Kompresi Data dengan Algoritma Huffman dan Algoritma Lainnya

Hary Fernando
Nim 13505113
Email: if15113@students.if.itb.ac.id

Program studi Teknik Informatika
Sekolah Teknik elektro dan informatika
Insttitut teknologi bandung
Jln ganesa 10 bandung 40132

Abstrak

Makalah ini mengemukakan tentang cara kompresi data dan implementasinya.
Kompresi data tersebut antara lain menggunakan kode/algoritma Huffman dan algoritma-algoritma sejenis yang telah menjelaskan dan membantu kita dalam menangani file-file besar. Termasuk format format file audio sepert mp3, format gambar seperti jpg dan contoh-contoh sederhana serta penerapannya

Diakhir makalah teradapat aplikasi-aplikasi yang bermula dari algoritma-algorima tersebut yang sampai sekarang masih dikembangkan softwarenya hingga menjadi lebih baik

***Kata kunci**: kompresi data, algoritma Huffman, Algoritma Lempel-Ziv-Welch, Shannon-Fano Algorithm, Lossy Compression, Loseless*

1. Pendahuluan

Kompresi ialah proses pengubahan ekumpulan data menjadi suatu bentuk kode untuk menghemat kebutuhan tempat penyimpanan dan waktu untuk transmisi data [1]. Saat ini terdapat berbagai tipe algoritma kompresi [2,3], antara lain: Huffman, LIFO, LZHUF,LZ77 dan variannya (LZ78, LZW, GZIP), Dynamic Markov Compression (DMC), Block-Sorting Lossless, Run-Length, Shannon-Fano, Arithmetic, PPM (Prediction by Partial Matching), Burrows-Wheeler Block Sorting, dan Half Byte.

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

- **Metode static**: 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.

- 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 1-kali pembacaan terhadap isi file.
  Contoh: algoritma LZW dan DMC

Kompresi data

- Kompresi berarti memampatkan /mengecilkan ukuran

- Kompresi data adalah proses mengkodekan informasi menggunakan bit atau information-bearing unit yang lain yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu sistem enkoding tertentu

- Contoh kompresi sederhana yang biasa kita lakukan misalnya adalah

menyingkat kata-kata yang sering digunakan tapi sudah memiliki konvensi umum. Misalnya: kata “yang” dikompres menjadi kata “yg”.

- Pengiriman data hasil kompresi dapat dilakukan jika pihak pengirim/yang melakukan kompresi dan pihak penerima memiliki aturan yang sama dalam hal kompresi data.

- Pihak pengirim harus menggunakan algoritma kompresi data yang sudah baku dan pihak penerima juga menggunakan teknik dekompresi data yang sama dengan pengirim sehingga data yang diterima dapat dibaca/di-dekode kembali dengan benar

- Kompresi data menjadi sangat penting karena memperkecil kebutuhan penyimpanan data, mempercepat pengiriman data, memperkecil kebutuhan bandwidth

- Teknik kompresi bisa dilakukan terhadap data teks/biner, gambar (JPEG, PNG, TIFF), audio (MP3, AAC, RMA, WMA), dan video (MPEG, H261, H263).

Contoh kebutuhan data selama 1 detik pada layar resolusi 640 x 480:

- Data text
    1. 1 karakter = 2 bytes (termasuk karakter ASCII Extended)
    2. Setiap karakter ditampilkan dalam 8x8 pixels
    3. Jumlah karakter yang dapat ditampilkan per halaman = (640 x 480) / (8 x 8) =4.800 karakter.
       Kebutuhan tempat penyimpanan per halaman = 4.800×2 byte = 9.600 byte = 9.375 Kbyte.
- Data Grafik Vektor
    1. 1 still image membutuhkan 500 baris
    2. Setiap 1 baris direpresentasikan dalam posisi horisontal, vertikal, dan field atribut sebesar 8-bit
    3. sumbu Horizontal direpresentasikan dengan $\log_2 640 = 10$ bits
    4. sumbu Vertical direpresentasikan dengan $\log_2 480 = 9$ bits
    5. Bits per line = 9bits + 10bits + 8bits = 27bits
    6. Storage required per screen page = 500 × (27 / 8) = = 1687,5 byte = 1,65 Kbyte

- Color Display
    1. Jenis : 256, 4.096, 16.384, 65.536, 16.777.216 warna
    2. Masing-masing warna pixel memakan tempat 1 byte
    3. Misal 640 x 480 x 256 warna x 1 byte = 307.200 byte = 300 Kbyte

Kebutuhan tempat penyimpanan untuk media kontinyu untuk 1 detik Playback:

- Sinyal audio tidak terkompres dengan kualitas suara telepon dengan sample 8 kHz dan dikuantisasi 8 bit per sample, pada bandwidth 6 Kbits/s, membutuhkan storage:

- Storage yang diperlukan = (64 Kbit/s / 8 bit/byte)x (1 s / 1.024 byte/Kbyte) = 8 Kbyte

- Sinyal audio CD disample 44,1 kHz, dikuantisasi 16 bits per sample.

