*Media Informatika, Vol. 5, No. 2, Desember 2007, 129-139*
*ISSN: 0854-4743*

# DATA COMPRESSION CODING USING STATIC AND DYNAMIC METHOD OF SHANNON-FANO ALGORITHM

**Romi Wiryadinata**
*Mahasiswa Sekolah Pascasarjana Universitas Gadjah Mada, Yogyakarta*
*Website: http://wiryadinata.web.id; E-mail: romi_wiryadinata@yahoo.com*

**ABSTRAK**

*Tulisan ini membahas tentang teknik kompresi data dengan menggunakan metode shannon-fano dengan membandingkan antara teknik statik dan teknik dinamik dengan menggunakan data yang sama. Data yang dibandingkan adalah berupa data pesan dalam bentuk teks 'gadjahmada' yang diolah sedemikian rupa menjadi kode ASCII dan runtun biner. Untuk pesan teks yang pendek metode statik lebih cocok digunakan agar menghasilkan kode dan runtun data yang lebih sedikit tetapi membutuhkan waktu yang cukup lama untuk scanning huruf, sedangkan untuk pesan teks yang panjang metode dinamik lebih cocok digunakan untuk mempersingkat waktu komputasi tetapi dengan hasil runtun data keluaran yang sedikit lebih besar dari metode statik. Secara keseluruahan algoritma Shannon-Fano menghasilkan runtun data yang lebih sedikit dari total runtun data yang masuk (dengan asumsi satu data/huruf membutuhkan alokasi 8 bit). Total bit keluaran dengan metode statik dapat mencapai 24 bit dengan rasio kompresi 0.3 dan dengan menggunakan metode dinamik dapat menghasilkan 34 bit runtun data output dengan rasio kompresi 0.425.*

*Kata kunci: Enkripsi, Kompresi data, Shannon-Fano statik dan dinamik*

## 1. PENDAHULUAN

Perkembangan IT (*Information Technology*) yang semakin cepat memberikan tantangan yang cukup besar akan terbatasnya media penyimpanan (*storage*), *bandwith*, dan kecepatan *time-delay*. Untuk menjawab tantangan tersebut dibutuhkan teknik-teknik pegolahan data, salah satu teknik pengolahan data tersebut adalah dengan kompresi data agar data hasil yang dikeluarkan dapat dimampatkan tanpa mengurangi besarnya informasi yang ada antara data sebelum dan sesudah pengolahan.

Media penyimpanan atau biasa dikenal dengan HDD (*Hard Disk Drive*) yang semakin besar tidak akan menjawab kebutuhan teknologi informasi jika data berkas (*file*) yang digunakan juga semakin besar. Begitu juga dengan *bandwith*, untuk data audio membutuhkan sekitar 4 kHz dan data video membutuhkan sekitar 6 MHz. Terlebih saat ini internet, telepon, dan telemetri telah menjadi suatu trend dan kebutuhan sebagai media komunikasi dalam bidang IT, komputer dan elektronika untuk selalu membutuhkan dan mencari koneksi yang cepat, tetapi koneksi yang cepat tidak akan berpengaruh jika data yang ditransferkan

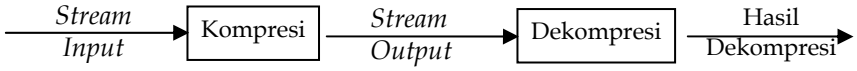
sebanding atau justru malah lebih besar. Hal tersebut akan lebih terasa jika internet digunakan untuk *download/upload* data, *browsing*, transaksi data, ftp, dan lain-lain.

Secara spesifik, kompresi data bertujuan untuk mereduksi tempat (*space*) penyimpanan data dan mereduksi waktu untuk mentransmisikan *file* atau data yang memiliki kapasitas besar.

