Kompresi Data dengan Kode Huffman dan Variasinya

I.Y.B. Aditya Eka Prabawa W.
Teknik Informatika Institut Teknologi Bandung, Bandung 40116, email: aditya_eka@students.itb.ac.id

Abstract *– Makalah ini membahas definisi, jenis-jenis dan beberapa teori kompresi data secara umum. Terutama metode kompresi dengan menggunakan kode Huffman, serta aplikasi konsep pohon dalam pembentukan kode Huffman. Selain itu, juga dibahas secara singkat aplikasi kompresi data dengan Kode Huffman dan beberapa variasi kode Huffman dalam teori kompresi data.*

Kata Kunci*: kompresi data, lossy, lossless, kode huffman, pohon huffman, variasi huffman.*

1. PENDAHULUAN

Pada dasarnya, data apapun sebenarnya merupakan rangkaian *bit* 0 dan 1. Yang membedakan antara suatu data tertentu dengan data yang lain adalah ukuran dari rangkaian bit dan bagaimana 0 dan 1 itu ditempatkan dalam rangkaian bit tersebut. Misalnya data berupa teks dan data yang berupa gambar, dalam data teks suatu rangkaian bit tertentu mewakili satu karakter, sedangkan dalam data gambar suatu rangkaian bit mewakili suatu warna dalam satu *pixel*. Semakin kompleks suatu data, ukuran rangkaian bit yang diperlukan semakin panjang, dengan demikian ukuran keseluruhan data juga semakin besar.

Dalam penyimpanan dan pengiriman data komputer, selain isi dari data tersebut parameter yang tidak kalah pentingnya adalah ukurannya. Kerap kali data yang disimpan dalam suatu media penyimpanan berukuran sangat besar sehingga memerlukan tempat yang lebih banyak dan tidak efisien. Apalagi bila data tersebut akan dikirim, semakin besar ukurannya, waktu yang diperlukan untuk pengiriman akan lebih lama. Untuk itu, diperlukan kompresi data (*data compression*) untuk memperkecil ukuran suatu data tanpa merubah isi atau informasi yang terkandung dalam data tersebut.

Ada banyak sekali teori dan metode untuk kompresi data. Salah satu teori yang cukup sederhana adalah dengan menggunakan kode Huffman (*Huffman coding*). Dalam enkoding kode Huffman, digunakan konsep struktur data pohon biner (*binary tree*). Teori kode Huffman ini sendiri tidak hanya ada satu, tetapi ada beberapa variasi, optimasi, dan kombinasinya.

2. KOMPRESI DATA

2.1. Definisi Kompresi Data

Kompresi data adalah proses enkoding suatu informasi dalam data menjadi rangkaian bit yang lebih pendek daripada rangkaian bit yang diperlukan data tersebut sebelum dikompresi dengan menggunakan metode enkoding tertentu.

Keuntungan kompresi data adalah penghematan tempat pada media penyimpanan (misalnya *Hard Disk* dan DVD (*Digital Versatile Disc*)), dan penghematan *bandwidth* (lebar pita) pada pengiriman data. Namun kompresi data juga memiliki sisi negatif. Bila data yang terkompresi ingin dibaca, perlu dilakukan proses dekompresi terlebih dahulu, pada pengiriman data, penerima data harus mengerti proses enkoding dalam data terkompresi yang dikirim (memiliki perangkat lunak (*software*) yang dapat mendekompresinya) [3].

Beberapa ekstensi file individual yang umum dijumpai sebenarnya sudah terkompresi. Misalnya file gambar: .PNG (*Portable Network Graphic*), .GIF (*Graphic Interchange Format*), .JPEG (*Joint Photographic Experts Group*); File *audio* dan *video*: .MP3 (MPEG-1 *Audio Layer* 3), .AAC (*Advanced Audio Coding*) sebagai penerus MP3, dan .MPEG (*Moving Picture Experts Group*). File-file Microsoft® Office 2007: (.DOCX, XLSX, PPTX); Memiliki kompatibilitas yang rendah, namun format-format yang baru ini memiliki kompresi yang lebih baik dibandingkan dengan format-format yang lama (.DOC, .XLS, .PPT). Ketika file-file tersebut dibuka, sebenarnya program akan mendekompresinya terlebih dahulu, kemudian membaca isi file tersebut.

