Analisis Penerapan Kompresi dan Dekompresi Data dengan Menggunakan Metode Statistik dan Kamus

Tjatur Kandaga
Jurusan Teknik Informatika
Fakultas Teknologi Informasi, Universitas Kristen Maranatha
Jl. Prof. Drg. Suria Sumantri No. 65 Bandung 40164
Email: tjatur.kandaga@eng.maranatha.edu

Abstract

Data compression attempts to reduce data size without losing the contained information. Smaller data size will take less space to store the data, and smaller bandwith when the data is transmitted through the network. There are two categories of data compression, which are lossy and lossless data compression. Uncompressed data mostly stored as ASCII character which use 8 bit for every character regardless the occurrence of that character. Lossless data compression techniques uses statistical or dictionary-based methods to recode the data and get smaller data size from the original (uncompressed) data. This is done by assigning fewer bits for characters that has high frequency of occurrence, and larger number of bits for rarely occurred characters. From experiments it is obvious that certain data compression techniques works best with certain data types. LZW15V has best overall result from other data compesion techniques we tested so far.

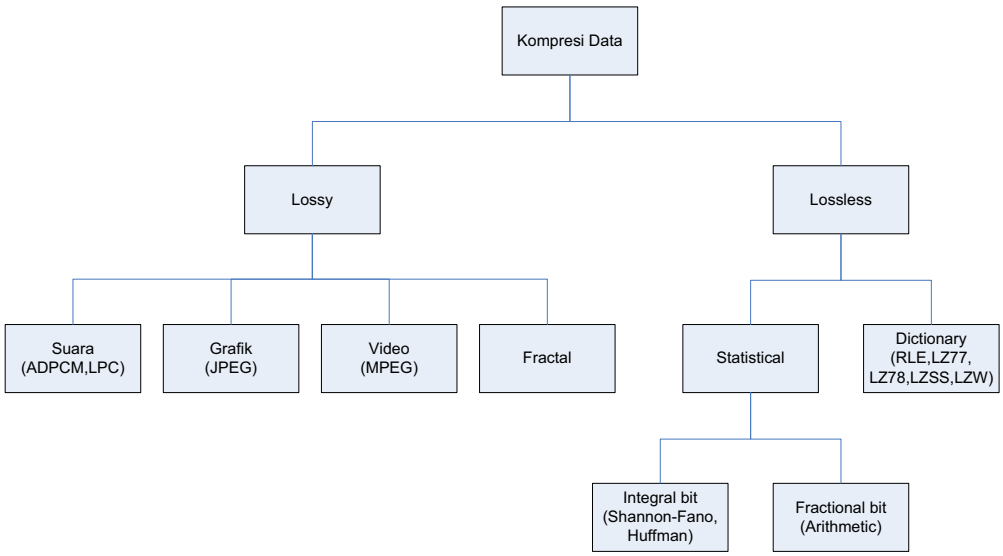
Keywords *:Data compression, Lossless, Statistical, Dictionary-based, Data decompression, Huffman Coding, Adaptive Huffman Coding, Arithmetic Coding, LZSS, LZW*

1. Pendahuluan

Tulisan ini akan membahas mengenai konsep dasar kompresi data, cara kerja dari beberapa teknik kompresi data *lossless* yang umum digunakan, analisa sederhana terhadap cara kerjanya, dan melakukan pengujian secara langsung terhadap teknik-teknik tersebut.

Kompresi data merupakan suatu upaya untuk mengurangi jumlah bit yang digunakan untuk menyimpan atau mentransmisikan data. Kompresi data meliputi berbagai teknik kompresi yang diterapkan dalam bentuk perangkat lunak *(software)* maupun perangkat keras *(hardware)*. Bila ditinjau dari sisi penggunaannya, kompresi data bisa bersifat umum untuk segala keperluan atau bersifat khusus untuk keperluan tertentu. Keuntungan data yang terkompresi antara lain: mengurangi *bottleneck* pada proses I/O dan transmisi data, penyimpanan data lebih hemat ruang, mempersulit pembacaan data oleh pihak yang tidak berkepentingan, dan memudahkan distribusi data dengan media *removable* seperti *flash* disk, CD, DVD, dll.

