_3) = (mk + g(e_2)) - f(v_2) = (0 + 0) - 1 = -1$. Karena $-1 \leq 0$, maka -1 tidak bisa untuk melabel simpul v_3. Kemudian untuk $k = 1$, $f(v_3) = (mk + g(e_2)) - f(v_2) = (5 + 0) - 1 = 4$. Karena label simpul $4 \geq 0$ dan $4 \leq n - 1 = 4$, dan tersedia, maka $f(v_3) = 4$ dan dinyatakan avail $[f(v_3) = 4] \coloneqq false$ dan avail $[g(e_2) = 0] \coloneqq false$ (Gambar 3.16e). Sehingga label simpul yang tersedia menjadi $V = \{2,3\}$ dan label busur yang tersedia menjadi $E = \{2,3,4\}$. Kemudian memanggil fungsi extendPath(*t*) untuk *t*= 4 untuk melabel busur e_3 dan simpul v_4. Misalkan $g(e_3) = 2$ (Gambar 3.16f). Dengan mengambil $k = 0$, label simpul $f(v_4) = \big(mk + g(e_3)\big) - f(v_3) = (0 + 2) - 4 = -2$, Karena $-2 \leq 0$, maka -2 tidak bisa untuk melabel simpul v_4. Sehingga diambil $k =$

1, $f(v_4) = (mk + g(e_3)) - f(v_3) = (5 + 2) - 4 = 3$. Karena $3 \geq 0$ dan $3 \leq n - 1 = 4$, dan tersedia, maka digunakan dan dinyatakan avail $[f(v_4) = 3] := false$ dan $avail\ [g(e_3) = 2] := false$ (Gambar 3.16g). Sehingga label simpul yang tersedia menjadi $V = \{2\}$ dan label busur yang tersedia menjadi $E = \{3,4\}$ (Gambar 3.16h).



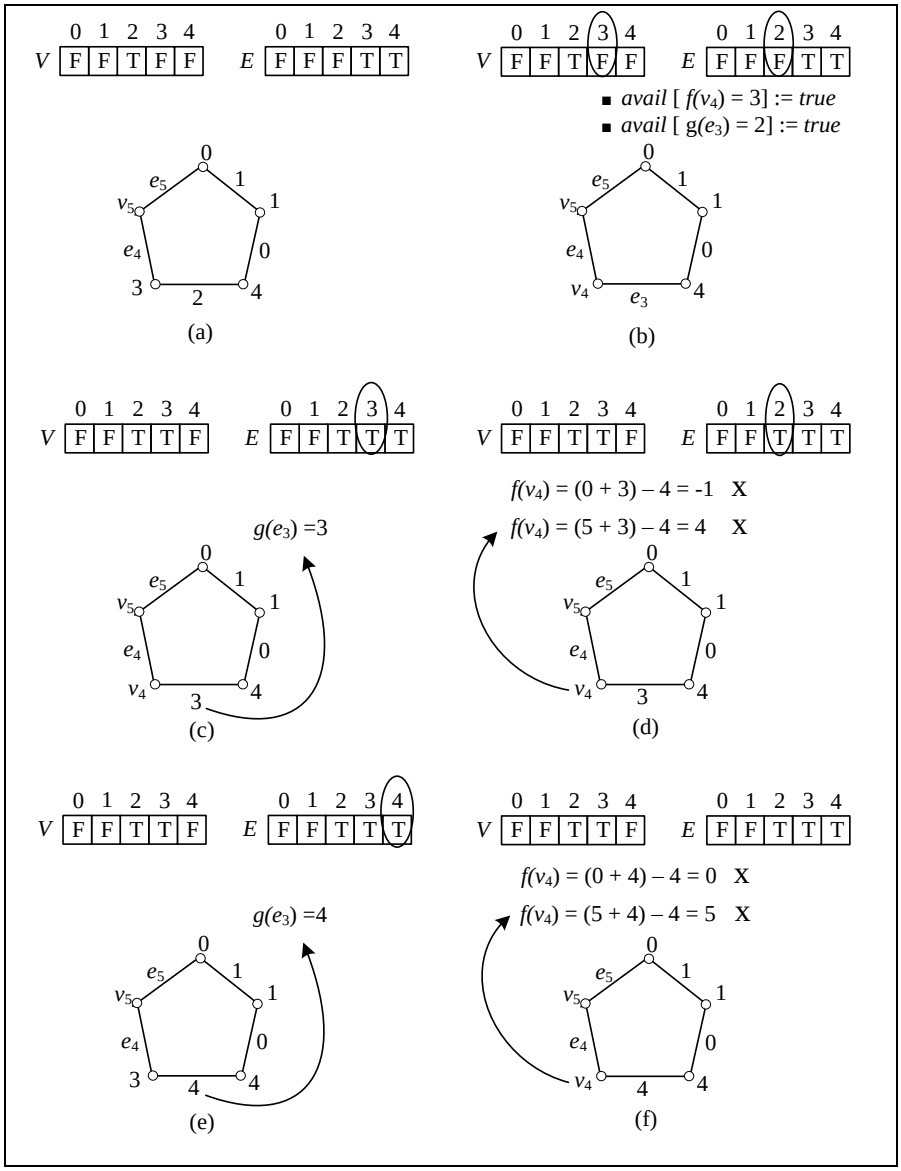
Gambar 3.16 Tahap perluasan algoritma pelabelan harmonis saat *t* = 2,3,4



Gambar 3.17 Tahap perluasan algoritma pelabelan harmonis saat $t = 5$

Kemudian memanggil fungsi extendPath(t) untuk $t = 5 = n$, yaitu melabel busur e_4, simpul v_5,dan busur e_5. Misalkan $g(e_4) = 3$ (Gambar 3.17b). Dengan mengambil $k = 0, f(v_5) = (mk + g(e_4)) - f(v_4) = (0 + 3) - 3 = 0$, karena 0 tidak tersedia. Sehingga diambil $k = 1, f(v_5) = (mk + g(e_4)) - f(v_4) = (5 + 3) - 3 = 5$. Karena $5 \geq n - 1 = 4$, maka 5 tidak bisa untuk melabel simpul v_5 (Gambar 3.17c). Kemudian diganti label busur $g(e_4)$. Misalkan $g(e_4) = 4$ (Gambar 3.17d). Untuk $k = 0, f(v_5) = (mk + g(e_4)) - f(v_4) = (0 + 4) - 3 = 1$, label 1 tidak tersedia. Kemudian ditentukan label simpul v_5 untuk $k = 1, f(v_5) = (mk + g(e_4)) - f(v_4) = (5 + 4) - 3 = 6$. Karena $6 \geq n - 1 = 4$, maka 6 tidak bisa untuk

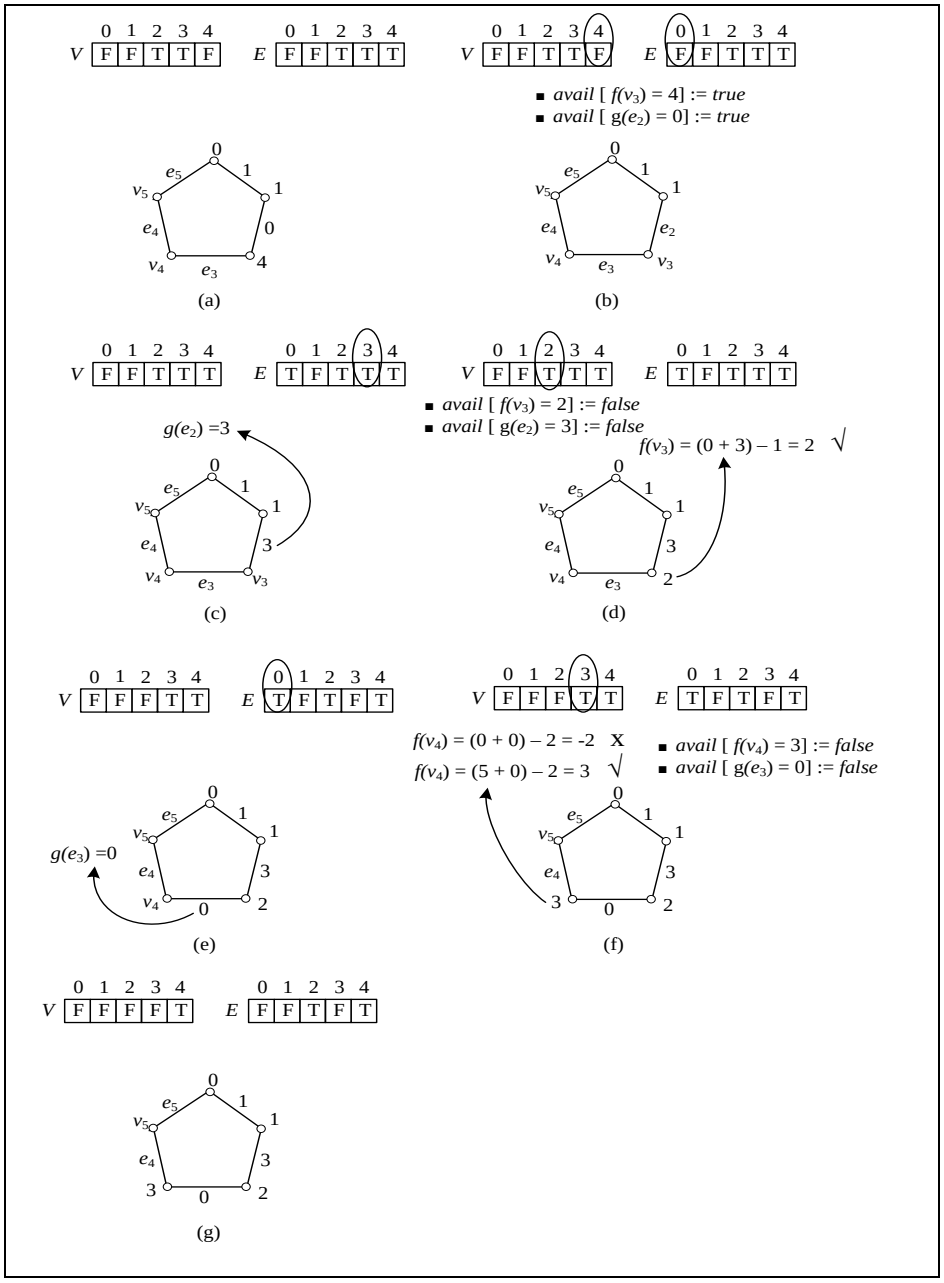
melabel simpul v_5 (Gambar 3.17e) Karena semua kemungkinan label busur pada $t = n = 5$, tidak ada yang memenuhi, maka perlu dilakukan *backtracking* untuk mengubah simpul v_4 dan label busur e_3. Kemudian label simpul v_4 dan label busur e_3 dibuat tersedia kembali, yaitu avail$[f(v_4) = 3] :=$ *true* dan avail $[g(e_3) = 2] :=$ *true* (Gambar 3.18b). Sehingga label simpul yang tersedia menjadi $V = \{2,3\}$ dan label busur yang tersedia menjadi $E = \{2,3,4\}$. Kemudian ganti label busur $g(e_3)$.



Gambar 3.18 *Backtracking* ke simpul v_4

Misalkan $g(e_3) = 3$ (Gambar 3.18c). Dengan mengambil $k = 0$, $f(v_4) = (mk + g(e_3)) - f(v_3) = (0 + 3) - 4 = -1$. Karena $-1 \leq 0$, maka -1 tidak bisa untuk melabel simpul v_4. Kemudian ditentukan label simpul v_4 untuk $k = 1$, $f(v_4) = (mk + g(e_3)) - f(v_3) = (5 + 3) - 4 = 4$, label 4 tidak tersedia. Saat $g(e_3) = 3$, tidak diperoleh label simpul $f(v_4)$ yang tersedia (Gambar 3.18d). Oleh karena itu, $g(e_3) = 3$ diganti menjadi $g(e_3) = 4$ (Gambar 3.18e). Kemudian ditentukan label simpul v_4. Dengan mengambil $k = 0$, label simpul $f(v_4) = (mk + g(e_3)) - f(v_3) = (0 + 4) - 4 = 0$, label 0 tidak tersedia. Sehingga diambil $k = 1$, $f(v_4) = (mk + g(e_3)) - f(v_3) = (5 + 4) - 4 = 5$. Karena $5 \geq n - 1 = 4$ maka 5 tidak bisa untuk melabel simpul v_4 (Gambar 3.18f). Karena semua kemungkinan label busur pada *t* = 4 tidak ada yang memenuhi, maka perlu dilakukan *backtracking* ke tahap *t* = 3. Kemudian label simpul v_3 dan label busur e_2 dibuat tersedia kembali, yaitu avail$[f(v_3) = 4] :=$ *true* dan avail $[g(e_2) = 0] :=$ *true* (Gambar 3.19b). Sehingga label simpul yang tersedia menjadi $V = \{2,3,4\}$ dan label busur yang tersedia menjadi $E = \{0,2,3,4\}$.

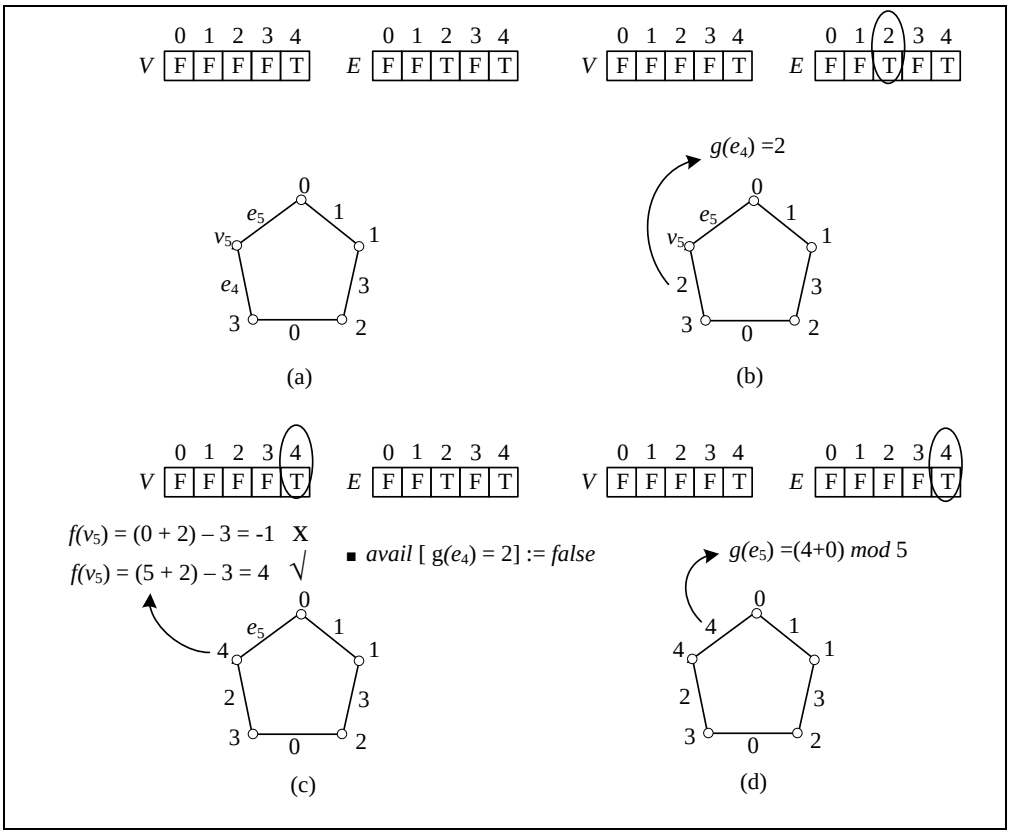
Kemudian, diganti label busur $g(e_2)$. Misalkan $g(e_2) = 3$ (Gambar 3.19c). Dengan mengambil $k = 0, f(v_3) = (mk + g(e_2)) - f(v_2) = (0 + 3) - 1 = 2$. Karena $2 \geq 0$ dan $2 \leq n - 1 = 4$, dan tersedia, maka label 2 digunakan dan dinyatakan avail $[f(v_3) = 2] :=$ *false* dan avail$[g(e_2) = 3] :=$ *false* (Gambar 3.19d). Sehingga label simpul yang tersedia menjadi $V = \{3,4\}$ dan label busur yang tersedia menjadi $E = \{0,2,4\}$. Kemudian memanggil fungsi extendPath(*t*) untuk $t = 4$, yaitu melabel busur e_3 dan simpul v_4. Misalkan $g(e_3) = 0$ (Gambar 3.19e). Dengan mengambil $k = 0$, $f(v_4) = (mk + g(e_3)) - f(v_3) = (0 + 0) - 2 = -2$, Karena $-2 \leq 0$, maka -2 tidak bisa untuk melabel simpul v_4. Kemudian ditentukan label simpul v_4 untuk $k = 1, f(v_4) = (mk + g(e_3)) - f(v_3) = (5 + 0) - 2 = 3$. Karena $3 \geq 0$ dan $f(v_4) = 3 \leq n - 1 = 4$, dan tersedia, maka label 3 digunakan dan dinyatakan avail $[f(v_4) = 3] :=$ *false* dan avail $[g(e_3) = 0] :=$ *false* (Gambar 3.19f). Sehingga label simpul yang tersedia menjadi $V = \{4\}$ dan label busur yang tersedia menjadi $E = \{2,4\}$ (Gambar 3.19g).



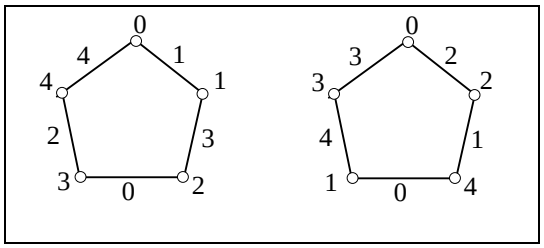
Gambar 3.19 *Backtracking* ke simpul v_3

Kemudian memanggil fungsi extendPath(t) untuk $t = 5 = n$,yaitu melabel busur e_4, simpul v_5,dan busur e_5. Misalkan $g(e_4) = 2$ (Gambar 3.20b). Dengan mengambil $k = 0$, label simpul $f(v_5) = (mk + g(e_4)) - f(v_4) = (0 + 2) - 3 = 0$, label 0 tidak tersedia. Sehingga diambil $k = 1$, $f(v_5) = (mk + g(e_4)) - f(v_4) = (5 + 2) - 3 = 4$. Karena $4 \geq 0$ dan $4 \leq n - 1 = 4$, dan tersedia, maka digunakan dan dinyatakan avail $[g(e_4) = 2] :=$ *false* (Gambar 3.20c). Kemudian ditentukan label busur e_5, yaitu $g(e_5) = \big(f(v_5) + f(v_1)\big) \bmod m = (4 + 0) \bmod 5 = 4$. Karena

memeuhi syarat $g(e_5) = 4 > g(e_1) = 1$ dan tersedia, maka terbentuk satu pelabelan harmonis pada graf lingkaran C_5 dimana V={ 0, 1, 2, 3, 4} dan E = {1, 3, 0, 2, 4} (Gambar 3.20d). Kemudian memanggil fungsi print() untuk mencetak elemen pelabelan harmonis tersebut.