Sebaigan besar format file kompresi juga merangkap sebagai *archiver* (dapat menyimpan banyak file input ke dalam satu file output yang sudah terkompresi). Format-format file *archiver* yang umum dijumpai antara lain: .ZIP, .RAR, .CAB, .ISO, .NRG, .ACE, .BIN, .TAR, .GZ, .BZ2, .ARC, .7Z, dan banyak format-format lainnya. Untuk membangun file-file tersebut, biasanya digunakan sebuah perangkat lunak *archiver*. Ada banyak sekali perangkat lunak semacam ini, ada beberapa yang berbayar, namun tidak sedikit yang gratis. Perangkat lunak yang populer digunakan untuk membangun file-file *archiver* di lingkungan sistem operasi Windows™ diantaranya: IZArc, WinAce, WinRar, dan WinZip®:

Gambar 1 : WinZip® merupakan perangkat lunak yang populer digunakan untuk kompresi data di lingkungan Windows™

2.2. Jenis-jenis Kompresi Data

2.2.1. *Lossy Data Compression*

Dalam kompresi jenis ini, data yang terkompresi tidak dapat dikembalikan (didekompres) menjadi data yang persis sama seperti sebelum dikompres (*irreversible*), karena ada informasi-informasi tertentu yang 'dibuang'. Kompresi jenis ini dapat mengurangi ukuran data secara signifikan, dan data yang sudah terkompresi dapat dikompres lagi sampai batas-batas tertentu.

Lossy data compression sangat banyak digunakan dalam bidang *Multimedia* karena sering digunakan untuk mengurangi ukuran data yang bersifat *audio – visual*. Misalnya untuk memperkecil ukuran gambar pada kamera digital, kompresi dalam enkoding musik digital, dan untuk enkoding film dari format *High Definition* seperti Blu-Ray menjadi format MPEG (*Moving Picture Experts Group*) seperti DVD (*Digital Versatile Disc*), dengan demikian kualitasnya dikurangi, selama masih dalam batas toleransi.

Kompresi jenis ini masih banyak yang menerapkan prinsip Kode Huffman, terutama sebagai langkah akhir proses kompresi.

2.2.2. *Lossless Data Compression*

Tidak seperti *lossy data compression*, dalam kompresi jenis ini, data yang terkompresi dapat dikembalikan menjadi data yang persis sama seperti sebelum dikompres (*reversible*), karena dengan metode ini informasi yang dibuang hanya informasi-informasi yang tidak perlu (*redundant*). Misalnya pengulangan karakter yang sama pada suatu file teks. Kompresi jenis ini tidak selalu dapat mengurangi ukuran data secara berarti, karena tidak semua data mengandung informasi yang redundan. Selain itu, dengan *lossless data compression*, data yang sudah terkompresi tidak dapat dikompres lagi. Biasanya mengompres data yang sudah terkompresi malah akan membengkakkan ukuran data.

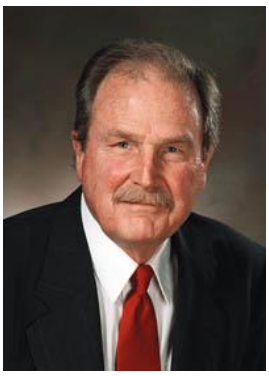
Lossless data compression digunakan untuk mengurangi ukuran data-data yang penting, data-data yang isinya tidak boleh berubah sedikitpun. Misalnya data-data berbentuk dokumen, teks, *spreadsheet*, kode sumber (*source code*), dan data-data program. File-file *archiver* juga menggunakan kompresi data jenis ini.

Salah satu teori yang juga dapat digunakan untuk *lossless data compression* adalah kode Huffman.

3. KOMPRESI DATA DENGAN KODE HUFFMAN

3.1. Penentuan Kode Huffman

Salah satu teori yang dapat digunakan untuk mengompresi data adalah dengan kode Huffman. Kode ini dikemukakan oleh David A. Huffman, seorang doktor teori informasi (*information theory*) lulusan MIT (*Massachusets Institute of Technology*) pada tahun 1952.