## 2. DASAR TEORI

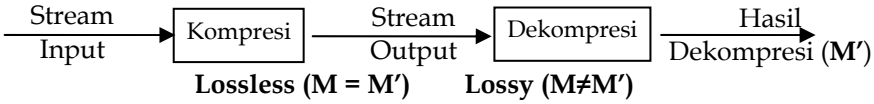
### 2.1 Kompresi data

Kompresi data adalah proses mengubah *stream* data masukan menjadi *stream* data keluaran agar menjadi lebih kecil, sedangkan proses pembalikan data yang sudah terkompres menjadi data semula disebut dengan dekompresi. Gambar berikut mengilustrasikan proses kompresi dan dekompresi data.




Gambar 1. Proses pengolahan data (Held, G., 1998)

Secara umum kompresi dibedakan menjadi *lossless compression* dan *lossy compression*. *Lossless compression* tidak terjadi perubahan antara *stream* data masukan dan *stream* data keluaran, proses kompresi secara *lossless* ini merupakan salah satu klasifikasi yang sering ditemukan pada kompresi jenis *text*, *executable file*, dan beberapa data citra (.gif, .png, .tiff, .bmp, .pcx dan lain-lain) (Richardson, I., 2007). Sedankan *lossy compression* memiliki perubahan antara *stream* data masukan dan *stream* data keluaran, pada proses secara *lossy* ini banyak digunakan pada kompresi jenis citra dan audio. Gambar 2 berikut menjelaskan proses klasifikasi kompresi.




Gambar 2. Proses kompresi secara *lossless* (Held, G., 1998)

Ada beberapa metode kompresi yang dapat digunakan, metode-metode tersebut dikelompokan dalam empat pembagian berdasarkan metode atau langkah kerjanya, berikut ini adalah klasifikasi beberapa metode kompresi.

a. algoritma kompresi dasar
   - *Run-length encoding*
   - *Differential encoding*
b. *statistical compression*
   - Huffman (*static* dan *dynamic*)
   - Shannon-Fano (*static* dan *dynamic*)
   - Arithmatic (*static* dan *dynamic*)

c. *dictionary based compression*
   - LZ77
   - LZ78
   - LZW

d. kompresi dengan transformasi
   - DCT (*Discrete Cosine Transform*)
   - *Wavelet* (gelombang-singkat)

## 2.2 Teori informasi

Setelah pemilihan metode yang akan digunakan telah diketahui maka selanjutnya mempertimbangkan beberapa persamaan-persamaan yang berhubungan dalam kompresi data. Menurut Shannon, entropi dari sebuah sumber informasi H(S) dinyatakan pada persamaan (3) dimana E adalah suatu ukuran untuk menentukan berapa banyak informasi dikodekan menjadi sebuah pesan dengan kata lain entropi adalah suatu ukuran ketidakpastian dalam sebuah pesan dan memberikan jumlah aktual bit isi informasi dalam sumber pesan (Menezes, A.J., et al., 1997). Dalam hal ini yang perlu dicermati adalah karena informasi dan ketidakpastian merupakan suatu konsep yang sepadan. Satuan entropi (*coding theory*) dinyatakan dalam *bits per symbol*, satuan ini ditentukan oleh dasar algoritma, bahwa bilangan basis dua adalah berbentuk *binary* (*bit*) dan bilangan basis 10 adalah berbentuk *decimal* (*digit*) (Rhee, M.Y., 2003).

Semua media baik berupa teks, grafik, audio, atau video mempunyai redudansi. Redudansi (berulang) terjadi ketika representasi dari sebuah media yang berkapasitas x bytes dan y bytes (y<x), maka x merupakan suatu redudansi relatif terhadap y. Dalam bentuk lain, redudansi terjadi jika representasi dari isi *capture* suatu media tidak dapat dipahami oleh manusia kemudian dihapus tetapi tidak akan mempengaruhi isi media aslinya, seperti dalam kasus penangkapan frekuensi audio di luar jangkauan pendengaran manusia dapat dihindari tanpa banyak mempengaruhi kualitas isinya.

