Pemampatan Data dengan Kode Huffman pada Perangkat Lunak WinZip

Amelia Natalie (13509004)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509004@std.if.itb.ac.id

Abstrak – *Makalah ini membahas tentang pemampatan data, teknik pemampatan data secara umum, teori kode Huffman, aplikasi pohon biner pada kode Huffman, perangkat lunak WINZIP. Selain itu, makalah ini akan menyorot penggunaan kode Huffman yang dikombinasikan dengan teori algoritma LZ77 dalam algoritma Deflate yang digunakan pada pemampatan data pada WinZip.*

Kata kunci: *kode Huffman, WinZip, algoritma LZ77, pemampatan data, dan algoritma Deflate.*

1. PENDAHULUAN

Pemampatan data merupakan salah satu hal yang berguna dalam penyimpanan dan pengiriman data. Jika data yang disimpan dalam suatu media penyimpanan berukuran besar maka diperlukan tempat yang lebih banyak. Bayangkan bila semua data yang disimpan memiliki ukuran yang besar, maka media penyimpanan hanya dapat menyimpan data dengan jumlah yang lebih sedikit dibandingkan bila menyimpan data dengan ukuran yang tidak besar. Dalam hal pengiriman data, ukuran juga berpengaruh terhadap waktu proses pengiriman. Jika data yang dikirim berukuran besar, maka waktu pengiriman data akan memakan waktu yang lama dan begitu juga sebaliknya. Ukuran dari suatu data dipengaruhi oleh kekompleksan data tersebut. Semakin kompleks suatu data maka ukuran rangkaian bit yang diperlukan semakin panjang sehingga ukuran keseluruhan data juga semakin besar. Dengan adanya pemampatan data, ukuran suatu data dapat diperkecil tanpa merubah isi ataupun informasi yang terkandung dalam data tersebut sehingga masalah-masalah seperti yang dijabarkan di atas dalam penyimpanan dan pengiriman data dapat diatasi.

Salah satu teori pemampatan data yang terkenal dan sering dipelajari dalam pelajaran Struktur Diskrit adalah teori kode Huffman. Kode Huffman merupakan salah satu algoritma kompresi tertua yang disusun oleh David Huffman pada tahun 1952. Kode Huffman menggunakan konsep struktur data pohon biner dalam proses enkoding suatu data. Teknik kode Huffman ini dapat memapatkan data sampai dengan tiga puluh persen dari ukuran semula.

Ada beberapa perangkat lunak yang digunakan untuk pemampatan data dan salah satunya adalah WinZip. WinZip secara umum memiliki empat fungsi utama yaitu untuk mengarsipkan data, memampatkan data, membuka rasip, dan mendekompresi. Namun, yang akan lebih dibahas pada makalah kali ini adalah peran WinZip sebagai perangkat lunak untuk memampatkan data.

2. DASAR-DASAR TEORI

2.1 Pemampatan Data

2.1.1 Pengertian Pemampatan Data

*Menurut Kamus Komputer dan Teknologi Informasi, "*Pemampatan adalah pemampatan ukuran data dengan suatu cara tertentu, sehingga menghemat media yang digunakan. Pada file data agar menghemat ruang penyimpanan file dan memperpendek waktu transfeer file."

*Menurut Kamus Komputer dan Teknologi Informasi, "*Pemampatan data adalah proses pemampatan data sehingga bisa berukuran lebih kecil dari data yang sebenarnya, terutama membuang informasi yang tidak penting. Digunakan terutama untuk mempercepat proses dan mempergunakan ruang data seefisien mungkin."

Menurut Oky Dwi Nurhayati, dalam makalahnya Kompresi Data, "Pemampatan berarti memapatkan atau mengecilkan ukuran. Pemampatan data adalah suatu proses mengkodekan informasi menggunakan bit (*information bearing unit*) yang lain yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu sistem encoding tertentu."

2.1.2 Jenis-Jenis Pemampatan Data

Pemampatan data dapat dibedakan menjadi dua berdasarkan hasil proses pemampatan yaitu *Lossy Compression* dan *Loseless.*

Lossy Compression adalah teknik pemampatan data yang menghasilkan data yang berbeda dengan data sebelum mengalami proses pemampatan data tetapi informasi yang terkandung di dalamnya masih dapat digunakan oleh si penerima. Teknik ini memanfaatkan keterbatasan indra manusia dengan membuang bagian-bagian data yang tidak berguna sehingga si penerima masih beranggapan bahwa data tersebut masih dapat digunakan. Contoh penggunaan : MP3, *streaming media*, JPEG, MPEG, dan WMA.