Gambar 2 : David A. Huffman (1925-1999)

Dalam kompresi data, kode Huffman adalah kode-kode biner yang mengodekan suatu simbol tertentu pada suatu data. Kode-kode tersebut dibentuk dengan memperhatikan frekuensi kemunculan simbol tertentu pada data tersebut. Kode Huffman tidak bersifat unik, kode untuk setiap simbol berbeda pada setiap data berbeda yang dikompres [2].

Dalam pembentukannya, Kode Huffman menerapkan konsep kode awalan (*prefix code*), yang merupakan himpunan kode biner, sedemikian sehingga tidak ada anggota himpunan yang merupakan awalan dari anggota yang lain, supaya pada proses dekoding, tidak ada keambiguan antara satu simbol dengan simbol yang lain. Kode awalan yang merepresentasikan simbol yang lebih sering muncul menggunakan rangkaian biner yang lebih pendek daripada kode yang digunakan untuk merepresentasikan simbol yang lebih jarang muncul. Dengan demikian jumlah bit yang digunakan untuk menyimpan informasi pada suatu data bisa lebih pendek [1].

Urutan algoritma untuk membentuk kode Huffman adalah sebagai berikut:

1. Mula-mula dihitung terlebih dahulu frekuensi kemunculan tiap simbol di dalam data.
2. Pembentukan kode Huffman dilakukan dengan membangun pohon biner dengan panjang lintasan berbobot minimum, (yang dinamakan pohon Huffman):

    a. Pertama-tama dipilih dua simbol dengan peluang kemunculan paling kecil (terdapat dengan jumlah paling sedikit di dalam data).
    b. Kedua simbol tadi digabungkan membentuk simpul orang tua dari kedua simbol itu sendiri, dengan peluang kemunculan sebesar jumlah dari peluang kemunculan kedua simbol itu.
    c. Simbol baru ini diperlakukan sebagai simpul baru dan diperhitungkan dalam mencari simbol selanjutnya yang memiliki peluang kemunculan terkecil.
    d. Kemudian, dipilih dua simbol lainnya yang

juga memiliki peluang kemunculan paling kecil (termasuk simbol yang baru dibuat).

e. Prosedur yang sama dilakukan pada dua simbol berikutnya yang mempunyai peluang kemunculan terkecil.
f. Langkah nomor 2 diluangi terus sampai semua simbol habis dibuat pohon biner.

3. Daun pada pohon Huffman menyatakan simbol yang terdapat di dalam data yang dikompres.
4. Setiap simbol dikodekan dengan memberi label 0 pada setiap cabang kiri pohon biner dan label 1 untuk setiap cabang kanannya.
5. Dibuat lintasan dari akar ke daun, sambil membaca label 0 atau 1 yang terdapat pada setiap cabang.
6. Kode Huffman untuk simbol pada suatu daun adalah rangkaian biner yang dibaca dari akar hingga daun yang bersangkutan.

3.2. Contoh Sederhana Penentuan Kode Huffman

Misalnya terdapat data berupa teks yang isinya sebagai berikut: "Selamat Natal *2008*". Kode ASCII (*American Standard Code for Information Interchange*) dari karakter-karakter pada string tersebut diberikan pada Tabel 1 berikut ini:

Tabel 1. Kode ASCII

Karakter	Kode ASCII (Biner)
'S'	0101 0011
'e'	0110 0101
'l'	0110 1100
'a'	0110 0001
'm'	0110 1101
't'	0111 0100
'N'	0100 1110
'2'	0011 0010
'0'	0011 0000
'8'	0011 1000
' ' (spasi)	0010 0000
'*'	0010 1010