Pembagian teknik-teknik kompresi dan dekompresi data, sbb:



Gambar 1 Pembagian Teknik Kompresi Data

Lossless compression merupakan kompresi data dimana jika pada data hasil kompresi tersebut dilakukan dekompresi akan menghasilkan data yang sama persis dengan data asalnya yang belum dikompresi.

Lossy compression merupakan kebalikan *lossless compression* dimana hasil kompresi jika di dekompresi hasilnya tidak akan sama dengan data sebelum dikompresi. *Lossy compression* dilakukan pada data analog yang disimpan secara digital seperti grafik dan suara. *Lossy compression* mengorbankan sedikit ketepatan untuk mendapatkan perbandingan kompresi yang lebih baik. Pada model statistik probabilitas simbol yang masuk menentukan kode yang akan dihasilkan.

Teknik-teknik kompresi data yang dibahas meliputi tiga teknik yang menggunakan metode statistik dan dua teknik dengan metode kamus, yaitu :

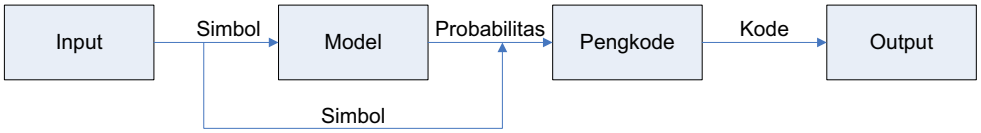
- Huffman coding (pengkodean Huffman – metode statistik)
- Adaptive Huffman coding (pengkodean Huffman secara *Adaptive*/dinamis – metode statistik)
- Arithmetic coding (pengkodean Aritmetik – metode statistik)
- LZSS (Lempel Ziv dengan *sliding window* - metode kamus)
- LZW (Lempel Ziv Welch - metode kamus)

2. Teknik-Teknik Kompresi Data *Lossless*

Secara umum kompresi data terdiri dari pengambilan simbol-simbol sebagai masukan dan mengubahnya menjadi kode-kode. Jika kompresi tersebut efektif maka kode-kode yang dihasilkan akan berukuran lebih kecil daripada simbol-

simbol asalnya. Keputusan untuk mengubah simbol atau kumpulan simbol menjadi kode tertentu diambil berdasarkan sebuah model. Karena itu elemen-elemen dasar dari kompresi data ialah pemodelan dan pengkodean.

Sebagai contoh pada Huffman Coding, proses kompresinya adalah sbb :



Gambar 2 Model statistik dengan pengkode Huffman

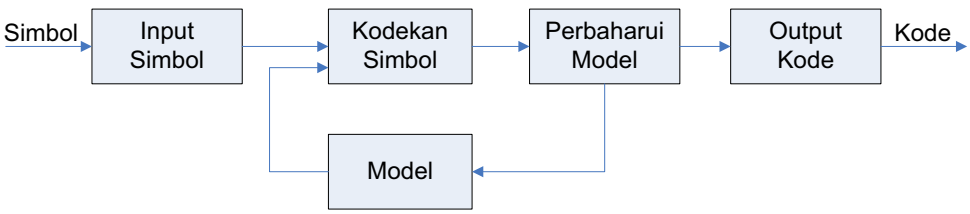
Pada Huffman Coding pertama kali seluruh data / file akan dibaca untuk mendapatkan probabilitas dari setiap simbol. Berdasarkan probabilitas tersebut dapat dibentuk tabel kode yang memiliki karakteristik sebagai berikut :

- Kode-kode yang berbeda dapat memiliki jumlah bit yang berbeda.
- Kode-kode untuk simbol-simbol yang probabilitasnya rendah memiliki jumlah bit yang lebih banyak, dan kode-kode untuk simbol-simbol yang probabilitasnya tinggi memiliki jumlah bit yang lebih sedikit.
- Kode-kode dapat didekode secara unik / tepat meskipun memiliki jumlah bit yang berbeda-beda.

Karakteristik yang pertama dan kedua secara otomatis akan menyebabkan data terkompresi pada saat dilakukan proses pengkodean. Kode-kode yang berbeda jumlah bit-nya dapat didekode dengan cara mengatur kode-kode tersebut didalam pohon Huffman. Supaya hasil kompresi dapat didekompresi, maka tabel frekuensi/ probabilitas simbol-simbol harus disimpan pada file hasil kompresi.