Redudansi dapat diketahui dengan terlebih dahulu mengetahui secara bertahap nilai-nilai *Compression Factor* (FK), *Compression Ratio* (RK), *Enthropy* (E) adalah jumlah informasi rata-rata dalam setiap huruf, baru kemudian diketahui nilai *Redudancy* (R) atau dapat dikatakan sebagai tingkat optimalisasi suatu metode. Berikut ini adalah penjelasan dan penjabaran persamaannya:

$$FK = \frac{ukuran\ runtun\ bit\ input}{ukuran\ runtun\ bit\ output} \tag{1}$$

$$RK = \frac{ukuran\ runtun\ bit\ output}{ukuran\ runtun\ bit\ input} \tag{2}$$

$$E = \sum_{i=1}^{n} P_i.\log_2(P_i) = \sum_{i=1}^{n} P_i.\log_2(\frac{1}{P_i}) \tag{3}$$

$$R = \log_2(n) + \sum_{i=1}^{n} P_i.\log_2(P_i) \tag{4}$$

Misalkan dalam sebuah pesan memiliki probabilitas karakter "a" muncul sebanyak 1/16 maka isi informasinya adalah 4 bit, sehingga karakter string "aaaaa" mempunyai total isi 20 bit. Berbeda dengan kode ASCII yang menggunakan 8 bit untuk merepresentasikan karakter "aaaaa" sehingga total isi menjadi 40 bit. Dalam proses kompresi data, kemunculan probabilitas suatu data yang paling banyak harus dikodekan menjadi bit biner yang paling sedikit untuk memperkecil nilai perhitungan yang menyebabkan semakin kecilnya hasil bit informasi yang diproses.

### 2.3 Shannon-Fano *coding*

Algoritma Shannon-Fano *coding* ditemukan oleh Claude Shannon (bapak teori informasi) dan Robert Fano pada tahun 1949. Pada saat itu metode ini merupakan metode yang paling baik tetapi hampir tidak pernah digunakan dan dikembangkan lagi setelah kemunculan algoritma Hufman. Pada dasarnya metode ini menggantikan setiap simbol dengan sebuah alternatif kode biner yang panjangnya ditentukan berdasarkan probabilitas dari simbol tersebut (Rhee, M.Y., 2003).

## 3. LANGKAH PERCOBAAN

Langkah percobaan terbagi atas dua, yang pertama menggunakan metode statik dan kemudian dibandingkan dengan metode adaptif. Berikut ini algoritma langkah percobaan dari algoritma Shannon-Fano.

a. Algoritma Shannon-Fano tak adaptif (statik), algoritma ini menggunakan metode statistika untuk menentukan jumlah probabilitas huruf. Berikut ini adalah urutan langkah algoritma secara statik
   1. Menentukan jumlah probabilitas huruf,
   2. Mengurutkan huruf pada suatu tabel,
   3. Meletakan pada baris pertama untuk huruf yang memiliki probabilitas tertinggi,
   4. Melakukan pembagian terhadap tabel atas dua kelompok dengan probabilitas yang sama atau saling mendekati,
   5. Memberikan kode nol pada kelompok pertama dan kode nol pada kelompok kedua,
   6. Mengulangi langkah empat dan lima sampai tidak ada lagi kemungkinan untuk membagi kelompok-kelompok tersebut dalam dua sub kelompok lagi.

b. Algoritma Shannon-Fano adaptif (dinamik), hampir sama dengan algoritma statik hanya memiliki tambahan *escape* pada setiap langkah data masuk. Berikut ini adalah urutan langkah algoritma secara dinamik.
   1. Menyimpan statistik dari huruf pertama sampai posisi aktual pada sebuah tabel,
   2. Menggabungkan perhitungan statistik dan kode dalam pengkodean pesan,

3. Menganggap kosong pada saat awal tabel, karena itu digunakan satu huruf khusus yang disebut sebagai ***escape***.
4. Mengurutkan data berdasarkan runtun masukan,
5. Menyusun kembali berdasarkan kode ASCII,
6. Menambah satu pada jumlah *escape* setiap ada pemunculan huruf yang baru, sedangkan jika huruf tersebut pernah muncul maka jumlah *escape* tidak bertambah yang bertambah hanya pada huruf yang muncul kembali.

