PENERAPAN TEORI GRAF PADA KODE PENGULANGAN

SKRIPSI

Oleh :

MOHAMAD AMSUJUDI
NIM : 02510042

JURUSAN MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MALANG
2008

PENERAPAN TEORI GRAF PADA KODE PENGULANGAN

SKRIPSI

Diajukan Kepada :
Universitas Islam Negeri Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Sains (S.Si)

Oleh :

MOHAMAD AMSUJUDI
NIM : 02510042

JURUSAN MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MALANG
2008

PENERAPAN TEORI GRAF PADA KODE PENGULANGAN

SKRIPSI

Oleh :

MOHAMAD AMSUJUDI
NIM : 02510042

Telah disetujui oleh :
Dosen Pembimbing

Dr. Yus Mochamad Cholily, M.Si

Malang, 02 Pebruari 2008
Mengetahui
Ketua Jurusan Matematika

Sri Harini M.Si
NIP. 150318321

PENERAPAN TEORI GRAF PADA KODE PENGULANGAN

SKRIPSI

Oleh :

MOHAMAD AMSUJUDI
NIM : 02510042

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

Malang, 12 April 2008

Susunan Dewan Penguji : Tanda Tangan

1. Penguji Utama : Wahyu H. Irawan, M.Pd. (   )
2. Ketua : Abdussakir, M.Pd. (   )
3. Sekretaris : Dr. Yus Mochamad Cholily, M.Si. (   )

Mengetahui dan Mengesahkan
Dekan Fakultas Sains dan Teknologi

<u>Prof. Dr. Sutiman Bambang Sumitro, D.Sc.</u>
NIP. 130 809 153

Motto :

Ciri orang berakal dan berbudaya adalah tidak menetap disuatu tempat.
Jika kau tinggalkan tempat kelahiran, kau akan temui derajat mulia di tempat baru.
Engkau bagaikan emas yang sudah terangkat dari tempatnya.
Merantaulah mencari kemuliaan karena dengan merantau kau dapatkan lima manfaat yaitu : 1) Menghilangkan kesedihan, 2) mendapatkan kehidupan, 3) mendapatkan ilmu, 4) mengukuhkan jiwa dan 5) berkenalan dengan banyak orang.

Kupersembahkan untuk

Yang Maha atas segalanya
Kedua Orang Tuaku, kakak-kakakku, orang special disampingku (Ika), seluruh keluargaku, teman-teman seperjuanganku dan Almamaterku tercinta

KATA PENGANTAR

Puji syukur kepada Allah SWT, penulis panjatkan atas segala rahmat dan hidayah-Nya, sehingga skripsi yang berjudul “PENERAPAN TEORI GRAF PADA KODE PENGULAGAN” dapat terselesaikan sebagaimana mestinya. Skripsi ini dimaksudkan sebagai syarat untuk memperoleh gelar Sarjana Sains di Universitas Islam Negeri Malang. Adapun yang disajikan dalam penulisan skripsi ini merupakan hasil dari pemikiran untuk mengaplikasikan Teori Graf pada kode pengulangan yang meliputi encoding, decoding, dan pengoreksian kata kode dalam pengiriman pesan.

Keberhasilan dalam menyelesaikan skripsi ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu penulis mengharapkan terima kasih yang dalam dan kepada semua pihak yang telah membantu, khususnya kepada:

1. Bapak Dr. Yus Mochamad Cholily, M.Si selaku Pembimbing yang telah meluangkan waktunya untuk membimbing dan memberi petunjuk serta motivasi dalam penyusunan skripsi ini dengan sabar dan telaten memberikan bimbingan dan pengarahan kepada penulis, sehingga skripsi ini dapat terselesaikan dengan baik.
2. Ibu Sri. Hasrini, M.Si. selaku Ketua Jurusan Matematika Fakultas Sains dan Teknologi UIN Malang
3. Bapak dan Ibu tercinta yang telah membantu baik moril, materiil dan spirituil, sehingga penulis dapat menyelesaikan skripsi ini.

4. Bapak dan Ibu Dosen Jurusan Matematika yang telah mendidik dan mengajar kepada penulis, sehingga penulis mempunyai bekal ilmu pengetahuan dan pengalaman untuk menyelesaikan skripsi ini.
5. Kakak-kakakku dan Ika MS yang telah memberi semangat dalam penulisan skripsi ini. Dan semua pihak yang tidak dapat penulis sebutkan satu persatu.

Tak ada gading yang tak retak, penulis menyadari bahwa dalam penelitian dan penyusunan skripsi ini masih banyak kekurangannya, karena tidak ada manusia yang sempurna, kesempurnaan itu milik Allah SWT, sehingga penulis dengan rendah hati mengharapkan kritik dan saran yang bersifat membangun untuk penyempurnaan skripsi ini.

Penulis hanya bisa berharap semoga Allah SWT senantiasa melimpahkan kurnia dan syafaatnya kepada seluruh pihak yang telah membantu dalam proses penulisan skripsi ini.

Malang, Januari 2008

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

Gambar

ABSTRAK

Amsujudi,Mohamad. 2007. Penerapan Teori Graf Pada Kode Pengulangan. Skripsi Jurusan Matematika Fakultas Sains Dan Teknologi. Universitas Islam Negeri Malang. Pembimbing Dr. Yus Mochamad Cholily, M.Si.

Kata Kunci: Teori Graf, Digraf, Coding, Kode Pengulangan Dan Decoding

Dalam pengiriman pesan, selalu diupayakan agar pesan tersebut dapat diterima dengan cepat dan sesuai dengan aslinya. Skripsi ini mengkaji cara merubah suatu pesan dalam bentuk kata kode pengulangan. Pengkodean adalah metode yang mengubah suatu informasi menjadi kode (*encoding*), dan mengembalikan kode tersebut kedalam informasi semula (*decoding*). Dari proses pengkodean, pengiriman sampai dengan proses penguraian kata kode, kode-kode tersebut dimungkinkan mengalami perubahan (*error*), untuk itu diadakan pengoreksian.

Data yang di kaji dalam skripsi ini berupa kata pesan, kemudian kata tersebut dirubah ke dalam bentuk kata kode. Perubahan pesan menjadi kata kode ini dilakukan dengan menggunakan aturan graf dan pengkodean sehingga dapat menjawab rumusan permasalahan yang diteliti yaitu dapat mengetahui cara merubah suatu pesan dalam bentuk kata kode pengulangan.

BAB I

P E N D A H U L U A N

1.1 Latar Belakang

Secara informal suatu *graf* adalah himpunan benda yang disebut titik yang terhubung oleh sisi. Pada perkembangan zaman yang semakin modern banyak sekali struktur yang dipresentasikan dengan graf, dan banyak masalah yang dapat diseleseaikan dengan bantuan graf. Teori Graf diantaranya dapat digunakan menyeleseaikan permasalahan tentang pencarian rute terpendek suatu perjalanan, dan juga dapat digunakan untuk pemecahan masalah dalam pengaturan lampu lalulintas di persimpangan jalan sehingga setiap arus dapat bergerak dengan aman dan waktu tunggu minimal.

Teori Pengkodean adalah salah satu teori yang mempelajari cara pengiriman informasi dari satu tempat ketempat lain. Teori Pengkodean meliputi bagaimana cara mengubah suatu informasi menjadi kode (*encoding*), dan mengembalikan kode tersebut kedalam informasi semula (*decoding*) (Hankerson, 2000 : 1). Dari proses pengkodean, pengiriman sampai dengan proses penguraian kata kode, kode-kode tersebut sangat besar kemungkinanya untuk mengelami perubahan.

Teori Graf dapat dipadukan dengan Teori Pengkodean dalam kehidupan sehari-hari yaitu pada permasalahan pengiriman suatu pesan. Skripsi ini membahas cara mengubah suatu pesan menjadi suatu kata kode.

Pembahasan ini diarahkan pada pengkodean dengan kode pengulangan. Dalam kode pengulangan, kata kode diperoleh dengan cara mengubah suatu pesan yang sudah diubah dalam bentuk kode sebanyak k-kali dengan k anggota bilangan bulat positif.

Dalam pengiriman pesan selalu diupayakan agar pesan tersebut diterima dengan cepat, namun satu hal yang paling penting adalah pesan tersebut dapat diterima sesuai dengan aslinya. *Error corecting code* adalah suatu cara untuk memperbaiki kesalahan yang terjadi selama proses pengiriman, sehingga pembaca kode tetap dapat menguraikan pesan yang telah terkontaminasi kesalahan tersebut menjadi informasi yang benar, sehingga pesan tersebut sesuai dengan aslinya.