Pada teknik *Adaptive* Huffman Coding tidak diperlukan penyimpanan tabel frekuensi simbol-simbol, karena model berupa pohon Huffman akan dibentuk dan diperbaharui secara *real-time* selama berlangsungnya proses kompresi atau dekompresi.

Teknik kompresi yang bersifat *adaptive* memiliki alur proses sebagai berikut :



Gambar 3 Kompresi secara *adaptive*

Arithmetic Coding tidak menggunakan suatu kode spesifik untuk menggantikan suatu simbol tertentu. Arithmetic Coding menggantikan seluruh aliran simbol yang masuk dengan sebuah bilangan pecahan. Hasil dari proses pengkodean aritmetik

berupa sebuah bilangan pecahan yang lebih besar atau sama dengan nol dan lebih kecil dari satu. Bilangan ini dapat didekode kembali untuk menghasilkan simbol-simbol asalnya.

Contoh pengkodean teks "BILL GATES" dengan pengkodean Aritmatik:

Simbol	Probabilitas	Wilayah
Spasi	1/10	$0.00 \leq r < 0.10$
A	1/10	$0.10 \leq r < 0.20$
B	1/10	$0.20 \leq r < 0.30$
E	1/10	$0.30 \leq r < 0.40$
G	1/10	$0.40 \leq r < 0.50$
I	1/10	$0.50 \leq r < 0.60$
L	2/10	$0.60 \leq r < 0.80$
S	1/10	$0.80 \leq r < 0.90$
T	1/10	$0.90 \leq r < 1.00$

Simbol Baru	Nilai low	Nilai high
	0.0	1.0
B	0.2	0.3
I	0.25	0.26
L	0.256	0.258
L	0.2572	0.2576
Spasi	0.25720	0.25724
G	0.257216	0.257220
A	0.2572164	0.2572168
T	0.25721676	0.2572168
E	0.257216772	0.257216776
S	0.2572167752	0.2572167756

Dari contoh diatas bilangan pecahan 0.2572167752, yaitu bilangan yang terakhir diperoleh setelah teks input habis, merupakan kode hasil dari proses kompresi teks “BILL GATES”.

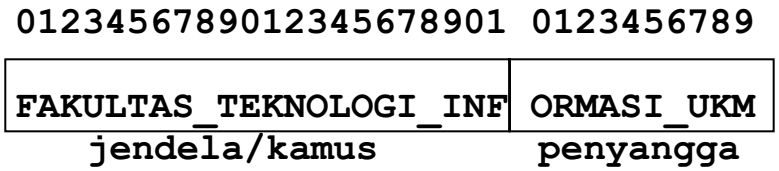
Algoritma penghitungan nilai low dan high adalah :

```
low  = 0.0;
high = 1.0;
while ((c = getc(input)) != EOF) {
        range = high - low;
        high  = low + range * high_range(c);
        low   = low + range * low_range(c);
}
```

```
output (low);
```

LZSS merupakan salah satu teknik kompresi yang menggunakan model berdasarkan kamus *(dictionary based).* Pada model ini sebuah frase yaitu sekumpulan simbol-simbol didalam penyangga akan dikodekan menjadi sebuah penunjuk. Penunjuk tersebut mengacu pada suatu posisi didalam kamus yang menyimpan frase yang diwakilinya (frase yang sama dengan frase didalam penyangga yang akan dikodekan). Kompresi terjadi karena penunjuk menggunakan jumlah bit yang lebih sedikit daripada frase yang diwakilinya. Kamus berupa sebuah blok besar yang merupakan jendela terhadap teks yang terakhir diproses. Penunjuk terdiri dari posisi frase didalam jendela/kamus dan panjang dari frase.