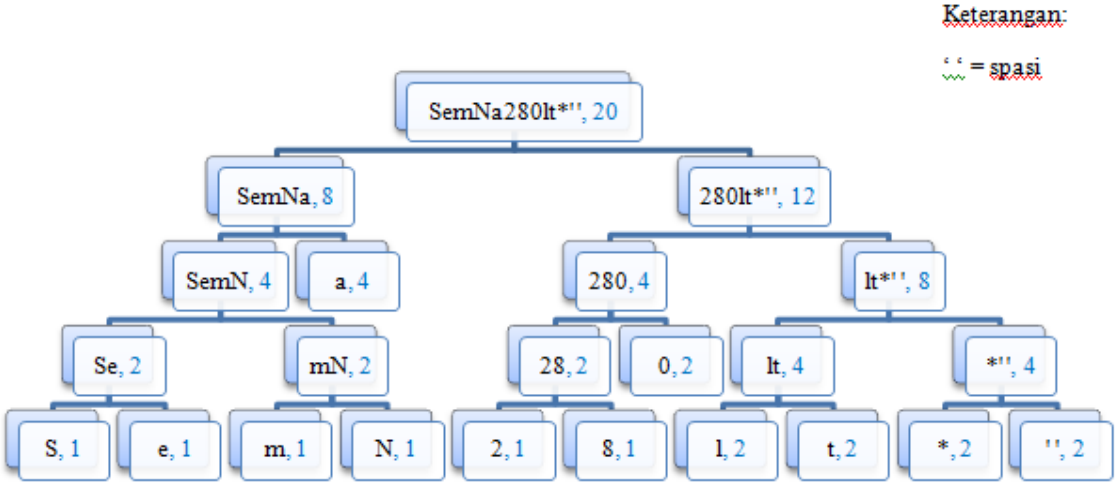
Dengan menggunakan kode ASCII, string tersebut direpresentasikan menjadi rangkaian bit:
"01010011011001010110110001100001011011010110000010111010000100000010011100110000101110100000110000010111000010110110000100000000101010001100100011000000110000001110000010101010".

Dengan menggunakan metode pengkodean ASCII, dibutuhkan 160 bit (20 byte) untuk menyimpan string tersebut. Teks tersebut akan dikompresi dengan Kode Huffman. Mula-mula dihitung terlebih dahulu frekuensi kemunculan tiap simbol di dalam string. Tabel frekuensi dan probabilitas kemunculan setiap simbol diberikan pada Tabel 2 berikut:

Tabel 2. Frekuensi dan probabilitas kemunculan karakter untuk string "Selamat Natal *2008*"

Karakter	Frekuensi	Probabilitas
'S'	1	1/20
'e'	1	1/20
'l'	2	2/20
'a'	4	4/20
'm'	1	1/20
't'	2	2/20
'N'	1	1/20
'2'	1	1/20
'0'	2	2/20
'8'	1	1/20
' ' (spasi)	2	2/20
'*'	2	2/20

Dengan menggunakan algoritma pembangunan pohon Huffman, dapat dibangun suatu pohon Huffman seperti pada Gambar 3:



Gambar 3 : Pohon Huffman untuk string "Selamat Natal *2008*"

Sehingga didapatkan kode Huffman untuk masing-masing karakter:

Tabel 3. Kode Huffman untuk string "Selamat Natal *2008*"

Karakter	**Kode Huffman**
'S'	0000
'e'	0001
'l'	1101
'a'	01
'm'	0001
't'	1101
'N'	0011
'2'	1000
'0'	101
'8'	1001
' ' (spasi)	1111
'*'	1110

Setelah dikompres dengan menggunakan Kode Huffman, string tersebut dapat direpresentasikan menjadi rangkaian bit:
"00000001110101000101110111110011011101010111011111111010001011011001111 0". (70 bit). Dengan rata-rata jumlah bit yang digunakan untuk enkoding satu karakter 3,1. Dengan demikian kompresi data teks sederhana itu telah 'menyelamatkan' 90 bit (atau 61,25% dari ukuran data). Ini adalah contoh kompresi data yang amat sangat sederhana, bila data yang dikompres jauh lebih besar, tentunya ukuran yang bisa diselamatkan juga jauh lebih besar.

3.3. Aplikasi Kompresi Data dengan Kode Huffman

Kode Huffman bukan metode kompresi dengan kinerja yang terbaik, namun demikian, Kode Huffman masih digunakan secara luas karena kesederhanaannya, kecepatannya yang tinggi, dan sedikitnya hak paten yang terkait dengannya.

Saat ini Kode Huffman sering digunakan sebagai lengkah akhir dari beberapa metode kompresi yang lain. Contoh-contoh implementasi Kode Huffman antara lain:

- Metode DEFLATE (kombinasi LZ77 dan Kode Huffman). Metode ini digunakan dalam format file .ZIP, .GZ (gzip), dan .PNG (*Portable Network Graphic*), dan .CAB (*Microsoft® Cabinet*).

- *Utility* '*pack*' pada sistem Unix. Dalam format file .Z.

