Analisis Perbandingan Kompresi dan Dekompresi Menggunakan Algoritma Shannon-Fano 2 Gram Dan Lempel Ziv Welch Pada Terjemahan Hadits Shahih Muslim

Yudistira Yoga Aji[1], Eko Darwiyanto, ST. MT[2], Gia Septiana, S.Si., M.Sc[2]

Departemen Teknik Informatika Universitas Telkom, Bandung
[1]yudisirayogaaji@gmail.com, [2]ekodarwiyanto@yahoo.com, [3]gia.septiana@gmail.com

Abstrak :
Pesatnya perkembangan teknologi komunikasi pada era globalisasi saat ini telah mendorong berkembangnya teknologi serta kebutuhan akan telekomunikasi di bidang-bidang lainnya. Salah satu kebutuhan tersebut adalah dalam penyimpanan data. Oleh karena itu, telah dikembangkan algoritma untuk memampatkan data (kompresi data). Kompresi adalah proses mengkonversikan sebuah *input* data *stream* (*stream* sumber, atau data mentah asli) menjadi data *stream* lainnya (*bit stream* hasil, atau *stream* yang telah terkompresi) yang berukuran lebih kecil. Kompresi terdapat 2 jenis lossless atau lossy. Dalam kompresi lossless, teks asli dapat disusun kembali dari data terkompresi. Algoritma Shannon-Fano 2 Gram dan algoritma Lempel Ziv Welch dapat digunakan dalam kompresi *lossless*. Hadits adalah perkataan (sabda), perbuatan, ketetapan dan persetujuan dari Nabi Muhammad SAW yang dijadikan landasan syariat Islam. Ada enam koleksi hadits, salah satunya adalah hadits Shahih Muslim yang terdiri dari 56 kitab. Tulisan ini mengimplementasikan pemampatan data teks dari terjemahan hadits Shahih Muslim menggunakan algoritma Shannon-Fano 2 Gram dan algoritma Lempel Ziv Welch. Beberapa kitab yang bervariasi ukurannya telah dipilih sebagai data uji untuk kompresi. Berdasarkan hasil pengujian dan analisis didapat kesimpulan bahwa secara rata-rata, algoritma Lempel Ziv Welch menghasilkan rasio file yang lebih baik sekitar ± 45,72 % dibandingkan dengan Algoritma Shannon-Fano 2 Gram yang hanya menghasilkan ± 58,50 %.
K**ata Kunci:** Kompresi Data, Algoritma Shannon-Fano 2 Gram, Algoritma Lempel Ziv Welch, Hadits Shahih Muslim, *lossless*

Abstract :
The rapid development of communications technology in the era of globalization has encouraged the development of technology and the need for telecommunications in other fields. One of those needs is in data storage. Therefore, it has developed algorithms to compress the data (data compression). Compression is the process of converting an input data stream (stream source, or the original raw data) into other data stream (bit stream results, or stream that has been compressed) smaller. There are two types of compression, which is lossless or lossy compression. In lossless compression, the original text can be reconstituted from the compressed data. Shannon-Fano algorithm 2 Gram and Lempel Ziv Welch algorithm can be used in lossless compression. Hadith is a word (the word), acts, statutes and approval of the Prophet Muhammad were used as the basis of Islamic law. There are six collections of hadith, one of which is the hadith of Sahih Muslim, consisting of 56 books. This paper implements compression of text data from Sahih Muslim hadith translation using Shannon-Fano algorithm 2 Gram and Lempel Ziv Welch algorithm. Several books of varying sizes have been as test data for compression. Based on test results and analysis concluded that on average , Lempel Ziv Welch algorithm produces better file ratio of about ± 45.72 % compared with the Shannon – Fano 2 Gram algorithm which produces only ± 58.50 %
.