Gambar 3.20 Tahap perluasan algoritma pelabelan harmonis saat $t = 5$



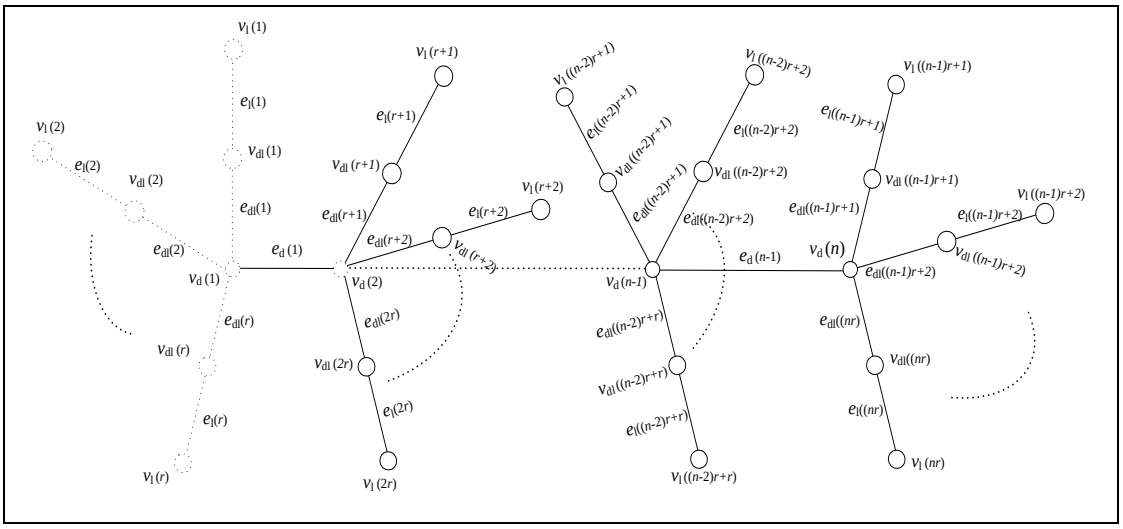
Gambar 3.21 Semua Pelabelan Harmonis yang mungkin dan berbeda pada Graf Lingkaran C_5

Untuk memperoleh pelabelan harmonis lainnya pada graf lingkaran C_5 maka dilakukan *backtracking* ke tahap sebelumnya sesuai dengan urutan langkah-langkah sebelumnya. Algoritma ini akan berhenti hingga diperoleh semua

pelabelan harmonis yang mungkin dan berbeda pada graf lingkaran C_5. Terdapat 2 pelabelan harmonis yang berbeda pada graf lingkaran C_5 (Gambar 3.21).

3.3 Algoritma Pelabelan Harmonis pada Graf Lobster Teratur $L_{n,r,1}$

Pada pembahasan Subbab 2.2 telah diberikan definisi dari graf lobster teratur $L_{n,r,1}$ ($n \geq 2$ dan $r \geq 1$) yaitu graf yang dibangun dengan menambahkan satu simpul daun pada setiap simpul daun dari graf caterpillar teratur. $|V(L_{n,r,1})| = 2nr+n$ dan $|E(L_{n,r,1})|= (2nr+n)-1$. Untuk mempermudah pembuatan algoritma pelabelan harmonis pada graf lobster teratur $L_{n,r,1}$, penamaan simpul dan busur pada graf lobster teratur $L_{n,r,1}$ diberikan pada Gambar 3.22.



Gambar 3.22 Graf lobster teratur $L_{n,r,1}$

Algoritma pelabelan harmonis untuk graf ini bekerja secara iteratif. Label simpul pada graf lobster teratur menggunakan {0, 1, 2, ..., $|V(L_{n,r,1})|$-1 } dan label busur menggunakan bilangan modulo m, yaitu {0, 1, 2, ..., m-1} dengan syarat label busur adalah jumlah (mod m) dari label dua simpul yang dihubungkan. Label-label tersebut digunakan tepat 1 kali. Label simpul dan label busur masing-masing dinyatakan sebagai *array*. *Array* label simpul dan *array* label busur masing-masing berukuran $|V(L_{n,r,1})|$ dan berukuran $|E(L_{n,r,1})|$, dimana *array* tersebut berisi status dari label-label tersebut bernilai *true* atau *false*. Status dari label-label mula-mula bernilai *true* yang menandakan label tersedia.

Masukan dari algoritma ini adalah *n* (banyak simpul pada *backbone* lobster, P_n) dan *r* (banyak simpul daun dari simpul-simpul pada *backbone* caterpillar teratur). Label simpul $f(v_d(1))$, $g(e_d(t))$, $g(e_{dl}(s))$ dan $g(e_l(s))$ akan diberikan sesuai dengan label yang tersedia, dimana *s*= 1,2,...,*nr* dan *t*= 1,2,...,*n*. Sedangkan label simpul $f(v_d(t))$, $f(v_{dl}(s))$, dan $f(v_l(s))$ ditentukan, yaitu :

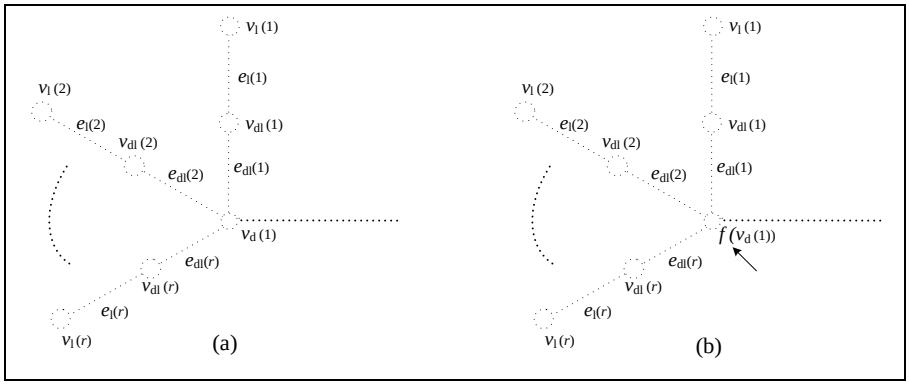
$$f(v_d(t)) = (mk + g(e_d(t-1))) - f(v_d(t-1)) \quad (3.1)$$

$$f(v_{dl}(s)) = (mk + g(e_{dl}(s))) - f(v_d(t)) \quad (3.2)$$

$$f(v_l(s)) = (mk + g(e_l(s))) - f(v_{dl}(s)) \quad (3.3)$$

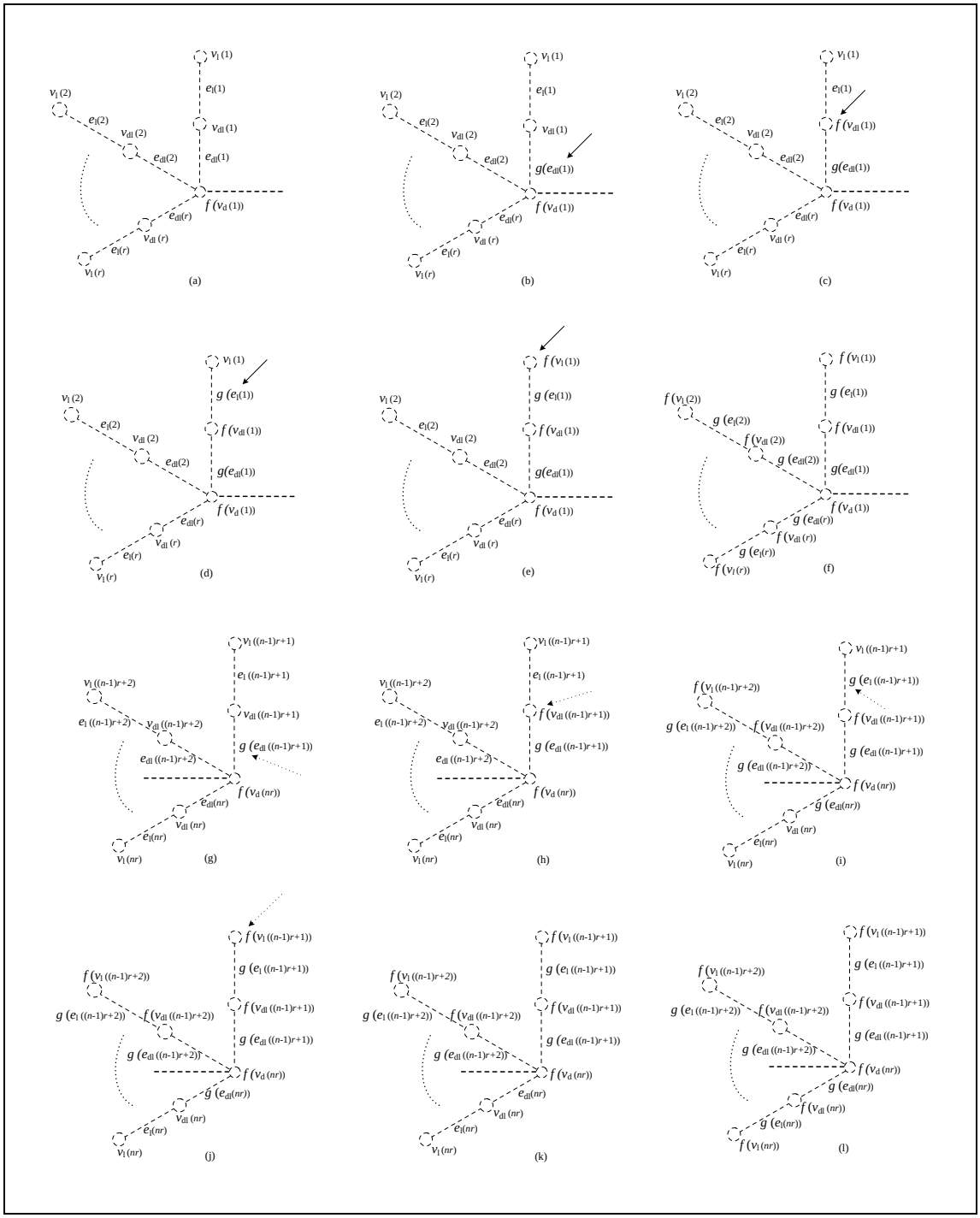
Dimana, *s* = 1, 2, ..., *nr*, *t* = 2, 3, ..., *n*, dan *m* = $|E(L_{n,r,1})|$, serta nilai k dapat bernilai *k* = 0 atau *k* = 1. Kondisi yang harus dipenuhi adalah $0 \leq f(v_d(t)) \leq |V(L_{n,r,1})| - 1$, $0 \leq f(v_{dl}(s)) \leq |V(L_{n,r,1})| - 1$, dan $0 \leq f(v_l(s)) \leq \left|V\left(L_{n,r,1}\right)\right| - 1$. Karena label simpul ditentukan dari Pesamaan 3.1, Pesamaan 3.2, dan Pesamaan 3.3, maka label simpul $f(v_d(t))$, $f(v_{dl}(s))$, dan $f(v_l(s))$ disebut *determined label*. Jika label simpul-simpul ($f(v_d(t))$, $f(v_{dl}(s))$, $f(v_l(s))$) dan label busur-busur ($g(e_d(t))$, $g(e_{dl}(s))$ dan $g(e_l(s))$) tersedia, maka masing-masing akan dinyatakan *avail* [$f(v_d(t))$] := *true*, *avail* [$f(v_{dl}(s))$] := *true*, *avail* $f(v_l(s))$:= *true*, *avail* $g(e_d(t))$:= *true*, *avail* $g(e_{dl}(s))$:= *true*, dan *avail* $g(e_l(s))$:= *true*. Setelah label simpul-simpul ($f(v_d(t))$, $f(v_{dl}(s))$, $f(v_l(s))$) dan label busur-busur ($g(e_d(t))$, $g(e_{dl}(s))$ dan $g(e_l(s))$) digunakan, maka masing-masing akan dinyatakan *avail* [$f(v_d(t))$] := *false*, *avail* [$f(v_{dl}(s))$] := *false*, *avail* $f(v_l(s))$:= *false*, *avail* $g(e_d(t))$:= *false*, *avail* $g(e_{dl}(s))$:= *false*, dan *avail* [$g(e_l(s))$] := *false*.

Langkah pertama pada algoritma ini adalah melabel simpul $v_d(1)$, yang merupakan tahap inisialisasi (Gambar 3.23). Setelah tahap inisialisasi dilakukan, masuk ke tahap perluasan. Tahap perluasan disini terdapat dua macam, yaitu tahap perluasan ke-1 dan tahap perluasan ke-2 (Gambar 3.24 dan Gambar 3.25).

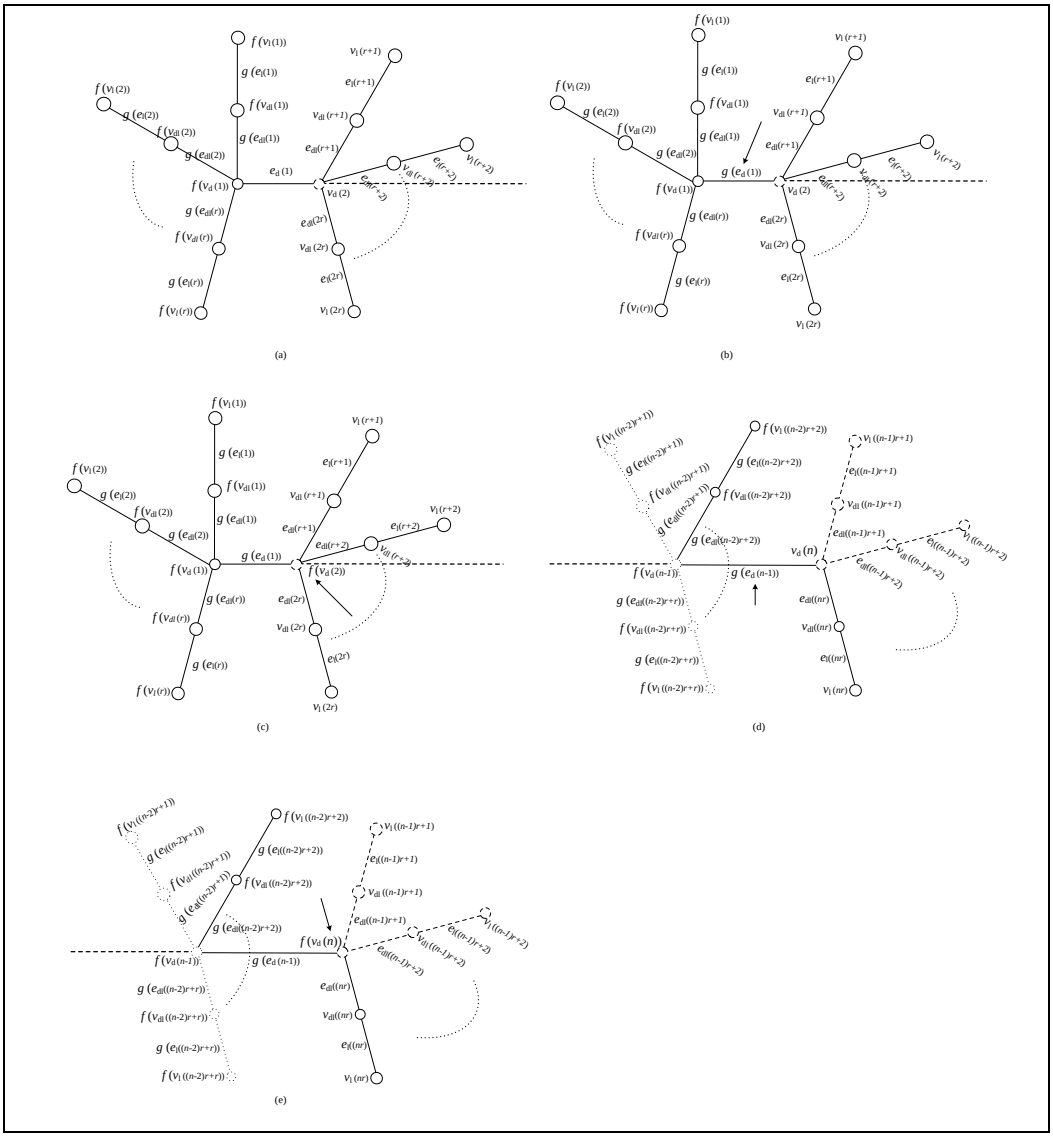


Gambar 3.23 Gambar tahap inisialisasi pada fungsi initializeLobster

Langkah pertama yang dilakukan pada tahap perluasan ke-1 adalah melabel simpul-simpul dan busur-busur pada cabang pertama yang berhubungan langsung dengan simpul $v_d(1)$ di lintasan P_n, yaitu dengan melabel busur $e_{dl}(1)$ terlebih dahulu, kemudian ditentukan label simpul $v_{dl}(1)$. Setelah melabel $v_{dl}(1)$, busur $e_l(1)$ diberikan label. Kemudian ditentukan label simpul $v_l(1)$. Satu cabang telah dilabel. Langkah tersebut dilakukan seterusnya hingga simpul-simpul dan busur-busur di cabang lainnya pada simpul $v_d(1)$ di *backbone* sudah berlabel. Kemudian masuk ke tahap perluasan ke-2, yaitu melabel busur $e_d(1)$. Setelah label busur $e_d(1)$ diberikan, ditentukan simpul $v_d(2)$. Setelah itu, masuk kembali ke tahap perluasan ke-1, yaitu melabel busur $e_{dl}(r+1)$, kemudian ditentukan label simpul $v_{dl}(r+1)$. Kemudian melabel busur $e_l(r+1)$, setelah itu ditentukan label simpul $v_l(r+1)$ dan seterusnya langkah tersebut dilakukan hingga busur $e_{dl}(2r)$, simpul $v_{dl}(2r)$, busur $e_l(2r)$ dan simpul $v_l(2r)$ dilabelkan. Kemudian masuk kembali ke tahap perluasan ke-2, yaitu melabel busur $e_d(2)$. Setelah itu, ditentukan simpul $v_d(3)$. Tahap perluasan ke-1 dan ke-2 dilakukan seterusnya, hingga langkah terakhir melabel busur $e_{dl}(nr)$, kemudian menentukan label simpul $v_{dl}(nr)$. Setelah diperoleh label simpul $v_{dl}(nr)$, untuk selanjutnya melabel busur $e_l(nr)$. Kemudian ditentukan label simpul $v_l(nr)$. Simpul $v_{dl}(nr)$, simpul $v_l(nr)$, busur $e_{dl}(nr)$, dan busur $e_l(nr)$ merupakan simpul-simpul dan busur-busur pada cabang terakhir yang berhubungan langsung dengan simpul $v_d(n)$ di lintasan P_n dengan nilai *n* dan *r* yang masing-masing diberikan.