1.2 Rumusan Masalah

Ada tiga macam jenis pengkodean yaitu pengkodean berbentuk matrik pembangkit, pengkodean berbentuk Hamming dan pengulangan. Skripsi ini mengkaji tentang permasalahan pengkodean yang berbentuk kode pengulangan. Secara rinci permasalahan kode pengulangan yang akan dikaji adalah sebagai berikut :

1. Bagaimana cara mengkodekan suatu pesan dengan kode pengulangan menggunakan pendekatan graf berarah?
2. Bagaimana cara menguraikan kata kode dengan kode pengulangan menggunakan pendekatan graf berarah?

3. Bagaimana cara mengoreksi suatu kata kode yang berbentuk kode pengulangan?

1.3 Fokus Masalah

Materi Teori Pengkodean sangatlah luas, namun dalam kajian ini difokuskan pada pembahasan tentang pengkodean yang berbentuk pengulangan. Kajian kode pengulangan ini lebih ditekankan pada :

1. Cara mengkodekan suatu pesan dengan kode pengulangan menggunakan pendekatan graf berarah.
2. Cara menguraikan kata kode dengan kode pengulangan menggunakan pendekatan graf berarah.
3. Cara mengoreksi suatu kata kode yang berbentuk kode pengulangan.

1.4 Tujuan Penelitian

Perhatikan kembali pada rumusan masalah. Sesuai permasalahan yang dibahas, pembahasan ini bertujuan untuk mengetahui bagaimana cara merubah suatu pesan dalam bentuk kata kode pengulangan. Secara lebih mendalam bertujuan sebagai berikut :

1. Untuk mengetahui cara mengkodekan suatu pesan dengan kode pengulangan menggunakan pendekatan graf berarah.
2. Untuk mengetahui cara menguraikan kata kode dengan kode pengulangan menggunakan pendekatan graf berarah.

3. Untuk mengetahui cara mengoreksi suatu kata kode yang berbentuk kode pengulangan.

1.5 Manfaat Penelitian

Hal yang dapat diharapkan dari hasil skripsi ini yaitu untuk mengetahui proses pengiriman pesan dalam bentuk kata kode yang bersifat kode pengulangan. Dipandang dari Teori Graf khususnya graf berarah untuk mengetahui penggunaan graf berarah dalam masalah pengkodean dan cara menguraikan kata kode yang berbentuk kode pengulangan. Sehingga pembaca dapat mengirim suatu pesan yang bersifat rahasia dimana pesan tersebut dapat dibaca oleh orang atau kelompok orang yang dikehandaki.

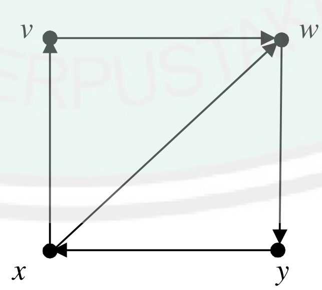
BAB II

KAJIAN TEORI

2.1 Digraf

Graf berarah (*digraf*) *D* adalah struktur yang terdiri dari himpunan yang unsurnya disebut *titik* (*vertex*)**,** dan himpunan pasangan terurut dua titik yang disebut *sisi berarah* (*arc*). Himpunan titik pada digraf *D* dinotasikan dengan $V(D)$, dan himpunan sisi berarahnya dinotasikan $A(D)$. Banyaknya unsur di $V(D)$, $|V(D)|$, disebut *order* dari *D* dan banyaknya unsur di $A(D)$, $|A(D)|$ disebut *ukuran* dari *D*. Jika titik *v* dan *w* adalah titik $V(D)$, maka sisi berarah (*vw*) dikatakan berarah dari *v* ke *w*, atau menghubungkan *v* ke *w* (Robin, 1992 : 85).

Misal *D* adalah digraf dengan himpunan titik $V(D)=\{v,w,x,y\}$ dan $A(D)=\{(vw),(wy),(yx),(xw),(xv)\}$. Digraf *D* ini dapat disajikan dalam bentuk gambar seperti ditunjukkan dalam Gambar 2.1.1

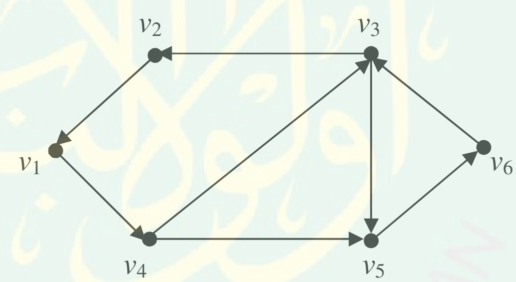


Gambar 2.1.1 Digraf *D*

Misal *v* adalah sebuah titik di digraf *D*. Banyaknya sisi berarah yang menuju ke titik *v* disebut *derajat masuk* (*in-degree*) dari *v* yang dinotasikan dengan id(v), dan banyaknya sisi berarah yang keluar dari titik v disebut *derajat keluar* (*out-degree*) dari v yang dinotasikan dengan od(v) (Seymour, 2002 : 103). Perhatikan Gambar 2.1.1, titik w mempunyai id(w) = 2 dan od(w) = 1. Selanjutnya id(v) = 1, id(y) = 1, id(x) = 1 dan od(v) = 1, od(y) = 1, od(x) = 2.

Lintasan pada digraf D adalah barisan titik-titik $v_1v_2v_3...v_n$ anggota $V(D)$ sedemikian hingga $(v_iv_{i+1}) \in A(D)$, untuk i = 1,2,3,...,n-1, sedangkan yang dinamakan *lintasan tertutup* pada digraf D adalah barisan titik-titik $v_1v_2v_3...v_n$, v_1 anggota $V(D)$ sedemikian hingga $(v_iv_{i+1}v_i) \in A(D)$, untuk i = 1,2,3,...,n-1 (Robin, 1992 : 34)

Contoh :



Gambar 2.1.2 Digraf D

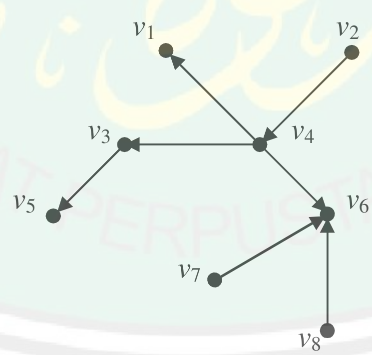
Perhatikan digraf D dengan $V(D) = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ dan $A(D) = \{(v_1v_4), (v_4v_5), (v_4v_3), (v_5v_6), (v_6v_3), (v_3v_5), (v_3v_2), (v_2v_1)\}$, yang gambarnya ditunjukkan pada Gambar 2.1.2. Barisan $v_1v_4v_3v_5v_6$ merupakan salah satu contoh lintasan. Lintasan $v_1v_4v_5v_6v_3v_2v_1$ dinamakan lintasan tertutup karena dimulai dari titik v_1 dan kembali lagi ke titik v_1.

Jarak (*distance*) antara titik v_1 dan v_2 pada digraf *D*, dinotasikan dengan $d(v_1, v_2)$ adalah panjang lintasan (banyaknya sisi yang menghubungkan) terpendek dari v_1 ke v_2 pada digraf *D*. Dari Gambar 2.1.2 dapat dihitung bahwa $d(v_1, v_2) = d(v_1, v_6) = 3$, $d(v_1, v_3) = d(v_1, v_5) = 2$ dan $d(v_1, v_4) = 1$.

2.2 Pohon Berakar

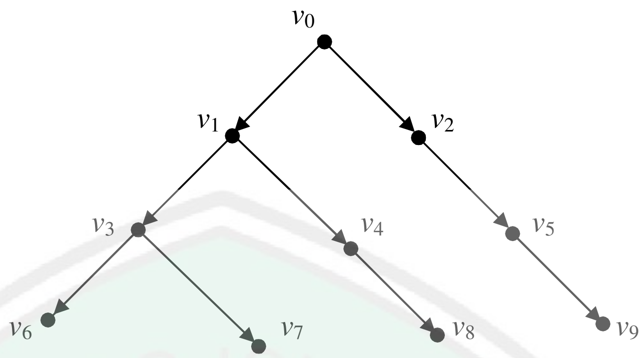
Pohon berarah T adalah digraf terhubung yang tidak memuat sikel. Pohon berarah *T* disebut *pohon berakar T* jika terdapat tepat satu titik v_0, dengan id(v_0) = 0 dan od(v_0) minimal 1 disebut akar (titik yang berada paling atas dalam pohon) (Robin, 1992 : 60). Misal *T* adalah pohon berakar dan v_0 sebagai akar dari pohon maka pohon berakar *T* dapat dinotasikan (T, v_0).