Keywords*: Data compression, Shannon-Fano 2 Gram algorithm, Lempel Ziv Welch Algorithm, Shahih Muslim, Hadith, lossless*

1. Pendahuluan

Meningkatnya penggunaan komputer dan piranti *mobile* dalam kegiatan sehari-hari, secara tidak langsung juga membuat kebutuhan akan penyimpanan data semakin meningkat. Data tersebut, dapat berupa *file text*, gambar, suara, dan video. Semakin besar ukuran *file*, semakin besar pula tempat penyimpanan yang dibutuhkan. Untuk keperluan pengiriman data melalui media transmisi, akan semakin lama juga waktu yang dibutuhkan untuk mengirimkan data tersebut. Oleh karena itu, mulailah dikembangkan algoritma-algoritma kompresi yang bertujuan untuk memampatkan data.

Kompresi data dikenal sebagai ilmu atau seni merepresentasikan informasi dalam bentuk yang lebih *compact*.[1] Berbagai algoritma telah dikembangkan untuk keperluan kompresi data. Namun, algoritma tersebut sebagian besar lebih efisien digunakan untuk tipe data tertentu saja. Misalnya untuk kompresi *text*, terdapat algoritma *Huffman, Ziv and Lempel* 77 (LZ77), *Ziv and Lempel* 78(LZ78), Lempel *Ziv Welch* (LZW), *Dinamic Markov Compression* (DMC), *Run Length Encoding* (RLE), *Shannon Fano* dan lain-lain.

Dalam tugas akhir ini, akan dibandingkan dua algoritma kompresi, yaitu Shannon Fano 2 Gram dan Lempel Ziv Welch (LZW), yang dimana algoritma Shannon Fano2 Gram dan LZW menggunakan metode kompresi yang berbeda. Shannon Fano2 Gram menggunakan metode kompresi entropikompresi sedangkan LZW memakai metode kompresi dictionary. Untuk mengetahui algoritma mana yang lebih baik, proses kompresi dan dekompresi akan diterapkan pada *file text* (ASCII).

File yang akan dikompresikan adalah *file text* (ASCII). Dimana Teks adalah kumpulan dari karakter – karakter atau *string* yang menjadi satu kesatuan. Teks yang memuat banyak karakter didalamnya selalu menimbulkan masalah pada media penyimpanan dan kecepatan waktu pada saat transmisi data. Media penyimpanan yang terbatas, membuat semua orang mencoba berpikir untuk menemukan sebuah cara yang dapat digunakan untuk mengompresi teks.

Hadits adalah perkataan (sabda), perbuatan, ketetapan dan persetujuan dari Nabi Muhammad SAW yang dijadikan landasan syariat Islam.[2] Hadist Shahih Muslim adalah hadits yang disusun oleh Imam Muslim.Hadits Shahih Muslim sendiri memiliki 56 kitab.[3] Sebagai salah satu dari enam Hadits yang paling direferensikan, hal ini sangat berguna untuk pengguna ponsel muslim jika ada aplikasi Hadits Shahih Muslim.

Dalam menentukan algoritma yang tepat, ada 3 faktor yang akan dipertimbangkan, yaitu: rasio perbandingan file hasil kompresi, waktu yang dibutuhkan untuk melakukan kompresi, dan waktu yang dibutuhkan untuk dekompresi.

2. Kompresi Data

Kompresi data adalah ilmu atau seni merepresentasikan informasi dalam bentuk yang lebih *compact*.[4] David Salomon mengatakan bahwa data kompresi adalah proses mengkonversikan sebuah *input data stream* (*stream* sumber, atau data mentah asli) menjadi data *stream* lainnya (*bit stream* hasil, atau *stream* yang telah terkompresi) yang berukuran lebih kecil.[5] Pemampatan merupakan salah satu dari bidang teori informasi yang bertujuan untuk menghilangkan redundansi dari sumber. Pemampatan bermanfaat dalam membantu mengurangi konsumsi sumber daya yang mahal, seperti ruang *hard disk* atau perpindahan data melalui internet.[6]

Tujuan dari kompresi data adalah untuk merepresentasikan suatu data *digital* dengan sesedikit mungkin *bit*, tetapi tetap mempertahankan kebutuhan minimum untuk membentuk kembali data aslinya. Data *digital* ini dapat berupa *text*, gambar, suara, dan kombinasi dari ketiganya, seperti video.

