Studi Informatika: Jurnal Sistem Informasi , 7(2), 2014, 1-6

PERBANDINGAN KOMPRESI FILE DATA DENGAN ALGORITMA HUFFMAN, HALF BYTE DAN RUN LENGTH

Nuryasin
Fakultas Sains dan Teknologi Program Studi Sistem Informasi
Universitas Islam Negeri Syarif Hidayatullah Jakarta
Email: tri3zyn@yahoo.com,

ABSTRACT

A company or organization in general often relates to document file. The documents usually are in the paper, and day by day it will be increased and use many space of room. To day to put a document can use disk storage with large capacity. The documents usually are used to change information from one departement to other and sent by modem or internet. Data file document will use large capacity and big cost, then for it must be compresed.

***Keywords** : Algoritma Huffman, Half Byte and Run Length*

ABSTRAK

Sebuah perusahaan atau organisasi pada umumnya sering berkaitan dengan dokumen berkas. Dokumen biasanya di kertas, dan hari demi hari itu akan meningkat dan menggunakan banyak tempat penyimpanan. Untuk sekarang, menempatkan dokumen dapat menggunakan penyimpanan disk dengan kapasitas besar. Dokumen biasanya digunakan untuk mengubah informasi dari satu departemen ke lain dan dikirim oleh modem atau internet. dokumen data file akan menggunakan kapasitas besar dan biaya yang besar, maka untuk itu harus dikompresi.

***Kata kunci**: Algoritma Huffman, Half Byte dan Run Length*

1. Pendahuluan

Pada umumnya suatu organisasi selalu berurusan dengan dokumen yang ditulis pada kertas. Dokumen tersebut disimpan pada filing cabinet. Dengan bertambahnya dokumen yang harus disimpan tiap hari, sehingga filing cabinet tersebut penuh.

Untuk mengatasinya perlu ditambah filing cabinet, tetapi ruang yang ada akan semakin sempit jika filing cabinet terus bertambah. Dengan menggunakan komputer dapat diatasi masalah ruang penyimpanan dokumen yang tidak lagi ditulis pada kertas melainkan pada magnetic disk seperti yang terlihat sekarang ini.

Dalam dunia komputer dan internet, pemampatan file digunakan dalam berbagai keperluan, jika kita ingin mem-backup data, kita tidak perlu menyalin semua file aslinya, dengan memampatkan (mengecilkan ukurannya) file tersebut terlebih dahulu, maka kapasitas tempat penyimpanan yang diperlukan akan menjadi lebih kecil. Jika sewaktu-waktu data tersebut akan diperlukan, baru dikembalikan lagi ke file aslinya.

Down-load dan Up-load file suatu pekerjaan yang kadang mengesalkan pada dunia internet, setelah menghabiskan beberapa waktu kadang-kadang hubungan terputus dan kita harus melakukannya lagi dari awal, hal ini sering terjadi pada file-file yang berukuran besar. Untunglah file-file tersebut dapat dimampatkan terlebih dahulu sehingga waktu yang diperlukan akan menjadi lebih pendek dan kemungkinan pekerjaan down-load dan up-load gagal akan menjadi lebih kecil.

Pada saat ini kebanyakan software memerlukan media penyimpanan (disk) dengan kapasitas yang besar, yang akan menimbulkan banyak biaya untuk penyimpanan.

Selain itu juga jika orang hendak mengirimkan suatu file data melalui modem dari satu komputer ke komputer lain akan diperlukan waktu yang lama karena ukuran filenya besar, sehingga juga mengakibatkan biaya semakin besar (biaya pengiriman melalui modem dihitung dalam pulsa). Untuk keperluan tersebut maka perlu adanya software yang akan memampatkan suatu file , tanpa mengubah isi aslinya. Berdasarkan latar belakang tersebut penulis melakukan penelitian dengan melakukan perbandingan untuk kompresi file data dengan judul : ***"Perbandingan Kompresi File Data Dengan Algoritma Huffman, Half Byte Dan Run Length"***

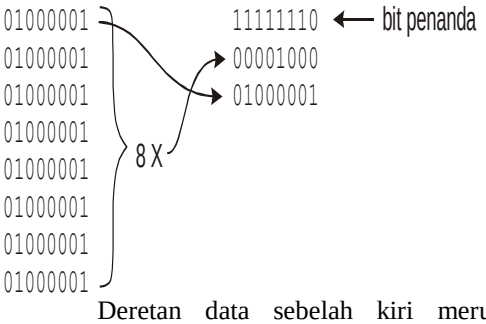
2. Landasan Teori

A. Teori Algoritma Run-Length

Algoritma Run-Lenngth digunakan untuk memampatkan data yang berisi karakter-karakter berulang. Saat karakter yang sama diterima secara berderet empat kali atau lebih (lebih dari tiga), algoritma ini mengkompres data dalam suatu tiga karakter berderetan. Algoritma Run-Length paling efektif pada file-file grafis, dimana biasanya berisi deretan panjang karakter yang sama.