Gambar 3.24 Gambar tahap perluasan ke-1 pada fungsi extendLobster2(*s*,*p*) dan fungsi extendLobster3(*s*,*p*)

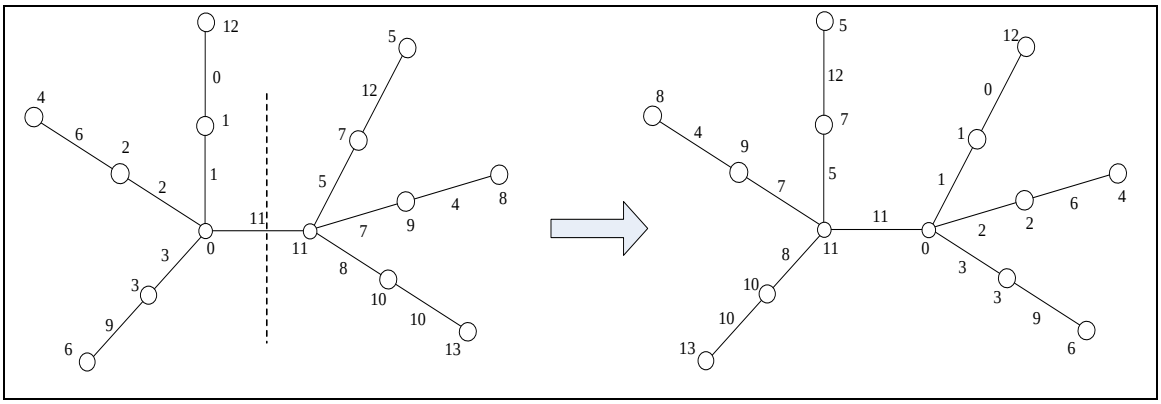


Gambar 3.25 Gambar tahap perluasan ke-2 pada fungsi extendLobster1(*t*)

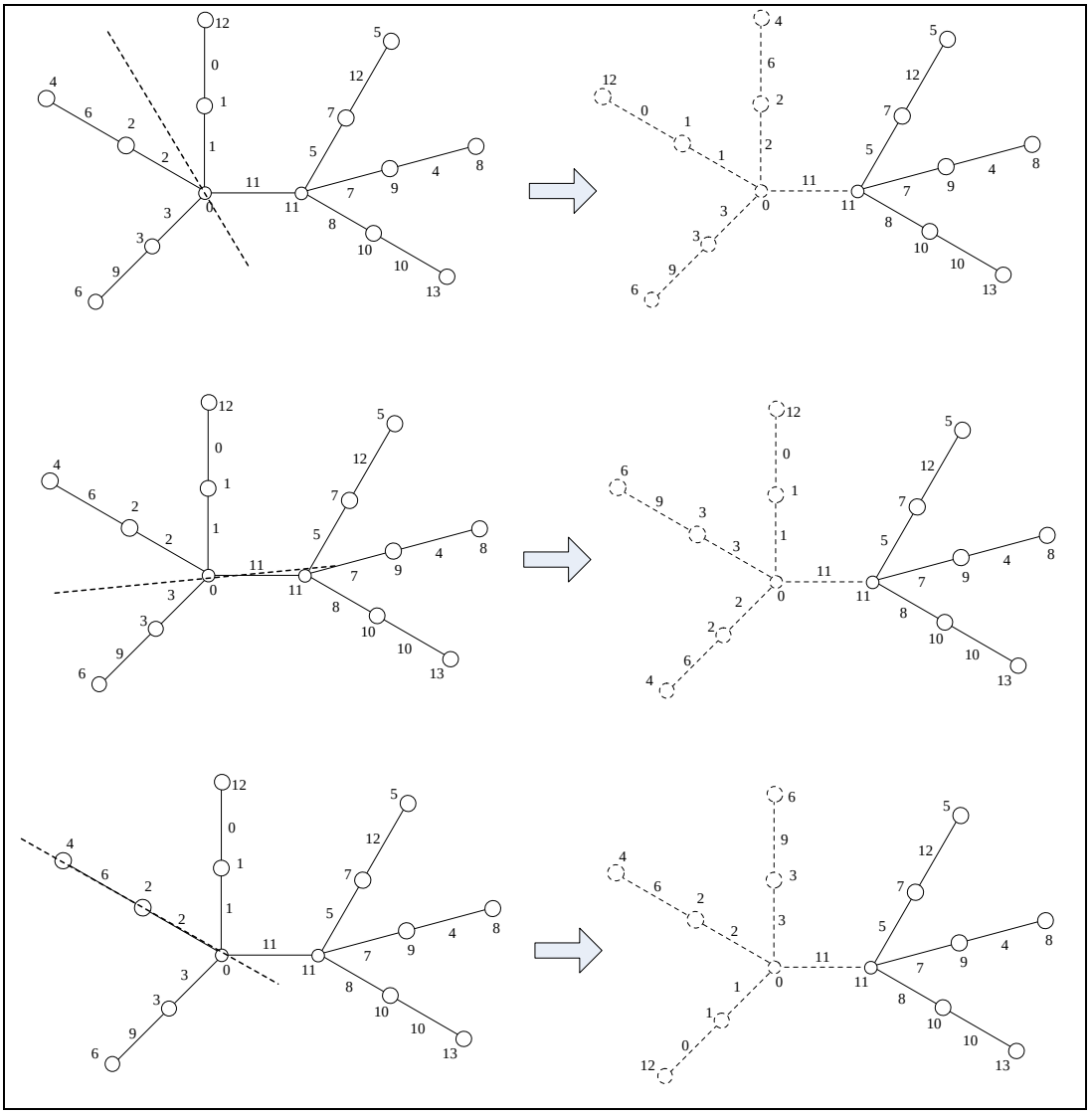
Algoritma ini terdiri dari 4 fungsi yaitu fungsi initializeLobster, extend-Lobster1(*t*), extendLobster2(*s*,*t*), dan extendLobster3(*s*,*t*). Tahap inisialisasi berada dalam fungsi initializeLobster. Tahap perluasan ke-1 berada dalam fungsi extendLobster2(*s*,*t*) (untuk kasus *r* >1), dan extendLobster3(*s*,*t*) (untuk kasus *r* =1). Sedangkan tahap perluasan ke-2 berada dalam fungsi extendLobster1(*t*), dimana parameter, *s* = 1, 2, 3, ..., *nr* dan *t* = 1, 2, 3, ..., *n*. Algoritma ini terang-kum pada Algoritma 3.1, Algoritma 3.2, Algoritma 3.3 dan Algoritma 3.4.

Algoritma ini bertujuan untuk membangun algoritma pelabelan harmonis yang menghasilkan semua pelabelan yang berbeda (tidak isomorfik) pada graf lobster teratur. Pelabelan harmonis yang berbeda dapat diperoleh dengan meng-

hindari pelabelan harmonis yang isomorfik. Pelabelan harmonis yang isomorfik pada graf lobster teratur dapat terjadi karena kasus refleksi terhadap pelabelan yang ada. Pada Gambar 3.26 dan Gambar 3.27, diberikan contoh kasus refleksi untuk $L_{2,3}$. Kasus refleksi pada graf lobster teratur yang diberikan pada Gambar 3.26, dapat dihindari dengan pemberian syarat, yaitu label simpul $f(v_d(1))$ lebih kecil atau lebih besar dari label simpul terakhir $f(v_d(n))$. Disini diambil $f(v_d(1))$ lebih kecil dari $f(v_d(n))$, diberikan pada Algoritma 3.2 baris 11. Sedangkan kasus refleksi pada graf lobster teratur yang diberikan pada Gambar 3.27, dapat dihindari dengan pemberian syarat, yaitu label simpul $f(v_d(s))$ lebih kecil atau lebih besar dari label simpul terakhir $f(v_d(s+1))$, dimana $s = 1, 2, ..., r, r+1, ..., 2r, ..., (n-1)r+1, (n-1)r+2, ..., nr$. Disini diambil $f(v_d(p))$ lebih kecil dari $f(v_d(p+1))$, diberikan pada Algoritma 3.3 baris 6, 22, dan 62. Kasus refleksi pada Gambar 3.27, hanya terjadi untuk $r > 1$. Sehingga untuk tahap perluasan ke-1 terbagi menjadi dua macam, yaitu untuk kasus $r = 1$ dan untuk kasus $r > 1$.



Gambar 3.26 Kasus refleksi pada graf lobster teratur $L_{2,3,1}$



Gambar 3.27 Kasus refleksi pada cabang graf lobster teratur $L_{2,3,1}$

Algoritma 3.1
initializeLobster
1. **for** each available label *i* **do**
2. $f\left(v_d(1)\right) = i$
3. avail [*i*] := ***false***
4. **extendLobster1(1)**
5. avail [*i*] := ***true***

end for

Algoritma 3.2
extendLobster1(*t*)

```
1.  if t = 1
2.      if r = 1
3.          extendLobster3(1, t)
4.      else
5.          extendLobster2(1, t)
        end if
6.  elseif t = n
7.      for each available label j do
8.          g(e_d(t − 1)) = j
9.          for k = 0 or k = 1 do
10.             f(v_d(t)) = [mK + g(e_d(t − 1))] − f(v_d(t − 1))
11.             if 0 ≤ f(v_d(t)) ≤ ((2nr + n) − 1) and avail [f(v_d(t))]
                and f(v_d(t)) > f(v_d(1)) then
12.                 avail [f(v_d(t))] := false
13.                 avail [ j ] := false
14.                 if r = 1
15.                     extendLobster3((t-1) r + 1, t)
16.                 else
17.                     extendLobster2((t-1) r + 1, t)
                    end if
18.                 avail [f(v_d(t))] := true
19.                 avail [ j ] := true
                    end if
            end for
        end for
20. else
21.     for each available label j do
22.         g(e_d(t − 1)) = j
23.         for k = 0 or k = 1 do
24.             f(v_d(t)) = [mK + g(e_d(t − 1))] − f(v_d(t − 1))
25.             if 0 ≤ f(v_d(t)) ≤ ((2nr + n) − 1) and avail [f(v_d(t))]
                then
26.                 avail [f(v_d(t))] := false
27.                 avail [ j ] := false
28.                 if r = 1
29.                     extendLobster3((t-1) r + 1, t)
30.                 else
31.                     extendLobster2((t-1) r + 1, t)
                    end if
32.                 avail [f(v_d(t))] := true
33.                 avail [ j ] := true
                  end if
            end for
        end for
    end if
```

Algoritma 3.3
extendLobster2(*s*,*t*)

1. **if** $s = nr$
2. **for** each available label j **do**
3. $g(e_{dl}(s)) = j$
4. **for** $k = 0$ or $k = 1$ **do**
5. $f(v_{dl}(s)) = [mK + g(e_{dl}(s))] - f(v_d(t))$
6. **if** $0 \le f(v_{dl}(s)) \le (2nr + n) - 1)$ and avail $[f(v_{dl}(s))]$ and $f(v_{dl}(s)) > f(v_{dl}(s-1))$ **then**
7. avail $[f(v_{dl}(s))]$:= ***false***
8. avail $[\ j\ ]$:= ***false***
9. **for** each available label l **do**
10. $g(e_l(s)) = l$
11. **for** $k = 0$ or $k = 1$ **do**
12. $g(f(v_l(s)) = [mK + g(e_l(s))] - f(v_{dl}(s))$
13. **if** $0 \le f(v_l(s)) \le (2nr + n) - 1)$ and avail $[f(v_l(s))]$ **then**
14. **print()**
end if
end for
end for
15. avail $[f(v_{dl}(s))]$:= ***true***
16. avail $[\ j\ ]$:= ***true***
end if
end for
end for
17. **elseif** s mod $r = 0$
18. **for** each available label j **do**
19. $g(e_{dl}(s)) = j$
20. **for** $k = 0$ or $k = 1$ **do**
21. $f(v_{dl}(s)) = [mK + g(e_{dl}(s))] - f(v_d(t))$
22. **if** $0 \le f(v_{dl}(s)) \le (2nr + n) - 1)$ and avail $[f(v_{dl}(s))]$ and $f(v_{dl}(s)) > f(v_{dl}(s-1))$ **then**
23. avail $[f(v_{dl}(s))]$:= ***false***
24. avail $[\ j\ ]$:= ***false***
25. **for** each available label l **do**
26. $g(e_l(s)) = l$
27. **for** $k = 0$ or $k = 1$ **do**
28. $g(f(v_l(s)) = [mK + g(e_l(s))] - f(v_{dl}(s))$
29. **if** $0 \le f(v_l(s)) \le (2nr + n) - 1)$ and avail $[f(v_l(s))]$ **then**
30. avail $[f(v_l(s))]$:= ***false***
31. avail $[\ l\ ]$:= ***false***
32. **extendLobster1(*t*+1)**
33. avail $[f(v_l(s))]$:= ***true***
34. avail $[\ l\ ]$:= ***true***
end if
end for
end for

```
35.                     avail [f(v_dl(s))] := true
36.                     avail [ j ] := true
                    end if
                end for
            end for
37.     elseif  s = (t-1) r + 1
38.         for  each available label  j  do
39.             g(e_dl(s)) = j
40.             for  k = 0 or k = 1 do
41.                 f(v_dl(s)) = [mK + g(e_dl(s))] − f(v_d(t))
42.                 if  0 ≤ f(v_dl(s)) ≤ (2nr + n) − 1) and avail [f(v_dl(s))] then
43.                     avail [f(v_dl(s))] := false
44.                     avail [ j ] := false
45.                     for  each available label  l  do
46.                         g(e_l(s)) = l
47.                         for  k = 0 or k = 1 do
48.                             g(f(v_l(s)) = [mK + g(e_l(s))] − f(v_dl(s))
49.                               if  0 ≤ f(v_l(s)) ≤ (2nr + n) − 1) and
                                  avail [f(v_l(s))] then
50.                                       avail [f(v_l(s))] := false
51.                                       avail [ l ] := false
52.                                       extendLobster2( s+1, t)
53.                                       avail [f(v_l(s))] := true
54.                                       avail [ l ] := true
                                  end if
                             end for
                         end for
55.                     avail [f(v_dl(s))] := true
56.                     avail [ j ] := true
                    end if
                end for
            end for
57.     else
58.         for  each available label  j  do
59.             g(e_dl(s)) = j
60.             for  k = 0 or k = 1 do
61.                 f(v_dl(s)) = [mK + g(e_dl(s))] − f(v_d(t))
62.                 if  0 ≤ f(v_dl(s)) ≤ (2nr + n) − 1) and avail [f(v_dl(s))]
                    [f(v_dl(s)) > f(v_dl(s − 1))] then
63.                     avail [f(v_dl(s))] := false
64.                     avail [ j ] := false
65.                     for  each available label  l  do
66.                         g(e_l(s)) = l
67.                         for  k = 0 or k = 1 do
68.                             g(f(v_l(s)) = [mK + g(e_l(s))] − f(v_dl(s))
69.                               if  0 ≤ f(v_l(s)) ≤ (2nr + n) − 1) and
                                  avail [f(v_l(s))] then
70.                                       avail [f(v_l(s))] := false
71.                                       avail [ l ] := false
```

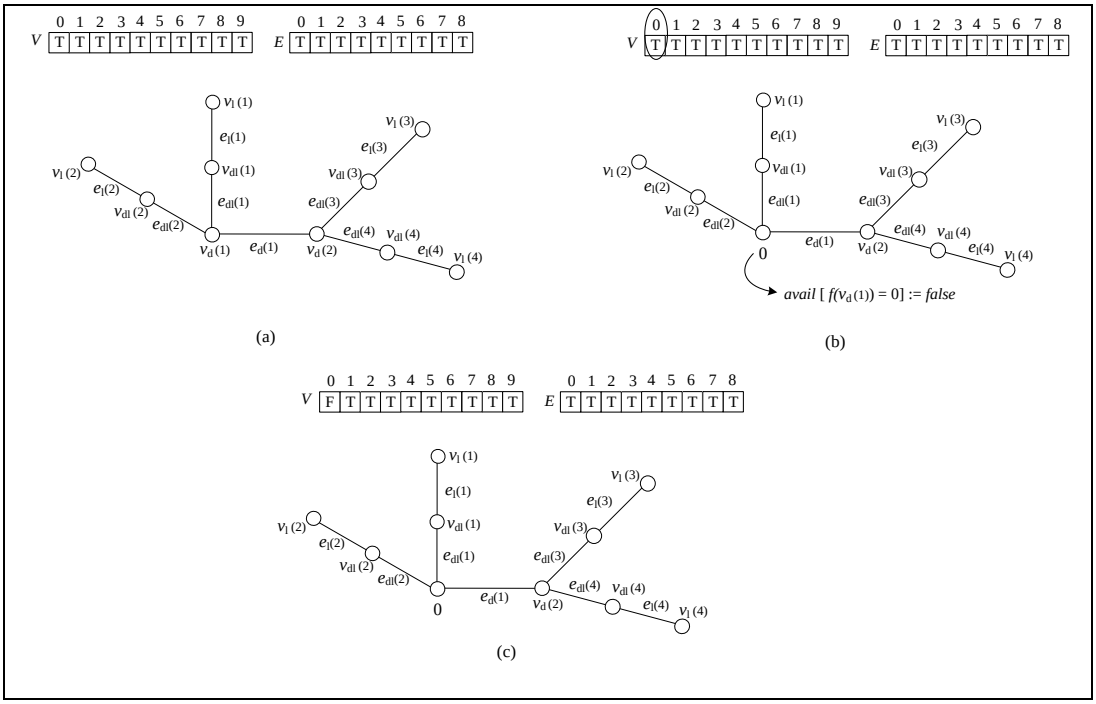
```
72.                                    extendLobster2( s+1, t)
73.                                    avail [f(v_l(s))] := true
74.                                    avail [ l ] := true
                                   end if
                               end for
                           end for
75.                        avail [f(v_dl(s))] := true
76.                        avail [ j ] := true
                       end if
                   end for
               end for
           end if
```

Algoritma 3.4
extendLobster3(*s*,*t*)

```
1.  if s = nr
2.      for each available label j do
3.          g(e_dl(s)) = j
4.          for k = 0 or k = 1 do
5.              f(v_dl(s)) = [mK + g(e_dl(s))] − f(v_d(t))
6.              if 0 ≤ f(v_dl(s)) ≤ (2nr + n) − 1) and avail [f(v_dl(s))] then
7.                  avail [f(v_dl(s))] := false
8.                  avail [ j ] := false
9.                  for each available label l do
10.                     g(e_l(s)) = l
11.                     for k = 0 or k = 1 do
12.                         g(f(v_l(s)) = [mK + g(e_l(s))] − f(v_dl(s))
13.                         if 0 ≤ f(v_l(s)) ≤ (2nr + n) − 1) and
                                avail [f(v_l(s))] then
14.                             print()
                            end if
                        end for
                    end for
15.                 avail [f(v_dl(s))] := true
16.                 avail [ j ] := true
                end if
            end for
        end for
17. else
18.     for each available label j do
19.         g(e_dl(s)) = j
20.         for k = 0 or k = 1 do
21.             f(v_dl(s)) = [mK + g(e_dl(s))] − f(v_d(t))
22.             if 0 ≤ f(v_dl(s)) ≤ (2nr + n) − 1) and avail [f(v_dl(s))] then
23.                 avail [f(v_dl(s))] := false
24.                 avail [ j ] := false
25.                 for each available label l do
26.                     g(e_l(s)) = l
27.                     for k = 0 or k = 1 do
```

```
28.                          g(f(v_l(s))) = [mK + g(e_l(s))] − f(v_dl(s))
29.                              if 0 ≤ f(v_l(s)) ≤ (2nr + n) − 1) and
                                 avail[f(v_l(s))] then
30.                                  avail [f(v_l(s))] := true
31.                                  avail [ l ] := true
32.                                  extendLobster1(t+1)
33.                                  avail [f(v_l(s))] := true
34.                                  avail [ l ] := true
                                 end if
                             end for
                         end for
35.                      avail [f(v_dl(s))] := true
36.                      avail [ j ] := true
                     end if
                 end for
             end for
         end if
```

Selanjutnya, akan diberikan contoh penggunaan algoritma ini untuk membangun pelabelan harmonis pada $L_{2,2,1}$. Banyak simpul dan busur pada $L_{2,2,1}$ adalah $|V| = 2nr+n = 10$ dan $m = |V|-1 = 9$ maka label simpul yang tersedia adalah $V = \{0,1,2,3,4,5,6,7,8,9\}$ dan label busur yang mungkin tersedia adalah $E = \{0, 1,2,3,4,5,6,7,8\}$.



