ALGORITMA HUFFMAN KANONIK UNTUK KOMPRESI TEKS SMS

Moch Ginanjar Busiri 13513041
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13513041@std.stei.itb.ac.id

***Abstract*— Algoritma Huffman adalah salah satu algoritma kompresi. Algoritma huffman merupakan algoritma yang paling terkenal untuk mengompres teks. Terdapat tiga fase dalam menggunakan algoritma Huffman untuk mengompres sebuah teks, pertama adalah fase pembentukan pohon Huffman, kedua fase encoding dan ketiga fase decoding. Prinsip yang digunakan oleh algoritma Huffman adalah karakter yang sering muncul di -encoding dengan rangkaian bit yang pendek dan karakter yang jarang muncul diencoding dengan rangkaian bit yang lebih panjang. Teknik kompresi algoritma Huffman mampu memberikan penghematan pemakaian memori sampai 30%. Algoritma Huffman mempunyai kompleksitas O(n log n) untuk himpunan dengan n karakter.**

Kata kunci: pohon biner, metode Huffman, metode Kanonik Huffman, encoding, decoding.

1. Pendahuluan

Short Message Service (SMS) merupakan salah satu aplikasi dari teknologi *sellular* yang paling banyak diminati, karena *user* dapat berkomunikasi dengan mengirimkan pesan-pesan pendek tanpa perlu khawatir akan biayanya dan waktu penyampainya pun relatif cepat. SMS adalah sebuah pesan yang dikirimkan dari *sender* ke *receiver* dengan panjang karakter maksimal adalah 160 karakter untuk setiap satu SMS. Untuk mengirimkan pesan yang lebih dari 160 karakter, maka biaya yang dikeluarkan pun berlipat sesuai jumlah karakter SMS. Dalam menulis pesan SMS, seorang pengguna biasa melakukannya dengan cara menyingkat isi pesan tersebut. Hal ini dilakukan selain karena kesulitan atau malas untuk mengetikkan isi pesan juga untuk menghemat tempat, sehingga terkadang, isi pesan yang ada, harus di edit untuk menyesuaikan agar dapat termuat dalam satu halaman SMS dan tentu akan merepotkan para pengguna ketika harus melakukan pengiriman SMS, sehingga perlu dibuat suatu aplikasi untuk menambah pemuatan karakter SMS yang akan dikirimkan dalam satu halaman, dengan cara mengkompresi isi pesan atau teks SMS tersebut. Kompresi merupakan proses pengubahan sekumpulan data menjadi suatu bentuk kode untuk menghemat kebutuhan tempat penyimpanan dan waktu untuk transmisi data. Contoh kompresi sederhana yang biasa dilakukan misalnya adalah menyingkat kata-kata yang sering digunakan tapi sudah memiliki konvensi umum. Misalnya kata “yang” dikompres menjadi kata “yg”. Saat ini terdapat berbagai tipe algoritma kompresi, antara lain: *Huffman*, *LZ77* dan variannya *(LZ78, LZW, GZIP), Dynamic Markov Compression (DMC), Run-Length, Shannon-Fano* dan lain-lain. Kompresi data menjadi sangat penting karena dapat memperkecil kebutuhan penyimpanan data, mempercepat pengiriman data, dan memperkecil kebutuhan *bandwidth*. Proses kompresi terhadap teks SMS dilakukan dengan cara melakukan proses encoding dan decoding terhadap data berupa teks SMS. Encoding data dilakukan dengan cara menyusun teks yang ada (mengkodekan informasi) menggunakan informasi lain yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu sistem encoding tertentu. Proses decoding merupakan kebalikan dari encoding yang berarti menyusun kembali data dari hasil encoding menjadi sebuah karakter kembali. Proses encoding dilakukan terhadap data SMS yang akan dikirimkan ke *handphone* tujuan sedangkan decoding dilakukan terhadap data SMS yang diterima oleh *handphone* penerima. Dengan adanya proses kompresi terhadap teks SMS maka akan terjadi pemampatan terhadap data SMS sehingga dapat menghemat biaya pengiriman SMS (pulsa). Selain itu dengan adanya proses encoding dan decoding terhadap teks SMS maka hal ini dapat memproteksi isi pesan SMS ketika terjadi penyadapan. Hal ini terjadi karena isi pesan akan sulit dibaca jika pesan tersebut belum dilakukan proses decoding.