Metode yang digunakan pada algoritma ini adalah dengan mencari karakter yang berulang lebih dari 3 kali pada suatu file untuk kemudian diubah menjadi sebuah bit penanda (marker bit) diikuti oleh sebuah bit yang memberikan informasi jumlah karakter yang berulang dan kemudian ditutup dengan karakter yang dikompres, yang dimaksud dengan bit penanda disini adalah deretan 8 bit yang membentuk suatu karakter ASCII.

Agar lebih jelas mengenai algoritma Run-Length dapat digambarkan sebagai berikut :



Deretan data sebelah kiri merupakan deretan data pada file asli, sedangkan deretan data sebelah kanan merupakan deretan data hasil pemampatan dengan algoritma Run-Length.

B. Algoritma Half Byte

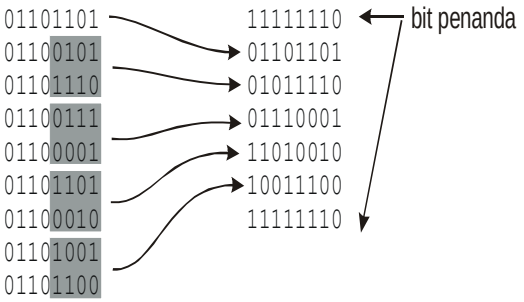
Algoritma *Half-Byte* meman-faatkan empat bit sebelah kiri yang sering sama secara berurutan terutama pada file-file text. Misalnya pada suatu file text berisi tulisan "mengambil", dalam heksadesimal dan biner karakter-karakter tersebut diterjemahkan sebagai :

Karakter	Heksadesimal	Biner
M	6D	01101101
e	65	01100101
n	6E	01101110
g	67	01100111
a	61	01100001
m	6D	01101101
b	62	01100010
i	69	01101001
l	6C	01101100

Jika anda perhatikan karakter-karakter tersebut memiliki empat bit sebelah kiri yang sama yaitu 0110. Gejala seperti inilah yang dimanfaatkan oleh Algoritma *Half-Byte*.

Saat karakter yang empat bit pertamanya sama diterima secara berderet tujuh kali atau lebih, algoritma ini mengkompres data tersebut dengan bit penanda kemudian karakter pertama dari deretan empat bit yang sama diikuti dengan pasangan empat bit terakhir deretan berikutnya dan ditutup dengan bit penutup. Algoritma ini paling

efektif pada file-file text dimana biasanya berisi text-text yang memiliki empat bit pertama yang sama. Agar lebih jelas algoritma *Half-Byte* dapat digambarkan sebagai berikut :



Deretan data sebelah kiri merupakan deretan data pada file asli, sedangkan deretan data sebelah kanan merupakan deretan data hasil pemampatan dengan algoritma *Half-Byte*.

C. Algoritma Huffman

Dasar pemikiran algoritma ini adalah bahwa setiap karakter ASCII biasanya diwakili oleh 8 bits. Jadi misalnya suatu file berisi deretan karakter "ABACAD" maka ukuran file tersebut adalah 6 x 8 bits = 48 bit atau 6 bytes. Jika setiap karakter tersebut di beri kode lain misalnya A=1, B=00, C=010, dan D=011, berarti kita hanya perlu file dengan ukuran 11 bits (10010101011), yang perlu diperhatikan ialah bahwa kode-kode tersebut harus unik atau dengan kata lain suatu kode tidak dapat dibentuk dari kode-kode yang lain. Pada contoh di atas jika kode D kita ganti dengan 001, maka kode tersebut dapat dibentuk dari kode B ditambah dengan kode A yaitu 00 dan 1, tapi kode 011 tidak dapat dibentuk dari kode-kode yang lain. Selain itu karakter yang paling sering muncul, kodenya diusahakan lebih kecil jumlah bitnya dibandingkan dengan karakter yang jarang muncul. Pada contoh di atas karakter A lebih sering muncul (3 kali), jadi kodenya dibuat lebih kecil jumlah bitnya dibanding karakter lain.

1. Penentuan Kode

Untuk menentukan kode-kode dengan kriteria bahwa kode harus unik dan karakter yang sering muncul dibuat kecil jumlah bitnya, kita dapat menggunakan algoritma Huffman.