Untuk membuat suatu data menjadi lebih kecil ukurannya daripada data asli, diperlukan tahapan-tahapan (algoritma) untuk mengolah data tersebut. Menurut Thomas H. Cormen algoritma adalah suatu prosedur komputasi yang didefinisikan secara baik, membutuhkan sebuah atau sekumpulan nilai sebagai *input*, dan menghasilkan sebuah atau sekumpulan nilai sebagai *output*.[1] Dalam algoritma kompresi data, tidak ada algoritma yang cocok untuk semua jenis data. Hal ini disebabkan karena data yang akan dimampatkan harus dianalisis terlebih dahulu, dan berharap menemukan pola tertentu yang dapat digunakan untuk memperoleh data dalam ukuran yang lebih kecil. Karena itu, muncul banyak algoritma-algoritma kompresi data.

a. Kompresi *Lossy*

Kompresi data yang menghasilkan file data hasil kompresi yang tidak dapat dikembalikan menjadi file data sebelum dikompresi secara utuh. Ketika data hasil kompresi di-decode kembali, data hasil decoding tersebut tidak dapat dikembalikan menjadi sama dengan data asli tetapi ada bagian data yang hilang.[4]

b. Kompresi *Lossless*

Kompresi data yang menghasilkan file data hasil kompresi yang dapat dikembalikan menjadi file data asli sebelum dikompresi secara utuh tanpa perubahan apapun. Kompresi jenis ini ideal untuk kompresi teks. Algoritma yang termasuk dalam metode kompresi lossless diantaranya adalah dictionary coding dan huffman coding.[3]

2.1 Algoritma Kompresi

Pada penelitian kali ini algoritma yang digunakan adalah :

a. Algoritma Shannon-Fano 2 Gram

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 *Huffman*. Sejatinya, algoritma Shannon-Fano 2 Gram mirip algoritma Shannon-Fano, namun dieksekusi per 2 simbol. Pada dasarnya metode ini menggantikan setiap 2 simbol dengan sebuah alternatif kode biner yang panjangnya ditentukan berdasarkan probabilitas dari simbol tersebut. Pada dasarnya proses *coding* dengan algoritma ini membutuhkan data akan frekuensi jumlah kemunculan suatu karakter pada sebuah pesan. Tiga prinsip utama yang mendasari algoritma ini adalah:

1. Simbol yang berbeda memiliki kode yang berbeda
2. Simbol dengan probabilitas kemunculan yang lebih kecil memiliki kode panjang bit yang lebih panjang dan simbol dengan probabilitas yang lebih besar memiliki panjang bit yang lebih pendek.
3. Meskipun kode yang dihasilkan memiliki panjang bit yang berbeda dengan kode pada karakter asli, tetapi dapat didekodekan secara unik (Nelson, 1996).

Sebagai contoh *string* "AAAAAAAAAaBaAaBBBBCcCcCc" akan di kompresi menggunakan algoritma *Shannon-Fano 2 Gram*, table di bawah ini merupakan simbol AA,Aa,Ba,BB, dan Cc dan kode yang diperoleh untuk setiap simbol.

Tabel 2 . 1 Shannon - Fano 2 Gram

Simbol	Kode
AA	00
Cc	01
Aa	10
BB	110
Ba	111

1. Langkah pertama yang harus dilakukan untuk memperoleh kode adalah dengan membuat daftar probabilitas atau frekuensi dari setiap simbol.

Tabel 2 . 2 Frekuensi Simbol

Simbol	Frekuensi
AA	4
Cc	3
Aa	2
BB	2
Ba	1

2. Urutkanlah daftar tersebut menurut frekuensi kehadiran simbol secara menurun (dari simbol yang frekuensi kemunculannya paling banyak sampai simbol dengan frekuensi kemunculan paling sedikit).