LZSS menggunakan sebuah bit pendahuluan untuk menunjukkan apakah kode merupakan sebuah karakter biasa atau sebuah penunjuk ke jendela. Bit 0 digunakan untuk mendahului sebuah karakter dan bit 1 digunakan untuk mendahului sebuah penunjuk. Berikut ini adalah contoh keadaan jendela dan penyangga :



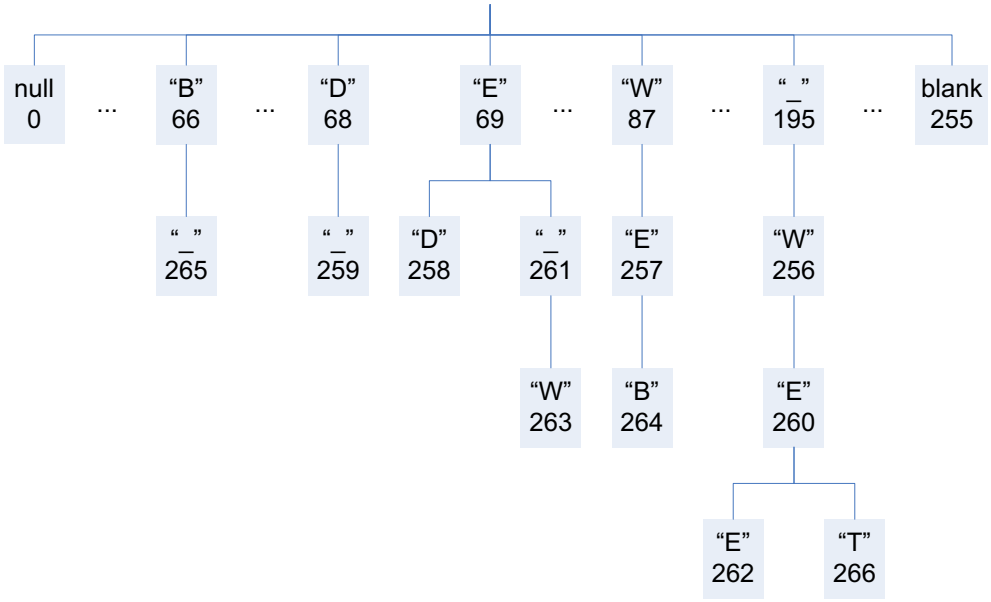
Gambar 4 Posisi frase didalam jendela dan penyangga

Dari contoh diatas, frase “AS” pada penyangga (*buffer*) akan dikodekan sebagai penunjuk terhadap posisi frase “AS” pada jendela/kamus.

Teknik kompresi LZW merupakan turunan dari teknik LZ78. Pada LZW digunakan kamus yang terdiri dari frase-frase yang terus-menerus dimasukkan kedalam kamus selama proses pengkodean/pendekodean. Frase baru akan terus-menerus dimasukkan kedalam kamus sampai tempat yang dicadangkan untuk kamus habis terpakai. Kamus pada LZW sudah diisi dengan frase-frase yang terdiri dari sebuah karakter yaitu seluruh karakter ASCII yang ada sebelum proses pengkodean/pendekodean dimulai. Frase-frase tersebut memiliki kode sesuai dengan kode ASCII dari karakter-karaktemya. Dengan cara ini maka tidak ada satu karakterpun yang tidak dapat langsung dikodekan oleh pengkode LZW. Kamus berupa sebuah pohon cabang-banyak *(multiway tree),* dan digunakan *hash* function dalam mencari simpul anak. Struktur yang digunakan LZW adalah sbb:

```
struct dictionary {
        int code_value;
        int parent_code;
        char character;
} dict[ TABLE_SIZE];
```

Contoh kamus LZW adalah sbb:



LZW12 mengubah karakter / substring input menjadi kode yang berukuran 12 bit. LZW15V merupakan modifikasi dari teknik kompresi LZW, dimana ukuran bit yang digunakan untuk menyimpan sebuah data karakter / substring dapat bervariasi, mulai dari 9 bit sampai dengan 15 bit.

Pada contoh kamus berbentuk *multiway tree* diatas, maka karakter 'W' akan menghasilkan kode berupa angka 87 yang merupakan kode ASCII untuk karakter 'W' itu sendiri. Kode yang dihasilkan berukuran 9 bit, hal ini berarti ukuran data bertambah sebanyak 1 bit. Tetapi sebuah substring "WEB" akan menghasilkan kode berupa angka 264 yang juga berukuran 9 bit, yang berarti penghematan (8 bit * 3 karakter) – 9 = 15 bit bila dibandingkan dengan data yang tidak dikompresi.