Contoh :



Gambar 2.2.1 Pohon Berarah *T*

Gambar 2.2.1 Pohon Berarah *T* adalah bukan pohon berakar, karena tidak terdapat satu titik khusus sebagai akar dari Pohon Berarah *T*.

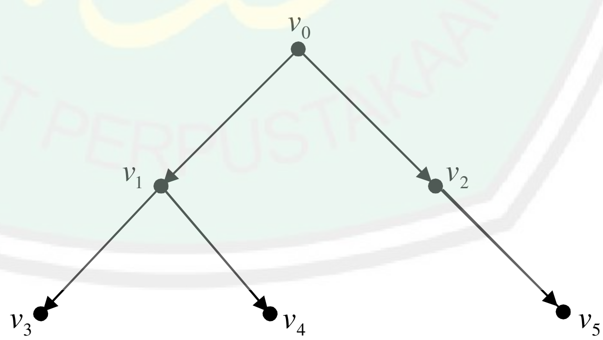


Gambar 2.2.2 (T, v_0)

Perhatikan gambar digraf yang ditunjukkan pada Gambar 2.2.2. Gambar tersebut menunjukkan sebuah pohon berarah dengan akar v_0.

Misal $(v_1\, v_2) \in A(D)$, maka v_1 disebut *tetangga dalam* (*in-neighbour*) dari v_2 dan himpunannya dinotasikan $N^-(v_2)$ dan v_2 disebut *tetangga luar* (*out-neighbour*) dari v_1 dan himpunannya dinotasikan $N^+(v_1)$ (Robin, 1992 : 31).

Contoh :

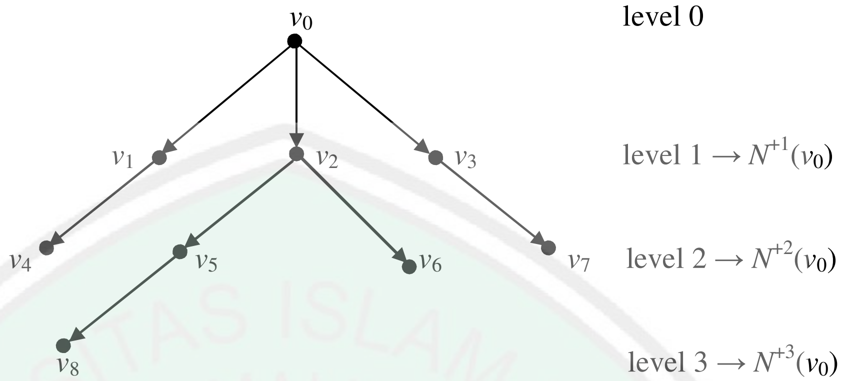


Gambar 2.2.3 (T, v_0)

Perhatikan Gambar 2.2.3, dari gambar tersebut diketahui bahwa v_1 tetangga luar dari v_0, begitu juga dengan v_2, sehingga $N^{+1}(v_0) = \{v_1, v_2\}$, v_3 tetangga luar kedua dari v_0 begitu juga dengan v_4 dan v_5, sehingga $N^{+2}(v_0) = \{v_3, v_4, v_5\}$, sedangkan v_0 merupakan tetangga dalam dari v_1 dan v_2, sehingga $N^{-1}(v_1)$ dan $N^{-1}(v_2)$ adalah v_0. Untuk memudahkan dalam pengamatan, pohon berakar *T* adalah menempatkan akar v_0 pada bagian paling atas dari suatu pohon. Titik yang terhubung langsung dari v_0 ditempatkan satu level (tingkat) dibawah v_0, dan seterusnya. Seperti pada gambar 2.2.3 Pohon Berakar, akar v_0 terletak pada level 0, titik v_1, v_2, v_3 terletak pada level 1, titik v_4, v_5, v_6, v_7 terletak pada level 2 dan titik v_8 terletak pada level 3. Pada umumnya, nilai level suatu titik v_n pada pokon berakar adalah panjang lintasan dari akar v_0 ke titik v_n.

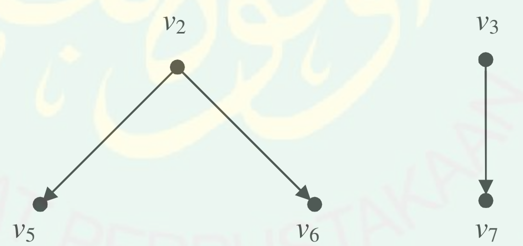
(T, v_0) adalah suatu pohon berakar dengan akar v_0, jika suatu titik v_2 dari *T* terhubung langsung ke titik v_5 dan titik v_5 terletak pada level di bawah titik v_2, maka titik v_5 disebut *child* (anak) dari v_2 dan titik v_2 adalah *parent* (orang tua) dari titik v_5. Sedangkan titik v_8 disebut *keturunan atau anak cucu* dari titik v_2, dan titik v_2 adalah *ancestor* (nenek, kakek, leluhur) dari titik v_8 jika lintasan $(v_5 v_8)$ pada (T, v_0) terletak di bawah titik v_5. Titik v_4, v_6, v_7, v_8 disebut *daun* (titik yang berada pada tingkat terendah dalam pohon), sedangkan titik v_1, v_2, v_3, v_5 disebut *titik internal* (titik dalam pohon yang hanya mempunyai anak).

Contoh :



Gambar 2.2.4 (T, v_0)

Misal T adalah pohon berakar dengan akar v_0 dan $v \in V(T)$. Pohon dengan akar v disebut *subpohon* dari (T, v_0). Sebagai contoh Gambar 2.2.5, berikut adalah subpohon yang ada di Gambar 2.2.4.



(i) (T, v_2)

(ii) (T, v_3)

Gambar 2.2.5

2.3 Pohon Biner

Pandang n adalah bilangan bulat positif, pohon T disebut n-pohon jika setiap titik internal mempunyai n-anak, oleh karena itu yang disebut dengan pohon biner adalah akar pohon yang mempunyai dua anak atau derajat keluar dua (Seymour, 2002 : 121).

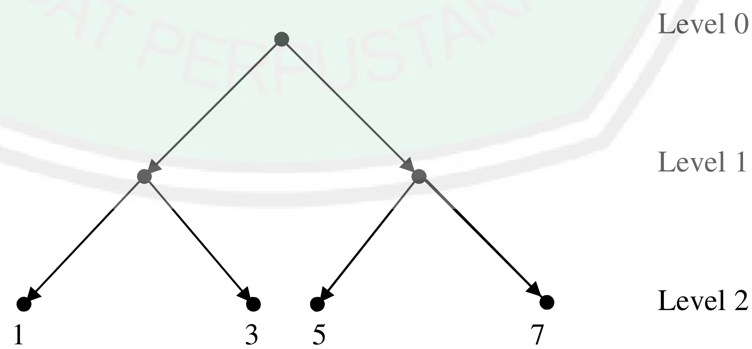
Pohon biner yang mempunyai daun sebanyak k dan masing-masing daun berlabel $w_1, w_2, w_3, ..., w_k$ adalah bilangan bulat positif, pohon biner tersebut dinamakan pohon biner berbobot (Grimaldi, 1985 : 641). Bobot pohon biner dapat dicari dengan cara menjumlahkan hasil kali dari bobot daun dengan panjang lintasan yang menuju daun tersebut, sehingga dapat dihitung dengan cara sebagai berikut:

$W(T) = \sum_{i=1}^{k} w_i l(w_i)$ dengan i = 1,2, . . .,k

w_i = label daun ke-i

$l(w_i)$ = panjang lintasan tunggal dari akar menuju w_i

Contoh:



Gambar 2.3.1 Pohon Biner T

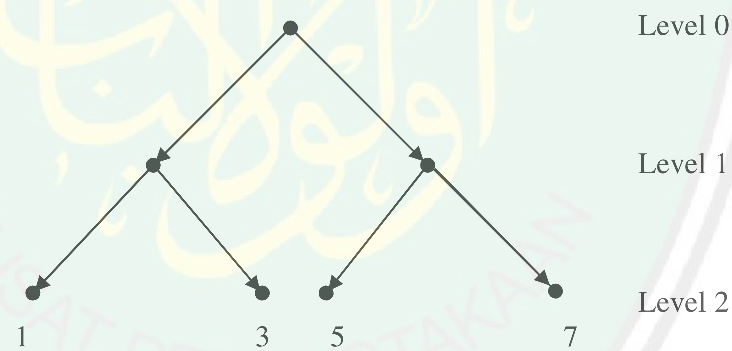
Pada Gambar 2.3.1 diketahui bobot dari masing-masing daun yaitu $w_1 = 1, w_2 = 3, w_3 = 5, w_4 = 7$ dan $l(w_1) = l(w_2) = l(w_3) = l(w_4) = 2$

Bobot pohon biner T adalah $W(T) = \sum_{i=1}^{k} w_i l(w_i) = 1 \cdot 2 + 3 \cdot 2 + 5 \cdot 2 + 7 \cdot 2 = 32$

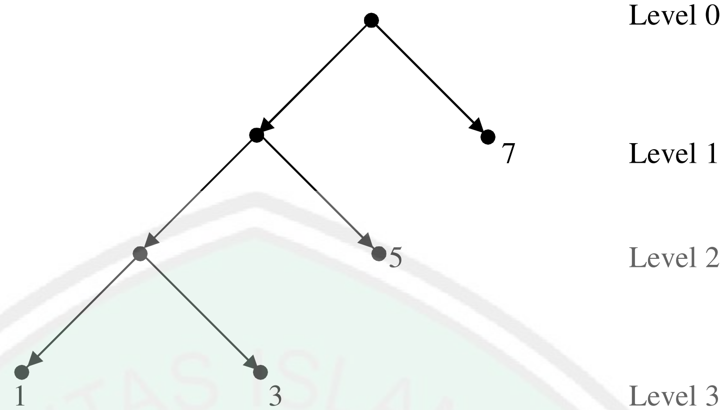
2.4 Pohon Biner Optimal

Pandang $w_1, w_2, w_3, ..., w_k$ adalah himpunan bilangan bulat positif pada pohon biner T, dengan $w_1 \le w_2 \le w_3 \le ... \le w_k$ dan $T = (V, E)$ memiliki ***n*** daun. Misalkan $T_1 = (V, E)$ memiliki bobot $w_1, w_2, w_3, ..., w_k$ dikatakan optimal jika $W(T_1) \le W(T)$ (Grimaldi, 1985 : 641).

Contoh :



Gambar 2.4.1(i)



Gambar 2.4.1(ii)

Pada Gambar 2.4.1(i) Pohon Biner *T*, diketahui bobot dari masing-masing daun yaitu $w_1 = 1, w_2 = 3, w_3 = 5, w_4 = 7$ dan $l(w_1) = l(w_2) = l(w_3) = l(w_4) = 2$.

Bobot pohon biner *T* adalah $W(T) = \sum_{i=1}^{k} w_i l(w_i) = 1 \cdot 2 + 3 \cdot 2 + 5 \cdot 2 + 7 \cdot 2 = 32$

Jadi bobot pohon biner *T* adalah 32.

Pada Gambar 2.4.1(ii) Pohon Biner T_1, diketahui bobot dari masing-masing daun yaitu $w_1 = 1, w_2 = 3, w_3 = 5, w_4 = 7$ dan $l(w_1) = 3, l(w_2) = 3, l(w_3) = 2, l(w_4) = 1$.

Bobot pohon biner T_1 adalah $W(T_1) = \sum_{i=1}^{k} w_i l(w_i) = 1 \cdot 3 + 3 \cdot 3 + 5 \cdot 2 + 7 \cdot 1 = 29$

Jadi bobot pohon biner T_1 adalah 29.

Dari Gambar 2.4.1(i) dan Gambar 2.4.1(ii) dapat disimpulkan bahwa $W(T_1) < W(T)$ sehingga pohon biner T_1 lebih optimal dari pada pohon biner *T*.

Tujuan membangun pohon biner optimal adalah untuk mendapatkan bobot pohon yang sekecil mungkin. Setelah mendapatkan pohon biner optimal maka kita menggunakan Algoritma Huffman untuk pohon biner optimal :

Misal $S = (w_1, w_2, w_3, ..., w_k), k \geq 2$ dan $w_1, w_2, w_3, ..., w_k$ anggota bilangan bulat positif. Algoritma Huffman diperoleh dengan mengikuti langkah-langkah sebagai berikut

1. Ambil $w_1, w_2 \in S$ dengan w_1, w_2 merupakan anggota yang memiliki bobot paling kecil dari *S*. Jika sudah diperoleh pohon biner optimal $T = (V, E)$ berarti selesai, jadi pohon yang dibangun adalah pohon biner optimal $T = (V, E)$ dengan bobot $w_1, w_2, w_3, ..., w_k$. Jika belum diperoleh pohon biner optimal $T = (V, E)$ lanjutkan ke langkah 2.
2. Membentuk pohon biner $T = (V, E)$ dengan akar $w_1 + w_2$, dengan cabang kiri berlabel w_1 dan cabang kanan berlabel w_2. Jika pohon biner $T = (V, E)$ dengan daun w_1 dan w_2 telah dibangun, masukkan ke dalam pohon biner $T = (V, E)$ berikutnya. Gantilah w_1 dan w_2 pada S dengan $w_1 + w_2$. Ulangi langkah 1 (Grimaldi, 1985 : 642).

2.5 Definisi Encoding dan Decoding

Misal *m,n* anggota bilangan bulat positif dan $n > m$, sedangkan *W* merupakan himpunan yang terdiri dari kata pesan yang dibentuk maka *encoding* didefinisikan proses penambahan bilangan biner sebanyak *n-m* karakter pada masing-masing $w \in W$, $W \subseteq Z_2^m$ agar terbentuk kata kode $C \in Z_2^n$ dan dinyatakan dalam bentuk fungsi $E : W \to C$ atau $E : Z_2^m \to Z_2^n$, dan ditulis $E(W):C$ atau $E(Z_2^m):Z_2^n$. Dari proses tersebut menghasilkan kata kode C yang diterima sebagai $C \in Z_2^n$.

Encoding berupa pengulangan adalah mengulang suatu pesan w = $w_1w_2w_3...w_m \in W \in Z_2^m$ sebanyak k-kali sehingga didapatkan kode kata berupa C=E(w)= $w_1w_2w_3...w_m$ $w_1w_2w_3...w_m$ $w_1w_2w_3...w_m$ dan C=E(w) = $\in Z_2^n = Z_2^m$. (Grimaldi, 1992 : 795).

Contoh 1:

Diketahui m = 5, k = 1 maka n = $1 \cdot 5$ = 5, sehingga $E: Z_2^5 \rightarrow Z_2^5$ adalah $E(W)$: $w_1w_2w_3 \ldots w_5w_1w_2w_3...w_5w_1w_2w_3 \ldots w_5$. Misalkan W : 11001 maka w_1 = 1, w_2 = 1, w_3 = 0, w_4 = 0, w_5 = 1, sehingga $E: Z_2^5 \rightarrow Z_2^5$ adalah $E(W)$ = E(11001) = 11001.

Contoh 2:

Diketahui m = 5, k = 2 maka n = $2 \cdot 5$ = 10, sehingga $E: Z_2^5 \rightarrow Z_2^{10}$ adalah $E(W)$: $w_1w_2w_3 \ldots w_5w_1w_2w_3...w_5w_1w_2w_3 \ldots w_5$. Misalkan W : 11001 maka w_1 = 1, w_2 = 1, w_3 = 0, w_4 = 0, w_5 = 1, sehingga $E: Z_2^5 \rightarrow Z_2^{10}$ adalah $E(W)$ = E(11001) = 1100111001.