- Storage = 44,1 kHz x 16 bits = 705,6 x 10^3 bits = 88.200 bytes untuk menyimpan 1 detik playback.

- Kebutuhan sistem PAL standar
    1. 625 baris dan 25 frame/detik
    2. 3 bytes/pixel (luminance, red chrom, blue chrom)

3. Luminance Y menggunakan sample rate 13,5 MHz
4. Chrominance (R-Y dan B-Y) menggunakan sample rate 6.75 MHz

5. Jika menggunakan 8 bit/sample, maka
   Bandwith=(13.4MHz + 6.75 MHz + 6.75 MHz) x 8 bit = $216x10^6$ bit/s
   Data rate= 640 x 480 x 25 x 3 byte/seconds = 23.040.000 byte/s
   Required storage space/s = $2.304 X 10^4$ byte/s x (1 s / 1.024 byte/Kbyte) = 22.500 Kbyte

Jenis Kompresi Data Berdasarkan Mode Penerimaan Data oleh Manusia

- Dialoque Mode: yaitu proses penerimaan data dimana pengirim dan penerima seakan berdialog (real time), seperti pada contoh video conference.
  - Dimana kompresi data harus berada dalam batas penglihatan dan pendengaran manusia. Waktu tunda (delay) tidak boleh lebih dari 150 ms, dimana 50 ms untuk proses kompresi dan dekompresi, 100 ms mentransmisikan data dalam jaringan

- Retrieval Mode: yaitu proses penerimaan data tidak dilakukan secara real time.
  - Dapat dilakukan fast forward dan fast rewind di client
  - Dapat dilakukan random access terhadap data dan dapat bersifat interaktif.

Jenis Kompresi Data Berdasarkan Output

- **Lossy Compression**
  - Teknik kompresi dimana data hasil dekompresi tidak sama dengan data sebelum kompresi namun sudah "cukup" untuk digunakan. Contoh: Mp3, streaming media, JPEG, MPEG, dan WMA.
  - Kelebihan: ukuran file lebih kecil dibanding loseless namun masih tetap memenuhi syarat untuk digunakan.
  - Biasanya teknik ini membuang bagian-bagian data yang sebenarnya tidak begitu berguna, tidak begitu dirasakan, tidak begitu dilihat oleh manusia sehingga manusia masih beranggapan bahwa data tersebut masih bisa digunakan walaupun sudah dikompresi.
  - 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%.

- **Loseless**
  - Teknik kompresi dimana data hasil kompresi dapat didekompres lagi dan hasilnya tepat sama seperti data sebelum proses kompresi.
    Contoh aplikasi: ZIP, RAR, GZIP, 7-Zip
  - Teknik ini digunakan jika dibutuhkan data setelah dikompresi harus dapat diekstrak/dekompres lagi tepat sama.
    Contoh pada data teks, data program/biner, beberapa image seperti GIF dan PNG.
  - Kadangkala ada data-data yang setelah dikompresi dengan teknik ini ukurannya menjadi lebih besar atau sama.

Kriteria Algoritma dan Aplikasi Kompresi Data

- Kualitas data hasil enkoding: ukuran lebih kecil, data tidak rusak untuk kompresi lossy
- Kecepatan, ratio, dan efisiensi proses kompresi dan dekompresi
- Ketepatan proses dekompresi data: data hasil dekompresi tetap sama dengan data sebelum dikompres (kompresi loseless)

Klasifikasi Teknik Kompresi

1. Entropy Encoding
   - Bersifat loseless
   - Tekniknya tidak berdasarkan media dengan spesifikasi dan karakteristik tertentu namun berdasarkan urutan data
   - Statistical encoding, tidak memperhatikan semantik data
   - Mis: Run-length coding, Huffman coding, Arithmetic coding Source Coding

2. Source Coding
   - Bersifat lossy
   - Berkaitan dengan data semantik (arti data) dan media
   - Mis: Prediction (DPCM, DM), Transformation (FFT, DCT), Layered Coding (Bit position, subsampling, sub-band coding), Vector quantization.