Gambar 3.28 Tahap inisialisasi algoritma pelabelan harmonis graf $L_{2,2,1}$

Algoritma ini dimulai dari fungsi initializeLobster untuk melabel simpul $v_d(1)$. Misalkan $f(v_d(1)) = 0$, karena $f(v_d(1)) = 0 \leq |V| - 2 = 8$, maka avail $[f(v_d(1)) = 0] \coloneqq false,$ yang menandakan bahwa label 0 sudah digunakan dan tidak tersedia lagi. Label simpul yang tersedia menjadi *V*={1,2,3,4,5,6,7,8,9} dan label busur yang tersedia tetap (Gambar 3.28). Kemudian memanggil fungsi extendLobster1(*t*) untuk *t* = 1. Karena *t*=1, maka memanggil fungsi berikutnya yaitu fungsi extendLobster2(1,*t*), yaitu melabel busur $e_{dl}(1)$ dan label simpul $v_{dl}(1)$. Misalkan label busur $g(e_{dl}(1)) = 1$ (Gambar 3.29a). Dengan mengambil $k = 0$, $f(v_{dl}(1)) = (mk + g(e_{dl}(1))) - f(v_d(1)) = (0 + 1) - 0 = 1$. Karena label $1 \geq 0$, $1 \leq |V| - 1 = 9$, dan tersedia, maka digunakan dan dinyatakan avail $[f(v_{dl}(1)) = 1] \coloneqq$ *false* dan avail $[g(e_{dl}(1)) = 1] \coloneqq$ *false* (Gambar 3.29b). Sehingga label simpul yang tersedia menjadi *V*={2,3,4,5,6,7,8,9} dan label busur yang tersedia menjadi *E*={0,2,3,4,5,6,7,8}. Selanjutnya melabel busur $e_l(1)$. Misalkan label busur $g(e_l(1)) = 3$ (Gambar 3.29c). Kemudian ditentukan label simpul $f(v_l(1))$. Dengan mengambil $k = 0$, $f(v_l(1)) = (mk + g(e_l(1))) - f(v_{dl}(1)) = (0 + 3) - 1 = 2$. Karena $2 \geq 0$, $2 \leq |V| - 1 = 9$, dan tersedia, maka digunakan dan dinyatakan avail $[f(v_l(1)) = 2] \coloneqq$ *false* dan avail $[g(e_l(1)) = 3] \coloneqq$ *false* (Gambar 3.29d). Sehingga label simpul yang tersedia menjadi *V*={3,4,5, 6,7,8,9} dan label busur yang tersedia menjadi *E*={0,2, 4,5,6,7,8}.

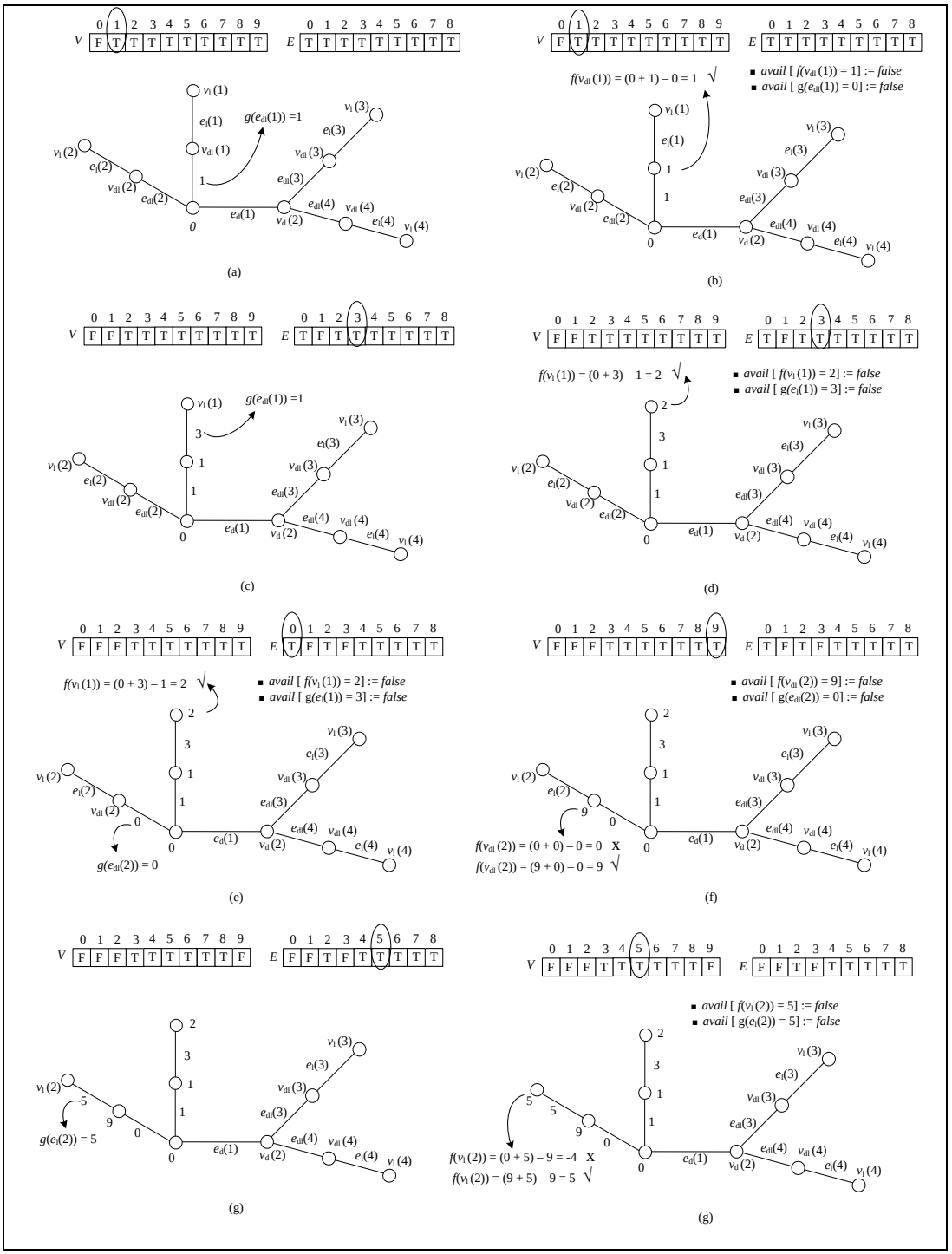
Kemudian secara rekursif memanggil fungsi fungsi extendLobster2(*s*,*t*) untuk *s*=2 dan *t*=1. Kemudian melabel busur $e_{dl}(2)$. Misalkan $g\big(e_{dl}(2)\big) = 0$ (Gambar 3.29e). Kemudian ditentukan label simpul $f\big(v_{dl}(2)\big)$. Dengan mengambil $k = 0$, $f\big(v_{dl}(2)\big) = \big(mk + g\big(e_{dl}(2)\big)\big) - f\big(v_d(1)\big) = (0 + 0) - 0 = 0$, label 0 tidak tersedia. Sehingga diambil $k = 1$, label simpul $f\big(v_{dl}(2)\big) = \big(mk + g\big(e_{dl}(2)\big)\big) - f\big(v_d(1)\big) = (9 + 0) - 0 = 9$. Karena $9 \geq 0$, $9 \leq |V| - 1 = 9$, dan tersedia, maka digunakan dan dinyatakan avail $\big[f\big(v_{dl}(2)\big) = 9\big] \coloneqq$ *false* dan avail $[g\big(e_{dl}(2)\big) = 0] \coloneqq$ *false* (Gambar 3.29f). Sehingga label simpul yang tersedia menjadi *V*={3,4,5,6,7,8} dan label busur yang tersedia menjadi *E*={2,4,5,6,7,8}. Selanjutnya melabel busur $e_l(2)$. Misalkan $g\big(e_l(2)\big) = 5$

(Gambar 3.29g). Kemudian ditentukan label simpul $f(v_l(2))$. Dengan mengambil $k = 0$, label simpul $f(v_l(2)) = (mk + g(e_l(2))) - f(v_{dl}(2)) = (0 + 5) - 9 = -4$. Karena $4 \leq 0$, maka tidak bisa label -4 untuk melabel simpul $v_l(2)$. Sehingga diambil $k = 1$, label simpul $f(v_l(2)) = (mk + g(e_l(2))) - f(v_{dl}(2)) = (9 + 5) - 9 = 5$. Karena $5 \geq 0$, $5 \leq |V| - 1 = 9$, dan tersedia, maka digunakan dan dinyatakan avail $[f(v_l(2)) = 5]$:= *false* dan avail $[g(e_l(2)) = 5]$:= *false* (Gambar 3.29h). Sehingga label simpul yang tersedia menjadi *V*={3,4, 6,7,8} dan label busur yang tersedia menjadi *E*= {2,4, 6,7,8}.

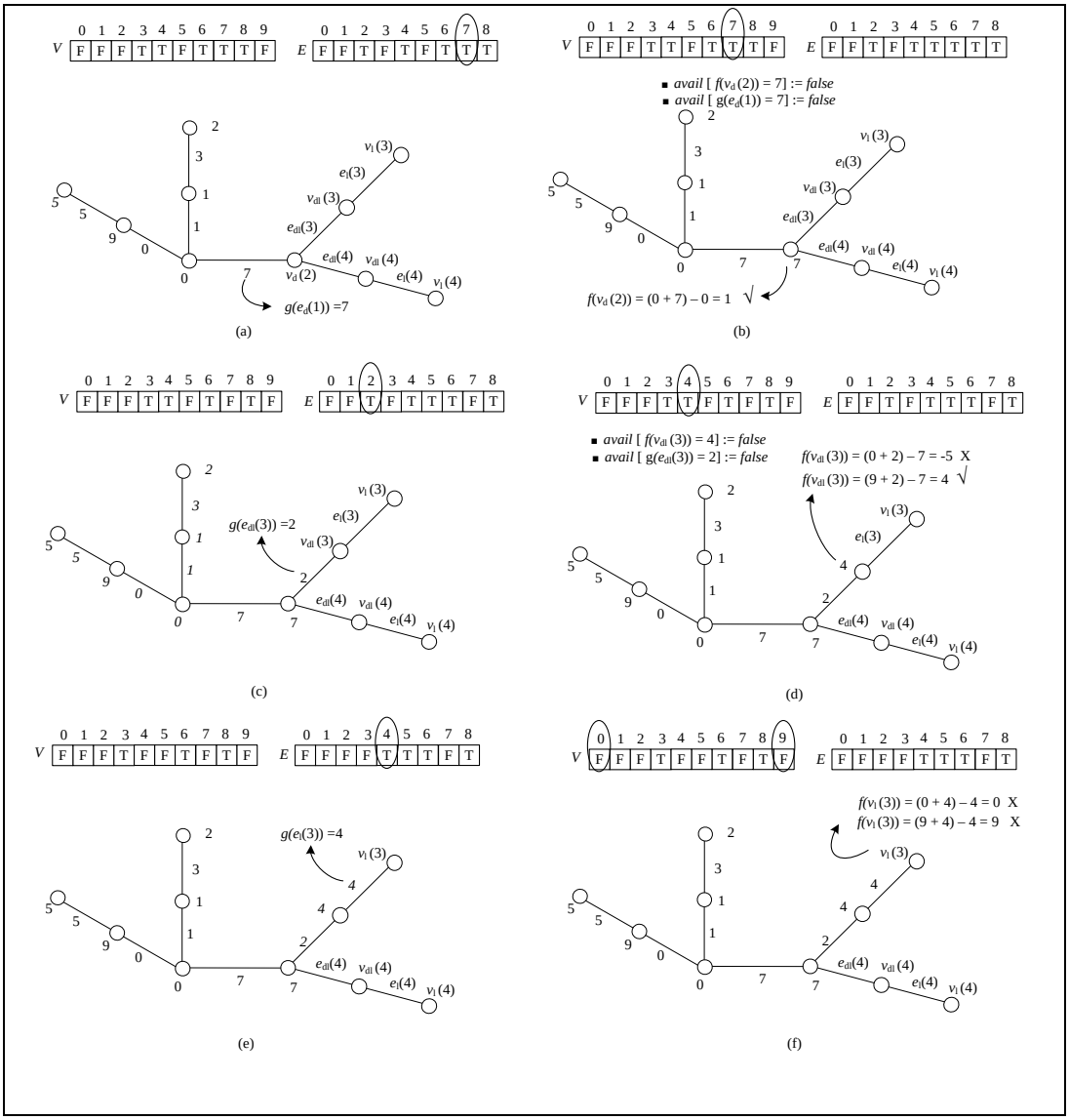
Kemudian secara rekursif memanggil fungsi extendLobster1(*t*) untuk *t*=2, yaitu melabel busur $e_d(1)$ dan simpul $v_d(2)$. Misalkan label busur $g(e_d(1)) = 7$ (Gambar 3.30a). Dengan mengambil $k = 0$, $f(v_d(2)) = (mk + g(e_d(1))) - f(v_d(1)) = (0 + 7) - 0 = 7$. Karena $7 \geq 0$, $7 \leq |V| - 1 = 9$, dan tersedia, maka digunakan dan dinyatakan avail $[f(v_d(2)) = 7]$:= *false* dan avail $[g(e_d(1)) = 7]$:= *false* (Gambar 3.30b). Sehingga label simpul yang tersedia menjadi *V*={3,4, 6, 8} dan label busur yang tersedia menjadi *E*= {2,4, 6, 8}.

Kemudian memanggil fungsi extendLobster2(*s*,*t*) untuk *s*=3 dan *t*=2. Kemudian melabel busur $e_{dl}(3)$. Misalkan $g(e_{dl}(3)) = 2$ (Gambar 3.30c). Kemudian ditentukan label simpul $f(v_{dl}(3))$. Dengan mengambil $k = 0$, $f(v_{dl}(3)) = (mk + g(e_{dl}(3))) - f(v_d(2)) = (0 + 2) - 7 = -5$. Karena -5 ≤ 0, maka -5 tidak bisa untuk melabel simpul $v_{dl}(3)$. Sehingga diambil $k = 1$, $f(v_{dl}(3)) = (mk + g(e_{dl}(3))) - f(v_d(2)) = (9 + 2) - 7 = 4$. Karena $4 \geq 0$, $4 \leq |V| - 1 = 9$, dan tersedia, maka digunakan dan dinyatakan avail $[f(v_{dl}(3)) = 4]$:=*false* dan avail $[g(e_{dl}(3))=2]$:= *false* (Gambar 3.30d). Sehingga label simpul yang tersedia menjadi *V*={3, 6, 8} dan label busur yang tersedia menjadi *E*= {4, 6, 8}. Selanjutnya melabel busur $e_l(3)$. Misalkan label busur $g(e_l(3)) = 4$ (Gambar 3.30e). Kemudian ditentukan label simpul $v_l(3)$. Dengan mengambil $k = 0$, $f(v_l(3)) = (mk + g(e_l(3))) - f(v_{dl}(3)) = (0 + 4) - 4 = 0$, label 0 tidak tersedia. Sehingga diambil $k = 1$, label simpul $f(v_l(3)) = (mk + g(e_l(3))) - f(v_{dl}(3)) = (9 + 4) - 4 = 9$, label 9 tidak tersedia (Gambar 3.30f). Sehingga dilakukan kembali langkah tersebut pada label busur yang tersedia, namun tidak

terdapat label busur yang memenuhi. Sehingga perlu dilakukan *backtracking* ke langkah sebelumnya, yaitu mengubah label simpul $f(v_{dl}(3))$ dan label busur $g(e_{dl}(3))$, sehingga dibuat tersedia kembali (avail $[f(v_{dl}(3))] = 4]$:=*true* dan avail $[g(e_{dl}(3)) = 2] :=$ *true*) (Gambar 3.31b).



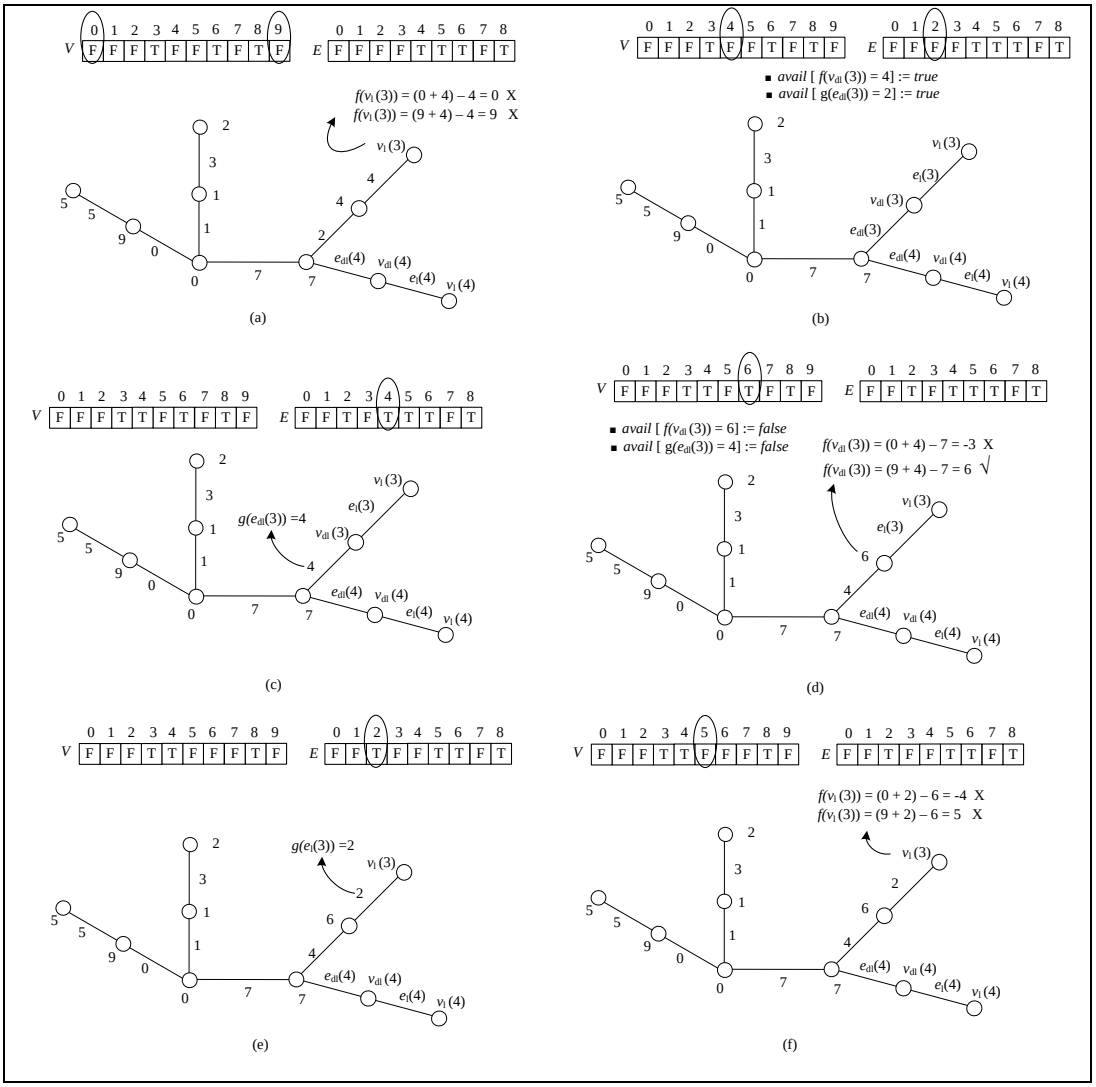
Gambar 3.29 Tahap perluasan algoritma pelabelan harmonis untuk *s*=1 dan *s*=2