- Kombinasi transformasi Burrows-Wheeler dan Kode Huffman. Dalam format file .BZ2 (bzip2).

- Kompresi gambar dengan format .JPEG (*Joint Photographic Experts Group*). Kompresi gambar ini menggunakan transformasi kosinus diskrit, lalu kuantisasi, kemudian diakhiri dengan Kode Huffman sebagai langkah akhir.

- Kompresi *audio* dengan format .MP3. Kompresi ini adalah bagian dari standar MPEG-1 untuk kompresi suara dan musik, menggunakan *subbanding*, MDCT (*Modified Discrete Cosine Transform*), *perceptual modeling*, kuantisasi, dan diakhiri dengan Kode Huffman sebagai langkah akhir.

- Kompresi *audio* dengan format .AAC (*Advanced Audio Coding*). Kompresi ini adalah bagian dari spesifikasi enkoding audio MPEG-2 dan MPEG-4 menggunakan MDCT (*Modified Discrete Cosine Transform*), *perceptual modeling*, kuantisasi, dan juga diakhiri dengan Kode Huffman sebagai langkah akhir.

- Selain dalam kompresi data, Kode Huffman yang telah dimodifikasi juga digunakan pada mesin fax untuk mengodekan warna hitam pada warna putih.

- HDTV (*High-Definition Television*) dan Modem (Modulator-Demodulator) juga menggunakan prinsip Kode Huffman.

3.4. Variasi Kode Huffman

Ada banyak variasi dari Kode Huffman, beberapa diantaranya menggunakan algoritma yang sama dengan algoritma pembentukan Kode Huffman, beberapa yang lain menemukan kode awalan yang optimal.

3.4.1. Kode Huffman n-ary

Algoritma Huffman n-ary menggunakan alfabet {0, 1, …, n - 1} untuk enkoding suatu pesan dan membuat pohon n –ary. Pendekatan ini adalah ide awal David Huffman. Algoritmanya sama dengan yang digunakan untuk Kode Huffman biner (n = 2), kecuali bahwa n buah simbol yang paling sedikit kemunculannya diambil secara bersamaan, tidak diambil 2 yang paling sedikit saja. Untuk n > 2, tidak semua himpunan kata-kata sumber dapat membentuk pohon n-ary untuk Kode Huffman dengan benar. Pada kasus ini, tambahan probabilitas 0 harus ditambahkan. Hal ini dilakukan karena pohonnya harus membentuk kontraktor n ke 1. Kode biner memiliki kontraktor 2 ke 1, himpunan kata berukuran berapapun dapat membentuk kontraktor ini.

3.4.2. Kode Huffman Adaptif

Variasi ini menghitung probabilitas secara dinamis berdasarkan frekuensi terakhir dari string sumber (adaptif).

3.4.3. Algoritma lain yang menggunakan dasar algoritma Huffman

Sering kali, bobot yang digunakan dalam implementasi Kode Huffman merepresentasikan probabilitas numerik, tapi contoh pembentukan Kode

Huffman di atas tidak demikian; yang diperlukan pada contoh di atas hanyalah suatu cara untuk mengurutkan bobot kemudian menjumlahkannya. Beberapa algoritma lain yang menggunakan dasar dari algoritma Huffman, bobotnya bisa apapun (misalnya harga, frekuensi, bobot sepasang, bobot yang tidak numerik) dan metode kombinasi yang lain (tidak hanya penjumlahan seperti contoh di atas). Algoritma semacam ini dapat menyelesaikan masalah minimalisasi yang lain, misalnya minimalisasi pada desain rangkaian digital.

3.4.4. Kode Huffman dengan panjang dibatasi

Ini adalah salah satu varian Kode Huffman yang tujuannya sama yaitu membangun pohon biner dengan panjang lintasan berbobot minimum, tetapi ada tambahan aturan, yaitu panjang Kode Huffman harus lebih kecil daripada konstanta tertentu.

3.4.5. Kode Huffman dengan bobot sumber tidak sama

Pada masalah penentuan Kode Huffman yang biasa, diasumsikan bahwa setiap simbol pada sumber memiliki bobot yang sama: kode biner dengan panjang N akan selalu berbobot N, tidak peduli berapa banyak digitnya yang 0 dan berapa banyak digitnya yang 1. Dengan asumsi seperti ini, meminimalisasi harga total suatu pesan sama dengan meminimalisasi jumlah digit total.