3. Hybrid Coding
   - Gabungan antara lossy + loseless
   - mis: JPEG, MPEG, H.261, DVI

Contoh-contoh Teknik Kompresi Teks

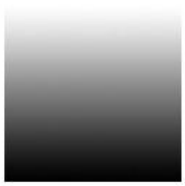
1 Run-Length-Encoding (RLE)

- Kompresi data teks dilakukan jika ada beberapa huruf yang sama yang ditampilkan berturut-turu:
  Mis: Data : ABCCCCCCCCDEFGGGG = 17 karakter
- RLE tipe 1 (min. 4 huruf sama) : ABC!8DEFG!4 = 11 karakter
- RLE ada yang menggunakan suatu karakter yang tidak digunakan dalam teks tersebut seperti misalnya ‘!’ untuk menandai.
- Kelemahan? Jika ada karakter angka, mana tanda mulai dan akhir?
  Misal data : ABCCCCCCCCDEFGGGG = 17 karakter
  RLE tipe 2: -2AB8C-3DEF4G = 12 karakter

  Misal data : AB12CCCCDEEEF = 13 karakter
  RLE tipe 2: -4AB124CD3EF = 12 karakter
- RLE ada yang menggunakan flag bilangan negatif untuk menandai batas sebanyak jumlah karakter tersebut.
- Berguna untuk data yang banyak memiliki kesamaan, misal teks ataupun grafik seperti icon atau gambar garis-garis yang banyak memiliki kesamaan pola
- Best case: untuk RLE tipe 2 adalah ketika terdapat 127 karakter yang sama sehingga akan dikompres menjadi 2 byte saja.

- Worst case: untuk RLE tipe 2 adalah ketika terdapat 127 karakter yang berbeda semua, maka akan terdapat 1 byte tambahan sebagai tanda jumlah karakter yang tidak sama tersebut.

- Menggunakan teknik loseless
- Contoh untuk data image, seperti dibawah ini:

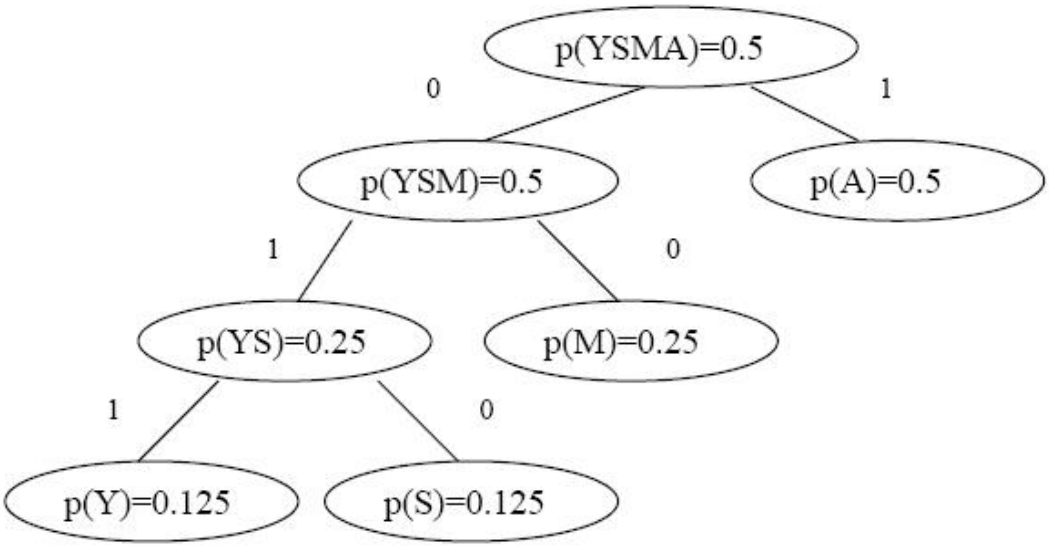
		
Original size: 10000 bytes Compressed size: 5713 bytes Ratio: 1.75	Original size: 10000 bytes Compressed size: 10100 Ratio: 0.99	Original size: 10000 bytes Compressed size: 200 Ratio: 50

2 Static Huffman Coding

- Frekuensi karakter dari string yang akan dikompres dianalisa terlebih dahulu. Selanjutnya dibuat pohon huffman yang merupakan pohon biner dengan root awal yang diberi nilai 0 (sebelah kiri) atau 1 (sebelah kanan), sedangkan selanjutnya untuk dahan kiri selalu diberi nilai 1(kiri) - 0(kanan) dan di dahan kanan diberi nilai 0(kiri) – 1(kanan)
- A bottom-up approach = frekuensi terkecil dikerjakan terlebih dahulu dan diletakkan ke dalam leaf(daun)
- Kemudian leaf-leaf akan dikombinasikan dan dijumlahkan probabilitasnya menjadi root diatasnya.