Loseless adalah teknik pemampatan data yang menghasilkan data yang sama dengan data sebelum mengalami proses pemampatan data. Teknik ini

digunakan pada data-data yang setelah mengalami proses pemampatan harus dapat diekstrak kembali tepat seperti sebelumnya (data-data penting yang isinya tidak boleh berubah sedikitpun) seperti dokumen, teks, data program atau biner, *spreadsheet*, data-data program, dan beberapa *image* seperti GIF dan PNG. Contoh penggunaan : ZIP, RAR, GZIP, 7-ZIP.

2.1.3 Metode Pemampatan Data

Metode-metode pada proses pemampatan data dapat dibedakan sebagai berikut:

[1]Berdasarkan tipe peta kode yang digunakan untuk mengubah pesan awal (isi file input) menjadi sekumpulan *codeword*, metode kompresi terbagi menjadi dua kelompok, yaitu :

(a) Metode statik : menggunakan peta kode yang selalu sama. Metode ini membutuhkan dua fase (*two-pass*): fase pertama untuk menghitung probabilitas kemunculan tiap simbol/karakter dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan ditransmisikan. Contoh: algoritma Huffman statik.

(b) Metode dinamik (adaptif) : menggunakan peta kode yang dapat berubah dari waktu ke waktu. Metode ini disebut adaptif karena peta kode mampu beradaptasi terhadap perubahan karakteristik isi file selama proses kompresi berlangsung. Metode ini bersifat *onepass*, karena hanya diperlukan satu kali pembacaan terhadap isi file. Contoh: algoritma LZ77 dan DMC.

Berdasarkan teknik pengkodean/pengubahan simbol yang digunakan, metode kompresi dapat dibagi ke dalam tiga kategori, yaitu :

(a) Metode *symbolwise* : menghitung peluang kemunculan dari tiap simbol dalam file input, lalu mengkodekan satu simbol dalam satu waktu, dimana simbol yang lebih sering muncul diberi kode lebih pendek dibandingkan simbol yang lebih jarang muncul, contoh: algoritma Huffman.

(b) Metode *dictionary* : menggantikan karakter/fragmen dalam file input dengan indeks lokasi dari karakter/fragmen tersebut dalam sebuah kamus (*dictionary*), contoh: algoritma LZ77.

(c) Metode *predictive* : menggunakan model *finite-context* atau *finite-state* untuk memprediksi distribusi probabilitas dari simbol-simbol selanjutnya; contoh: algoritma DMC.

2.2 Kode Huffman

2.2.1 Definisi Kode Huffman

Menurut Aditya Eka Prabawa, dalam makalahnya Kompresi Data dengan Kode Huffman dan Variansinya, "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. "

[1] jbptunikompp-gdl-s1-2005-faisal1010-1909-bab-ii-a-m.

Kode Huffman tidak bersifat unik, kode untuk setiap simbol berbeda pada setiap data berbeda yang dikompres. Kode Huffman digunakan untuk mentransformasi informasi-informasi yang akan ditransfer melalui jaringan. Informasi- informasi tersebut perlu diubah ke bentuk yang lebih sederhana, dalam hal ini biner, agar selain lebih mudah dalam proses transfer, juga akan membuat informasi menjadi lebih aman.

2.2.2 Aplikasi Pohon Biner pada Kode Huffman

Pohon biner adalah pohon n-ary khusu yang memiliki n=2 atau dapat dikatakan setiap simpul yang mempunyai maksimal dua anak. Anak pertama disebut anak kiri dan anak kedua disbut anak kanan.

Proses pembentukan kode Huffman secara garis besar dapat dibagi menjadi tiga bagian :

1. Pembentukan Pohon Huffman

   Sebelum membentuk pohon Huffman, tiap karakter yang ada akan dihitung frekuensinya di dalam teks. Kemudian, menggabungkan dua pohon yang mempunyai frekuensi terkecil pada sebuah akar. Setelah digabungkan akar tersebut akan mempunyai frekuensi yang merupakan jumlah dari dua pohon penyusunnya. Proses tersebut diulangi sampai tersisa satu pohon Huffman.
2. Proses *Encoding*

   Proses ini menyusun string biner dari data yang ada. Kode untuk satu karakter yang ada dibuat dengan menyusun nama string biner yang dibaca dari akar sampai ke daun pohon Huffman. Caranya dengan menentukan karakter yang akan di-*encoding* kemuduain membaca setiap bit yagn ada pada cabang sampai menemukan daun dimana karakter itu berada sampai seluruh karakter dalam teks tersebut di-*encoding.*
3. Proses *Decoding*

   Proses ini menyusun kembali data dari string biner menjadi karakter kembali. Caranya dengan membaca sebuah bit dari string biner dimulai dari akar. Untuk setiap bit lakukan transversal pada cabang yang bersesuaian. Kemudian kodekan rangkaian bit yang telah dibaca menjadi karakter sampai semua bit di dalam string habis.