2. Landasan Teori

2.1 Kompresi Data

Kompresi data merupakan cabang dari Teori Informasi. Teori Informasi sendiri adalah salah satu cabang Matematika yang berkembang sekitar akhir 1940an. Tokoh utama dari Teori Informasi adalah Claude Shannon dari Bell *Laboratory*. Teori Informasi memfokuskan pada berbagai metode tentang informasi termasuk penyimpanan dan pemrosesan pesan. Teori Informasi mempelajari pula tentang *redundancy* pada pesan. Semakin banyak *redundancy* semakin besar pula ukuran pesan, upaya mengurangi *redundancy* ini yang akhirnya melahirkan subyek ilmu tentang kompresi data. Kompresi data merupakan sebuah cara untuk memadatkan data sehingga hanya memerlukan ruangan penyimpanan lebih kecil sehingga lebih efisien dalam penyimpanannya dan mempersingkat waktu pertukaran data tersebut. Keuntungan kompresi data adalah penghematan tempat pada media penyimpanan dan penghematan *bandwidth* pada pengiriman data. Namun kompresi data juga

memiliki sisi negatif, bila data yang terkompresi ingin dibaca, perlu dilakukan dekompresi terlebih dahulu. Pada pengiriman data, penerima data harus mengerti proses decoding dalam data terkompresi yang diterima atau dengan kata lain, penerima juga harus memiliki perangkat lunak yang sama dengan pengirim sehingga dapat melakukan proses decoding tersebut.
Berikut ini adalah beberapa teknik kompresi data yang dikategorikan menurut jenis data yang akan dikompresi, yaitu:

1. Teknik kompresi untuk citra diam (*still image*)
Contoh: JPEG, GIF dan *run-length*.

2. Teknik kompresi untuk citra bergerak (*motion picture*)
Contoh: MPEG.

3. Teknik kompresi untuk data teks
Contoh: *half byte.*

4. Teknik kompresi untuk data umum
Contoh: LZW, *half byte* dan *Huffman.*

5. Teknik kompresi untuk data sinyal *speech*
Contoh: PCM, SBC, LPC dan CELP.

Ada empat pendekatan yang digunakan pada kompresi suatu data, yaitu:

1. Pendekatan statistik
Kompresi didasarkan pada frekuensi kemunculan derajat keabuan *pixel* didalam seluruh bagian.
Contoh: *Huffman coding*.

2. Pendekatan ruang
Kompresi didasarkan pada hubungan spasial antara *pixel-pixel* didalam suatu kelompok yang memiliki derajat keabuan yang sama dalam suatu daerah gambar atau data.
Contoh: *Run-Length encoding*

3. Pendekatan kuantisasi
Kompresi dilakukan dengan mengurangi jumlah derajat keabuan yang tersedia.
Contoh: Kompresi kuantisasi (CS&Q)

4. Pendekatan Fraktal
Kompresi dilakukan pada kenyataan bahwa kemiripan bagian-bagian di dalam data atau citra atau gambar dapat dieksploitasi dengan suatu matriks transformasi
Contoh: *Fractal image compression*.

Berdasarkan Output, Algoritma Kompresi Data dapat diklasifikasikan menjadi dua jenis, yaitu:

1. Algoritma Kompresi *Lossless*

2. Algoritma Kompresi *Lossy*

2.1.1 Algoritma Kompresi *Lossless*

Lossless data kompresi adalah kelas dari algoritma data kompresi yang memungkinkan data yang asli dapat disusun kembali dari data kompresi. *Lossless* data kompresi digunakan dalam berbagai aplikasi seperti format ZIP dan GZIP. *Lossless* juga sering digunakan sebagai komponen dalam teknologi kompresi data *lossy*. Kompresi *Lossless* digunakan ketika sesuatu yang penting pada kondisi asli. Beberapa format gambar sperti PNG atau GIF hanya menggunakan kompresi *lossless*, sedangkan yang lainnya seperti TIFF dan MNG dapat menggunakan Algoritma *lossy* atau *lossless*. Algoritma *lossless* menghasilkan data yang identik dengan data aslinya, hal ini dibutuhkan untuk banyak tipe data, contohnya: *executable code*, *word processing files*, *tabulated numbers*, dan sebagainya. Misalnya pada citra atau gambar dimana Algoritma ini akan menghasilkan hasil yang tepat sama dengan citra semula, *pixel* per *pixel* sehingga tidak ada informasi yang hilang akibat kompresi. Namun ratio kompresi (Rasio kompresi yaitu, ukuran file yang dikompresi dibanding yang tak terkompresi dari file) dengan Algoritma ini sangat rendah. Algoritma ini cocok untuk kompresi citra yang mengandung informasi penting yang tidak boleh rusak akibat kompresi, misalnya gambar hasil diagnosa medis. Contoh Algoritma *lossless* adalah metode *run-length*, *Huffman*, *delta* dan *LZW*. Kompresi *Lossless* dibagi menjadi 2 yaitu :

1. Algoritma berbasis *Entropi*
Algoritma berbasis *Entropi*, atau disebut juga berbasis statistik, menggunakan model statistik dan probabilitas untuk memodelkan data, keefisienan kompresi bergantung kepada berapa banyak karakter yang digunakan dan seberapa besar distribusi probabilitas pada data asli. Contoh algoritma yang berbasis entropi adalah: *Huffman Coding*, *Adaptive Huffman*, dan *Shannon Fano*, *Run Length*, Burrows wheeler transform

2. Algoritma berbasis *Dictionary*
Algoritma berbasis *dictionary*, bekerja dengan cara menyimpan pola masukan sebelumnya, dan menggunakan *index* dalam mengakses pola tersebut jika terdapat perulangan. Contoh algoritma yang berbasis *dictionary* adalah: LZ77, LZ78, LZW, DEFLATE, dan LZMA.

2.1.2 Algoritma Kompresi *Lossy*

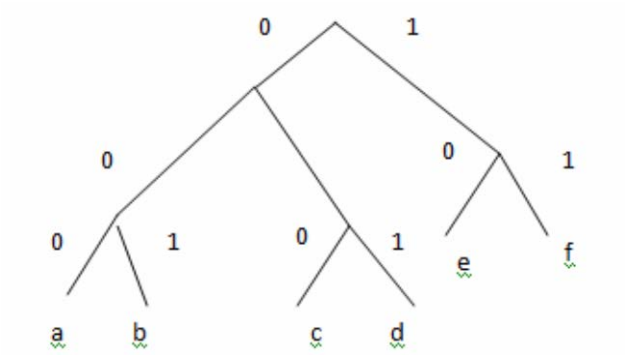
Lossy kompresi adalah suatu Algoritma untuk mengkompresi data dan men-dekompresinya, data yang diperoleh mungkin berbeda dari yang aslinya tetapi cukup dekat perbedaaanya. *Lossy* kompresi ini paling sering digunakan untuk kompres data multimedia (Audio, gambar diam). Sebaliknya, kompresi *lossless* diperlukan untuk data teks dan file, seperti catatan bank, artikel teks dll. Format kompresi *lossy* mengalami *generation loss* yaitu jika melakukan berulang kali kompresi dan dekompresi file akan menyebabkan kehilangan kualitas secara progresif. hal ini berbeda dengan kompresi data *lossless*. ketika pengguna yang menerima file terkompresi secara lossy (misalnya untuk mengurangi waktu *download*) file yang diambil dapat sedikit berbeda dari

yang asli dilevel bit ketika tidak dapat dibedakan oleh mata dan telinga manusia untuk tujuan paling praktis. Algoritma ini menghasilkan ratio kompresi yang lebih besar daripada Algoritma *lossless*. Misal terdapat *image* asli berukuran 12,249 bytes, kemudian dilakukan kompresi dengan JPEG kualitas 30 dan berukuran 1,869 bytes berarti image tersebut 85% lebih kecil dan ratio kompresi 15%. Contoh Algoritma *lossy* adalah Algoritma CS&Q (*coarser sampling and/or quantization*), JPEG, dan MPEG. Ada dua skema dasar *lossy* kompresi :