Huffman Tree:

Mis: MAMA SAYA
A = 4 -> 4/8 = 0.5
M = 2 -> 2/8 = 0.25
S = 1 -> 1/8 = 0.125
Y = 1 -> 1/8 = 0.125
Total = 8 karakter



Sehingga w(A) = 1, w(M) = 00, w(S) = 010, dan w(Y) = 011

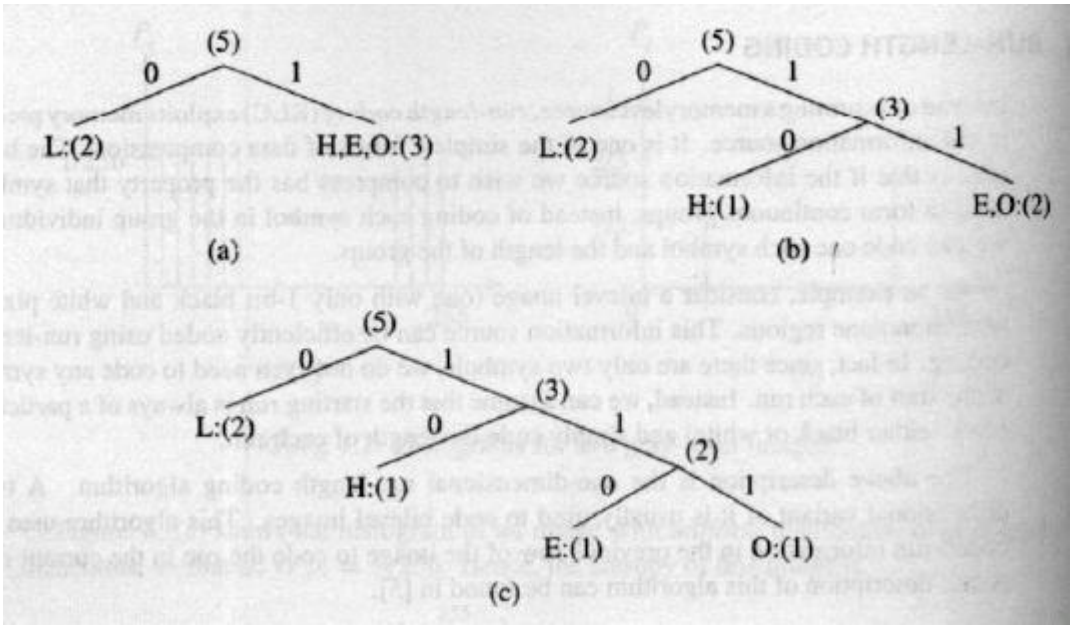
Shannon-Fano Algorithm

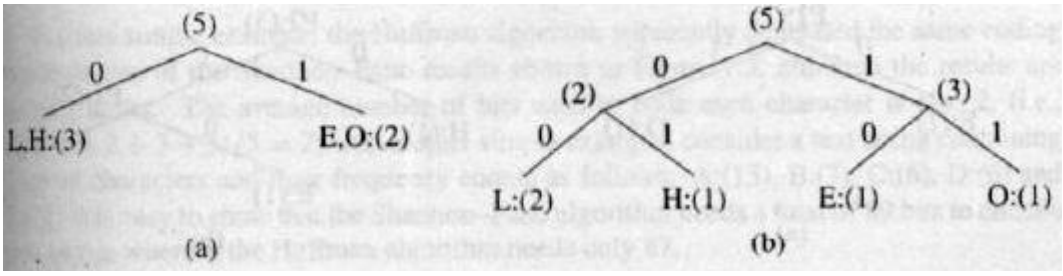
- Dikembangkan oleh Shannon (Bell Labs) dan Robert Fano (MIT)
- Contoh:

H E L L O

Simbol	H	E	L	O
Jumlah	1	1	2	1

- Algoritma
  - Urutkan simbol berdasarkan frekuensi kemunculannya
  - Bagi simbol menjadi 2 bagian secara rekursif, dengan jumlah yang kira-kira sama pada kedua bagian, sampai tiap bagian hanya terdiri dari 1 simbol.
- Cara yang paling tepat untuk mengimplementasikan adalah dengan membuat binary tree.