Kode Huffman dengan bobot sumber tidak sama adalah generalisasinya, di mana asumsi tersebut tidak berlaku lagi: simbol-simbol pada sumber panjangnya bisa berbeda-beda, karena adanya karakterisitk tertentu dari medium pengiriman. Contohnya adalah enkoding alfabet pada Kode Morse, di mana sebuah '-' (garis) memakan waktu pengiriman yang lebih lama daripada sebuah '.' (titik), dengan demikian bobot garis dalam pengiriman lebih besar. Tujuannya masih sama, yaitu meminimalisasi panjang kode rata-rata, tetapi tidak cukup hanya dengan meminimalisasi jumlah simbol yang digunakan dalam suatu pesan. Belum ada algoritma yang diketahui dapat menyelesaikan masalah ini dengan tingkat efisiensi yang sama dengan Kode Huffman yang konvensional.

3.4.6. Pohon biner alfabetis (Kode Hu-Tucker)

Pada masalah penentuan Kode Huffman yang biasa, diasumsikan bahwa setiap kode dapat mengacu pada simbol apapun pada sumber. Pada versi alfabetis ini, urutan alfabetis pada sumber dan hasil enkoding harus sama. Kode Hu-Tucker ini sering digunakan sebagai pohon pencarian biner (BST/*Binary Search Tree*).

3.4.7. Kode Huffman kanonik

Kode yang didapatkan dari sumber yang diurutkan kembali, yang disebut Kode Huffman kanonik ini adalah kode yang sering digunakan, karena kemudahannya dalam enkoding/dekoding. Teknik menentukan kode ini sering disebut Kode Huffman-Shannon-Fano, karena optimalitasnya seperti Kode Huffman dan probabilitas bobotnya yang alfabetis seperti Kode Shannon-Fano.
(P.S. Robert M. Fano adalah dosen David A. Huffman ketika ia mahasiswa doktoral; Sedangkan Claude Shannon adalah kolega Fano yang sekaligus penemu keilmuan teori informasi (*information theory*)).

4. KESIMPULAN

Kompresi data sangat diperlukan pada dunia komputasi, terutama dewasa ini di mana data-data menjadi semakin besar. Walaupun media penyimpanan dan *bandwidth* juga semakin membesar, tetap diperlukan suatu metode yang efektif untuk mengompres data.

Salah satu metode yang cukup sederhana untuk kompresi data adalah dengan Kode Huffman. Pembentukan kode Huffman sederhana dapat dilakukan dengan menentukan kode awalan menggunakan pohon biner dengan panjang lintasan berbobot minimum.

Kode Huffman hanya optimal untuk kompresi data yang berupa simbol-simbol dengan distribusi peluang yang diketahui, dan walapupun sekarang telah ditemukan metode kompresi yang lebih baik seperti Kode Aritmetik dan LZW (*Lempel-Ziv-Welch*), Kode Huffman tetap digunakan secara luas karena kesederhanaannya, kecepatannya yang tinggi, dan sedikitnya hak paten yang terkait dengan Kode Huffman.

Saat ini Kode Huffman masih sering digunakan sebagai langkah akhir kompresi data. Untuk menyempurnakannya, dibuat banyak sekali variasi dari Kode Huffman yang dibuat terspesialisasi untuk tipe-tipe file tertentu.

DAFTAR REFERENSI

[1] Ir. Rinaldi Munir, *Struktur Diskrit*, Program Studi Teknik Informatika, 2003.

[2] "Huffman coding", Wikipedia: http://en.wikipedia.org/wiki/Huffman_coding Waktu akses: 11 Desember 2008 11:14

[3] "Data compression", Wikipedia: http://en.wikipedia.org/wiki/Data_compression Waktu akses: 11 Desember 2008 11:16

[4] Soen I Siaw, *Belajar Sendiri Personal Computer*, Elex Media Komputindo, 1987.

[5] http://cats.ucsc,edu Waktu akses: 28 Desember 2008 20:42

[6] "Huffman Coding", Scientific American: http://www.huffmancoding.com Waktu akses: 28 Desember 2008 20:43