Contoh 3:

Diketahui $m = 4$, k = 3 maka n = $3 \cdot 4$ = 12, sehingga $E: Z_2^4 \rightarrow Z_2^{12}$ adalah $E(W)$: $w_1w_2w_3w_4w_1w_2w_3w_4w_1w_2w_3w_4$. Misalkan W : 0101 maka w_1 = 0, w_2 = 1, w_3 = 0, w_4 = 0, sehingga E : $Z_2^4 \rightarrow Z_2^{12}$ adalah $E(W)$ = E(0101) = 010101010101

Contoh 4:

Diketahui m = 3, k = 4 maka n = $4 \cdot 3$ = 12, sehingga $E : Z_2^3 \to Z_2^{12}$ adalah $E(W)$: $w_1w_2w_3ww_1w_2w_3ww_1w_2w_3$. Misalkan W : 110 maka w_1 = 1, w_2 = 1, w_3 = 0, sehingga $E : Z_2^3 \to Z_2^{12}$ adalah $E(W) = E(110)$ = 110110110110

Untuk mengetahui pesan sebenarnya maka hal yang harus dilakukan adalah *decoding*, yaitu menguraikan kata kode $C : E(W) \in Z_2^n$ menjadi sebuah pesan $W \in Z_2^m$ yang bisa dinyatakan dalam bentuk fungsi $D : C \to W$ atau $D : Z_2^n \to Z_2^m$ ditulis $D(C)$ atau $D(Z_2^n) = Z_2^m$. Decoding dilakukan dengan cara mengambil $w_1w_2...w_m$ dari $C = E(W) = w_1w_2...w_m w_{m+1} w_{m+2}...w_n \in Z_2^n$ atau menghapus $w_{m+1}w_{m+2}...w_n$ dari $C = E(W)$ tersebut sehingga diperoleh $W = w_1w_2...w_m \in Z_2^m$ yang merupakan sebuah pesan.

Contoh 1:

Diketahui n = 10, k = 2 dengan decoding berupa kode pengulangan maka $\frac{n}{k}, \frac{10}{2}$= 5, sehingga $D : Z_2^{10} \to Z_2^5$ adalah $D(C) = w_1w_2w_3w_4w_5$. misalkan C = 1100111001 maka $D(C) = D(1100111001)$ = 11001.

Contoh 2:

Diketahui n = 12, k = 3 dengan decoding berupa kode pengulangan maka $\frac{n}{k}, \frac{12}{3}$= 4, sehingga $D : Z_2^{12} \to Z_2^4$ adalah $D(C) = w_1w_2w_3w_4$. misalkan C = 110111011101 maka $D(C) = D(110111011101)$ = 1101

2.6 Pengoreksian Kata Kode dengan Kode Pengulangan

Pengoreksian kesalahan kata kode dilakukan untuk mengetahui atau mengecek bahwa kata yang diterima sudah benar dan sudah sama dengan pesan sesungguhnya. Pendeteksian ini dilakukan dengan cara melihat melihat kata kode yang diterima secara keseluruhan, kemudian membaginya sebanyak k-bagian, k bilangan bulat positif. Karena proses pengkodean berupa kode pengulangan yaitu $E : Z_2^m \rightarrow Z_2^{n=mk}$ dengan k-banyaknya pengulangan suatu pesan. Setiap bagian mempunyai kata kode sebanyak m-karakter dan antara bagian satu dengan lainnya harus mempunyai kata kode yang sama. Jika ada salah satu bagian berbeda, berarti telah terjadi kesalahan pengkodean. Ambil kata kode pertama sebagai acuan karena pada dasarnya $w_{m+1}w_{m+2}...w_n$ merupakan kode pengulangan dari $w_1w_2...w_m$. Kemudian dilakukan dilakukan decoding ulang berupa kode pengulangan untuk memastikan kata kode yang sebenarnya sehingga diperoleh kode baru $D(C): w_1w_2...w_m$. Kata kode baru tersebut dengan menggunakan pohon biner optimal. Jika $D(C) = W$ maka kata kode pertama $w_1w_2...w_m$ adalah kata kode yang benar, dan sebaliknya jika $D(C) \neq W$ maka pilih kata kode yang mendekati kata kode yang dimaksud.

Cara membuat kata kode $D(C)$ yang mendekati kata kode yang dimaksud adalah sebagai berikut:

1. Pandang pohon biner optimal dari pesan yang dimaksud dan kode setiap huruf yang berbeda diperoleh dari pohon biner optimal
2. Buat lintasan tunggal dari $D(C)$ pada pohon biner tersebut

3. Jika terdapat *n* pilihan kata kode maka pilih kata kode yang memuat paling banyak karakter yang sama dengan kode dari pohon biner optimal tersebut

BAB III

METODE PENELITIAN

3.1 Metode Penelitian

Penelitian merupakan rangkaian kegiatan ilmiah dalam rangka pemecahan suatu permasalahan. Penelitian ini merupakan studi literatur yang bertujuan untuk mengetahui bagaimana cara mengubah suatu pesan dalam membentuk kata kode pengulangan dengan menggunakan Teori Graf dan Pengkodean. Dalam proses penelitian ini lebih dikhususkan untuk mengetahui cara mengkodekan suatu pesan serta mengetahui cara menguraikan kata kode dengan kode pengulangan dari sudut pandang teori graf yang menggunakan pendekatan graf berarah (*digraf*) dan untuk mengetahui cara mengkoreksi suatu kata kode yang berbentuk kode pengulangan. Fungsi dari penelitihan adalah mencari penjelasan dan jawaban terhadap permasalahan serta memberikan alternatif bagi kemungkinan yang dapat digunakan untuk pemecahan masalah. Kegiatan penelitian tidak lepas dari kerangka tujuan pemecahan permasalahan. Walaupun penelitian tidak memberikan jawaban langsung terhadap permasalahan yang diteliti akan tetapi hasilnya harus memberi kontribusi dalam usaha pemecahan masalah.

Pendekatan lain yang digunakan adalah pendekatan kualitatif. Dalam hal ini adalah merubah kata pesan menjadi kata kode yang berbentuk kode pengulangan. Pendekatan kualitatif bukan berarti sama sekali tidak menggunakan dukungan data kuantitatif akan tetapi penekanannya tidak pada pengujian hipotesis melainkan pada usaha menjawab pertanyaan penelitian melalui cara berfikir normal dan argumentatif. Sedangkan praktek dalam penelitihan ini akan dibahas tentang cara mengubah kata pesan menjadi kata kode yang berbentuk kode pengulangan dengan bantuan Teori Graf.

3.2 Pengumpulan Data

Proses yang dilakukan yang digunakan dalam penelitian ini diperoleh dengan cara mengkaji buku-buku yang berkaitan dengan materi pada skripsi ini yaitu yang berhubungan dengan Teori Graf dan pengkodean. Tahap – tahap yang dilakukan sebelum manganalisis data adalah sebagai berikut :

1. Mengetahui cara mengkodekan suatu pesan dengan kode pengulangan.
2. Mengetahui cara mengoreksi suatu kata kode dengan kode pengulangan.
3. Mengetahui cara mengoreksi suatu kata dengan berbentuk kode pengulangan.
4. Aplikasinya melalui contoh soal yang meliputi proses pengkodean, penguraian kata kode dan pengoreksian kesalahan kata kode yang diterima apabila kode tersebut tidak terbaca.

3.3 Kajian Data

Data yang dikaji dalam penulisan skripsi ini adalah berupa kata pesan, kemudian kata dalam bentuk pesan itu dirubah ke dalam bentuk kata kode. Perubahan pesan menjadi kata kode ini dilakukan dengan menggunakan aturan graf dan pengkodean sehingga dapat menjawab rumusan permasalahan yang dibahas yaitu dapat mengetahui cara merubah suatu pesan yang bersifat rahasia dalam bentuk kata kode pengulangan.

BAB IV

PEMBAHASAN

Pada Bab IV ini dibahas tentang permasalahan penerepan Teori Graf untuk masalah pengkodean yang berbentuk kode pengulangan dengan cara menerapkannya pada contoh soal. Contoh soal ini meliputi proses pengkodean, penguraian kata kode dan pengoreksian kesalahan kata kode yang diterima apabila kode tersebut tidak terbaca.