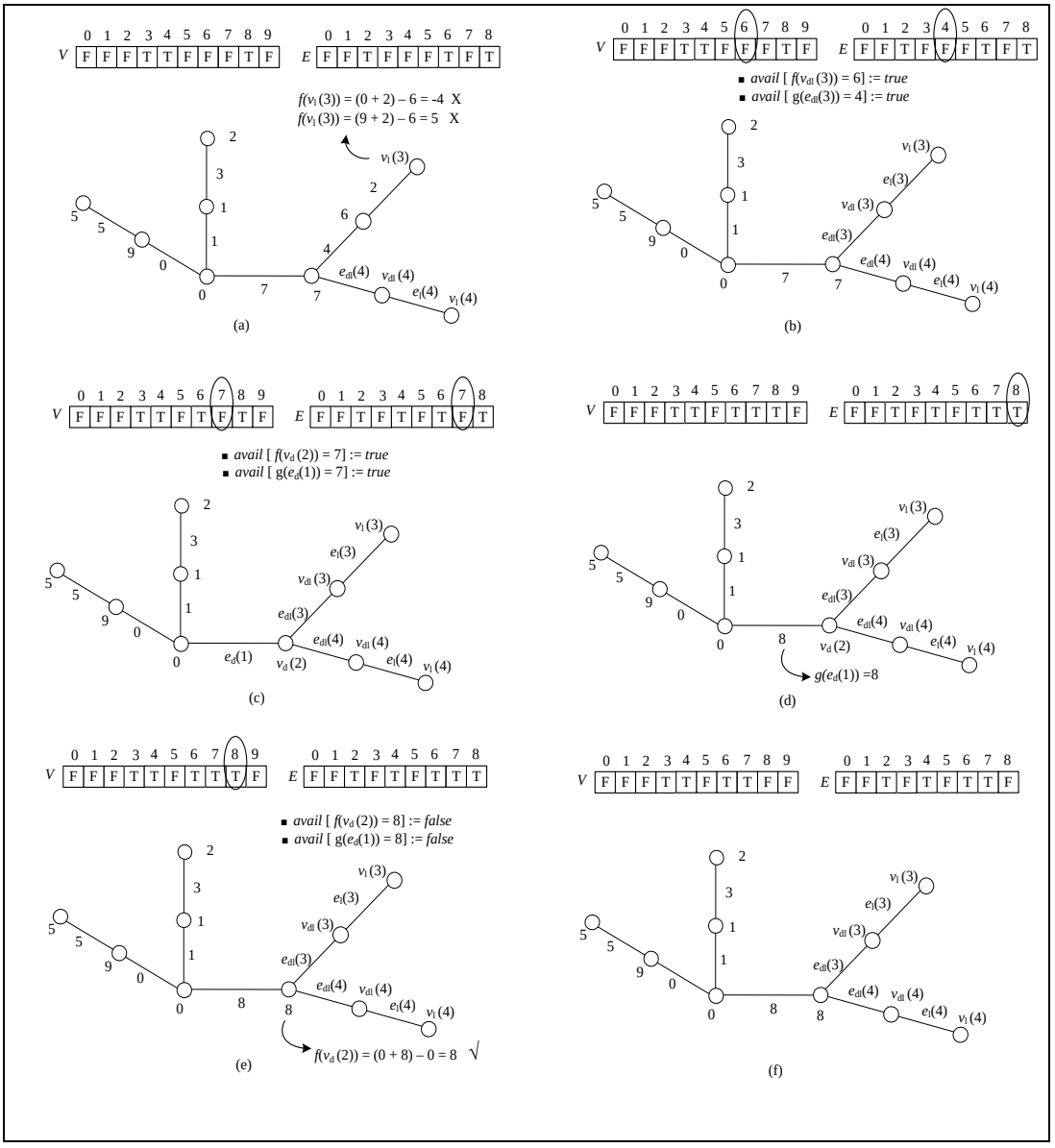
Gambar 3.30 Tahap perluasan algoritma pelabelan harmonis untuk *s*=3

Kemudian melabel kembali busur $e_{dl}(3)$. Misalkan $g(e_{dl}(3)) = 4$ (Gambar 3.31c). Kemudian ditentukan label simpul $v_{dl}(3)$. Dengan mengambil $k = 0$, $f(v_{dl}(3)) = \left(mk + g\left(e_{dl}(3)\right)\right) - f\left(v_d(2)\right) = (0 + 4) - 7 = -3$. Karena label -3≤ 0, maka -3 tidak bisa untuk melabel simpul $v_{dl}(3)$. Sehingga diambil $k = 1$, $f(v_{dl}(3)) = \left(mk + g\left(e_{dl}(3)\right)\right) - f\left(v_d(2)\right) = (9 + 4) - 7 = 6$. Karena $6 \geq 0$, $6 \leq |V| - 1 = 9$, dan tersedia, maka digunakan dan dinyatakan $avail\ [f(v_{dl}(3)) = 6] := false$ dan avail $[g(e_{dl}(3)) = 4] :=\ false$ (Gambar 3.31d).

Selanjutnya melabel busur $e_l(3)$. Misalkan label busur $g(e_l(3)) = 2$ (Gambar 3.31e). Kemudian ditentukan label simpul $v_l(3)$. Dengan mengambil $k = 0$, $f(v_l(3)) = \left(mk + g\left(e_l(3)\right)\right) - f\left(v_{dl}(3)\right) = (0 + 2) - 6 = -4$. Karena label -4≤ 0, maka -4 tidak bisa untuk melabel simpul $v_l(3)$. Sehingga diambil $k = 1$, label simpul $f(v_l(3)) = \left(mk + g\left(e_l(3)\right)\right) - f\left(v_{dl}(3)\right) = (9 + 2) - 6 = 5$, label 5 tidak tersedia (Gambar 3.31f). Sehingga perlu dilakukan *backtracking* ke langkah sebelumnya.



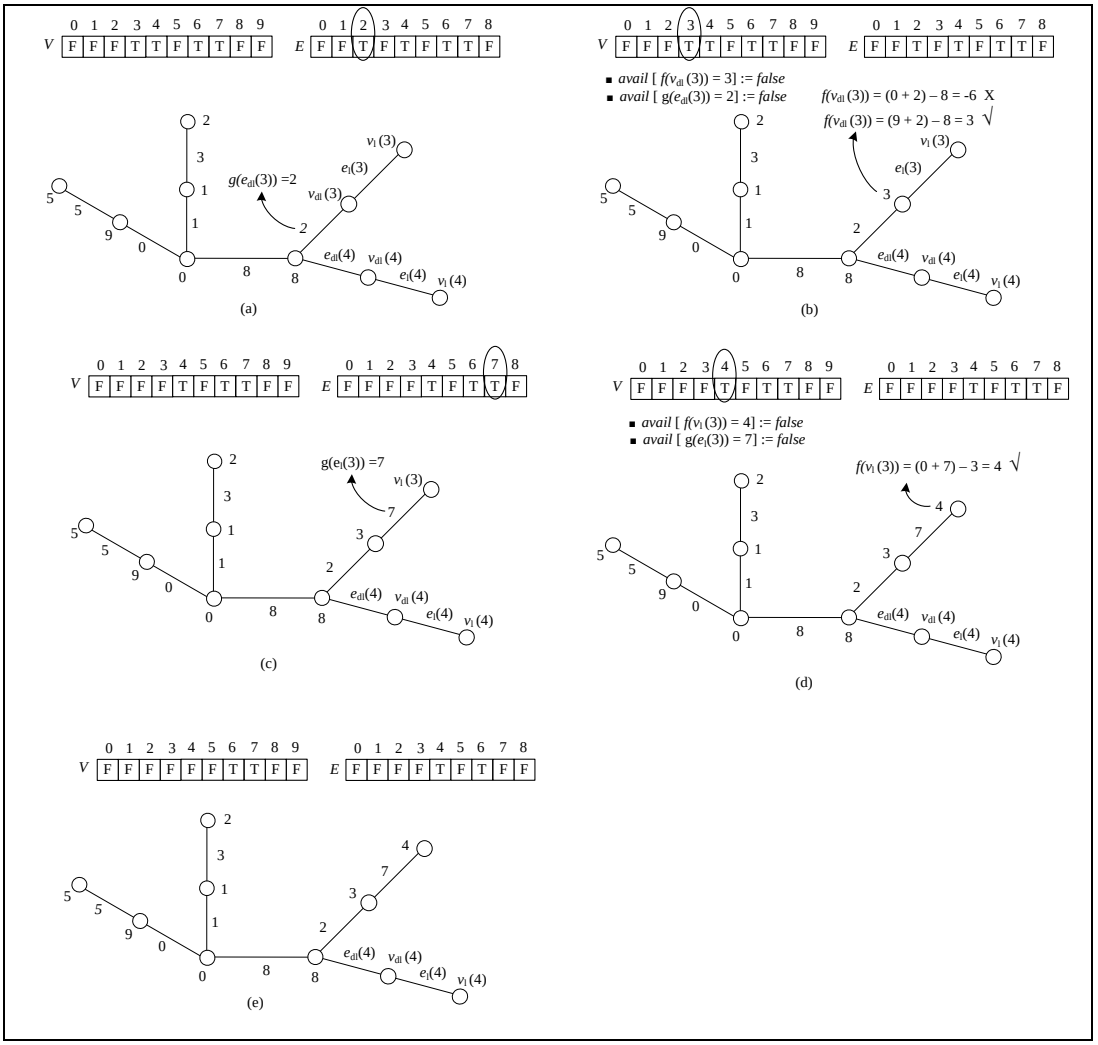
Gambar 3.31 *Backtracking* ke simpul $v_{dl}(3)$



Gambar 3.32 *Backtracking* ke simpul $v_{dl}(3)$, kemudian ke simpul $v_d(2)$

Label simpul $f(v_{dl}(3))$ dan label busur $g(e_{dl}(3))$ diubah dan dibuat tersedia kembali, yaitu avail $[f(v_{dl}(3)) = 6] :=$ *true* dan avail $[g(e_{dl}(3)) = 4] :=$ *true* (Gambar 3.32b). Kemudian dilakukan kembali pemberian label pada simpul $v_{dl}(3)$ dan label busur $e_{dl}(3)$, tetapi pada langkah selanjutnya terjadi beberapa kali *backtracking*. Sehingga label simpul $f(v_d(2))$ dan label busur $g(e_d(1))$ harus diubah dan dibuat tersedia kembali, yaitu avail $[f(v_d(2)) = 7] :=$ *true* dan avail $[g(e_d(1)) = 7] :=$ *true* (Gambar 3.32c). Sehingga label simpul yang tersedia menjadi *V*={3,4, 6,7, 8} dan label busur yang tersedia menjadi *E*= {2,4, 6, 7,8}.

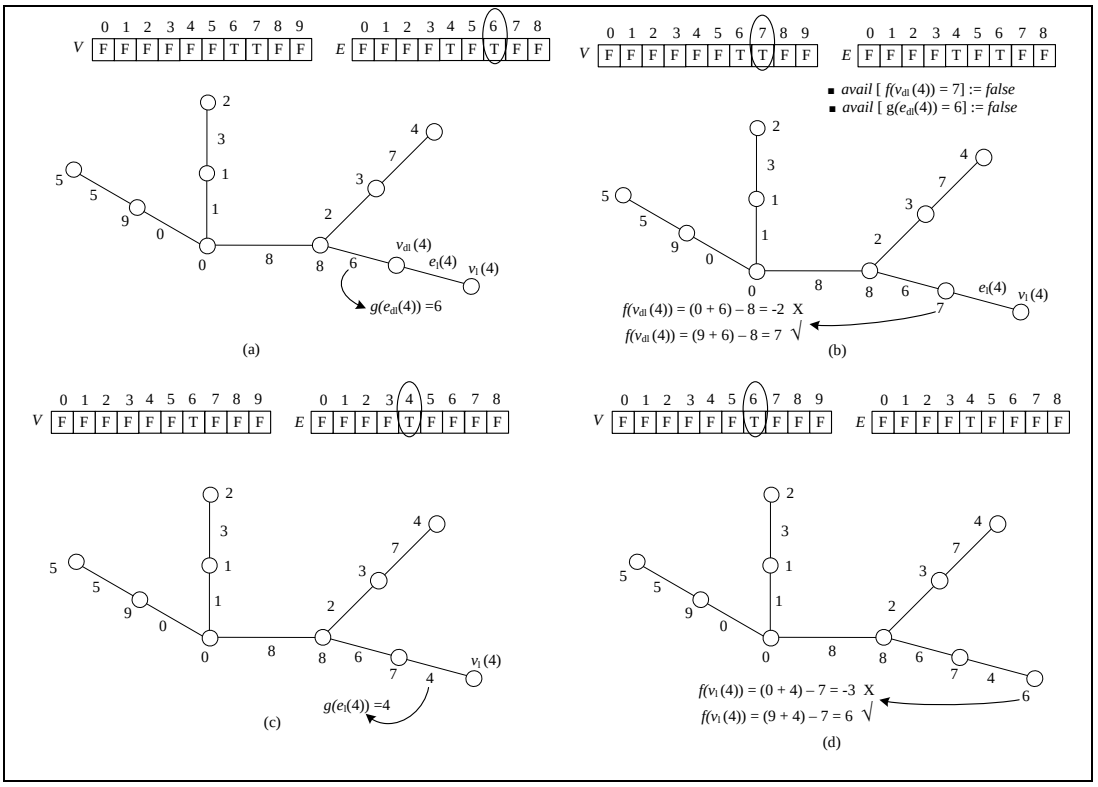
Kemudian diberi label kembali busur $e_d(1)$. Misalkan $g(e_d(1))$ = 8 (Gambar 3.32d). Setelah itu, ditentukan label simpul $v_d(2)$. Dengan mengambil $k = 0$, label simpul $f(v_d(2)) = (mk + g(e_d(1))) - f(v_d(1)) = (0 + 8) - 0 = 8$. $f(v_d(2)) = 8 \geq 0$, $f(v_d(2)) = 8 \leq |V| - 1 = 9$, dan tersedia, maka diguna-kan dan dinyatakan avail $[f(v_d(2)) = 8] :=$ *false* dan avail $[g(e_d(1)) = 8] :=$ *false* (Gambar 3.32e). Sehingga label simpul yang tersedia menjadi *V*={3,4, 6, 7} dan label busur yang tersedia menjadi *E*= {2,4, 6, 7}.



Gambar 3.33 Tahap perluasan algoritma pelabelan harmonis untuk *s*=3

Kemudian memanggil fungsi extendLobster2(*s*,*t*) untuk *s*=3 dan *t*=2. Kemudian melabel busur $e_{dl}(3)$. Misalkan label busur $g(e_{dl}(3)) = 2$ (Gambar 3.33a). Kemudian ditentukan label simpul $v_{dl}(3)$. Dengan mengambil $k = 0$,

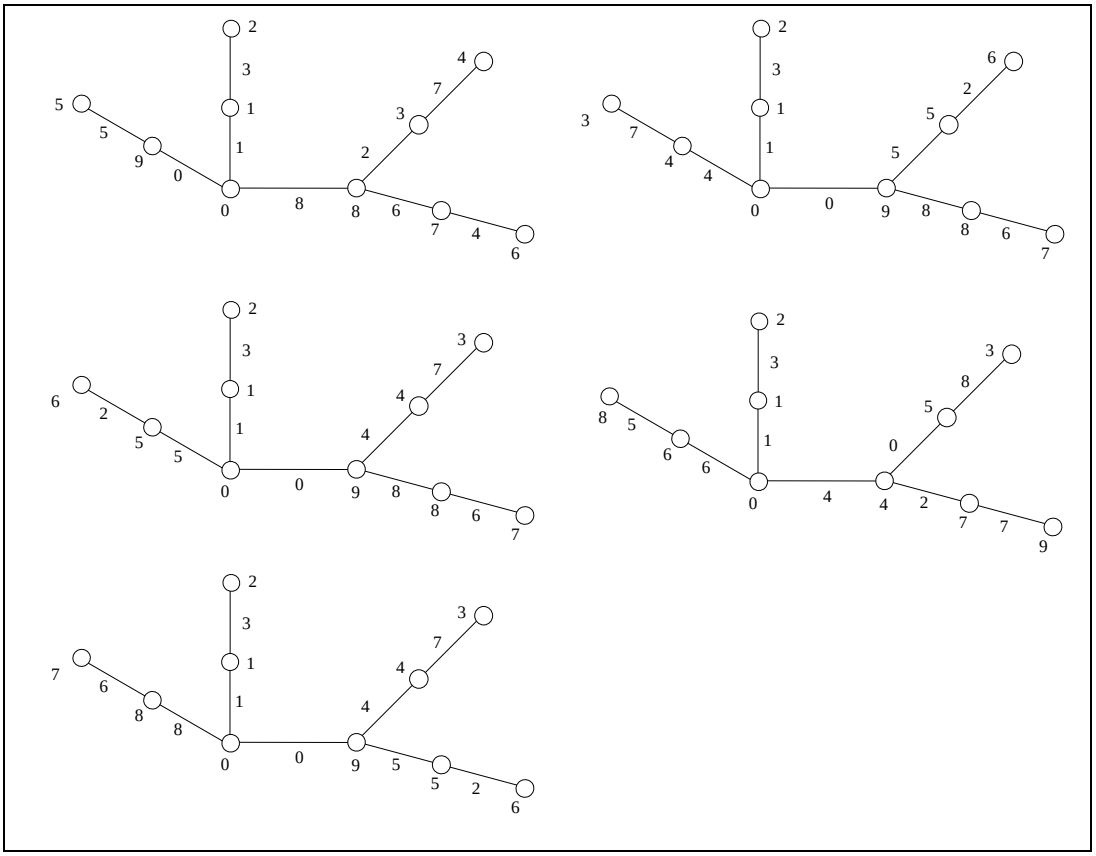
$f(v_{dl}(3)) = \left(mk + \ g\left(e_{dl}(3)\right)\right) - f\left(v_d(2)\right) = (0 + 2) - 8 = -6$. Karena -6 ≤ 0, maka -6 tidak bisa untuk melabel simpul $v_{dl}(3)$. Sehingga diambil $k = 1$, $f(v_{dl}(3)) = \left(mk + \ g\left(e_{dl}(3)\right)\right) - f\left(v_d(2)\right) = (9 + 2) - 8 = 3$. Karena label $3 \geq 0$, $3 \leq |V| - 1 = 9$, dan tersedia, maka digunakan dan dinyatakan avail $[f(v_{dl}(3)) = 3] :=$ *false* dan avail $[g(e_{dl}(3)) = 2] :=$ *false* (Gambar 3.33b). Sehingga label simpul yang tersedia menjadi *V*={4,6,7} dan label busur yang tersedia menjadi *E*= {4, 6, 7}. Selanjutnya melabel busur $e_l(3)$. Misalkan label busur $g(e_l(3)) = 7$ (Gambar 3.33c). Kemudian ditentukan label simpul $v_l(3)$. Dengan mengambil $k = 0, f(v_l(3)) = \left(mk + \ g\left(e_l(3)\right)\right) - f\left(v_{dl}(3)\right) = (0 + 7) - 3 = 4$. Karena $4 \geq 0, 4 \leq |V| - 1 = 9,$ dan tersedia, maka digunakan dan dinyatakan *avail* $[f(v_l(3)) = 4] :=$ *false* dan avail $[g(e_l(3)) = 7] :=$ *false* (Gambar 3.33d).