Untuk melakukan proses I/O file berorientasi bit, digunakan teknik *blocked I/O,* dengan struktur sbb :

```
typedef  struct bit_file {
        FILE *file;
        unsigned char mask;
        int rack;
        int pacifier_counter;
} BIT_FILE;
```

3. Analisis Teknik Kompresi Data

Huffman Coding merupakan teknik kompresi data yang paling hemat didalam penggunaan memori dan prosesor. Karena Huffman Coding bersifat statis maka

sebelum memulai proses pengkodean, file input harus dibaca terlebih dahulu untuk membentuk tabel frekuensi simbol-simbol. Kemudian tabel frekuensi simbol ini juga harus disimpan pada file output untuk dipergunakan pada proses pendekodean. Ada satu lagi kelemahan Huffman Coding yang juga berlaku pada semua teknik kompresi data yang menggunakan model statistik, yaitu tidak dapat atau sangat lemah dalam mengkompresi data yang betul-betul berdistribusi seragam *(purely uniform distribution)*.

Pada *Adaptive* Huffman Coding tidak diperlukan langkah pendahuluan membaca file input sebelum proses pengkodean dan tidak perlu menyimpan tabel frekuensi simbol didalam file hasil kompresi. Kekurangannya selama proses pengkodean dan pendekodean program harus melakukan pemeliharaan dan penyesuaian-penyesuaian yang cukup kompleks terhadap pohon Huffman.

Arithmetic Coding memiliki tingkat kompresi yang lebih baik atau minimal sama dengan Huffman Coding, tetapi waktu kompresi dan dekompresinya lambat karena banyaknya proses perhitungan yang harus dilakukan.

Teknik kompresi LZSS memiliki proses dekompresi yang sangat cepat, dan menghasilkan rasio kompresi yang sangat baik ketika digunakan untuk mengkompresi file teks.

Pada teknik kompresi LZW, kamus awalnya telah diisi dengan 256 karakter ASCII sehingga dapat langsung melakukan pengkodean sejak frase pertama dari file input dibaca. Karena pengkode selalu dapat menghasilkan kode, dan tidak pernah menyimpan karakter biasa ke file output maka tidak diperlukan *prefix* bit seperti pada LZSS. Berdasarkan model kamus yang digunakan, pada saat pengkodean teknik LZW tidak perlu menyimpan informasi panjang frase pada kode output, karena informasi tersebut tidak diperlukan oleh pendekode.

Jika teknik kompresi data berbasis kamus LZSS dan LZW akan digunakan pada domain data yang spesifik, maka dapat diperoleh rasio kompresi yang lebih baik lagi dengan cara mengisi terlebih dahulu (*preloaded*) kamus yang akan digunakan dengan data-data yang sering digunakan dalam domain tersebut.

4. Pengujian

Untuk keperluan pengujian dibuat sebuah program yang secara otomatis akan mengeksekusi program-program *executable* dari masing-masing metode kompresi data yang diuji. Pengujian yang dilakukan meliputi rasio kompresi, waktu yang diperlukan pada saat proses kompresi, waktu yang diperlukan pada saat proses dekompresi dan verifikasi kebenaran proses kompresi dan dekompresi.

Spesifikasi perangkat keras yang digunakan pada saat pengujian adalah sbb :

- Prosesor : AMD Athlon XP 2100
- Memory : 256 MB SDRAM

- Hardisk : Seagate 40 GB 5400 RPM

Ketika pengujian diperoleh waktu yang diperlukan pada saat kompresi dan dekompresi sedikit berubah-ubah, maka pengujian untuk setiap metode dilakukan sebanyak sepuluh kali untuk file data yang sama, kemudian diambil nilai rata-ratanya.

Rasio kompresi dihitung menggunakan rumus :

rasio = 100% - (ukuran terkompresi / ukuran asal) x 100%

yang berarti semakin tinggi nilai rasio semakin baik pula tingkat kompresinya. Jika nilainya nol berarti tidak terjadi kompresi, dan jika nilainya negatif berarti proses kompresi telah membuat file output semakin besar.
Waktu yang diperlukan pada saat proses kompresi dan dekompresi disajikan dalam satuan detik, dengan ketepatan sampai seperseratus detik.