Tabel 2 . 3 Tabel Frekuensi

Simbol	Frekuensi
AA	4
Cc	3
Aa	2
BB	2
Ba	1
Total	12

3. Bagilah daftar tersebut menjadi dua bagian dengan pembagian didasari pada jumlah total frekuensi suatu bagian (disebut bagian atas) sedekat mungkin dengan jumlah frekuensi dengan bagian yang lain (disebut bagian bawah).

Tabel 2 . 4 Tabel Pembagian Frekuensi

Simbol	Frekuensi
A	4
Cc	3
Aa	2
B	2
Ba	1

4. Daftar bagian atas dengan digit 0 dan bagian bawah dinyatakan dengan digit 1. Hal tersebut berarti kode untuk simbol-simbol pada bagian atas akan dimulai dengan 0 dan kode untuk simbol-simbol pada bagian bawah akan dimulai dengan 1.

Tabel 2 . 5 Tabel Pembentukan Shannon - Fano 2 Gram

Simbol	Frekuensi	Kode
A	4	00
Cc	3	01
Aa	2	10
B	2	11)
Ba	1	11 l

5. Lakukanlah proses secara rekursif langkah 3 dan 4 pada bagian atas dan bawah. Bagilah menjadi kelompok-kelompok dan tambahkanlah bit-bit pada kode sampai setiap simbol memperoleh kode.

Hasil perhitungan ukuran file setelah dikompresi dapat dilihat pada Tabel 2.6

Tabel 2 . 6 Tabel Perhitungan

Simbol	Frekuensi	Kode	Panjang Kode	Total Bit (Panjang Kode x Frekuensi)
AA	4	00	2	8
Cc	3	01	2	6
Aa	2	10	2	4
BB	2	110	3	6
Ba	1	111	3	3
TOTAL	**12**			**27**

Algoritma Shannon-Fano 2 Gram menggunakan metode statik, sehingga memiliki 2 fase (*two pass*), setelah menghitung probabilitas kemunculan tiap simbol/karakter dan menentukan peta kodenya, kemudian fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan ditransmisikan. Sehingga dari *string* "AAAAAAAAAaBaAaBBBBCcCcCc" didapat runtun bit 000000001011110110110010101. Total bit yang dibutuhkan untuk mewakili pesan diatas adalah 27 bit. Sedangkan bila menggunakan ASCII 8 bit, dibutuhkan 12 x 8 bit = 96 bit, sehingga kompresi rasionya adalah:

$$\text{Rasio Kompresi} = \frac{27}{96} x\ 100\% = 28{,}125\%$$

Gambar 2 . 1 Rasio Kompresi Algoritma Shannon - Fano 2 Gram

Sedangkan decoding Algoritma Shannon-Fano 2 Gram sebagai berikut:

1. Baca bit pertama dari serangkaian kode yang dihasilkan.
2. Cocokkan bit kode secara berulang sampai ditemukan kode simbol pada tabel Shannon.
3. Mengulang pencocokkan runtun bit sampai semua kode berhasil diterjemahkan ke simbol file asli. Sehingga akan menghasilkan file *output* berupa file asli yang berisi rangkaian *string* yang sama dengan rangkaian *string* pada file asli.