## 4. PERCOBAAN DAN DISKUSI

Pada percobaan *input stream* menggunakan pesan teks "gadjahmada", pesan teks terdiri dari lima huruf **a**, **d**, **g**, **h**, **j**, dan **m**. Pemilihan teks gadjahmada karena memiliki rangkaian huruf yang pendek dan memiliki dua huruf perulangan, sehingga langkah algoritma sudah terpenuhi.

*a. Algoritma Shannon-Fano tak adaptif (statik)*

Berikut ini (Tabel 1) adalah hasil dari setiap langkah kompresi dengan metode statik yang disajikan berurutan secara langsung dalam bentuk tabel mulai dari langkah satu sampai langkah enam.

Tabel 1. Probabilitas huruf/data/pesan.

| *Huruf* | *Jumlah Probabilitas* |
|---|---|
| **a** | **4** |
| d | 2 |
| g | 1 |
| h | 1 |
| j | 1 |
| m | 1 |

Tabel 1 di atas adalah hasil langkah pertama yaitu mengurutkan data huruf berdasarkan probabilitas. Berikut ini adalah langkah selanjutnya yang ditunjukan pada (Tabel 2), huruf pertama dengan probabilitas tertinggi diberikan kode nol dan kode yang lainnya menjadi sama dengan satu.

Tabel 2. Pemberian kode kelompok pertama

| *Huruf* | *Jumlah Probabilitas* | *Kode* |
|---|---|---|
| **A** | **4** | **0** |
| D | 2 | 1 |
| G | 1 | 1 |
| H | 1 | 1 |
| J | 1 | 1 |
| M | 1 | 1 |

Untuk (Tabel 3) huruf berikutnya diberikan kode nol tetapi untuk huruf lainnya menyesuaikan berdasarkan pembagian jumlah huruf, untuk jumlah paling sedikit diletakan pada bagian atas dengan memberikan kode nol dan lainnya diberikan kode satu.

Tabel 3. Pemberian kode kelompok ke-dua

| *Huruf* | *Jumlah Probabilitas* | *Kode* | |
|---|---|---|---|
| a | 4 | 0 | |
| **d** | **2** | **1** | **0** |
| **g** | **1** | **1** | **0** |
| h | 1 | 1 | 1 |
| j | 1 | 1 | 1 |
| m | 1 | 1 | 1 |

Pada langkah berikutnya (Tabel 4) terjadi perpotongan kembali antara huruf d-g dan h-j-m, maka pemberian kode nol diletakan pada bagian paling atas untuk setiap perpotongan dan kode satu diberikan untuk perpotongan baris berikutnya. Berikut ini (Tabel 4) adalah hasil pengkodean pada kelompok ketiga.

Tabel 4. Pemberian kode kelompok ke-tiga

| *Huruf* | *Jumlah Probabilitas* | *Kode* | | |
|---|---|---|---|---|
| A | 4 | 0 | | |
| **d** | **2** | **1** | **0** | **0** |
| g | 1 | 1 | 0 | 1 |
| **h** | **1** | **1** | **1** | **0** |
| j | 1 | 1 | 1 | 1 |
| m | 1 | 1 | 1 | 1 |

Tabel 5. Pemberian kode kelompok ke-empat

| *Huruf* | *Jumlah Probabilitas* | *Kode* | | | |
|---|---|---|---|---|---|
| a | 4 | 0 | | | |
| d | 2 | 1 | 0 | 0 | |
| g | 1 | 1 | 0 | 1 | |
| h | 1 | 1 | 1 | 0 | |
| **j** | **1** | **1** | **1** | **1** | **0** |
| m | 1 | 1 | 1 | 1 | 1 |

Tabel 5 hanya menyisakan dua huruf sehinga perpotongan tidak dilakukan perpotongan serumit (Tabel 4) dan pemberian kode hanya menyisakan nol dan satu. Sampai langkah ini semua huruf yang masuk untuk satu pesan teks sudah lengkap dan berikut ini (Tabel 6) adalah hasil pengkodean berrdasarkan metode statik.