Berbagai jenis file yang diuji adalah sbb :

Kelompok	**Jumlah File**	**Ukuran Total**	**File Extension**
Grafik	9	632698 byte	BMP, ICO, WMF
Suara	8	611281 byte	WAV, MID
Program	8	628767 byte	COM, EXE, SYS, DLL, 386
Teks	9	618113 byte	TXT, DOC, INI, C
Jumlah	34	2490859 byte	

Berikut ini contoh output pengujian untuk salah satu teknik, yaitu LZW15V :

File Name	Compress Time	Expand Time	Original Size	Packed Size	Ratio	Result
T\G\SETUP.BMP	1.06	0.74	308280	63995	80%	Passed
T\G\BUBBLES.BMP	0.21	0.09	2118	437	80%	Passed
T\G\SCRSHOT.BMP	0.59	0.48	184248	34279	82%	Passed
T\G\FACTORY.BMP	0.38	0.33	44278	28138	37%	Passed
T\G\WINUPD.ICO	0.25	0.23	10134	4357	58%	Passed
T\G\AGREE.WMF	0.32	0.29	26454	18511	31%	Passed
T\G\BUILDING.WMF	0.24	0.23	10806	4678	57%	Passed
T\G\CHAMPGNE.WMF	0.35	0.21	34806	10828	69%	Passed
T\G\EAGLE.WMF	0.28	0.26	11574	9185	21%	Passed
T\S\TADA.WAV	1.14	0.92	171100	164594	4%	Passed
T\S\FLY2MOON.MID	0.41	0.31	56521	36414	36%	Passed
T\S\HAPPYBDY.MID	0.25	0.23	9151	7758	16%	Passed
T\S\NOTIFY.WAV	0.8	0.7	119384	116570	3%	Passed
T\S\BABYBABY.MID	0.27	0.26	24515	9237	63%	Passed
T\S\SONGBIRD.MID	0.36	0.34	52738	28427	47%	Passed
T\S\DING.WAV	0.62	0.53	80856	82492	-2%	Passed
T\S\CHORD.WAV	0.66	0.59	97016	89808	8%	Passed

File Name	Compress Time	Expand Time	Original Size	Packed Size	Ratio	Result
T\E\CDPLAYER.EXE	0.56	0.51	106496	56726	47%	Passed
T\E\EDIT.COM	0.53	0.46	69902	57496	18%	Passed
T\E\COMMAND.COM	0.57	0.45	93880	55014	42%	Passed
T\E\WIN.COM	0.29	0.27	24791	12404	50%	Passed
T\E\FREECELL.EXE	0.3	0.29	28576	19966	31%	Passed
T\E\HIMEM.SYS	0.34	0.29	33191	14338	57%	Passed
T\E\SYMEVNT.386	0.62	0.53	116283	64853	45%	Passed
T\E\GDI32.DLL	0.17	0.73	155648	106322	32%	Passed
T\T\PROGRAMS.TXT	0.31	0.33	47714	21234	56%	Passed
T\T\TES1ERC	0.24	0.23	14430	6455	56%	Passed
T\T\SYS1EM.INI	0.25	0.22	2501	1645	35%	Passed
T\T\ WIN.INI	0.25	0.22	9892	5509	45%	Passed
T\T\SETUPLOG.TXT	0.52	0.41	104718	32708	69%	Passed
T\T\GENERAL.TXT	0.34	0.31	39907	17978	55%	Passed
T\T\HARDWARE.TXT	0.31	0.3	39527	17372	57%	Passed
T\T\BAB2.DOC	0.68	0.62	188416	66543	65%	Passed
T\T\BAB3.DOC	0.86	0.62	171008	72009	58%	Passed

Total elapsed time : 321.00000 seconds

Total files: 34
Total passed: 34
Total failed: 0

Jika kita perhatikan, rincian output pengujian untuk teknik LZW15V diatas, terdapat rasio kompresi yang negatif, yang berarti ukuran file hasil kompresi malahan menjadi lebih besar. Hal seperti itu terjadi pula pada teknik-teknik kompresi lainnya ketika mengkompresi file-file tertentu. Walaupun demikian file tetap dapat didekompresi dengan baik, yang ditunjukkan dengan hasil Passed.