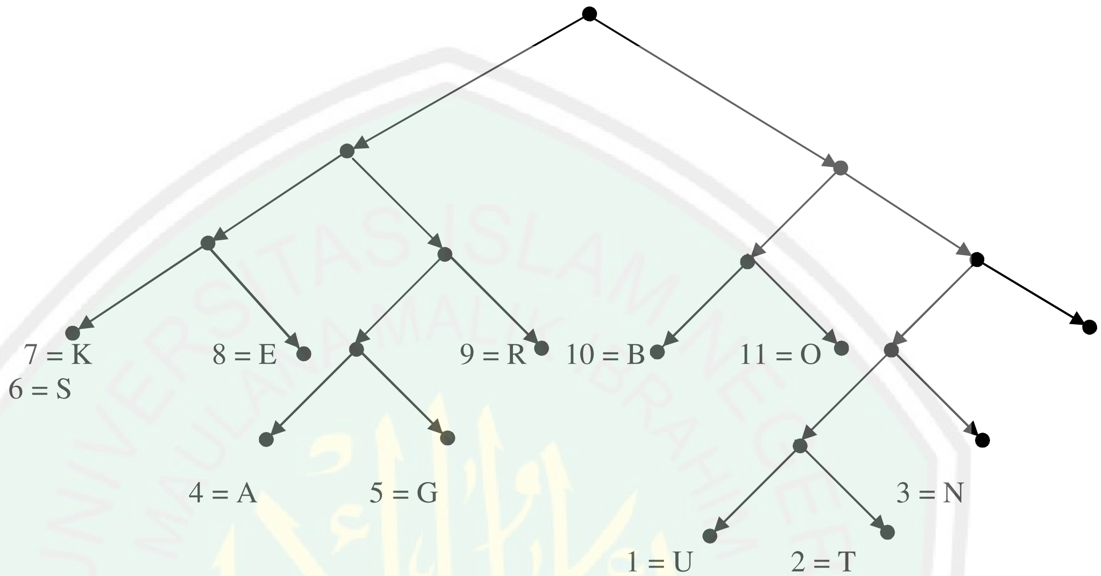
4.1 Encoding

Diberikan fungsi encoding $E : Z_2^m \rightarrow Z_2^n$, dengan bantuan pohon biner optimal dapat dibuat kata kode dari kalimat : "TUNGGU AKU SEBENTAR OK" dengan bobot-bobot hurufnya sebagai berikut:

Tabel 1
Daftar masing-masing bobot huruf

HURUF	**BOBOT**	**DITULIS**
A	4	A = 4
B	10	B = 10
E	8	E = 8
G	5	G = 5
K	7	K = 7
N	3	N = 3
O	11	O = 11
R	9	R = 9
S	6	S = 6
T	2	T = 2
U	1	U = 1

Dari bobot-bobot huruf tersebut maka dapat dibuat pohon biner optimal sebagai berikut :



Gambar 4.1.1

Gambar 4.1.1 merupakan pohon biner optimal karena memiliki bobot yang paling kecil, setelah pohon biner optimal terbentuk, maka perlu diperhatikan Algoritma Huffman yang dinyatakan dengan langkah-langkah sebagai berikut :

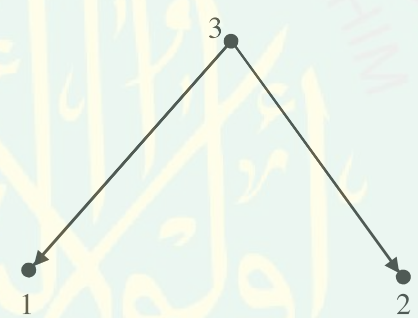
1. Mula-mula diurutkan bobot huruf yang tersedia, dimulai dari bobot terkecil hingga terbesar yaitu :

U = 1	T = 2	N = 3
A = 4	G = 5	S = 6
K = 7	E = 8	R = 9
B = 10	O = 11	

Bobot-bobot huruf tersebut disusun menjadi anggota S dan dianggap sebagai titik-titik terisolasi, sehingga S = {1,2,3,4,5,6,7,8,9,10,11}.

2. Dari titik internal tersebut dibuat sebuah pohon, dengan memilih dua anggota S yang terkecil yaitu 1 dan 2. Masing-masing sebagai daun sebelah kiri dan daun sebelah kanan, sedangkan akarnya diperoleh dari penjumlahan bobot kedua daun tersebut yaitu 1 + 2 = 3.

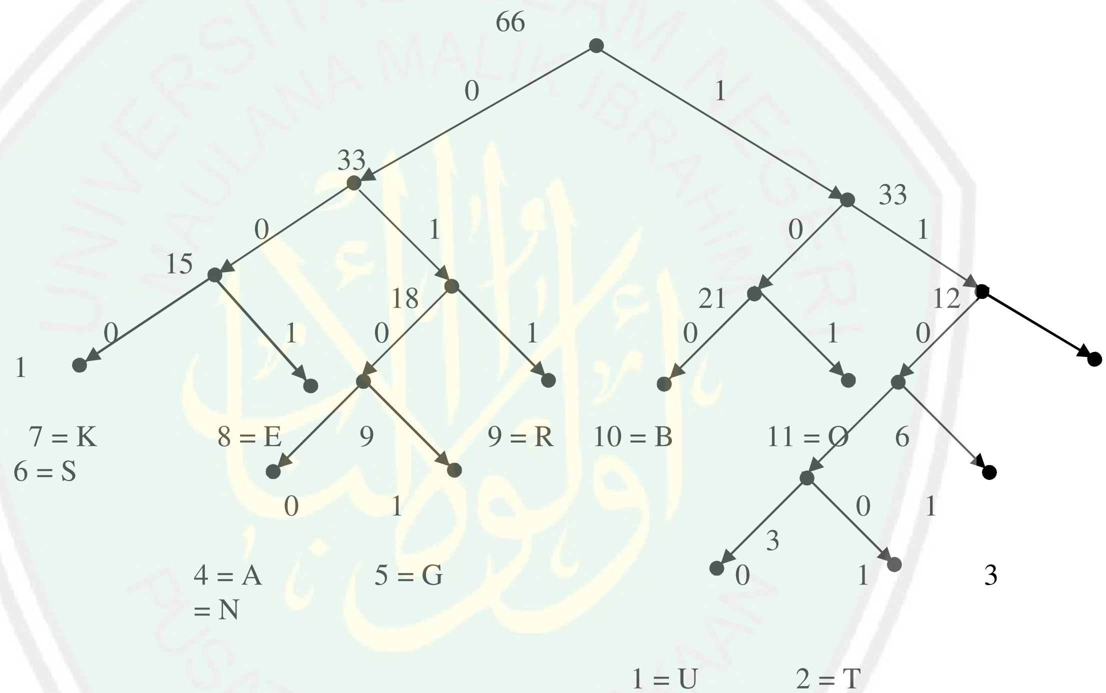
Hasilnya :



Gambar 4.1.2

3. Masukkan bobot akar pohon yang baru ke dalam anggota S dengan terlebih dahulu menghapus bilangan 1 dan 2 yang telah dijadikan daun, sehingga S yang baru menjadi S'= {3,3,4,5,6,7,8,9,10,11}.
4. Lakukan seperti langkah nomor 2 dan 3 di atas untuk setiap anggota S hingga didapatkan pohon biner optimal.

5. Berilah nilai 1 untuk setiap derajat keluar ke kanan dan 0 untuk derajar keluar ke kiri.
6. Untuk mendapatkan kode dari huruf yang dimaksud, maka dibuatlah lintasan tunggal dari akar menuju bobot huruf yang sesuai.
7. Hasil dari keenam langkah tersebut adalah sebagai berikut :



Gambar 4.1.3

Berdasarkan pohon biner optimal di atas, maka dapat diperoleh kode huruf dengan cara membuat lintasan tunggal dari akar menuju huruf tertentu, sehingga didapatkan kode huruf sebagai berikut :

T = 11001 U = 11000 N = 1101

G = 0101 A = 0100 K = 000

S = 111 E = 001 B = 100

R = 011 O = 101

Setelah kode huruf terbentuk, maka selanjutnya dilakukan proses pengkodean berupa kode pengulangan yang didefinisikan $E : Z_2^m \rightarrow Z_2^n$, untuk m, n anggota bilangan bulat positif sehingga didapat pengkodean yang berupa kode pengulangan sebagai berikut :

Diketahui $k = 2$, sehingga $E : Z_2^m \rightarrow Z_2^n$ dan $E(W) = w_1 w_2 \ldots w_m w_1 w_2 \ldots w_m$.