Tabel 6. Hasil kompresi statik

| *Huruf* | *Jumlah Probabilitas* | *Kode* | | | | *Hasil* |
|---|---|---|---|---|---|---|
| a | 4 | 0 | | | | **0** |
| d | 2 | 1 | 0 | 0 | | **1 0 0** |
| g | 1 | 1 | 0 | 1 | | **1 0 1** |
| h | 1 | 1 | 1 | 0 | | **1 1 0** |
| j | 1 | 1 | 1 | 1 | 0 | **1 1 1 0** |
| m | 1 | 1 | 1 | 1 | 1 | **1 1 1 1** |

Tanpa kompresi, teks ”gadjahmada” memiliki 10 huruf dikali 8 bit sama dengan 80 bit dengan asumsi 1 huruf membutuhkan alokasi sekitar 8 bit tetapi dengan kompresi metode static, teks “gadjahmada” berubah menjadi 101 0 100 1110 0 110 1111 0 100 0, sehingga total bit dalam satu pesan teks menjadi sama dengan 24 bit. Kode pesan yang dihasilkan untuk pesan teks ‘gadjahmada’ dengan algoritma Shannon-Fano statik adalah A9CDE8.

*b. Algoritma Shannon-Fano adaptif (dinamik)*

Tabel 7. Inisialisasi nilai awal

| *Huruf* | *Jumlah probabilitas* | *Kode* |
|---|---|---|
| Escape | 0 | 0 |

Tabel 7 menjelaskan bahwa pada saat awal sebelum pesan teks masuk, dianggap tabel masih sebagai tabel kosong sehingga *escape* diberikan kode nol.

Tabel 8. Data huruf pertama (g)

| *Huruf* | *Jumlah probabilitas* | *Kode* |
|---|---|---|
| Escape | 1 | 0 |
| g | 1 | 1 |

Pada (Tabel 8) di atas, setelah huruf pertama masuk maka nilai *escape* berubah menjadi satu dan huruf g diberikan kode satu. Untuk huruf/data berikutnya, kode ASCII huruf a lebih kecil dari kode ASCII g sehingga pada (Tabel 9) huruf a diletakan di atas huruf g. Kode yang diberikan berikutnya dimulai pada baris setelah escape dengan menambahkan kode nol dan huruf berikutnya diberikan kode satu, nilai *escape* ditambah satu menjadi sama dengan dua.

Tabel 9. Data huruf ke-dua (a)

| *Huruf* | *Jumlah probabilitas* | *Kode* |
|---|---|---|
| Escape | 2 | 0 |
| a | 1 | 10 |
| g | 1 | 11 |

Seperti pada langkah atau tabel sebelumnya, pada (Tabel 10) huruf yang masuk diurutkan berdasarkan kode ASCII kemudian ditambahkan kode nol dan baris berikutnya ditambahkan kode satu, nilai *escape* ditambah satu menjadi sama dengan tiga. Perubahan kode disesuaikan berdasarkan logika biner.

Tabel 10. Data huruf ketiga (d)

| *Huruf* | *Jumlah probabilitas* | *Kode* |
|---|---|---|
| Escape | 3 | 0 |
| a | 1 | 10 |
| d | 1 | 110 |
| g | 1 | 111 |

Tabel 11. Data huruf ke-empat (j)

| *Huruf* | *Jumlah probabilitas* | *Kode* |
|---|---|---|
| Escape | 4 | 0 |
| a | 1 | 100 |
| d | 1 | 101 |
| g | 1 | 110 |
| j | 1 | 111 |

Kode ASCII huruf j lebih kecil dari huruf-huruf sebelumnya sehingga pada (tabel 11) huruf j tetap pada baris terkahir, nilai *escape* ditambah satu menjadi sama dengan empat. Untuk pengkodean empat data masukan, maka kode disesuaikan dengan logika biner menggunkan dua bit pada huruf a-d dan dua bit berikutnya pada huruf g-j.