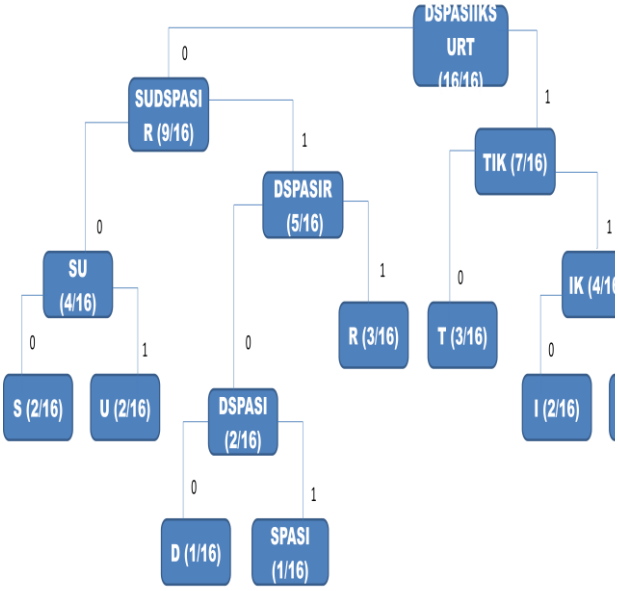
2.2.3 Contoh Penerapan Kode Huffman

Karakter-karakter dalam kalimat "Struktur Diskrit" akan dibuat kode Huffmannya.

Langkah I :

Karakter	Frekuensi	Peluang
D	1	1/16
Spasi	1	1/16
I	2	2/16
K	2	2/16
S	2	2/16
U	2	2/16
R	3	3/16
T	3	3/16

Langkah II :



Gambar 2.2.3 Pohon Huffman

Langkah III :

Karakter	Frekuensi	Kode Huffman
D	1	0100
Spasi	1	0101
I	2	110
K	2	111
S	2	000
U	2	001
R	3	011
T	3	10

2.3 Algoritma LZ77

2.3.1 Definisi Algoritma LZ77

LZ77 merupakan algoritma lain selain kode Huffman yang dapat memampatkan karakter. LZ77 bekerja dengan menemukan rangkaian karakter yang diulang. Istilah "*jendela luncur*" digunakan, pada setiap titik yang diberikan pada karakter, terdapat sebuah pencatat tentang karakter apa sebelumnya. Ketika rangkaian karakter berikutnya sama dengan yang ditangkap pada "*jendela luncur*", maka rangkaian karakter tersebut digantikan dengan dua angka yaitu :

- Angka pertama menunjukan jarak : jarak menggambarkan berapa karakter yang dilewati antara karakter pertama yang ditangkap "*jendela luncur*" dengan satu karakter sebelum karakter awal dimana rangkaian karakter sama dengan yang ditangkap "*jendela luncur*".
- Angka kedua menunjukan panjang : panjang menggambarkan banyaknya karakter dimana rangkaian karakter tersebut sama dengan yang ditangkap "*jendela luncur".*

2.3.2 Contoh Penerapan Algoritma LZ77

1. Kata-kata pada kalimat :

Blah blah blah blah blah!

2. Kita mempunyai karakter: 'B','l','a','h',' ', dan 'b'.
3. Lihat lima kalimat setelah karakter 'B' yaitu 'lah b'

vvvvv Blah blah blah blah blah! ^^^^^

4. Lihat karakter pada kalimat selanjutnya dan ternyata kalimat selanjutnya merupakan perulangan lima karakter tersebut yaitu 'lah b'
5. Kita dapat menghitung berapa panjang karakter yang sama
6. Data yang telah dijelajah : Blah blah b, dan hasil yang didapat :

Blah b[D=5,L=5]