Untuk $m = 5$ dan $n = 5 \cdot 2 = 10$, sedemikian sehingga :

Jika T = 11001 maka E (T) = E(11001) = 1100111001

Jika U = 11000 maka E (U) = E(11000) = 1100011000

Untuk $m = 4$ dan $n = 4$ x 2 = 8 sedemikian sehingga :

Jika N = 1101 maka E (N) = E(1101) = 11011101

Jika G = 0101 maka E (G) = E(0101) = 01010101

Jika A = 0100 maka E (A) = E(0100) = 01000100

Untuk $m = 3$ dan $n = 3 \cdot 2 = 6$, sedemikian sehingga :

Jika K = 000 maka E (K) = E(000) = 000000

Jika S = 111 maka E (S) = E(111) = 111111

Jika E = 001 maka E (E) = E(001) = 001001

Jika B = 100 maka E (B) = E(100) = 100100

Jika R = 011 maka E (R) = E(011) = 011011

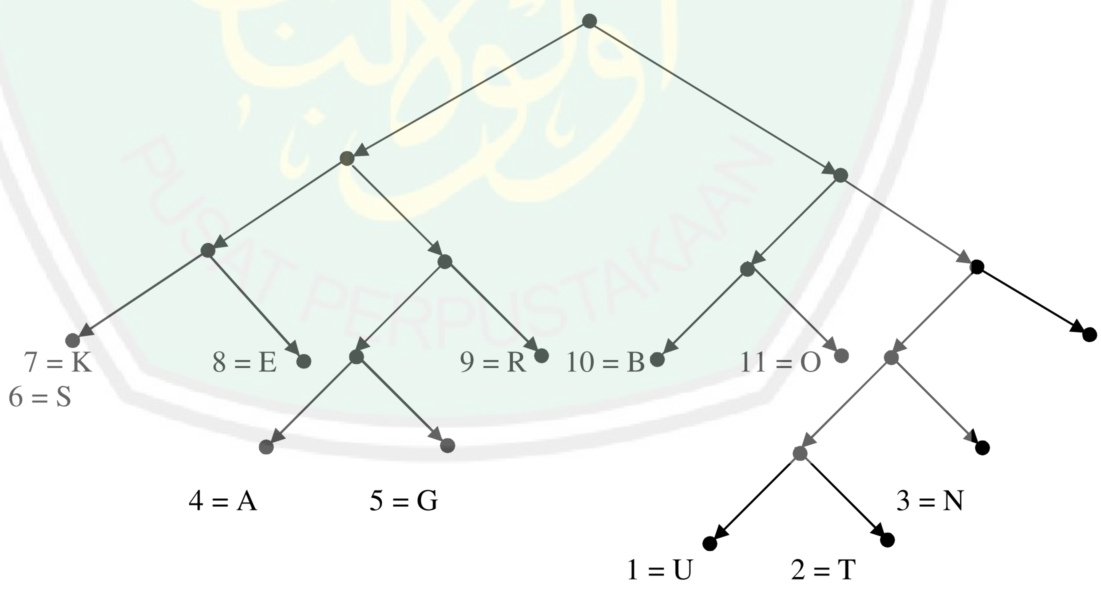
Jika O = 101 maka E (O) = E(101) = 101101

4.2 Decoding

Telah diberikan fungsi decoding $D : Z_2^n \rightarrow Z_2^m$, dengan bobot-bobot huruf sebagai berikut :

A = 4	B = 10	E = 8
G = 5	K = 7	N = 3
O = 11	R = 9	S = 6
T = 2	U = 1	

Dari bobot-bobot huruf tersebut maka dapat dibuat pohon biner optimal sebagai berikut :



Gambar 4.2.1

Setelah pohon biner optimal dibentuk, maka perlu diperhatikan Algoritma Huffman yang dinyatakan dengan langkah-langkah sebagai berikut :

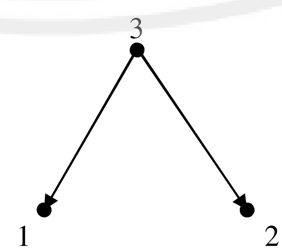
1. Mula-mula urutkanlah bobot huruf yang tersedia, dimulai dari bobot terkecil hingga terbesar yaitu :

U = 1	T = 2	N = 3
A = 4	G = 5	S = 6
K = 7	E = 8	R = 9
B = 10	O = 11	

   Bobot-bobot huruf tersebut disusun menjadi anggota *S* dan dianggap sebagai titik-titik terisolasi sehingga *S* = {1,2,3,4,5,6,7,8,9,10,11}.

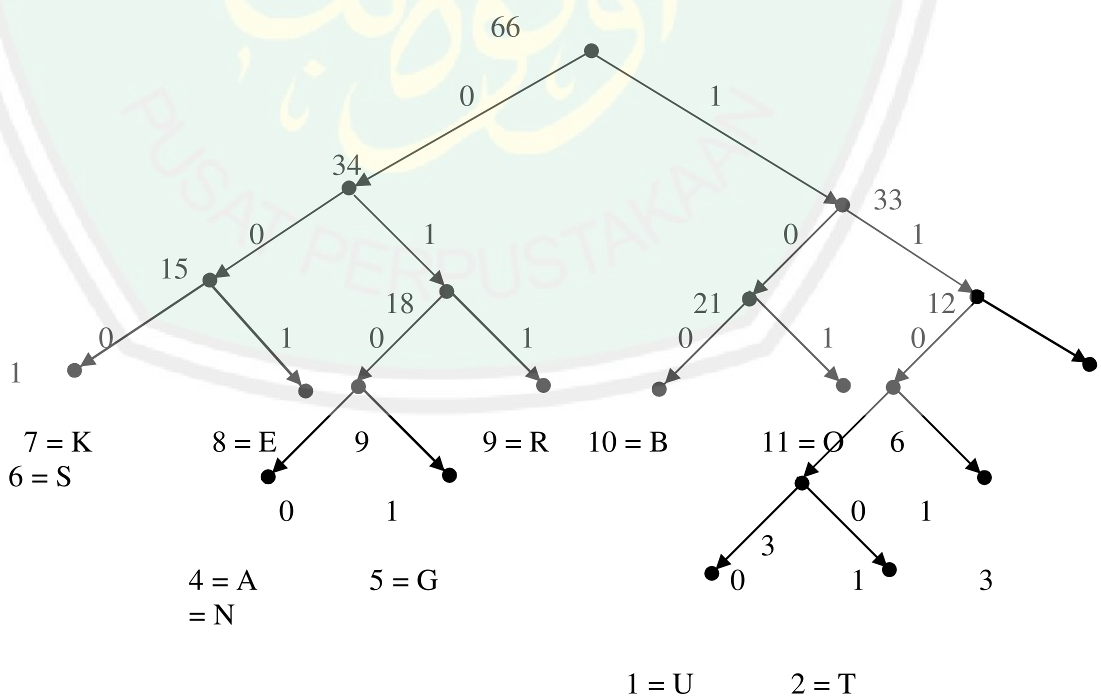
2. Dari titik internal tersebut dibuat sebuah pohon, dengan memilih dua anggota *S* yang terkecil yaitu 1 dan 2. Masing-masing sebagai daun sebelah kiri dan daun sebelah kanan, sedangkan akarnya diperoleh dari penjumlahan bobot kedua daun tersebut yaitu 1 + 2 = 3.

   Hasilnya :



Gambar 4.2.2

3. Masukkan bobot akar pohon yang baru ke dalam anggota *S* dengan terlebih dahulu menghapus bilangan 1 dan 2 yang telah dijadikan daun, sehingga *S* yang baru menjadi *S*'= {3,3,4,5,6,7,8,9,10,11}.
4. Lakukan seperti langkah nomor 2 dan 3 di atas untuk setiap anggota *S* hingga didapatkan pohon biner optimal.
5. Berilah nilai 1 untuk setiap derajat keluar ke kanan dan 0 untuk derajat keluar ke kiri.
6. Untuk mendapatkan kode dari huruf yang dimaksud, maka dibuatlah lintasan tunggal dari akar menuju bobot huruf yang sesuai.
7. Hasil keenam langkah tersebut adalah sebagai berikut :



Gambar 4.2.3

Dari pohon biner optimal di atas didapatkan kode huruf yang berbentuk kode pengulangan dengan fungsi encoding $n = 2m$, seperti pada Tabel 2

Tabel 2
Daftar kode huruf yang berbentuk kode pengulangan

HURUF	KODE HURUF
T	1100111001
U	1100011000
N	11011101
G	01010101
A	01000100
K	000000
S	111111
E	001001
B	100100
R	011011
O	101101

Selanjutnya dilakukan proses decoding berupa pengulangan yang didefinisikan $D : Z_2^n \to Z_2^m$, dari contoh tersebut diketahui bahwa $n = 2m$, maka fungsi decodingnya menjadi $D : Z_2^{2m} \to Z_2^m$. Langkah yang harus dilakukan adalah membagi setiap kode menjadi dua bagian kemudian diambil bagian pertama. Bagian pertama itulah yang merupakan pesan sesungguhnya.sehingga didapat decoding berupa pengulangan sebagai berikut :

Untuk $n = 10$ maka $m = 10 : 2 = 5$ sedemikian hingga :

Jika E(T) = 1100111001 maka D(T) = 11001

Jika E(U) = 1100011000 maka D(U) = 11000

Untuk n = 8 maka m = 8 : 2 = 4 sedemikian hingga :