Berbeda dengan langakah sebelumnya, pada (Tabel 12) huruf a muncul untuk yang ke-dua kalinya sehingga nilai jumlah *escape* tetap dan jumlah probabilitas huruf a ditambah satu menjadi sama dengan dua.

Tabel 12. Data huruf ke-lima (a)

| *Huruf* | *Jumlah probabilitas* | *Kode* |
|---|---|---|
| Escape | 4 | 0 |
| a | 2 | 100 |
| d | 1 | 110 |
| g | 1 | 1110 |
| j | 1 | 1111 |

Huruf yang muncul berikutnya adalah huruf h, maka langkah pengerjaan pada (Tabel 13) sama dengan seperti pada tabel sebelumnya (tabel 7) sampai (Tabel 11) yang tidak ada perubahan pada jumlah probabilitas.

Begitu juga dengan huruf berikutnya pada (Tabel 14) yang muncul adalah huruf m dengan jumlah probabilitas satu sehingga langkah perubahan kode sama dengan pada sebelumnya (Tabel 7) sampai (Tabel 11) dan (Tabel 13).

Tabel 13. Data huruf ke-enam (h)

| *Huruf* | *Jumlah probabilitas* | *Kode* |
|---|---|---|
| Escape | 5 | 0 |
| a | 2 | 100 |
| d | 1 | 101 |
| g | 1 | 110 |
| h | 1 | 1110 |
| j | 1 | 1111 |

Tabel 14. Data huruf ke-tujuh (m)

| *Huruf* | *Jumlah probabilitas* | *Kode* |
|---|---|---|
| Escape | 6 | 0 |
| a | 2 | 100 |
| d | 1 | 101 |
| g | 1 | 1100 |
| h | 1 | 1101 |
| j | 1 | 1110 |
| m | 1 | 1111 |

Tabel 15. Data huruf ke-delapan (a)

| *Huruf* | *Jumlah probabilitas* | *Kode* |
|---|---|---|
| Escape | 6 | 0 |
| a | 3 | 100 |
| d | 1 | 101 |
| g | 1 | 1100 |
| h | 1 | 1101 |
| j | 1 | 1110 |
| m | 1 | 1111 |

Huruf yang muncul pada (Tabel 15) adalah huruf a sehingga nilai *esacape* tetap dan jumlah probabilitas huruf a ditambah satu dari yang sebelumnya sama dengan dua menjadi sama dengan tiga.

Tabel 16. Data huruf ke-sembilan (d)

| *Huruf* | *Jumlah probabilitas* | *Kode* |
|---|---|---|
| Escape | 6 | 0 |
| a | 3 | 10 |
| d | 2 | 1100 |
| g | 1 | 1101 |
| h | 1 | 1110 |
| j | 1 | 11110 |
| m | 1 | 11111 |

Begitu juga dengan kemunculan huruf berikutnya (Tabel 16) adalah huruf d untuk yang kedua kalinya, maka langkah pengerjaan sama dengan (Tabel 12) dan (Tabel 15) dengan menambahkan jumlah probabilitas huruf d menjadi sama dengan dua dan nilai *esacape* tidak memiliki perubahan.