Sebagai contoh, sebuah file yang akan dimampatkan berisi karakter-karakter "PERKARA". Dalam kode ASCII masing-masing karakter dikodekan sebagai :

P = 50H = 01010000B
E = 45H = 01000101B
R = 52H = 01010010B
K = 4BH = 01001011B
A = 41H = 01000001B

Maka jika diubah dalam rangkaian bit, "PERKARA" menjadi :

01010000 = P
01000101 = E
01010010 = R
01001011 = K
01000001 = A
01010010 = R
01000001 = A

3. Metode Penelitian

Dalam penulisan ini, digunakan beberapa metode, yaitu :

A. Pengamatan dan Penelitian

Pada penulisan penelitian ini penulis menggunakan metode dengan cara mengadakan pengamatan terhadap hasil dari file yang telah dikompresi dan mengadakan penelitian dari proses berjalannya kompresi file data.

B. Riset Perpustakaan

Pada riset perpustakaan ini, penulis mengumpulkan data dari buku-buku literature serta bahan lain yang menunjang penulisan skripsi ini.

4. IMPLEMENTASI PROGRAM

A. Implementasi Algoritma *Run-Length*

File hasil pemampatan dengan algoritma *Run-Length* harus ditandai pada awal datanya sehingga sewaktu pengembalian ke file asli dapat dikenali apakah file tersebut benar merupakan hasil pemampatan dengan algoritma ini.

Pada program ini format pengenal file tersebut ditulis pada byte pertama, kedua dan ketiga dengan karakter R, U, dan N. Pembaca dapat mengganti format tersebut dengan karakter lain yang diinginkan, demikian juga dengan jumlahnya.

Karakter berikutnya (keempat) berisi karakter bit penanda yang telah ditentukan dengan mencari karakter dengan frekuensi kemunculan terkecil. Jika misalnya pada suatu file bit penandanya adalah X, maka 4 byte pertama isi file pemampatan adalah :

					. . .
it ke-					5 . . .

Karakter kelima dan seterusnya berisi hasil pemampatan dengan algoritma *Run-Length* seperti yang telah dijelaskan pada bab sebelumnya.

B. Implementasi Algoritma *Half-Byte*

Implementasi pada algoritma Half-Byte dengan algoritma Run-Length mempunyai

perbedaan terutama pada metode untuk proses pemampatan file dan jenis file yang dimampatkan oleh kedua algoritma tersebut.

Pada algoritma Run-Length metode yang digunakan adalah dengan mencari karakter yang berulang lebih dari 3 (tiga) kali pada suatu file untuk kemudian diubah menjadi sebuah bit penanda (marker bit) diikuti oleh sebuah bit yang memberikan informasi jumlah karakter yang berulang dan kemudian ditutup dengan karakter yang dikompres.

Algoritma Run-Length ini paling efektif untuk file-file grafis, dimana biasanya berisi deretan panjang karakter yang sama.

Sedangkan metode pada algoritma Half-Byte yaitu dengan memanfaatkan empat bit sebelah kiri yang sering sama secara berurutan. Pada algoritma ini, pemampatan file yang paling efektif digunakan untuk file – file text.

Seperti pada algoritma *Run-Length*, file hasil pemampatan dengan algoritma *Half-Byte* harus ditandai pada awal datanya sehingga sewaktu pengembalian ke file asli dapat dikenali apakah file tersebut benar merupakan hasil pemampatan dengan algoritma ini.

Pada program ini format pengenal file tersebut ditulis pada byte pertama, kedua dan ketiga dengan karakter H, A, dan L. Pembaca dapat mengganti format tersebut dengan karakter lain yang diinginkan, demikian juga dengan jumlahnya.

Karakter berikutnya (keempat) berisi karakter bit penanda yang telah ditentukan dengan mencari karakter dengan frekuensi kemunculan terkecil. Jika misalnya pada suatu file bit penandanya adalah Q, maka 4 byte pertama isi file pemampatan adalah :

					. .
it ke-					. . .

Karakter kelima dan seterusnya berisi hasil pemampatan dengan algoritma *Half-Byte* seperti yang telah dijelaskan pada bab sebelumnya.

C. Implementasi Algoritma *Huffman*

Seperti pada kedua algoritma sebelumnya, file hasil pemampatan dengan algoritma *Huffman* harus ditandai pada awal datanya sehingga sewaktu pengembalian ke file asli dapat dikenali apakah file tersebut benar merupakan hasil pemampatan dengan algoritma ini.