b. Algoritma Lempel Ziv Welch

Algoritma Lempel-Ziv-Welch (LZW), dikembangkan oleh Abraham Lempel, Jacob Ziv, dan Terry Welch. Dan dipublikasikan pada tahun 1984 oleh Terry Welch. Algoritma ini mereduksi jumlah token yang dibutuhkan menjadi 1 simbol saja. Simbol ini merujuk kepada index dalam dictionary. LZW mengisi dictionary ini dengan seluruh simbol alphabet yang dibutuhkan. Pada kasus yang umum, 256 index pertama dari dictionary akan diisi dengan karakter ASCII dari 0-255. Karena dictionary telah diisi dengan semua kemungkinan karakter terlebih dahulu, maka karakter inputan pertama akan selalu dapat ditemukan dalam dictionary. Inilah yang menyebabkan token pada LZW hanya memerlukan 1 simbol saja yang merupakan pointer pada dictionary. Algoritma ini mengeluarkan output berupa indeks untuk string yang ada dalam dictionary. Jika kombinasi string tersebut tidak ada, maka kombinasi tersebut akan ditambahkan dalam dictionary dan algoritma akan mengeluarkan output dari kombinasi yang ada.[7]. Prinsip kerja LZW, dimulai dengan membaca karakter input satu persatu dan diakumulasi pada sebuah string A. Lalu dilakukan pencarian dalam dictionary, apakah terdapat string A. Selama string A ditemukan didalam dictionary, string ini ditambahkan dengan satu karakter berikutnya, lalu dicari lagi dalam dictionary.

Pada saat tertentu, menambahkan satu karakter x pada string A akan menyebabkan tidak ditemukan dalam dictionary. String A ditemukan, tetapi string Ax tidak. Dalam tahap ini, algoritma akan menulis index dari string A sebagai output, menambahkan string Ax kedalam dictionary, dan menginisialisasikan string A dengan x, lalu proses dimulai lagi dari awal. Kumpulan output yang berupa angka-angka inilah sebagai hasil kompresi..Algoritma ini melakukan kompresi dengan menggunakan kamus, di mana fragmen-fragmen teks digantikan dengan indeks yang diperoleh dari sebuah "kamus". Pendekatan ini bersifat adaptif dan efektif karena banyak karakter dapat dikodekan dengan mengacu pada string yang telah muncul sebelumnya dalam teks. Prinsip kompresi tercapai jika referensi dalam bentuk pointer dapat disimpan dalam jumlah bit yang lebih sedikit dibandingkan string aslinya.

```
Step 1: Dictionary terlebih dahulu diinisialisasi dengan karakter ASCII
Step 2: CurrentChars ← Karakter pertama dari input stream
Step 3: DictIndex ← 255
Step 4:        while not EOF character do begin
Step 4a:   NextChar ← Karakter selanjutnya dari CurrentChars
Step 4b:   ConcatStr ← CurrentChars + NextChar
Step 4c:   if ConcatStr ada pada Dictionary then begin
Step 4d:        CurrentChars ← ConcatStr
Step 4e:   end else begin
Step 4f:       Output ← Index dari CurrentChars pada
Step 4g:       dictionary
Step 4h:       Isi ke dalam Dictionary (DictIndex, ConcatStr)
Step 4i:       DictIndex ← DictIndex + 1
Step 4j:       CurrentChars ← NextChar
Step 4k:   end
Step 5: end
Step 6: EncodedStream ← Output
```

Gambar 2 . 2 Algoritma Dasar LZW Encoding

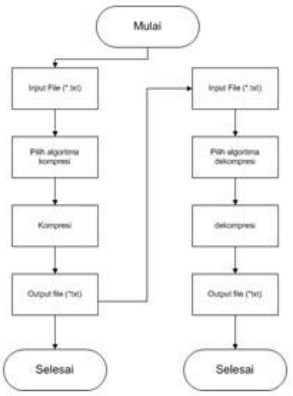
Tidak seperti encoding, pada decoding isi dictionary selalu ditambahkan setiap pembacaan codeword, hal inilah yang memungkinkan string asli dapat dikembalikan.

```
Step 1: Dictionary terlebih dahulu diinisialisasi dengan karakter ASCII
Step 2: PreviousCodeWord ← Codeword pertama dari encoded stream
Step 3: String ← toString(PreviousCodeWord);
Step 4: Char ← toChar(first input codeword);
Step 5: DictIndex ← 256;
Step 6: while NOT EOF encoded stream
Step 6a: CurrentCodeWord ← codeword selanjutnya pada encoded
        stream
Step 6b: if CurrentCodeWord ada pada Dictionary then begin
Step 6c:   String ← toString(CurrentCodeWord)
Step 6d  end else begin
Step 6e:   String ← toString(PreviousCodeWord) + Char
Step 6f:   end;
Step 6g: Output ← Output + String;
Step 6h: Char ← Karakter pertama dari string sekarang
Step 6i: insertToDictionary(DictIndex,
            toString(PreviousCodeWord) + Char;
Step 6j:   PreviousCodeWord ← CurrentCodeWord;
Step 6k:   DictIndex++;
Step 7:    end
Step 8:    DecodedStream ← Output
```