Gambar 3.34 Tahap perluasan algoritma pelabelan harmonis untuk *s*=4

Secara rekursif, dipanggil fungsi extendLobster2(*s*,*t*) untuk *s*=4 dan *t*=2. Kemudian melabel busur $e_{dl}(4)$. Misalkan $g(e_{dl}(4)) = 6$ (Gambar 3.34a). Kemudian ditentukan label simpul $v_{dl}(4)$. Dengan mengambil $k = 0$, $f(v_{dl}(4)) = (mk + g(e_{dl}(4))) - f(v_d(2)) = (0 + 6) - 8 = -2$. Karena -2≤ 0, maka -2 tidak bisa untuk melabel simpul $v_{dl}(4)$. Sehingga diambil $k = 1$, $f(v_{dl}(4)) = \left(mk + g(e_{dl}(4))\right) - f(v_d(2)) = (9 + 6) - 8 = 7$. Karena $7 \geq 0$, $7 \leq |V| - 1 = 9$, dan tersedia, maka digunakan dan dinyatakan avail $[f(v_{dl}(4)) = 7] :=$ *false* dan avail $[g(e_{dl}(4)) = 6] :=$ *false* (Gambar 3.34b). Sehingga label simpul yang tersedia menjadi *V*={6} dan label busur yang tersedia menjadi *E*= {4}. Selanjutnya melabel busur $e_l(4)$. Karena label busur yang tersedia adalah 4, diberikan label busur $g(e_l(4)) = 4$ (Gambar 3.34c). Kemudian ditentukan label simpul $v_l(4)$. Dengan mengambil $k = 0$, $f(v_l(4)) = \left(mk + g(e_l(4))\right) - f(v_{dl}(4)) = (0 + 4) - 7 = -3$. Karena label -3≤ 0, maka -3 tidak bisa untuk melabel simpul $v_l(4)$. Sehingga diambil $k = 1$, label simpul $f(v_l(4)) = \left(mk + g(e_l(4))\right) - f(v_{dl}(4)) = (9 + 4) - 7 = 6$. Karena $6 \geq 0$, $6 \leq |V| - 1 = 9$, dan tersedia, maka label simpul $f(v_l(4)) = 6$. Sehingga terbentuklah satu pelabelan harmonis pada graf lobster teratur $L_{2,2,1}$, dimana $V_d = \{0,8\}$, $V_{dl} = \{1,9,3,7\}$, $V_l = \{2,5,4,6\}$, $E_d = \{8\}$, $E_{dl} = \{1,0,2,6\}$, dan $E_l = \{3,5,7,4\}$ (Gambar 3.34d). Kemudian memanggil fungsi print() untuk mencetak elemen pelabelan harmonis tersebut.

Untuk memperoleh pelabelan harmonis lainnya pada graf lobster $L_{2,2,1}$ maka dilakukan *backtracking* ke tahap sebelumnya dan seterusnya sesuai dengan urutan langkah-langkah sebelumnya. Algoritma ini akan berhenti sampai diperoleh semua pelabelan harmonis yang berbeda pada graf lobster teratur $L_{2,2,1}$. Semua pelabelan harmonis yang berbeda pada graf lobster teratur $L_{2,2,1}$ diperoleh 624. Pada Gambar 3.35, hanya ditunjukkan 5 pelabelan harmonis yang mungkin dan berbeda pada graf lobster teratur $L_{2,2,1}$.



Gambar 3.35 Lima pelabelan harmonis yang mungkin dan berbeda pada graf lobster teratur $L_{2,2,1}$

Pada bab selanjutnya, akan diberikan implementasi dan simulasi pada graf lintasan, lingkaran, dan lobster teratur.

BAB 4
IMPLEMENTASI DAN SIMULASI

Pada Bab 3 telah dijelaskan mengenai pembentukan algoritma pelabelan harmonis pada graf lintasan, graf lingkaran, dan graf lobster teratur. Selanjutnya pada Bab ini akan diberikan implementasi dan simulasi dari algoritma-algoritma tersebut dalam bentuk program. Program tersebut akan menggunakan MATLAB yang dijalankan pada *PC* dengan *processor Intel Core i3 @* 2.27 GHz, memori 2.00 GB, dan sistem operasi *Microsoft Windows 7 Home Basic*. Masukan dari program tersebut adalah *n* (banyak simpul) untuk graf lintasan dan graf lingakaran. Sedangkan masukan pada graf tersebut untuk graf lobster teratur adalah *n* (banyak simpul pada *backbone* lobster, P_n) dan *r* (banyak simpul daun dari simpul-simpul pada *backbone* caterpillar teratur). Program tersebut akan menghasilkan pelabelan harmonis yang mungkin dan tidak isomorfik pada graf terkait. Secara umum, simulasi ini dapat dijalankan untuk sembarang nilai *n* dan *r* (untuk graf lobster teratur), tetapi karena keterbatasan waktu simulasi baru dapat dilakukan sampai nilai *n* tertentu saja.

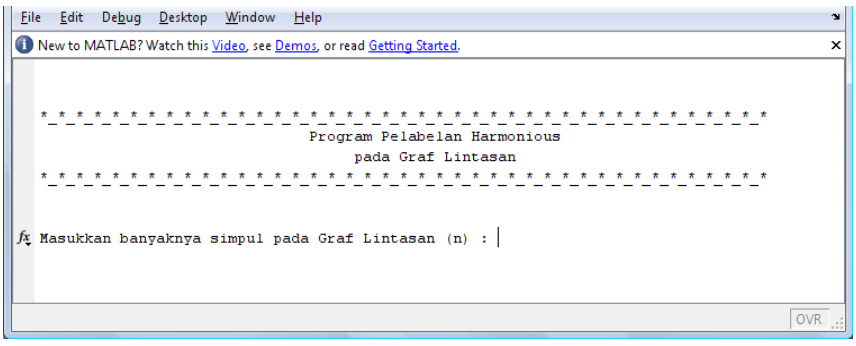
Pada Subbab 4.1, Subbab 4.2, dan Subbab 4.3, masing-masing diberikan implementasi dan simulasi algoritma pelabelan harmonis pada graf lintasan, graf lingkaran, dan graf lobster teratur.

4.1 Implementasi dan Simulasi Algoritma Pelabelan Harmonis pada Graf Lintasan

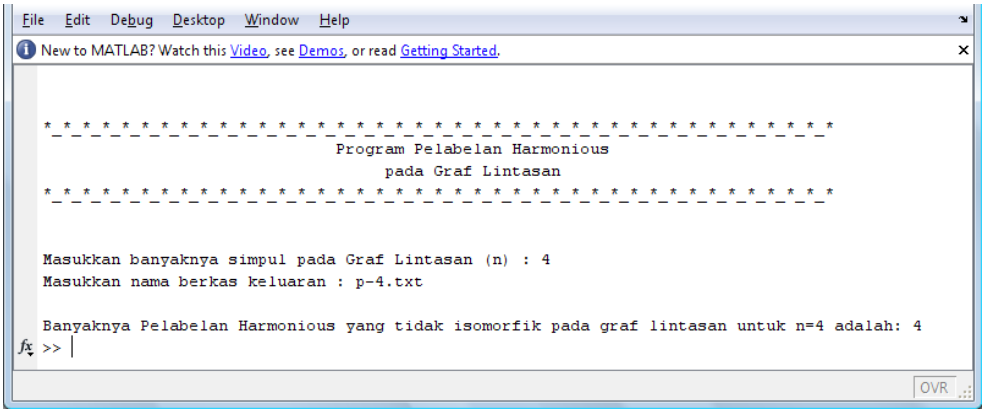
Seperti yang telah dijelaskan pada Subbab.3.1, algoritma pelabelan harmonis pada graf lintasan P_n terdiri dari dua fungsi yaitu fungsi initializePath (Algoritma 1.1) dan fungsi extendPath(*t*) (Algoritma 1.2). Dalam implementasi-nya, program untuk mecari pelabelan harmonis graf lintasan juga terdiri dari dua fungsi, yaitu pathH dan extendPathH. Fungsi pathH akan melakukan tahap-tahap pada Algoritma 1.1. Kemudian fungsi pathH akan memanggil fungsi extendpathH yang akan melakukan tahap-tahap pada Algoritma 1.2. *Array* label simpul pada program ini dinyatakan dengan *S*(*i*) dan *array* label busur dinyatakan dengan

$B(j)$. *Array* label simpul dan *array* label busur di algoritma masing-masing merupakan suatu *array* berukuran $|V|$ dan $|E|$ yang berisikan nilai *true* atau *false*, tetapi tidak demikian pada program. Pada program ini, *array* label simpul dan *array* label busur masing-masing merupakan suatu *array* berukuran $|V|$ dan $|E|$ yang berisi label-label untuk melabel elemen pada graf, yaitu $S(i) = i$ dan $B(j) = j$, dimana $i = 0, 1, 2, \ldots, |V|-1$ dan $j = 0, 1, 2, \ldots, |E|-1$. Setiap label-label ini digunakan tepat satu kali. Jika label simpul i (label busur j) sudah digunakan, maka tandai $S(i) = n$ dan $B(j) = n$. Jika ternyata label yang digunakan bukan merupakan label yang tepat untuk melabel elemen pada graf, maka label simpul i dan label busur j dibuat tersedia kembali dengan ditandai $S(i) = i$ dan $B(j) = j$, yang berarti label simpul i (label busur j) tersedia kembali. Masukan dari program ini adalah n (banyak simpul di graf lintasan P_n). Serta keluaran dari program ini adalah banyak pelabelan yang mungkin dan tidak isomorfik pada graf lintasan, P_n. *Listing* program ini dilampirkan pada Lampiran 1 (*CD*).

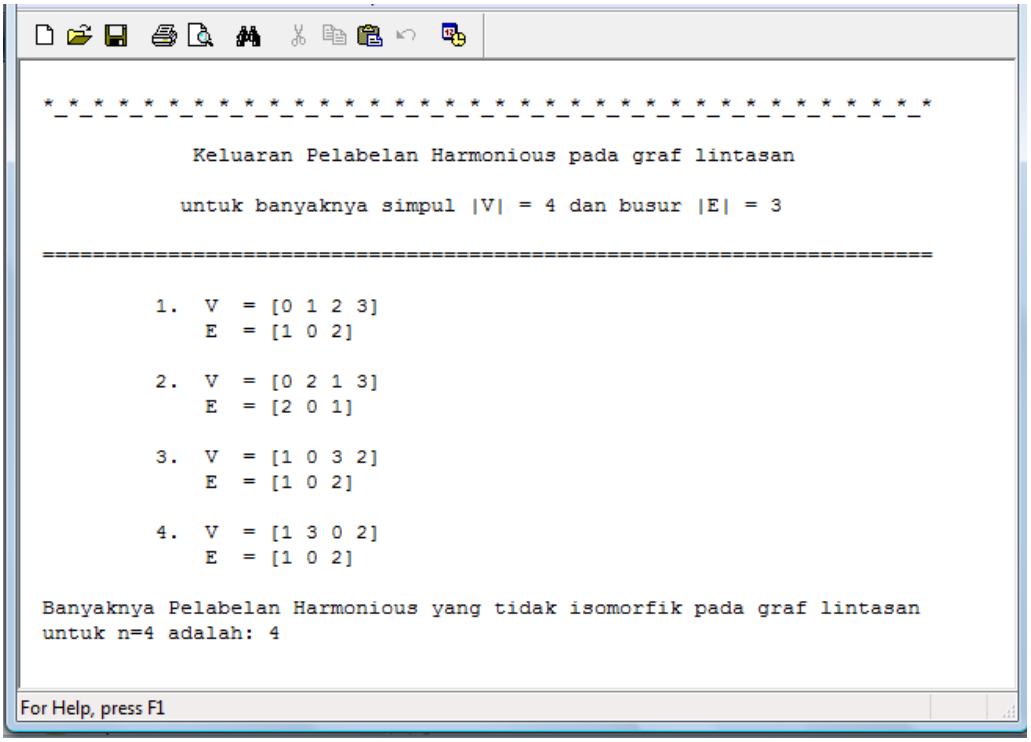
Pemanggilan program dilakukan dengan cara mengetik "pathH" pada *Command Window*. Pengguna akan diminta memasukkan nilai n (banyak simpul) dan nama berkas keluaran (*file output*). Berkas keluaran digunakan untuk menyimpan keluaran dari program yang berupa label simpul dan label busur beserta banyak pelabelan harmonis yang mungkin dan berbeda pada graf lintasan dengan nilai n yang diberikan oleh pengguna.



Gambar 4.1 Tampilan awal pada *Command Window*



Gambar 4.2 Tampilan keluaran pada *Command Window*



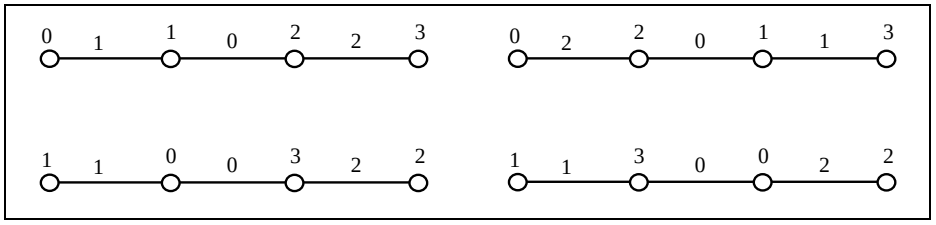
Gambar 4.3 Tampilan keluaran pada berkas

Selanjutnya, akan diberikan contoh penggunaan program ini pada pelabelan harmonis graf lintasan P_4. Langkah pertama menjalankan program yaitu ketik "pathH" di *command window*, kemudian muncul tampilan seperti pada Gambar 4.1. Kemudian pengguna harus memasukkan banyak simpul dan nama berkas keluarannya. Karena graf lintasan P_4, maka $n = 4$. Nama file sebagai berkas keluarannya adalah p-4.txt. Serta, hasilnya yaitu terdapat 4 pelabelan

harmonis yang mungkin dan tidak isomorfik pada graf lintasan (Gambar 4.2). Isi dari file 'p-4.txt' akan diperlihatkan pada Gambar 4.3.

Pada berkas keluaran, terdapat 4 pelabelan harmonis yang mungkin dan tidak isomorfik untuk P_4. Pelabelan yang pertama pada Gambar 4.3 diperoleh V = [0 1 2 3] menunjukkan label simpul v_1, v_2, v_3, v_4 secara berurutan, E = [1 0 2] menunjukkan label busur e_1, e_2, e_3 secara berurutan. Pelabelan harmonis yang diperoleh ini sama dengan yang diperoleh pada contoh di Subbab 3.1.

Semua pelabelan yang telah terbentuk pada Gambar 4.3, jika digambar-kan, akan tampak seperti pada Gambar 4.4.



Gambar 4.4 Representasi visual keluaran pelabelan harmonis pada graf P_4

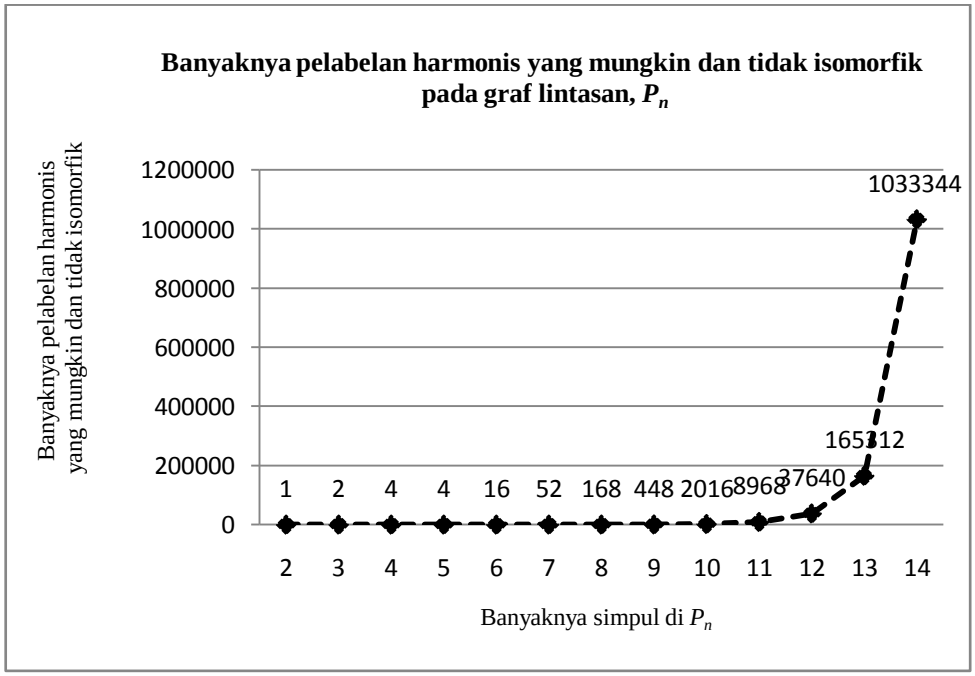
Simulasi dari program pelabelan harmonis pada graf lintasan P_n dijalankan untuk mengetahui berapa banyak pelabelan harmonis yang mungkin dan tidak isomorfik untuk nilai *n* yang diberikan. Program ini bisa disimulasikan untuk sembarang *n*, namun dikarenakan keterbatasan waktu, simulasi hanya dijalankan sampai *n* = 14. Hasil simulasi pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lintasan P_n, diberikan pada Tabel 4.1.

Tabel 4.1 Banyak pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lintasan P_n, untuk n = 2 s.d. n = 14

Graf Lintasan, P_n	Banyak pelabelan harmonis yang mungkin dan tidak isomorfik
P_2	1
P_3	2
P_4	4
P_5	4
P_6	16
P_7	52

P_8	168
P_9	448
P_{10}	2016
P_{11}	8968
P_{12}	37640
P_{13}	165312
P_{14}	1033344

Pada Tabel 4.1 terlihat bahwa untuk setiap n=2 s.d. n=14 selalu dapat ditemukan pelabelan harmonis untuk graf lintasan P_n. Banyak pelabelan harmonis yang terbentuk semakin bertambah seiring bertambahnya nilai n, kecuali saat n=4 dan n=5 memiliki jumlah pelabelan harmonis yang terbentuk adalah sama, yaitu 4.



Gambar 4.5 Grafik pertambahan pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lintasan P_n untuk n = 2 s.d. n = 14

Pada Gambar 4.5 ditunjukkan hubungan nilai n (banyak simpul) dengan banyak pelabelan harmonis yang mungkin dan tidak isomorfik. Terlihat bahwa semakin bertambahnya nilai n (banyak simpul), maka banyak pelabelan harmonis

yang tidak isomorfik juga bertambah dengan pertambahan yang cenderung eksponensial.