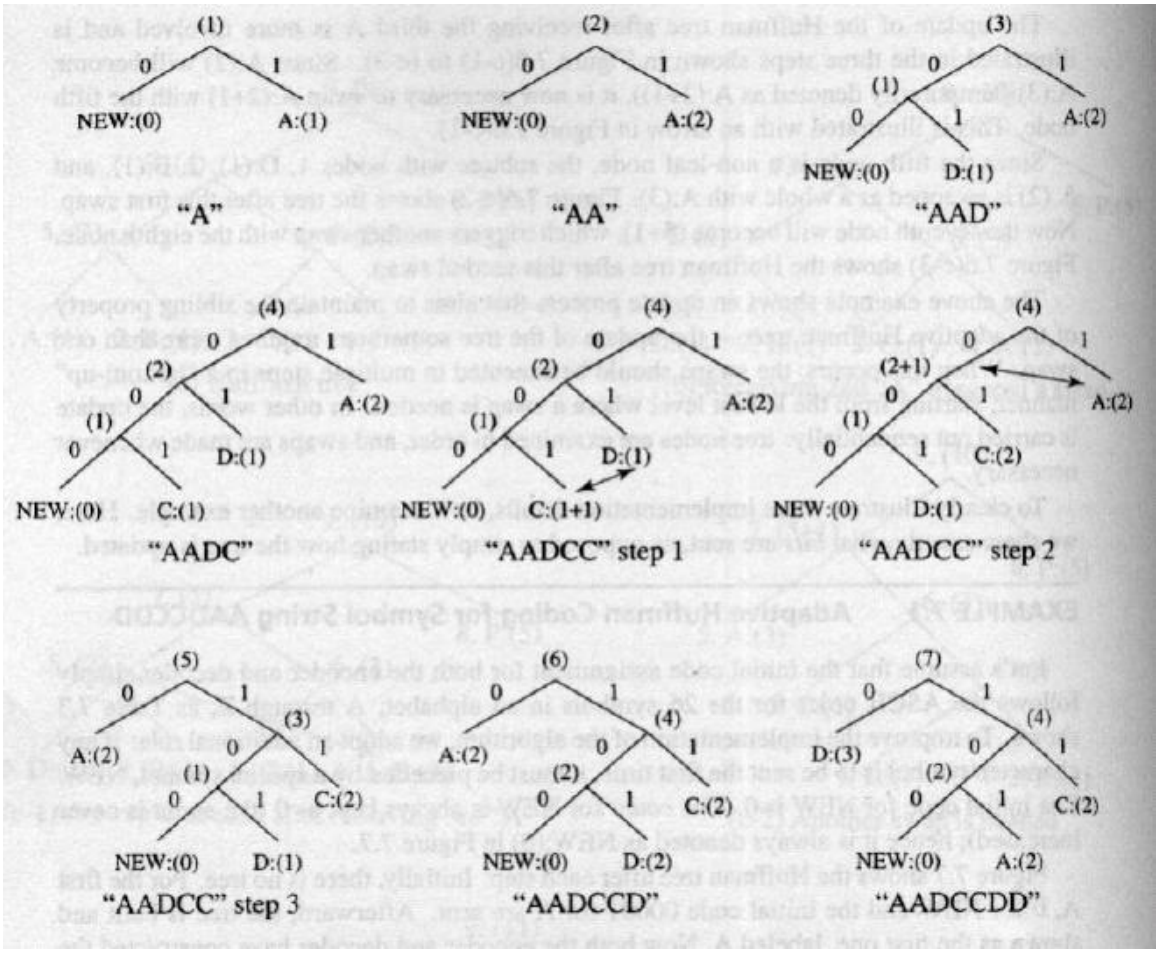
Adaptive Huffman Coding

- Metode SHC mengharuskan kita mengetahui terlebih dahulu frekuensi masing-masing karakter sebelum dilakukan proses pengkodean. Metode AHC merupakan pengembangan dari SHC dimana proses penghitungan

  frekuensi karakter dan pembuatan pohon Huffman dibuat secara dinamis pada saat membaca data

- Algoritma Huffman tepat bila dipergunakan pada informasi yang bersifat statis. Sedangkan untuk multimedia application, dimana data yang akan datang belum dapat dipastikan kedatangannya (audio dan video streaming), algoritma Adaptive Huffman dapat dipergunakan
- Metode SHC maupun AHC merupakan kompresi yang bersifat loseless