1. *Lossy Transform Codec*
Sampel suara atau gambar yang diambil, di potong kesegmen kecil, diubah menjadi ruang basis yang baru, dan kuantisasi. hasil nilai kuantisasi menjadi *entropy coded*.

2. *Lossy Predictive Codec*
Sebelum atau sesudahnya data di-*decode* digunakan untuk memprediksi sampel suara dan *frame picture* saat ini. kesalahan antara data prediksi dan data yang nyata, bersama-sama dengan informasi lain digunakan untuk mereproduksi prediksi, dan kemudian dikuantisasi dan kode.

2.2 Kode *Huffman* Kanonik
Kode *Huffman* kanonik merupakan suatu bentuk variasi dari kode *Huffman* yang memiliki sifat dapat dideskripsikan dengan sangat kompres. Hal ini disebabkan kode *Huffman* kanonik memiliki aturan-aturan penulisan yang baku sehingga beberapa hal dapat disepakati dan tidak perlu dituliskan pada deskripsi kode. Contohnya adalah kode harus terurut berdasarkan urutan simbol, panjang kode suatu simbol harus sesuai dengan aras simpul simbol tersebut pada pohon *Huffman*. Deskripsi kode yang *compact* membuat kode ini lebih efisien dalam beberapa hal dibandingkan dengan kode *Huffman*.



Gambar Pohon Huffman Kanonik

Simbol	Kode
A	000
B	001
C	010
D	011
E	10
F	11

Tabel Kode *Huffman* Kanonik dari pohon *Huffman*

2.2.1 Proses Encoding
Aturan-aturan proses encoding pada algoritma Huffman Kanonik adalah sebagai berikut:

1. Panjang kode untuk suatu simpul adalah sebesar aras+1 simpul tersebut.

2. String biner simpul paling dalam yang terletak paling kiri diberi nilai 0 semuanya. Untuk simpul berikutnya(bergeser dari kiri ke kanan) string binernya naik satu nilai dari simpul sebelumnya.

3. Apabila semua simpul pada kedalaman yang sama telah di-encode, maka proses encoding dilanjutkan ke aras yang lebih rendah. Hanya saja string binernya tidak dimulai dengan semuanya 0. String biner simpul paling kiri dimulai dengan kode baru. Kode baru itu merupakan kenaikan 1 nilai dari string biner simpul yang terakhir di-encode namun biner paling belakangnya dihilangkan sehingga panjang kodenya berkurang satu. Simpul berikutnya di-encode dengan string binernya naik satu nilai(bergeser dari kiri ke kanan).

4. Proses berhenti bila telah mencapai akar.

2.2.2 Proses Decoding
Proses decoding memiliki beberapa cara alternatif.

1. Alternatif pertama adalah dengan membuat tabel yang berisi karakter yang di-decode dan jumlah bit yang digunakan. Alternatif pertama ini cocok digunakan pada data-data dengan panjang kode maksimum yang pendek karena ukuran tabel sebanding dengan panjang kode.

2. Alternatif kedua adalah membuat tabel untuk setiap panjang kode. Lalu tabel yang sesuai akan dicari untuk setiap input yang dimasukkan (tabel ini adalah tabel yang dibentuk menggunakan algoritma yang ada pada proses encoding).