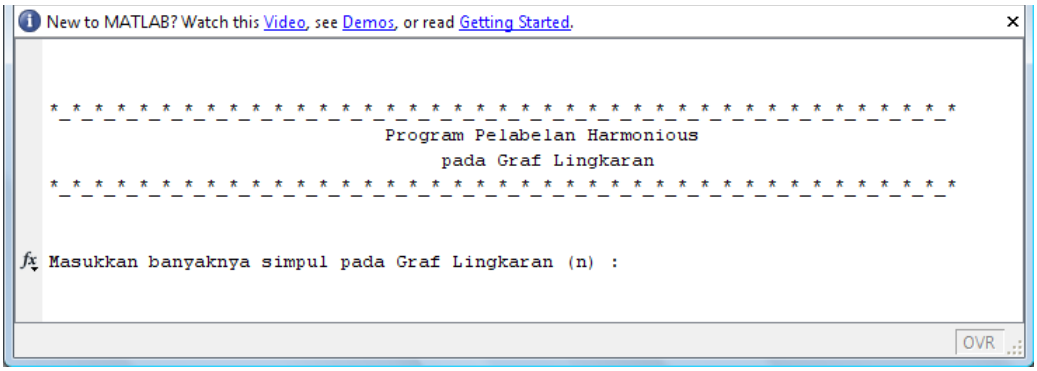
Pada subbab selanjutnya akan diberikan implementasi dan simulasi algoritma pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lingkaran C_n.

4.2 Implementasi dan Simulasi Algoritma Pelabelan Harmonis pada Graf Lingkaran

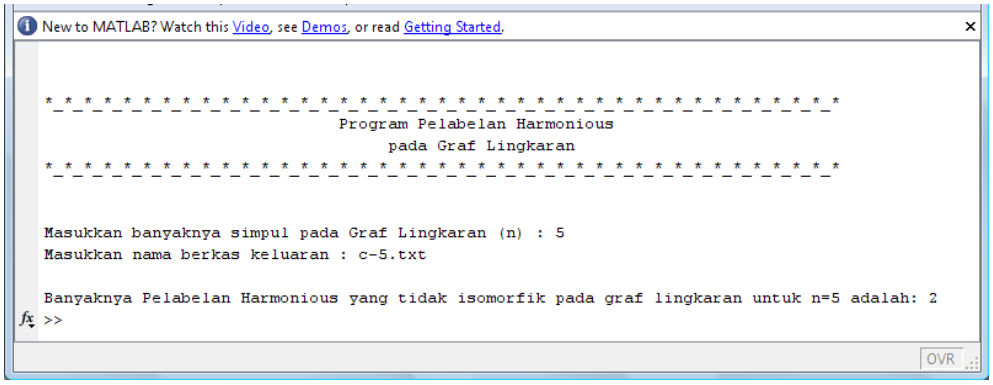
Seperti yang telah dijelaskan pada Subbab.3.2, algoritma pelabelan harmonis pada graf lingkaran C_n, terdiri dari dua fungsi yaitu fungsi initialize-Cycle (Algoritma 2.1) dan fungsi extendCycle (t) (Algoritma 2.2). Dalam implementasinya, program untuk graf lingkaran juga terdiri dari dua fungsi, yaitu cycle dan extendcycle. Fungsi cycle akan melakukan tahap-tahap pada Algoritma 2.1. Kemudian fungsi cycle akan memanggil fungsi extendCycle yang akan melakukan tahap-tahap pada Algoritma 2.2. *Array* label simpul pada program ini dinyatakan dengan $S(i)$ dan *array* label busur dinyatakan dengan $B(j)$. *Array* label simpul dan *array* label busur di algoritma merupakan suatu *array* berukuran $|V|$ yang berisikan nilai *true* atau *false*, tetapi tidak demikian pada program. Pada program ini *array* label simpul dan *array* label busur merupakan suatu *array* berukuran $|V|$ yang berisi label-label untuk melabel elemen pada graf, yaitu $S(i) = i$ dan $B(j) = j$, dimana $i = j = 0, 1, 2, \ldots, |V|-1$. Setiap label-label ini digunakan tepat satu kali. Jika label simpul i (label busur j) sudah digunakan, maka tandai $S(i) = n$ dan $B(j) = n$. Jika ternyata label yang digunakan bukan merupakan label yang tepat untuk melabel elemen graf maka label simpul i dan label busur j dibuat tersedia kembali dengan ditandai $S(i) = i$ dan $B(j) = j$, yang berarti label label simpul i (label busur j) tersedia kembali. Masukan dari program ini adalah n (banyak simpul). Serta keluaran dari program ini adalah banyak pelabelan yang mungkin dan tidak isomorfik pada graf lingkaran C_n. *Listing* program ini dilampirkan pada Lampiran 2 (*CD*).

Pemanggilan program dilakukan dengan cara mengetik “cycle” pada *Command Window*. Pengguna akan diminta memasukkan nilai n (banyak simpul)

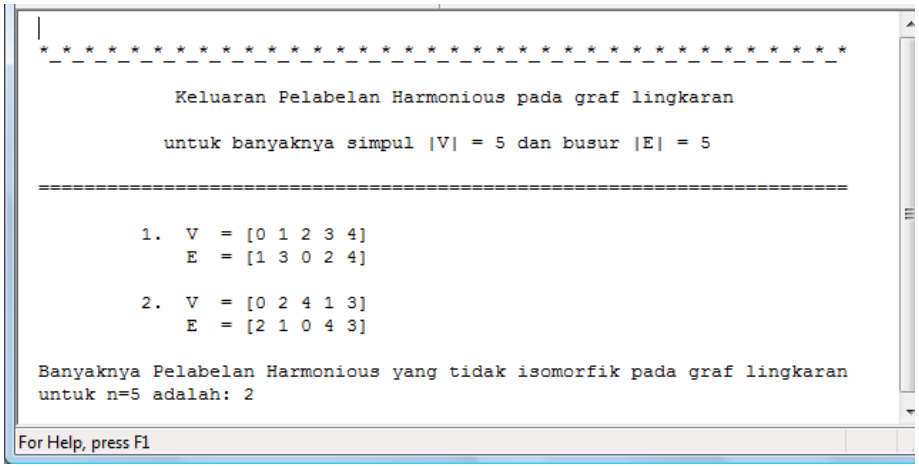
pada graf lingkaran dan nama berkas keluaran (*file output*). Berkas keluaran digunakan untuk menyimpan keluaran dari program yang berupa label simpul dan label busur beserta banyak pelabelan harmonis yang mungkin dan berbeda pada graf lingkaran dengan nilai *n* yang diberikan oleh pengguna.



Gambar 4.6 Tampilan awal pada *Command Window*



Gambar 4.7 Tampilan keluaran pada *Command Window*

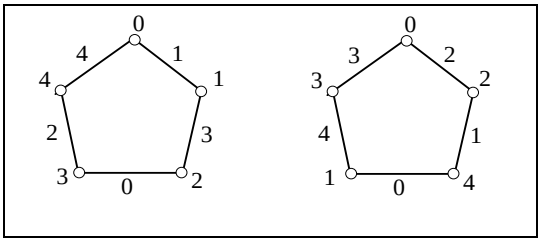


Gambar 4.8 Tampilan keluaran pada berkas

Selanjutnya, akan diberikan contoh penggunaan program ini pada pelabelan harmonis graf lingkaran C_5. Langkah pertama menjalankan program yaitu ketik "cycle" di *command window*, kemudian muncul tampilan seperti pada Gambar 4.6. Kemudian pengguna harus memasukkan nilai *n* (banyak simpul) dan nama berkas keluarannya. Karena graf lintasan C_5, maka *n* = 5. Nama file sebagai berkas keluarannya adalah c-5.txt. Serta, hasilnya yaitu terdapat 2 pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lingkaran C_5 (Gambar 4.7). Isi dari file 'c-5.txt' akan diperlihatkan pada Gambar 4.8.

Pada berkas keluaran, terdapat 2 pelabelan harmonis yang mungkin dan tidak isomorfik untuk C_5. Pelabelan yang pertama pada Gambar 4.8, diperoleh *V* = [0 1 2 3 4] menunjukkan label simpul v_1, v_2, v_3, v_4, v_5 secara berurutan, serta *E* = [1 3 0 2 4] menunjukkan label busur e_1, e_2, e_3, e_4, e_5 secara berurutan. Pelabelan harmonis yang diperoleh ini sama dengan pelabelan harmonis yang diperoleh pada contoh pelabelan harmonis pada graf lingkaran C_5 di Subbab 3.2.

Semua pelabelan yang telah terbentuk pada Gambar 4.8, jika digambar-kan, akan tampak seperti pada Gambar 4.9.



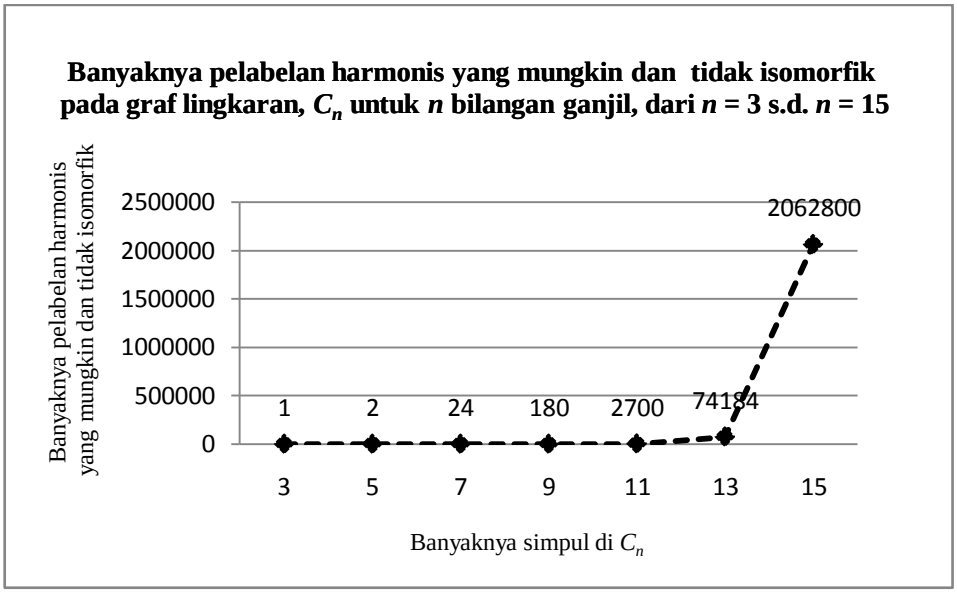
Gambar 4.9 Representasi visual keluaran pelabelan harmonis pada graf C_5

Simulasi dari program pelabelan harmonis pada graf lingkaran C_n dijalankan untuk mengetahui berapa banyak pelabelan harmonis yang mungkin dan tidak isomorfik untuk nilai *n* yang diberikan. Program ini bisa disimulasikan untuk sembarang *n*, namun dikarenakan keterbatasan waktu, simulasi hanya dijalankan sampai *n* = 15. Hasil simulasi pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lintasan C_n diberikan pada Tabel 4.2.

Tabel 4.2 Banyak pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lingkaran C_n untuk n = 3 s.d. n = 15

Graf Lingkaran, C_n	Banyak pelabelan harmonis yang mungkin dan tidak isomorfik
C_3	1
C_4	0
C_5	2
C_6	0
C_7	24
C_8	0
C_9	180
C_{10}	0
C_{11}	2700
C_{12}	0
C_{13}	74184
C_{14}	0
C_{15}	2062800

Pada Tabel 4.2 terlihat bahwa untuk n bernilai bilangan genap, tidak ditemukan pelabelan harmonis pada graf C_n. Hal ini sesuai dengan yang dibuktikan oleh Graham dan Sloane (1980) bahwa graf lingkaran C_n adalah harmonis jika dan hanya jika $n \equiv 1$ atau $3(mod\ 4)$ atau bilangan ganjil. Namun program ini tetap dijalankan untuk n genap, untuk memverifikasi kebenaran program.



Gambar 4.10 Grafik pertambahan pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lingkaran C_n, untuk n bilangan ganjil, dari n = 3 s.d. n = 15

Pada Gambar 4.10 ditunjukkan hubungan nilai n (banyak simpul) dengan banyak pelabelan harmonis yang mungkin dan tidak isomorfik. Untuk n ganjil terlihat bahwa semakin bertambahnya nilai n, maka banyak pelabelan harmonis yang tidak isomorfik juga bertambah dengan pertambahan yang cenderung eksponensial.

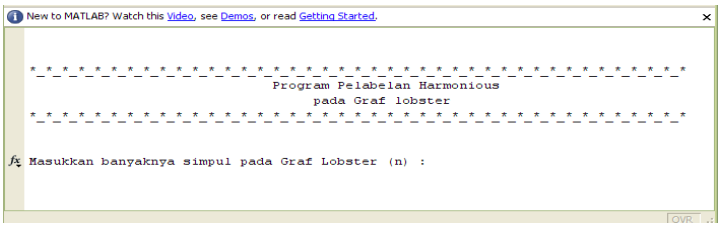
Pada subbab selanjutnya akan diberikan implementasi dan simulasi algoritma pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lobster teratur $L_{n,r,1}$.

4.3 Implementasi dan Simulasi Algoritma Pelabelan Harmonis pada Graf Lobster Teratur

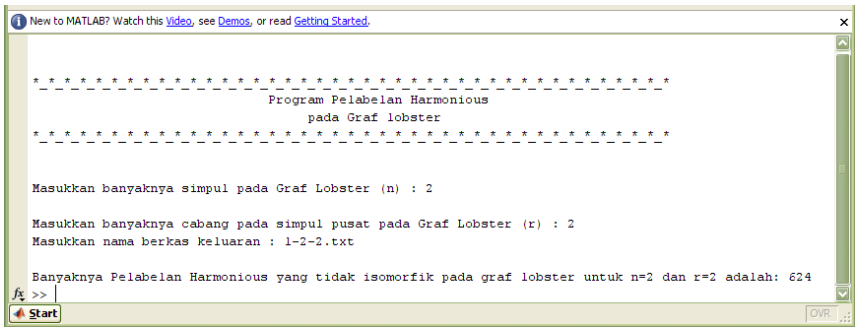
Seperti yang telah dijelaskan pada Subbab.3.3, algoritma pelabelan harmonis pada graf lobster teratur $L_{n,r,1}$, terdiri dari empat fungsi yaitu fungsi initializeLobster (Algoritma 3.1), extendLobster1(t) (Algoritma 3.2), extend-Lobster2(s,t) (Algoritma 3.3), dan extendLobster3(s,t) (Algoritma 3.4). Dalam implementasinya, program untuk mecari pelabelan harmonis pada graf lobster teratur juga terdiri dari empat fungsi, yaitu lobster, extendlob1(t), extendlob2(s,t), dan extendlob3(s,t). Fungsi lobster akan melakukan tahap-tahap pada Algoritma

3.1, fungsi extendlob1(t) akan melakukan tahap-tahap pada Algoritma 3.2, fungsi extendlob2(s,t) akan melakukan tahap-tahap pada Algoritma 3.3, dan fungsi extendlob3(s,t) akan melakukan tahap-tahap pada Algoritma 3.4. *Array* label simpul pada program ini dinyatakan dengan $S(i)$ dan *array* label busur dinyatakan dengan $B(j)$. *Array* label simpul dan *array* label busur di algoritma masing-masing merupakan suatu *array* berukuran $|V|$ dan $|E|$ yang berisikan nilai *true* atau *false*, tetapi tidak demikian pada program. Pada program ini *array* label simpul dan *array* label busur masing-masing merupakan suatu *array* berukuran $|V|$ dan $|E|$ yang berisi label-label untuk melabel elemen pada graf, yaitu $S(i) = i$ dan $B(j) = j$, dimana $i = 0, 1, 2, \ldots, |V|-1$ dan $j = 0, 1, 2, \ldots, |E|-1$. Setiap label-label ini digunakan tepat satu kali. Jika label simpul i (label busur j) sudah digunakan, maka tandai $S(i) = 2nr+n$ dan $B(j) = 2nr+n$. Jika ternyata label yang digunakan bukan merupakan label yang tepat untuk melabel elemen graf maka label simpul i dan label busur j dibuat tersedia kembali dengan ditandai $S(i) = i$ dan $B(j) = j$, yang berarti label label simpul i (label busur j) tersedia kembali. Masukan dari program ini adalah n (banyak simpul pada *backbone* lobster, P_n) dan r (banyak simpul daun dari simpul-simpul pada *backbone* caterpillar teratur). Serta keluaran dari program ini adalah banyak pelabelan yang mungkin dan tidak isomorfik pada graf lobster teratur $L_{n,r,1}$. *Listing* program ini dilampirkan pada Lampiran 3 (*CD*).

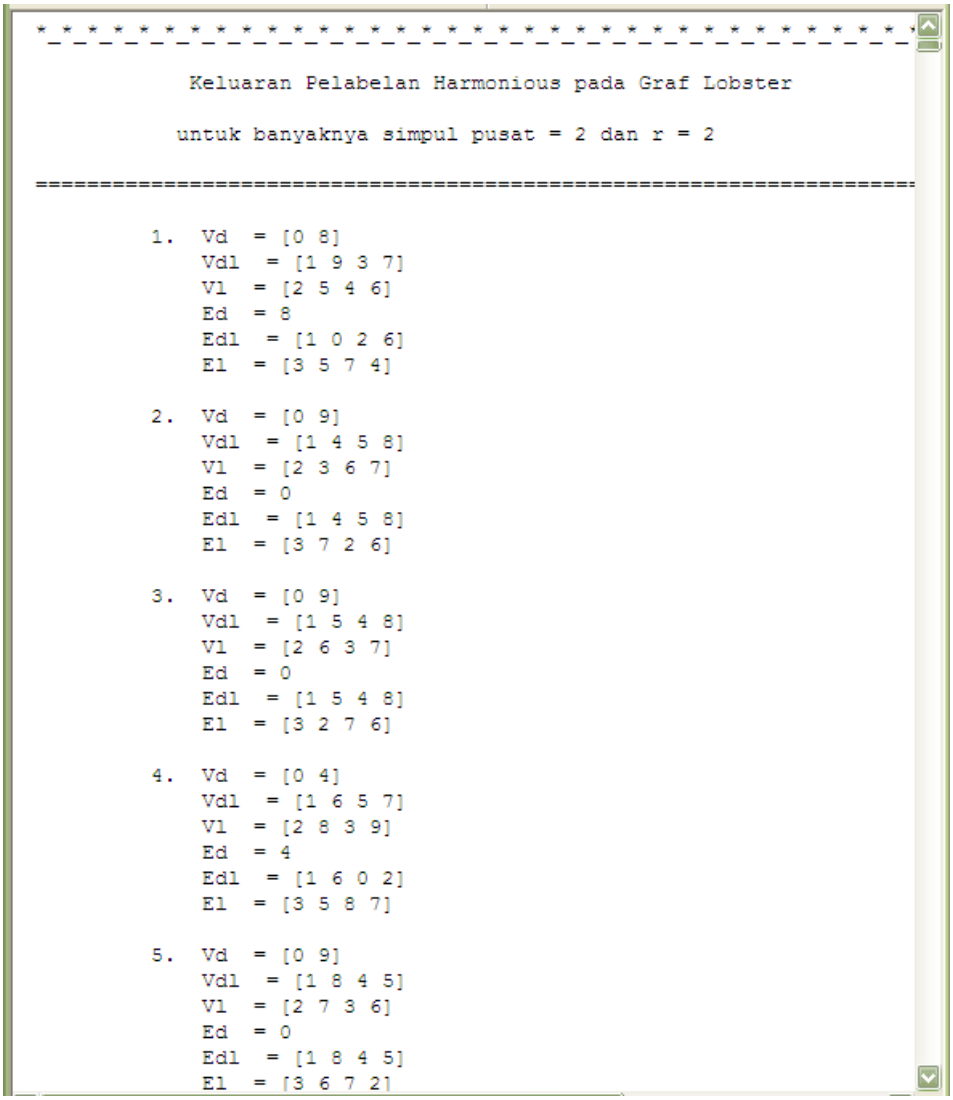
Pemanggilan program dilakukan dengan cara mengetik "lobster" pada *Command Window*. Pengguna akan diminta memasukkan nilai n (banyak simpul pada *backbone* lobster, P_n) dan r (banyak simpul daun dari simpul-simpul pada *backbone* caterpillar teratur), serta nama berkas keluaran (*file output*). Berkas keluaran digunakan untuk menyimpan keluaran dari program yang berupa label simpul dan label busur beserta banyak pelabelan harmonis yang mungkin dan berbeda pada graf lobster teratur dengan nilai n dan r yang diberikan oleh pengguna..