7. Lanjutkan penjelajahannya
8. Setelah semua karakter dijelajah maka didapatkan hasil totalnya: terdapat karakter yang ditangkap *"jendela seluncur"* pada karakter kedua dan sama dengan rangkaian karakter yang panjangnya 18 karakter yang sama dan dimulai pada karakter ketujuh.
9. Hasil pemampatan data:

Blah b[D=5, L=18]

2.4 Perangkat Lunak WinZip

WinZip adalah pengompres data untuk *Microsoft Windows* dan *Mac OS X*. WinZip pertama kali diluncurkan pada April 1991 dan dikembangkan oleh WinZip International LLC, Corel Corporation. Ditemukan oleh Phil Katz untuk program PKZIP kemudian dikembangkan untuk WinZip. Versi terbaru dari WinZip adalah WinZip 15 (diluncurkan pada November 2010) dengan kelebihan-kelebihan yang lebih baik dari versi sebelumnya seperti perbaikan penampilan dengan memperbaiki penataan menu, kemampuan *drag-and-drop*, dan semakin cepat dalam pemrosesannya.

WinZip merupakan pengompres berkas yaitu suatu perangkat lunak yang digunakan untuk menggabungkan beberapa data menjadi satu atau beberapa berkas untuk memudahkan pengiriman ataupun penyimpanan berkas tersebut. WinZip mengurangi ukuran data yang berguna untuk memindahkan sejumlah besar berkas melewati jaringan yang memiliki latensi tinggi seperti internet. WinZip dapat menggabungkan dan memampatkan beberapa file sekaligus menggunakan bermacam-macam algoritma, tetapi paling umum menggunakan *Deflate Algorithm*. *Deflate Algorithm* mengkombinasikan dua algoritma yaitu algoritma Huffman dan algoritma LZ77.

[2]Dari hasil pengujian, Algoritma Deflate menghasilkan rasio kompresi yang paling bagus yaitu dengan rata-rata rasio kompresi 401,2%, diikuti oleh algoritma LZW yaitu dengan rata-rata rasio 240,2% dan terakhir algoritma Huffman dengan rata-rata rasio kompresi 156,9%. Sedangkan dilihat dari segi kecepatan kompresi data, algoritma Huffman menghasilkan kecepatan rata-rata yang paling tingi yaitu dengan rata-rata kecepatan kompresi 6.560,3 Kbyte/s, diikuti oleh algoritma Deflate dengan 1.833,7 Kbyte/s dan algoritma LZW 483,6 Kbyte/s. Sedangkan untuk kecepatan dekompresi data algoritma Deflate menghasilkan kecepatan rata-rata 17.653,5 Kbyte/s kemudian algoritma Huffman 7.790,8 Kbyte/s dan algoritma LZW 622,4 Kbyte/s.

Gambar 2.4 Logo WinZip
Sumber : http://www.google.co.id/imgres?imgurl=http://3.bp.blogspot.com

3. ALGORITMA HUFFMAN DAN ALGORITMA LZ77 PADA WINZIP

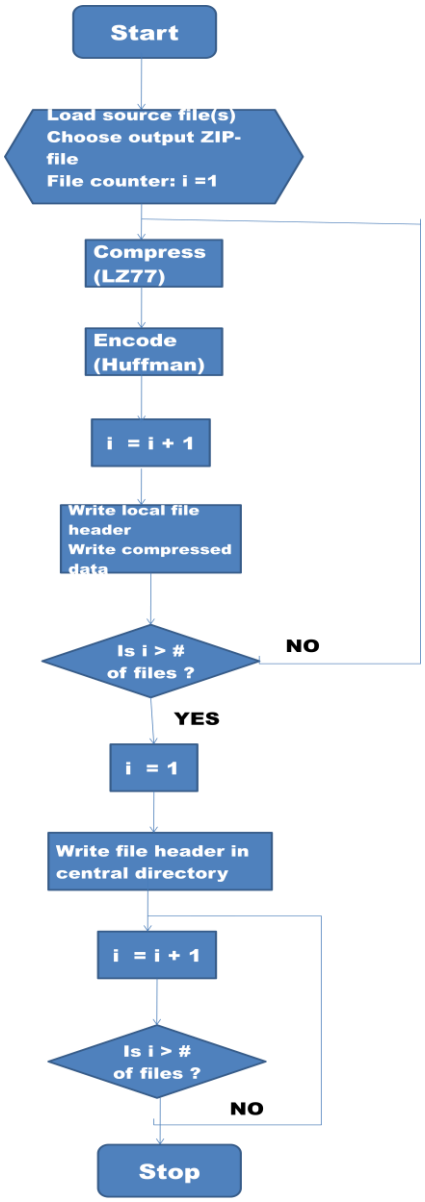
Terdapat tiga metode pemampatan data yaitu :