Pada program ini format pengenal file tersebut ditulis pada byte pertama, kedua dan ketiga dengan karakter H, U, dan F. Lagi-lagi anda dapat mengganti format tersebut dengan karakter lain yang diinginkan, demikian juga dengan jumlahnya.

Karakter keempat, kelima dan keenam berisi informasi ukuran file asli dalam byte, 3 karakter ini dapat berisi maksimal FFFFFF H atau 16.777.215 byte. Karakter ketujuh berisi informasi jumlah karakter yang memiliki kode *Huffman* atau dengan kata lain jumlah karakter yang frekuensi kemunculannya pada file asli lebih dari nol, jumlah tersebut dikurangi satu dan hasilnya disimpan pada karakter ke tujuh pada file pemampatan

Program ini dibuat dengan program Borland Delphi versi 5 yang berorientasi pada objek *(Object Orientation)*, mengingat saat ini hampir semua program sudah menggunakan windows sebagai base-nya. Bagi pembaca yang sudah terbiasa dengan program konvensional (under DOS), tidak akan menjadi masalah karena penulis akan menjelaskan langkah demi langkah pembuatan program ini, sebagai berikut :

1. Buatlah file baru dengan meng-klik File pada menu utama kemudian klik New Application.
2. Ganti Caption dari Form dengan Pemampatan File dan Name dengan FormUtama pada object properties.
3. Tambahkan 5 buah Speed Button pada FormUtama masing-masing :

a. Name : SButOpen
   Caption : Open
b. Name : SButSave
   Caption : Save
   Enabled : False
c. Name : SButSaveOrig
   Caption : Save to Original File
   Enabled : False
   Visible : False
d. Name : SButDecompress
   Caption : Decompress
   Enabled : False
   Visible : False
e. Name : SButExit
   Caption : Exit

4. Tambahkan sebuah GroupBox pada form utama dengan object properties :
   Name : GroupMetode
   Caption : Kompres File
   Enabled : False
5. Di dalam GroupMetode tambahkan 3 buaf Speed Button masing-masing :

a. Name : SButRunLength
   Caption : Run Length
b. Name : SButHalfByte
   Caption : Half Byte
c. Name : SButHuffman
   Caption : Huffman

6. Tambahkan 2 buah Radio Button pada FormUtama masing-masing :

a. Name : RadioCompress
   Caption : Compress
   Checked : True
b. Name : RadioDecompress
   Caption : Decompress

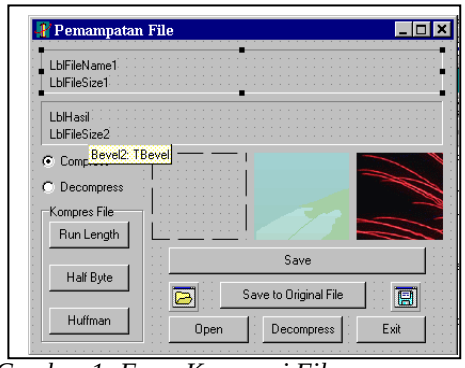
7. Tambahkan 4 buah Label pada FormUtama masing-masing :

a. Name : LblFileName1
b. Name : LblFileSize1
c. Name : LblHasil
d. Name : LblFileSize2

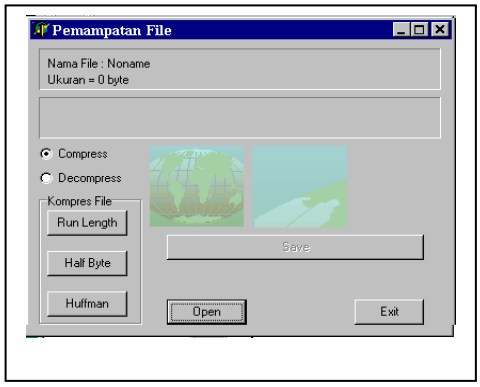
8. Tambahkan sebuah OpenDialog dan sebuah SaveDialog pada FormUtama.
9. Tambahkan 3 buah Image pada FormUtama masing-masing :

a. Name : ImageTampil
Stretch : True
b. Name : Image1
Picture : (Pilih sendiri gambar untuk compress)
c. Name : Image2
Visible : False
Picture : (Pilih sendiri gambar untuk decompress)
d. Tambahkan Bevel untuk memperindah tampilan