Jika E(N) = 11011101 maka D(N) = 1101

Jika E(G) = 01010101 maka D(G) = 0101

Jika E(A) = 01000100 maka D(A) = 0100

Untuk n = 6 maka m = 6 : 2 = 3

Jika E(K) = 000000 maka D(K) = 000

Jika E(S) = 111111 maka D(S) = 111

Jika E(E) = 001001 maka D(E) = 001

Jika E(B) = 100100 maka D(B) = 100

Jika E(R) = 011011 maka D(R) = 011

Jika E(O) = 101101 maka D(O) = 101

Tabel 3
Daftar decoding dari kode pengulangan

HURUF	KODE PENGULANGAN	DECODING
T	1100111001	11001
U	1100011000	11000
N	11011101	1101
G	01010101	0101
A	01000100	0100
K	000000	000
S	111111	111
E	001001	001
B	100100	100
R	011011	011
O	101101	101

Dengan demikian bunyi pesan 11001, 11000, 1101, 0101, 0101, 11000, 0100, 000, 11000, 111, 001, 100, 001, 1101, 11001, 0100, 011, 101, 000 adalah **"**TUNGGU AKU SEBENTAR OK**"**

4.3 Pengoreksian Kata Kode Pengulangan

Diketahui pada pohon biner optimal kode setiap huruf yaitu T=11001 dan U = 11000. Misal diketahui n = 2m dengan menggunakan encoding berupa kode pengulangan dilakukan pengoreksian kata kode sebagai berikut.

a. 11001

b. 11000

diketahui n = 2m maka $E : Z_2^m \to Z_2^{2.m}$ ditulis $E(Z_2^n) : Z_2^{2m}$ dengan w$\in Z_2^m$ $C \le Z_2^{2m}$ dan C: $E(Z_2^m) =$ $w_1 w_2 ... w_m w_{m+1} w_{m+2} w_n \in Z_2^{2.m}$,

langkah-langkahnya adalah sebagai berikut :

1. Membagi setiap kata kode C = $w_1 w_2 ... w_m w_{m+1} w_{m+2} ... w_n$ yang diterima menjadi 2 bagian yang sama sehingga diperoleh C' = $w_1 w_2 ... w_m$ dan C'' = $w_{m+1} w_{m+2} ... w_n$, C', C'' $\in Z_2^m$

   a. Diketahui C = 1100111000$\in Z_2^{10}$ dan n = 10 maka m = 5 sehingga C' = 11001 dan C'' = 11000

   b. Diketahui C = 1100011000 $\in Z_2^{10}$ dan n = 10 maka m = 5 sehingga C' = 11000 dan C'' = 11000

2. Melihat karakter setiap *C*’ dan *C*’’. Jika *C*’ = *C*’’ berarti setiap karakter pada *C*’ sama dengan setiap karakter pada *C*’’, maka tidak terjadi kesalahan pengkodean. Sedangkan *C*’ $\neq$ *C*’’ berarti terdapat 1 atau lebih karakter yang berbeda. Hal ini berarti telah terjadi kesalahan pengkodean.

   a. Hasil *C*’ $\neq$ *C*’’ karena karakter ke-5 dari *C*’ yaitu 1 tidak sama dengan ke-5 dari *C*’’ yaitu 0. $1 \neq 0$ maka telah terjadi kesalahan pengkodean.

   b. Hasil *C*’ = *C*’’ karena *C*’ = 11000 = *C*’’ maka tidak terjadi kesalahan dalam pengkodean.

3. Jika terjadi kesalahan maka tentukan kode yang seharusnya diterima dengan *C*’ sebagai acuan

   a. Kata kode yang diterima seharusnya adalah 1100111001 karena C’ sebagai acuan.

4. Melakukan proses decoding untuk memastikan kata kode yang sebenarnya

   a. $D : Z_2^{10} \rightarrow Z_2^5$ adalah $D(C) = w_1 w_2 ... w_5$, jika C = 1100111001 maka *D*(1100111001) = 11001.

5. Cek kata kode dengan menggunakan pohon biner optimal. Jika kode yang dimaksud memenuhi kata kode yang ada maka cari kata kode yang mendekati dengan kata kode yang dimaksud

   a. Dari pohon biner optimal diketahui T = 11001

   b. Dari pohon biner optimal diketahui U = 11000

BAB V

P E N U T U P

Dalam Bab V ini akan disajikan ringkasan dan saran – saran berhubungan dengan pembahasan yang telah dipaparkan dalam Bab sebelumnya.

5.1 Ringkasan

Ringkasan yang dapat diambil dari skripsi ini berkaitan dengan penerapan teori graf pada kode pengulangan adalah sebagai berikut

a. Encoding dengan cara menggunakan fungsi E : $Z_2^m \rightarrow Z_2^n$ dengan bantuan pohon biner optimal. Cara mengubah kalimat menjadi kata–kata kode dengan cara memberi bobot pada setiap hurufnya mulai dari yang terkecil sampai yang terbesar. Dengan menggunakan pohon biner optimal, out-degree ke kanan diberi nilai 1 dan out-degree ke kiri diberi nilai 0. Untuk mendapatkan kata kode dari huruf tersebut maka kita membuat lintasan tunggal dari akar menuju kebobot huruf yang sesuai. Setelah kata kode didapat maka kode tersebut kita masukan ke kode pengulangan dengan fungsi E : $Z_2^m \rightarrow Z_2^n$. Di mana $n = 2m$

b. Decoding (penguraian kata kode) dilakukan dengan cara membuat pohon biner optimal berdasarkan bobot yang telah disarankan dengan menggunakan Alogaritma Huffman. Dari pohon biner optimal ini didapat kata kode dengan cara membuat lintasan tunggal dari akar menuju kebobot huruf yang sesuai. Decoding D : $Z_2^m \rightarrow Z_2^n$ di mana $n =$

$2m$ maka fungsi decoding menjadi D : $Z_2^{2m} \rightarrow Z_2^n$. Sehingga kata kode pengulangan dibagi menjadi bagian yang sama bagian pertama itulah yang merupakan pesan sesungguhnya.

c. Pengoreksian Kata Kode

Pengoreksian dilakukan untuk mengatahui kebenaran suatu pesan. Proses pengoreksian berupa kode pengulangan yaitu E : $Z_2^m \rightarrow Z_2^{2m}$ dilakukan dengan cara membagi tiap kata kode menjadi 2 bagian yang sama sehingga diperoleh C' dan C''. Jika mulai C' dan C'' adalah sama maka kode tersebut sudah benar. Jika C' dan C'' tidak sama maka terjadi kesalahan. Untuk mengetahui kebenaran kode tersebut maka dilakukan proses decoding dan pengecekan dengan menggunakan pohon biner optimal selanjutnya dipilih kata kode yang mendekati kode yang dimaksud.

5.2. Saran

Sesuai dengan ringkasan di atas diharapkan skripsi ini dapat digunakan untuk mengetahui bagaimana cara merubah suatu pesan dalam bentuk kata kode pengulangan. Selain itu dapat bermanfaat bagi penulis maupun pembaca untuk mengetahui penggunaan graf berarah dalam masalah pengkodean yang berbentuk kode pengulangan. Khususnya dapat memahami masalah encoding dan decoding serta penerapan graf berarah dalam kode pengulangan, sehingga diharapkan pembaca dapat mengirim suatu pesan dengan menggunakan kata kode yang berbentuk pengulangan.

DAFTAR RUJUKAN

Grimaldi, Ralph. 1985. *Discrete And Combinatorial Mathematics An Applied Introduction*. Addison Wesley. Reading Mass.

Hankerson, D. dkk. 2000. *Coding Theory And Cryptograhy The Essentials*. New York. Penerbit Marce Deker,Inc.

Lipshutz, Seymour. 2002. *Matematika Diskrit Jilid 2*. Jakarta. Penerbit Salemba Teknika.

Wikipedia. *PohonBerakar*. http://id.wikipedia.org/wiki/pohon_%28struktur_data%29. [26 November 2007].

Wilson, Robin J Dan John J Wat Kins. 1992. *Graf Pengantar 1*. Terjemahan Theresia M.H Tirta. Surabaya. Penerbit University Press IKIP Surabaya.

Wilson, Robin J Dan John J Wat Kins. 1992. *Graf Pengantar 2*. Terjemahan Theresia M.H Tirta.Surabaya. Penerbit University Press IKIP Surabaya.