1. Tidak memampatkan data apapun. Hal ini disebabkan data tersebut telah dimampatkan.
2. Memampatkan, pertama kali dengan LZ77 lalu dengan kode Huffman. Pohon yang digunakan pada metode ini didefinisikan oleh spesifikasi Deflate sendiri, dan tidak ada ruang bebas yang diperlukan yang diambil untuk penyimpanan pohon tersebut.
3. Memampatkan, pertama kali dengan LZ77 kemudian kode Huffman dengan pohon yang dibuat kompresor dan disimpan bersama dengan data.

Secara sederhana proses pemampatan data pada algoritma *Deflate* yang digunakan oleh ZIP adalah proses pemampatan data gabungan antara algoritma LZ77 dengan algoritma Huffman. Algoritma *Deflate* sendiri termasuk jenis pemampatan data *Loselees* karena hasil keluarannya yang tidak berubah tetapi tetap mempertahankan bentuk asli arsip tersebut.

Proses ini dimulai dengan algoritma LZ77. Dari proses LZ77, dihasilkan tiga bagian yaitu karakter awal, sepasang panjang-jarak, dan akhir rangkaian karakter (seperti yang telah dijelaskan pada bagian 2.3.1 Definisi Algoritma LZ77). Ketiga hal tersebut harus dapat dibedakan satu sama lain dimana unsur yang menunjukan karakter awal, sepasang panjang-jarak, dan akhir rangkaian karakter karena semua ini akan menjadi dasar dari pohon Huffman. Hasil dari proses LZ77 akan mengalami proses *encoding* di pohon Huffman. Mungkin bagian paling sulit dari spesifikasi *Deflate* adalah memahami bagaimana cara pohon dikodekan untuk "pergi bersama" dengan data ketika data tersebut dikomprei dengan pohon-pohon khusus.

Alur kerja pemampatan data dari ZIP sebagai berikut:



Gambar 3 Zipping source file(s) to ZIP-file
Sumber : ZIP-file encoding & decoding using DEFLATE

[2] http://www.ittelkom.ac.id/library/index.php?option=com_repository&Itemid=34&task=detail&nim=111010095

4. KESIMPULAN

Pemampatan data diperlukan karena sangat bermanfaat terutama pada saat penyimpanan data dan pengiriman data. Jenis pemampatan data berdasarkan hasilnya dibedakan menjadi dua yaitu *Lossy Compression* dan *Loselees.* Algoritma Huffman memanfaatkan teori pohon biner pada pembentukan pohon Huffman. Pemampatan data pada perangkat lunak WinZip menggunakan algorima *Deflate* dan algoritma *Deflate* memanfaatkan algoritma LZ77 dan algoritma Huffman pada proses pemampatannya.Algoritma *Deflate* tergolong jenis pemampatan data *Loselees.* Pada algoritma *Deflate*, algoritma LZ77 dilakukan terlebih dahulu sebelum algoritma Huffman.

REFERENSI

Rinaldi Munir, "Matematika Diskrit", Penerbit Informatika, 2005.Hlm IX-26 sampai dengan IX-28.

Tanggal akses : Rabu, 15 Desember 2010.

http://www.zlib.net/feldspar.html.

Tanggal akses: Rabu, 15 Desember 2010.

Andreas Corneliussen dkk. "ZIP-file encoding & decoding using DEFLATE ".Penerbit Aalborg University Department of Electronic Systems.

Tanggal akses : Kamis, 16 Desember 2010

http://www.zlib.net/feldspar.html

Tanggal akses : Kamis, 16 Desember 2010

http://www.ittelkom.ac.id/library/index.php?option=com_repository&Itemid=34&task=detail&nim=111010095

Tanggal akses : Kamis, 16 Desember 2010

Oky Dwi Nurhayati, ST, MT. "kompresi data.pdf "

Tanggal Akses : Rabu, 15 Desember 2010.

jbptunikompp-gdl-s1-2005-faisal1010-1909-bab-ii-a-m.

Tanggal Akses : Rabu, 15 Desember 2010.

I.Y.B. Aditya Eka Prabawa W. "Kompresi Data dengan Kode Huffman dan Variansinya.pdf"

Tanggal Akses : Rabu, 15 Desember 2010.

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, 17 Desember 2010

Amelia Natalie (13509004)