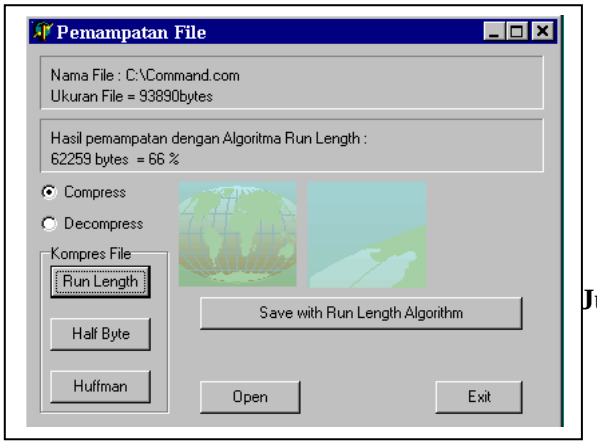
Setelah langkah-langkah tersebut dilakukan, kita akan mendapatkan sebuah form seperti gambar berikut ini :



Gambar 1: Form Kompresi File



Gambar 2 : Running Program



Gambar 3 : Hasil Compress Run Length

1. Kesimpulan Dan Saran

A. Kesimpulan

Dengan penjelasan yang telah diuraikan, maka penulis mengambil kesimpulan secara umum bahwa hasil pemampatan untuk semua jenis file dengan menggunakan algoritma Huffman ternyata lebih efisien dibandingkan dengan algoritma Run-Length dan algoritma Half-Byte.

Dari ketiga algoritma tersebut, dihasilkan perbandingan mengenai pemampatan file dalam bentuk tabel seperti dibawah ini, yang menunjukkan bahwa algoritma Huffman merupakan algoritma yang paling efisien dibanding dengan dua algoritma Run-Length dan algoritma Half-Byte.

ama file	kuran file asli bytes)	Ukuran File Hasil Pemampatan Dengan Algoritma : (bytes)		
		un-Length	alf-Byte	Huffman
uratkuasa.doc	04448	5134 62 %)	4581 80 %)	49277 (47 %)
levator.cdr	93750	90354 98 %)	92471 99 %)	194056 (100 %)
ifa2000.exe	605632	514726 94 %)	549043 96 %)	1305473 (81 %)
unga.bmp	60054	95691 54 %)	78159 77 %)	70890 (19 %)
vent.dat	3195	112 8 %)	067 53 %)	2427 (18 %)
eadme.txt	0756	717 81 %)	171 57 %)	6277 (58 %)

Tabel 1 Perbandingan Hasil Kompresi

Disamping itu dengan adanya kompresi file, akan mendapatkan keuntungan dan manfaat yang didapatkan antara lain :

- Untuk menghemat biaya pengiriman data melalui modem karena semakin kecil file yang dikirim semakin kecil juga pulsa yang dipakai.

- Untuk menghemat biaya penyimpanan ke disk karena ukuran file semakin kecil sehingga memerlukan sedikit disk.
- Untuk security karena selain dimampatkan, file juga dikodekan *(encoded)*

.

B. Saran

Dari kesimpulan di atas, penulis memberikan saran-saran sebagai berikut :

1. Sebaiknya untuk penyimpanan file dengan basis digital dilakukan kompresi file data terlebih dahulu sebelum disimpan secara permanen, karena akan mengurangi dari size data tersebut.
2. Penelitian ini dapat anda kembangkan dengan menggunakan algoritma selain dari ketiga algoritma yang penulis gunakan.

2. Daftar Pustaka

[1] *Coulouris George, Dollmore Jean, Kinberg Tim;* Distributed System Concepts and Design; London; *1994*

[2] *Widodo Priyono*; Kamus Istilah Internet dan Komputer; Lintas Media Jombang; *2001*

[3] *Suryanto;* Pemampatan File dengan Algoritma Huffman; Dnastindo Adiperkasa Internasional; *1995.*

[4] *Ir. Rinaldi Munir, Ir. Leoni Lidya*; Algoritma dan Pemrograman; Informatika Bandung; *1998*

[5] *John M. Echols, Hassan Shadily*; Kamus Inggris Indonesia; PT. Gramedia Jakarta; *1995*

[6] *M. agus J. alam;* Borland Delphi 5.0; Elex Media Komputindo; *2001*

[7] *Pusat Pembinaan dan Pengembangan*; Kamus Besar Bahasa Indonesia; Balai Pustaka Jakarta; *1990*

[8] *Binstock, Andrew and Rex John;* Practical algorithm for Programmer; Reading Addison-Wesley; *1995*

[9] Nelson, Mark and Gailly; The Compression Book; New York; *1996*

[10] *John J. Longkutoy, Drs;* Dasar-dasar Programming*;* Mutiara Sumber Widya, Jakarta Pusat; *1993*