Gambar 4.11 Tampilan awal pada *Command Window*



Gambar 4.12 Tampilan keluaran pada *Command Window*



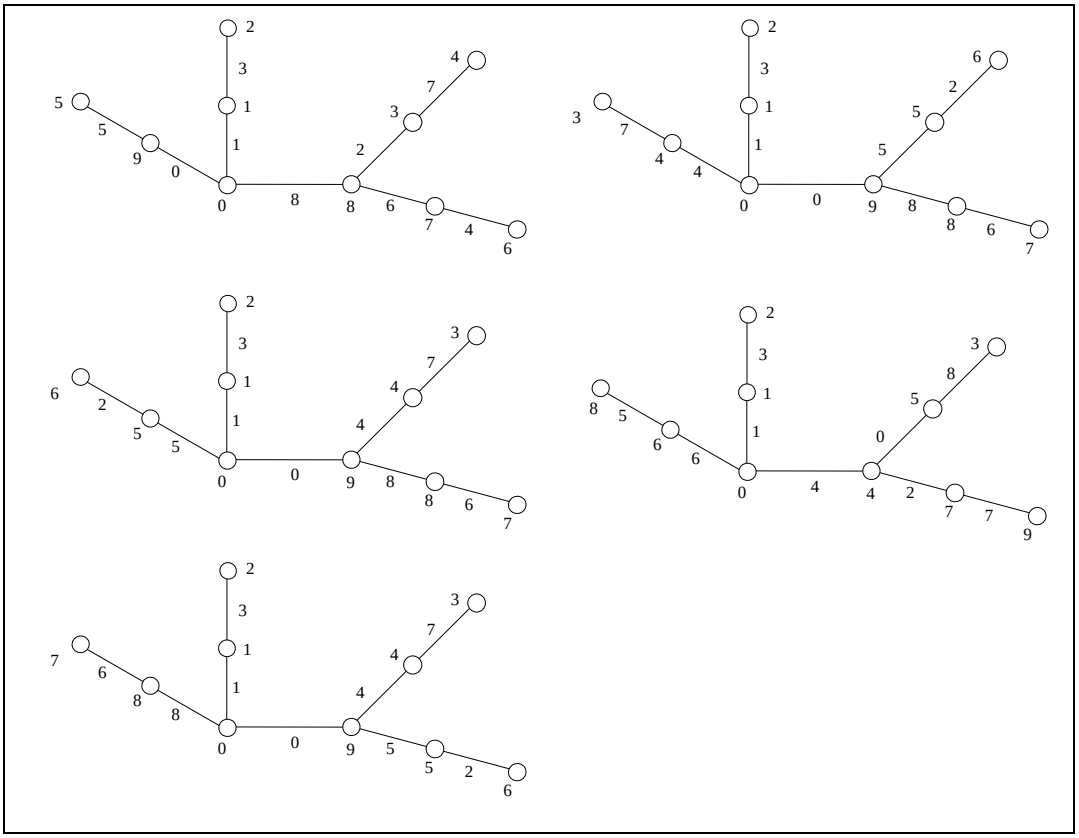
Gambar 4.13 Tampilan keluaran pada berkas

Selanjutnya, akan diberikan contoh penggunaan program ini pada pelabelan harmonis graf lobster teratur, $L_{2,2,1}$. Langkah pertama menjalankan program yaitu ketik "lobster" di *command window*, kemudian muncul tampilan seperti pada Gambar 4.11. Kemudian pengguna harus memasukkan *n*, *r*, dan nama berkas keluarannya. Karena graf lobster teratur, $L_{2,2,1}$, maka $|V| = 2nr+n = 10$ dan $|E| = |V|-1=9$. Nama file sebagai berkas keluarannya adalah l-2-1.txt. Serta, hasilnya yaitu terdapat 624 pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lobster teratur (Gambar 4.12). Isi 5 pelabelan pertama pada file "l-2-1.txt" akan diperlihatkan pada Gambar 4.13.

Pada berkas keluaran, terdapat 624 pelabelan harmonis yang mungkin dan tidak isomorfik untuk $L_{2,2,1}$. Pelabelan yang pertama pada Gambar 4.3, diperoleh V_d = [0 8] menunjukkan label simpul $v_d(1)$,$v_d(2)$ secara berurutan, V_{dl} = [1 9 3 7] menunjukkan label simpul $v_{dl}(1)$, $v_{dl}(2)$, $v_{dl}(3)$, $v_{dl}(4)$ secara berurutan, V_l = [2 5 4 6] menunjukkan label simpul $v_l(1)$, $v_l(2)$, $v_l(3)$, $v_l(4)$ secara berurutan, E_d = 8 menunjukkan label busur $e_d(1)$, E_{dl} = [1 0 2 6] menunjukkan label busur $e_{dl}(1)$, $e_{dl}(2)$, $e_{dl}(3)$, $e_{dl}(4)$ secara berurutan, serta E_l = [3 5 7 4] menunjukkan label busur $e_l(1)$, $e_l(2)$, $e_l(3)$, $e_l(4)$ secara berurutan. Pelabelan harmonis yang diperoleh ini sama dengan yang diperoleh pada contoh di Subbab 3.3.

Semua pelabelan yang telah terbentuk pada Gambar 4.13, jika digambarkan, akan tampak seperti pada Gambar 4.14.

Simulasi dari program pelabelan harmonis pada graf lobster teratur $L_{n,r,1}$, dijalankan untuk mengetahui berapa banyak pelabelan harmonis yang mungkin dan tidak isomorfik untuk nilai *n* dan *r* yang diberikan. Program ini bisa disimulasikan untuk sembarang *n* dan *r*. Untuk graf lobster teratur $L_{n,r,1}$, penambahan satu nilai *r* akan menambahkan 2 simpul dan 2 busur. Sedangkan penambahan satu nilai *n* akan menambahkan 3 simpul dan 3 busur. Pada simulasi hanya dijalankan sampai $n = 2$ dengan $r=1$, $r=2$, dan $r=3$, $n = 3$ dengan $r=1$ dan $r=2$, serta $n = 4$ dengan $r=1$. Hasil simulasi pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lobster teratur $L_{n,r,1}$, diberikan pada Tabel 4.3, Tabel 4.4, dan Tabel 4.5.

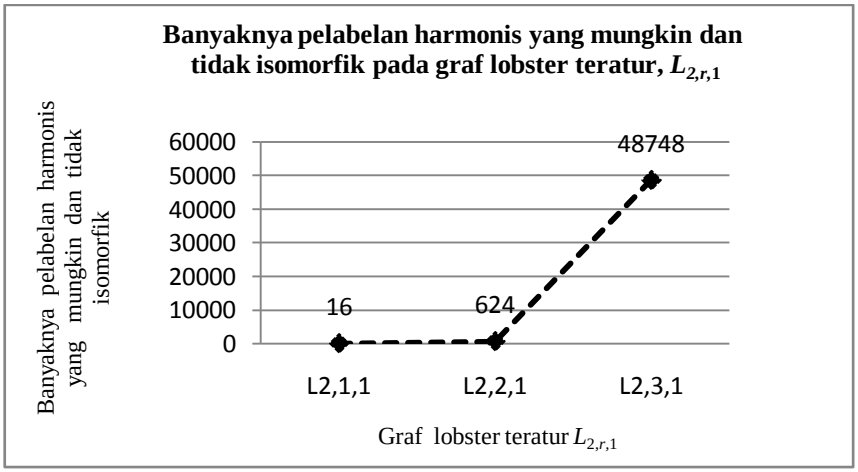


Gambar 4.14 Representasi visual keluaran 5 pelabelan harmonis pertama pada graf $L_{2,2,1}$

Tabel 4.3 Banyak pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lobster teratur $L_{2,r,1}$ untuk r = 1 s.d. r = 3

Graf Lobster Teratur, $L_{2,r,1}$	Banyak pelabelan harmonis yang mungkin dan tidak isomorfik
$L_{2,1,1}$	16
$L_{2,2,1}$	624
$L_{2,3,1}$	48748

Pada Tabel 4.3 terlihat bahwa untuk n = 2 dengan r=1, r=2, dan r=3 dapat ditemukan pelabelan harmonis untuk graf lobster teratur. Banyak pelabelan harmonis yang terbentuk, semakin bertambah seiring bertambahnya nilai r dengan n=2.



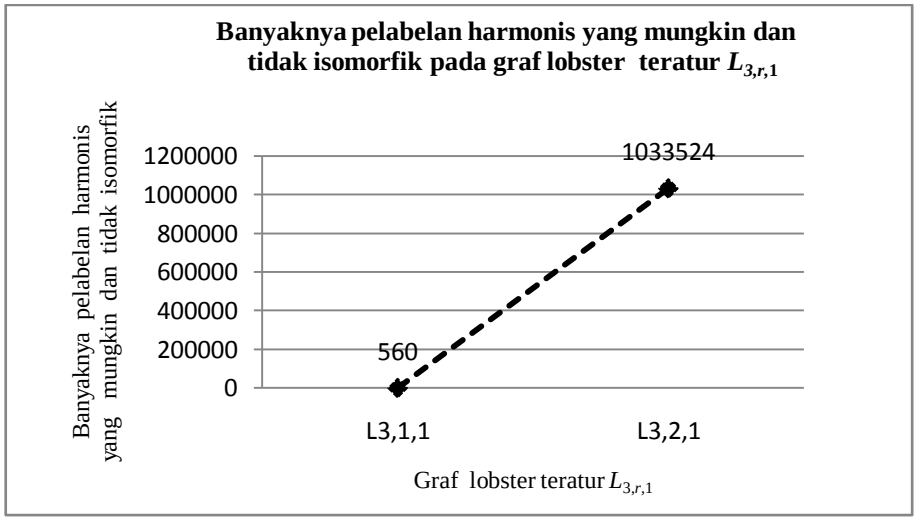
Gambar 4.15 Grafik pertambahan pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lobster teratur $L_{2,r,1}$ untuk r = 1 s.d. r = 3

Pada Gambar 4.15 ditunjukkan banyak pelabelan tidak isomorfik pada graf lobster teratur $L_{2,r,1}$ menurut pertambahan nilai r. Banyak pelabelan harmonis yang tidak isomorfik, saat r=2 adalah 239 kali lebih besar dibandingkan saat r=1 dan saat r=3 adalah sekitar 78 kali lebih besar dibandingkan saat r=2. Perubahan perbandingan ini menunjukkan tren yang eksponensial.

Tabel 4.4 Banyak pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lobster teratur $L_{3,r,1}$ untuk r = 1 dan r = 2

Graf Lobster Teratur $L_{3,r,1}$	Banyak pelabelan harmonis yang mungkin dan tidak isomorfik
$L_{3,1,1}$	560
$L_{3,2,1}$	1033524

Pada Tabel 4.4 terlihat bahwa untuk n = 3 dengan r=1 dan r=2 dapat ditemukan pelabelan harmonis untuk graf lobster teratur $L_{3,r,1}$. Banyak pelabelan harmonis yang terbentuk semakin bertambah seiring bertambahnya nilai r dengan n=3.



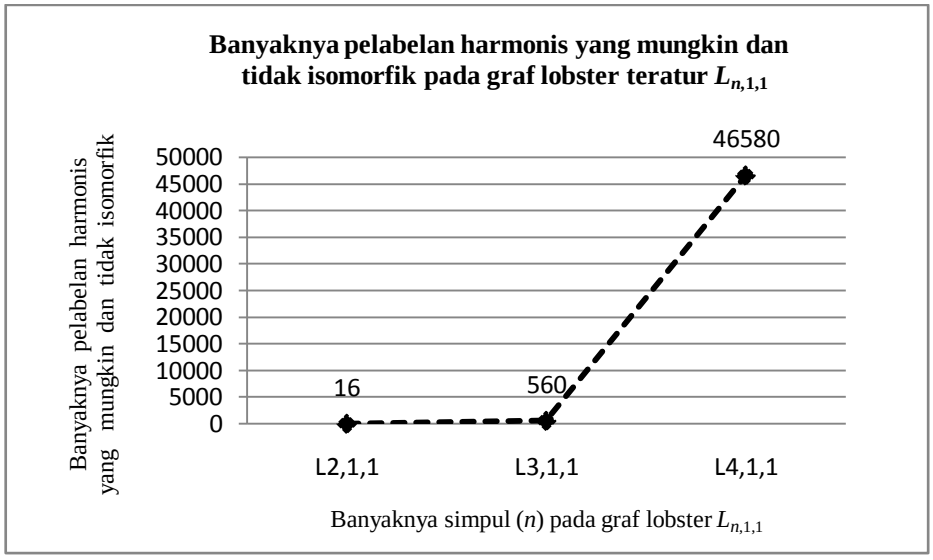
Gambar 4.16 Grafik pertambahan pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lobster teratur $L_{3,r,1}$ untuk $r = 1$ dan $r = 2$

Pada Gambar 4.16 ditunjukkan banyak pelabelan tidak isomorfik pada graf lobster teratur $L_{3,r,1}$ menurut pertambahan nilai r. Banyak pelabelan harmonis yang tidak isomorfik, saat r=2 adalah sekitar 1845 kali lebih besar dibandingkan saat r=1. Perubahan perbandingan ini menunjukkan tren yang eksponensial.

Tabel 4.5 Banyak pelabelan harmonis yang mungkin dan tidak isomorfik pada graf lobster teratur $L_{n,1,1}$ untuk $n = 2$ s.d. $n = 4$

Graf Lobster Teratur, $L_{n,1,1}$	Banyak pelabelan harmonis yang mungkin dan tidak isomorfik
$L_{2,1,1}$	16
$L_{3,1,1}$	560
$L_{4,1,1}$	46580

Pada Tabel 4.5 terlihat bahwa untuk n = 2,3,4 dengan r=1, dapat ditemukan pelabelan harmonis untuk graf lobster teratur $L_{n,r,1}$. Banyak pelabelan harmonis yang terbentuk, semakin bertambah seiring bertambahnya nilai n dengan r=1.



Gambar 4.17 Grafik pertambahan pelabelan harmonis yang mungkin dan tidak isomorfik pada graf graf lobster teratur $L_{n,1,1}$ untuk n = 2 s.d. n = 4

Pada Gambar 4.17 ditunjukkan banyak pelabelan tidak isomorfik pada graf lobster teratur $L_{n,1,1}$ menurut pertambahan nilai n. Banyak pelabelan harmonis yang tidak isomorfik, saat n=3 adalah 35 kali lebih besar dibandingkan saat n=2 dan saat n=4 adalah sekitar 83 kali lebih besar dibandingkan saat n=3. Perubahan perbandingan ini menunjukkan tren yang eksponensial.

Pada bab selanjutnya akan diberikan kesimpulan dari pembahasan skripsi ini.

BAB 5
KESIMPULAN

Dalam skripsi ini telah dibangun algoritma pelabelan harmonis pada graf lintasan, lingkaran dan lobster teratur. Dengan menggunakan algoritma-algoritma tersebut dapat diperoleh semua pelabelan harmonis yang mungkin dan tidak isomorfik pada masing-masing graf untuk nilai *n* dan *r* (untuk graf lobster teratur) tertentu. Algoritma-algoritma tersebut telah diimplementasikan dan disimulasikan dalam bentuk program. Secara teoritis, program-program ini dapat menghasilkan semua pelabelan harmonis yang mungkin dan tidak isomorfik untuk sembarang nilai *n*. Namun karena keterbatasan waktu, untuk graf lintasan hanya pada nilai n=2 s.d. n=14, untuk graf lingkaran hanya pada nilai n=3 s.d. n=15, dan untuk graf lobster teratur hanya pada nilai n=2 dengan r=1,2,3, n=3 dengan r=1,2, dan n=4 dengan r=1.

Hasil yang diperoleh pada skripsi ini, sebagai berikut :

- Graf Lintasan , P_n :
  - Algoritma fungsi intializePath dan fungsi extendPath(t).
  - Program pathH dan extendpathH, dengan masukan nilai *n* (banyak simpul) dan keluarannya adalah banyak pelabelan harmonis yang mungkin dan tidak isomorfik untuk nilai *n* tertentu.
  - Hasil simulasi selalu ada untuk n=2 s.d. n=14.

- Graf Lingkaran, C_n :
  - Algoritma fungsi intializeCycle dan fungsi extendCycle(t).
  - Program cycle dan extendcycle, dengan masukan nilai *n* (banyak simpul) dan keluarannya adalah banyak pelabelan harmonis yang mungkin dan tidak isomorfik untuk nilai *n* tertentu.
  - Hasil dari simulasi selalu ada untuk n=3 s.d. n=15.

- Graf Lobster teratur, $L_{n,r,1}$:
  - Algoritma fungsi intializeLobster, fungsi extendLobster1(t), fungsi extendLobster2(s,t), dan fungsi extendLobster3(s,t).
  - Program lobster, extendlob1(t), extendlob2(s,t), dan extendlob3(s,t), dengan masukan nilai n (banyak simpul pada *backbone* lobster, P_n) dan r (banyak simpul daun dari simpul-simpul pada *backbone* caterpillar teratur).
  - Hasil dari simulasi selalu ada untuk n=2 dengan r=1, r=2, dan r=3, n=3 dengan r=1 dan r=2, serta n=4 dengan r=1.

DAFTAR PUSTAKA

Ananda, A. I. (2009). *Algoritma Pelabelan Total Simpul Ajaib Pada Graf Lingkaran, Matahari, Dan Kecebong*. Skripsi, Departemen Matematika Universitas Indonesia.

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

Gallian, J.A.(2009).A Dynamic Survey of Graph Labeling.*The Electronic Journal of Combinatorics 16*, #DS6.

Graham, R. L. dan Sloane, N. J. A. (1980). On additive bases and harmonious graphs. *SIAM J. Alg. Discrete Meth*.

Baker, A. dan Sawada, J. (2008). Magic Labelings on Cycles and Wheels. *COCOA LNCS*.

Luan, H. C. & Fu, H. L.(2007). New Results on Harmonious Trees. *Utilitas Mathematica* 74.

Rachmawati, M. (2009). *Pelabelan Total (a, d)-Simpul Antiajaib Pada Graf Lintasan dan Graf Lingkaran*. Skripsi, Departemen Matematika Universitas Indonesia.

Surgandini, A. (2009). *Algoritma Pelabelan Total Busur Ajaib Pada Graf Lingkaran, Kipas, Dan Roda*. Skripsi, Departemen Matematika Universitas Indonesia.

Utami, B. (2009). *Algoritma Pelabelan Total Simpul Ajaib Pada Graf Friendship, Kipas, Dan Jahangir yang Diperumum*. Skripsi, Departemen Matematika Universitas Indonesia.

West, D. B. (2001). *Introduction to Graph Theory*. United States: Prentice-Hall, Inc.