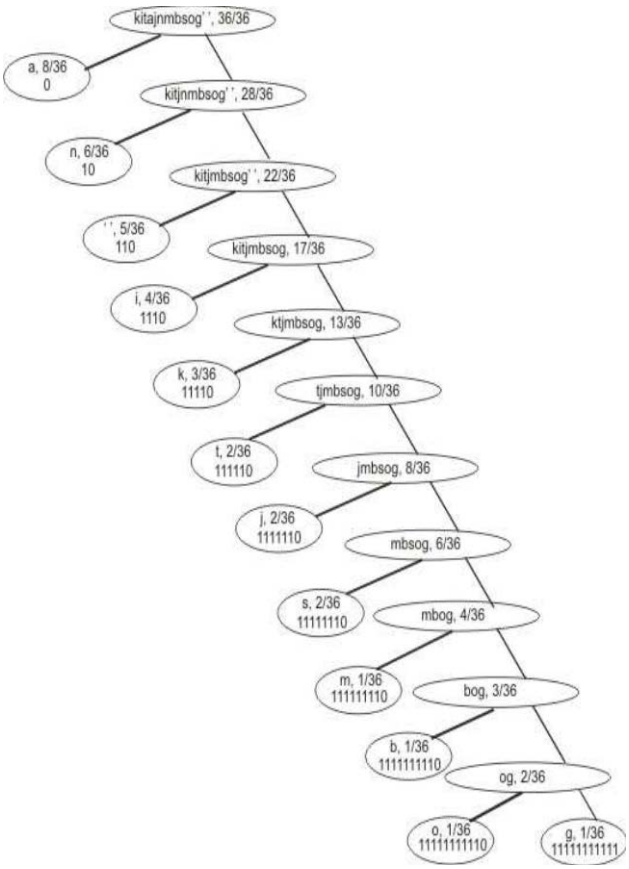
3. Alternatif ketiga adalah dengan membuat multilevel tabel. Pertama-tama dicek beberapa bit kode pertama, lalu tabel yang akan dipakai diberitahu. Kemudian pada tabel berikutnya akan dicari karakter yang sesuai berdasarkan panjang kodenya.

3. Analisis
Dalam analisis ini kita akan mencoba melakukan encoding sebuah string karakter (suatu teks SMS) menggunakan teknik dasar dan menggunakan Canonical Huffman Code kemudian membandingkan efisiensi antara keduanya. Misalkan kita memiliki sebuah data berupa sebuah string (SMS) karakter berikut "*kita janjian makan bakso nanti sian*" Kemudian melakukan satu kali pembacaan untuk mengitung kemunculan dari masing-masing karakter:

Karakter	Jumlah
k	3
i	4
t	2
a	8
j	2
n	6
m	1
b	1
s	2
o	1
g	1
‘ ‘ (spasi)	5
Total karakter	36

Tabel jumlah kemunculan masing-masing karakter dalam string

1. Menggunakan teknik dasar
Pertama kita membangun sebuah pohon Huffman menggunakan cara biasa :



Gambar Pohon biner Huffman didapatkan menggunakan teknik dasar

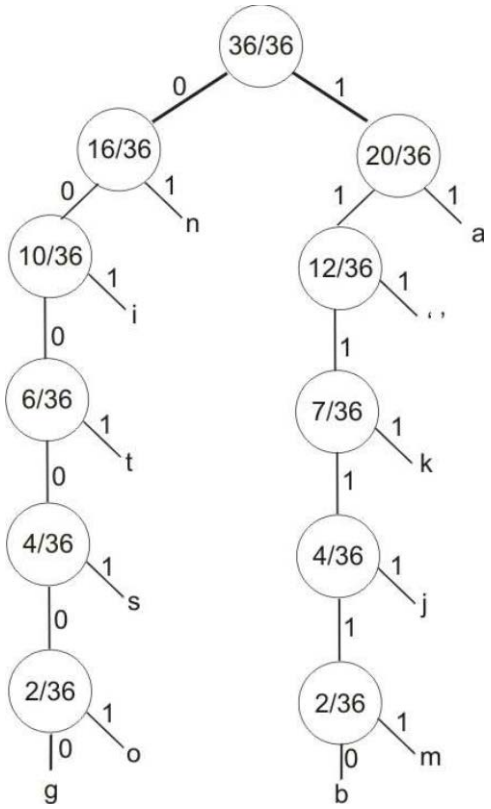
Kemudian coba ubah string asal menjadi kode dalam bit Huffman yang sudah didapat dari pohon Huffman di atas:
11110111011111001101111110010111111101110010110
11111110011110010110111111111001111011111111011
11111110110100101111110111011011111111011100101
1111111111