- Dibuat oleh David A. Huffman dari MIT tahun 1952 Huffman banyak dijadikan “back-end” pada algoritma lain, seperti Arithmetic Coding, aplikasi PKZIP, JPEG, dan MP3



DICTIONARY-BASED CODING

Algoritma Lempel-Ziv-Welch (LZW) menggunakan teknik adaptif dan berbasiskan "kamus" Pendahulu LZW adalah LZ77 dan LZ78 yang dikembangkan oleh Jacob Ziv dan Abraham Lempel pada tahun 1977 dan 1978. Terry Welch mengembangkan teknik tersebut pada tahun 1984. LZW banyak dipergunakan pada UNIX, GIF, V.42 untuk modem.

Algoritma Kompresi:

```
BEGIN
 S = next input character;
 While not EOF
 {
  C = next input character;
 If s + c exists in the diactionary
   S = s + c
 Else
 {
    Output the code for s;
    Add string s + c to the dictionary with a new
code
  S = c;
  }
 }
END
```

Algoritma Dekompresi:

```
BEGIN
 S = NULL;
 while not EOF{
  K = NEXT INPUT CODE;
  Entry = dictionary entry for K;
 Ouput entry;
 if(s != NULL)
    add string s + entry[0] to dictionary with new
code
 S = Entry;
 }
END
```

Aplikasi Kompresi

- ZIP File Format

- Ditemukan oleh Phil Katz untuk program PKZIP kemudian dikembangkan untuk WinZip, WinRAR, 7-Zip.
- Berekstensi *.zip dan MIME application/zip
- Dapat menggabungkan dan mengkompresi beberapa file sekaligus menggunakan bermacam-macam algoritma, namun paling umum menggunakan Katz's Deflate Algorithm.
- Beberapa method Zip:
  - Shrinking : merupakan metode variasi dari LZW
  - Reducing : merupakan metode yang mengkombinasikan metode same byte sequence based dan probability based encoding.
  - Imploding : menggunakan metode byte sequence based dan Shannon-Fano encoding.
  - Deflate : menggunakan LZW
  - Bzip2, dan lain-lain
- Aplikasi: WinZip oleh Nico-Mak Computing

- RAR File
  - Ditemukan oleh Eugene Roshal, sehingga RAR merupakan singkatan dari Roshal Archive pada 10 Maret 1972 di Rusia.
  - Berekstensi .rar dan MIME application/x-rar-compressed
  - Proses kompresi lebih lambat dari ZIP tapi ukuran file hasil kompresi lebih kecil.
  - Aplikasi: WinRAR yang mampu menangani RAR dan ZIP, mendukung volume split, enkripsi AES.

Kesimpulan

Penggunaan aplikasi pemampatan data dengan menggunakan algoritma Huffman dan algoritma-algoritma lainnya telah membantu dalam pekerjaan yang melibatkan file-file yang besar dalam bidang informasi dan telekomunikasi.
Dari kode-kode dan algoritma tersebut dihasilkanlah software-software pengompres yang selama ini kita kenal.

Daftar pustaka

[1] Howe, D., *"Free On-line Dictionary of Computing"*, http://www.foldoc.org/, 1993.

[2] Witten, I.H, *et al*., "*Managing Gigabytes"*, Van Nostrand Reinhold, New York, 1994.

[3] Ben Zhao, *et al*., "*Algorithm in the Real World - (Compression) Scribe Notes"*, http://www-2.cs.cmu.edu/~guyb/realworld/class-notes/all/, 1998, [17 Jan 2002]

[4] Compression Team, "*The LZW Algorithm"*, Data Compression Reference Center, http://www.rasip.fer.hr/research/compress/algorithms/fund/lz/LZW.html, 1997, [17 Jan 2002]

[5] Ziviani, N., de Moura, E. S., *"Compression: A Key for Next Generation Text Retrieval System"*, Department of Computer Science Univ. Federal de Minas Gerais, Brazil, 2000.

[6] University of Calgary, *Calgary Corpus*, http://links.uwaterloo.ca/calgary.corpus.html, [26 Maret 2002]

[7] University of Canterbury, "*The Canterbury Corpus"*, http://corpus.canterbury.ac.nz, [26 Feb 2002]

[8] Cormack, G.V., Horspool, R.N., *"Data Compression Using Dynamic Markov Compression"*, University of Waterloo and University of Victoria, 1986.