Ringkasan hasil pengujian beberapa teknik kompresi dan dekompresi data, sbb :

Teknik	Pengujian	Grafik	Suara	Program	Teks	Rata-2
Huffman	Rasio	44%	22%	29%	36%	33%
	Waktu Kompresi	0.38	0.44	0.48	0.43	0.43
	Waktu Dekompresi	0.30	0.39	0.42	0.32	0.36
Adaptive Huffman	Rasio	46%	23%	32%	38%	35%
	Waktu Kompresi	0.42	0.60	0.57	0.48	0.52
	Waktu Dekompresi	0.35	0.54	0.53	0.44	0.46
Aritmetik	Rasio	45%	22%	30%	36%	33%
	Waktu Kompresi	0.64	0.73	0.72	0.63	0.68
	Waktu Dekompresi	0.96	1.08	1.09	0.99	1.03
LZSS	Rasio	56%	23%	45%	58%	46%
	Waktu Kompresi	0.65	0.71	0.69	0.67	0.68
	Waktu Dekompresi	0.32	0.45	0.40	0.34	0.38
LZW12	Rasio	39%	-4%	17%	46%	25%
	Waktu Kompresi	0.46	0.58	0.52	0.43	0.50

Teknik	Pengujian	Grafik	Suara	Program	Teks	Rata-2
	Waktu Dekompresi	0.38	0.51	0.45	0.37	0.43
LZW15V	Rasio	57%	22%	40%	55%	44%
	Waktu Kompresi	0.41	0.56	0.50	0.42	0.47
	Waktu Dekompresi	0.32	0.49	0.44	0.36	0.41
Rata-rata	Rasio	51%	18%	32%	45%	36%
	Waktu Kompresi	0.49	0.60	0.58	0.51	0.55
	Waktu Dekompresi	0.44	0.58	0.55	0.47	0.51

Nilai yang diarsir pada tabel diatas menunjukkan nilai terbaik pada item pengujian tersebut, dibandingkan teknik kompresi data lainnya.

5. Kesimpulan dan Saran

Dari hasil pengujian, reliabilitas teknik-teknik kompresi cukup dapat diandalkan, dimana ketika proses kompresi menghasilkan nilai yang rasio yang negatif, file tetap dapat didekompresi dengan baik.

Bila rata-rata hasil pengujian diatas disusun dalam peringkat yang dimulai dari yang terbaik, diperoleh data sebagai berikut :

Peringkat	I	II	III	IV	V	VI
Rasio	LZSS	LZW15V	A. Huff	Aritmatik & Huffman	LZW12	-
Waktu Kompresi	Huffman	LZW15V	LZW12	A. Huff	Aritmatik & LZSS	-
Waktu Dekompresi	Huffman	LZSS	LZW15V	LZW12	A. Huff	Aritmatik

Terlihat bahwa waktu kompresi dan dekompresi yang paling cepat dihasilkan oleh teknik kompresi Huffman Coding, dan rasio kompresi yang terbaik diperoleh dari teknik LZSS.

Meskipun waktu kompresi dan dekompresinya paling baik, tetapi Huffman Coding memiliki rasio kompresi yang tidak terlalu baik. Demikian juga LZSS meskipun rasio kompresinya paling baik, tetapi waktu kompresinya lambat. Disini terlihat bahwa Huffman Coding lebih cocok untuk data kerja yang sering (atau secara *real-time*) harus dikompresi dan didekompresi. LZSS cocok untuk mengkompresi data yang akan disimpan sebagai arsip.

Terlihat juga bahwa LZWI5V paling stabil, sehingga untuk penggunaan secara umum lebih direkomendasikan.

Daftar Pustaka

[Sal04] Salomon, David, *Data Compression: The Complete Reference*, Springer, 2004.

[Nel95] Nelson, Mark, Gailly, Jean-loup, *The Data Compression Book 2nd ed*, M&T Books, 1995.

[Sch00] Schild, Herbert, *C The Complete Reference, 4th ed.*, osborne / McGraw-Hill, 2000.

[Wik06] Wikipedia, *Data Compression,* http://en.wikipedia.org/wiki/ Data_compression, Accessed Dec 2006.