Gambar 2 . 3 Algoritma Dasar LZW Decoding

3. Pengujian

Skenario pengujian dalam tugas akhir kali ini adalah dengan menganalisis pengaruh dari algoritma Shannon-Fano 2 Gram dan Lempel Ziv Welch yang digunakan dengan melihat dari sisi pengurangan ukuran file, rasio yang dihasilkan dan kecepatan dalam proses kompresi maupun kecepatan dalam waktu dekompresi file.



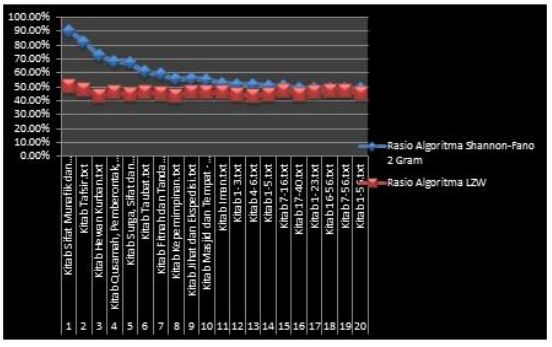
Gambar 3.1 Skenario Pengujian.

Data uji yang digunakan dalam tugas akhir ini adalah 20 *file* yang berisi text dari terjemahan "Hadits Sunan An-nasa'i" dengan format (*.*txt*) yang diatur dari ukuran terkecil sampai terbesar. Dibawah ini terdapat tabel yang berisi nama *file* yang akan di *input* ke dalam aplikasi beserta ukuran file tersebut.

Tabel 3.1 Data Uji

No.	Nama File	Ukuran File Asli (byte)
1	Kitab Sifat Munafik dan Hukumnya.txt	19.210
2	Kitab Tafsir.txt	26.124
3	Kitab Hewan Kurban.txt	37.723
4	Kitab Qusamah, Pemberontak, Qishah dan Diyat.txt	47.470
5	Kitab Surga, Sifat dan Penghuninya.txt	64.315
6	Kitab Taubat.txt	92.913
7	Kitab Fitnah dan Tanda Kiamat.txt	125.871
8	Kitab Kepemimpinan.txt	173.908
9	Kitab Jihat dan Ekspedisi.txt	203.357
10	Kitab Masjid dan Tempat - Tempat Shalat.txt	274.656
11	Kitab Iman.txt	385.436
12	Kitab 1-3.txt	477.746
13	Kitab 4-6.txt	560.995
14	Kitab 1-5.txt	764.085
15	Kitab 7-16.txt	1.213.048
16	Kitab 17-40.txt	1.693.695
17	Kitab 1-23.txt	2.743.953
18	Kitab 16-56.txt	3.358.425
19	Kitab 7-56.txt	4.164.421
20	Kitab 1-56.txt	5.203.161