Dengan jumlah bit : 149

Setelah mengetahui jumlah bit hasil kompresi, kita bandingkan rasio antara hasil kompresi dengan jumlah bit asal

rasio = 149/288 = 51,73%

2. Menggunakan Canonical Huffman Code
Menggunakan string yang sama, kita akan membangun sebuah pohon Huffman menggunakan Canonical Huffman Code.



Gambar Pohon biner Huffman didapatkan menggunakan Canonical Huffman Code

Kemudian mencoba untuk mengubah string asal ke kode bit Huffman yang baru saja kita dapatkan :
11110010001111111111111011111100111011111111111
11111111011111111110101111000010000011110111010
01001000010011101000000

Dengan jumlah bit : 119

Sama seperti sebelumnya, kita bandingkan rasio antara hasil kompresi dengan jumlah bit asal

rasio = 119/288 = 41,32%

4. Kesimpulan

1. Algoritma Huffman merupakan salah satu algoritma kompresi teks yang cukup efisien, dengan rasio minimal berkisar pada 50% (dalam analisis didapatkan rasio yang melebihi 50% karena banyaknya karakter dengan jumlah kemunculan yang sama dan kita memberikan kode Huffman yang berbeda untuk masing-masing karakter tersebut).

2. Jika dibandingkan, algoritma Huffman menggunakan Canonical Huffman Code menghasilkan hasil kompresi yang lebih efisien daripada metode menggunakan teknik dasar. Hal ini dapat diamati dari bentuk pohon kode Huffman untuk setiap metode. Pada pohon hasil pengodean menggunakan metode teknik dasar, setiap karakter memiliki jumlah representasi bit yang sama (kecuali dua karakter dengan jumlah kemunculan yang paling sedikit), perbedaan jumlah bit untuk masing-masing karakter ini juga menyebabkan tingginya pohon yang terbentuk (dalam contoh yang dianalisis pohon yang terbentuk memiliki 11 aras). Sedangkan, pada pohon hasil pengodean menggunakan Canonical Huffman dapat dilihat setiap karakter pada level yang sama direpresentasikan dengan jumlah bit yang sama, sehingga memang ada karakter-karakter yang direpresentasikan dengan jumlah bit yang sama tapi tetap unik. Dengan adanya karakter yang direpresentasikan dengan jumlah bit sama ini pohon yang terbentuk pun ‘lebih lebar’ dan memiliki aras yang lebih rendah (dalam contoh ada 6 aras).

DAFTAR PUSTAKA

[1]. Raharjo Budi, Haryanto Imam dan HaryonoArif, 2007, “ ***Tuntunan Pemrograman Java Untuk Handphone”***, Bandung, Informatika.
(6 Desember 2014)

[2]. Sholahudin, M., Rosa, A. S., 2006, “***Pemrograman J2ME Belajar Cepat Pemrograman Perangkat Telekomunikasi Mobile”***, Bandung, Informatika.
(6 Desember 2014)

[3]. Tim Penelitian Dan Pengembangan Wahana Komputer, 2005, “ ***Pengembangan Aplikasi Sistem Informasi Akademik Berbasis SMS Dengan Java”***, Jakarta, Salemba Infotek.
(7 Desember 2014)

[4]. Wayan Name Astagina (2010). Pengantar Kompresi Data
(7 Desember 2014)

[5]. Rd. Aditya Satrya Wibawa, 2009, p.4
(8 Desember 2014)

[6]. Second Edition 1999, “***Managing Gigabytes*** Compressing and Indexing Documents and Images”
(8 Desember 2014)

[7]. Donald E. Knuth, 1985, "***Dynamic Huffman Coding***", Journal of Algorithm, 6(2).
(8 Desember 2014)

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Desember 2014

Moch Ginanjar Busiri 13513041