Tabel 17. Data huruf terakhir (a)

| *Huruf* | *Jumlah probabilitas* | *Kode* |
|---|---|---|
| Escape | 6 | 0 |
| a | 4 | 10 |
| d | 2 | 1100 |
| g | 1 | 1101 |
| h | 1 | 1110 |
| j | 1 | 11110 |
| m | 1 | 11111 |

Untuk data terakhir huruf yang muncul adalah huruf a untuk yang ke-empat kalinya, sehingga (Tabel 17) di atas adalah tabel lengkap untuk satu pesan teks "gadjahmada"

Secara keseluruhan pengkodean huruf-huruf dari pesan teks "gadjahmada" dapat dilihat pada (Tabel 18) berikut, sehingga total bit pada satu pesan teks memiliki 34 bit. Hasil kompresi dengan metode dinamik ini lebih besar dari metode statik tetapi kelebihan dari metode dinamik ini, pesan teks tidak perlu dilakukan pembacaan terhadap setiap huruf dalam satu pesan, artinya data apapun yang masuk dalam metode statik dapat secara langsung diolah dan dikodekan. Hal inilah mengapa pada metode dinamik dikatakan juga sebagai metode adaptif dari algoritma Shannon-Fano.

Untuk pesan teks yang panjang metode dinamik lebih cocok digunakan karena tidak memakan waktu komputasi pada saat pembacaan pesan teks per satu huruf, tetapi untuk pesan teks yang tidak terlalu panjang maka metode statik lebih cocok digunakan karena lebih sedikit menghasilkan bit keluaran. Kedua metode pada kompresi data dengan algoritma Shannon-Fano membuktikan bahwa *output* yang dihasilkan setelah jauh lebih sedikit dibandingkan tanpa melalui kompresi. Dapat dibayngkan jika pesan teks yang panjangnya sudah mencapai ratusan sampai ribuan runtun data maka algoritma pengkompresian data akan sangat membantu dalam penyelesaian masalah seperti yang dijelaskan pada bagian pendahuluan. Kode pesan yang dihasilkan untuk pesan teks 'gadjahmada' dengan algoritma Shannon-Fano dinamik adalah 36CF5DFB2.

Tabel 18. Hasil kompresi metode dinamik

| *g* | *a* | *d* | *J* | *A* | *h* | *m* | *a* | *D* | *A* |
|---|---|---|---|---|---|---|---|---|---|
| 1101 | 10 | 1100 | 11110 | 10 | 1110 | 11111 | 10 | 1100 | 10 |
| 4 | 2 | 4 | 5 | 2 | 4 | 5 | 2 | 4 | 2 |

FK (Faktor Kompresi), metode statik: $\frac{80}{24} = 3.333$

metode dinamik: $\frac{80}{34} = 2.353$

RK (Rasio Kompresi), metode statik: $\frac{24}{80} = 0.3$

metode dinamik: $\frac{34}{80} = 0.425$

## 5. SIMPULAN

Untuk *input stream* yang sama menggunakan metode *Shannon-Fano Statik* jika dilakukan dengan cara mengurutkan simbol yang berbeda akan tetap menghasilkan jumlah bit *output* yang sama.

Jumlah bit *output* hasil kompresi menggunakan metode Shannon-Fano statik lebih sedikit dibanding jumlah bit *output* hasil kompresi metode Shannon-Fano dinamik untuk *input stream* yang sama.

Rasio kompresi metode statik lebih kecil dibandingkan metode dinamik tetapi dengan jumlah data yang berbeda perlu dipertimbangkan juga beban komputasi yang akan terjadi, terutama setelah diterapkan pada sebuah aplikasi *hardware* yang bergantung pada besarnya memori yang digunakan.

## PUSTAKA


Held, G. (1998). *Learn Encryption Techniques with Basic and C++*, Wordware Publishing Inc.

Menezes, A.J, P.C.V. Oorschot, dan S.A. Vanstone. (1997). *Handbook of Applied Cryptography*, CRC Press. Inc.

Rhee, M.Y. (2003). *Cryptograhpy Principles, Algorithms and Protocol*, John Wiley and Sons Inc.

Richardson, I. (2007). *An Overview of H.264 Advanced Video Coding*, terdapat di www.vcodex.com di akses pada bulan September 2005.