3.1 Analisis Hasil Rasio Kedua Algoritma

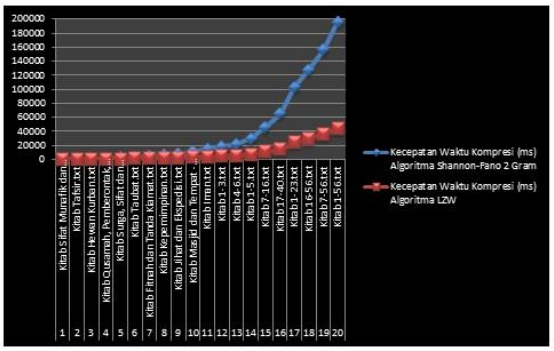


Gambar 3.2 Grafik rasio

Pada kasus ini algoritma Lempel Ziv Welch mempunyai persentasi rasio paling kecil dan dapat mengurangi ukuran file lebih baik dibanding algoritma Shannon-Fano 2 Gram. Dikarenakan banyaknya karakter yang terjadi pengulangan pada data file hadist seperti contoh nya adalah kata – kata "dan, bahwa, mengatakan" sehingga algoritma LZW tidak memerlukan dictionary baru sehingga bit yang dipakai sedikit. Sedangkan pada algoritma Shannon-Fano 2 Gram, terdapat beberapa karakter dan frekuensi nya yang sama membuat rangkaian bit tidak

begitu banyak terjadi pengurangan, yang membuat banyaknya kode Shannon yang terbentuk dan menjadikan algoritma Shannon-Fano 2 Gram menjadi kurang optimal.

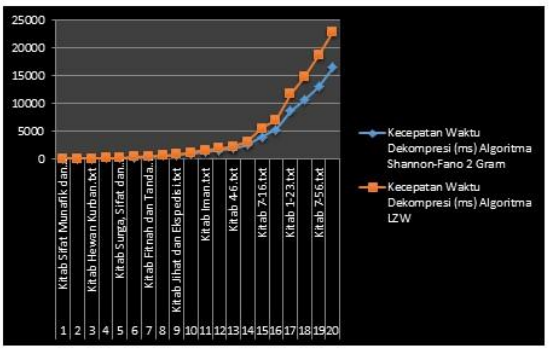
3.2 Analisis Hasil Kecepatan Kompresi Kedua Algoritma



Gambar 3.3 Hasil Kecepatan Kompresi

Pada hasil ini dapat dilihat bahwa waktu kompresi dari algoritma Lempel Ziv Welch jauh lebih baik dibanding dengan Algoritma Shannon-Fano 2 Gram. Algoritma LZW bersifat adaptif dan berbasis dictionary, dimana selama proses kompresi berlangsung algoritma ini menggunakan sebuah dictionary yang dibentuk selama pembacaan input stream. Dictionary dibentuk dengan tujuan untuk menyimpan karakter atau pola string tertentu yang berguna untuk mengkodekan simbol atau string pada input stream yang merujuk pada index dalam dictionary. Sedangkan algoritma Shannon-Fano 2 Gram menggunakan metoda statik. Metoda yang selalu menggunakan peta kode yang sama, metode ini membutuhkan dua fase (two-pass): fase pertama untuk menghitung probabilitas kemunculan tiap 2 karakter dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan di transmisikan. Di karenakan banyaknya kemungkinan 2 karakter yang muncul, sehingga waktu yang dibutuhkan dalam proses kompresi menjadi semakin banyak.

3.3 Analisis Hasil Kecepatan Dekompresi Kedua Algoritma.



Gambar 3.4 Hasil kecepatan Dekompresi

Pada hasil ini dapat dilihat bahwa dekompresi algoritma Shannon-fano sedikit lebih cepat jika dibandingkan dengan algoritma Lempel Ziv Welch. Pada algoritma Shannon-fano 2 Gram dalam proses dekompresi nya membaca simbol-simbol ASCII dari hasil kompresi, kemudian menerjemahkan kembali menjadi urutan runtun bit hasil kompresi yang telah di urutkan sebelumnya. Kemudian dari runtun bit tersebut diambil satu persatu bit dan mencocokkan dengan tabel Shannon, jika bit tidak cocok, ambil kembali bit selanjutnya dan jumlahkan bit tersebut sampai ditemukan bit yang cocok pada tabel Shannon. Sedangkan algoritma Lempel Ziv Welch pada saat dekompresi yang dibaca adalah kumpulan kode (angka) hasil kompresi. Selama proses pembacaan kode berlangsung, pada saat itu pula dilakukan pembentukan isi dictionary yang menjadi referensi untuk pembentukan string asli. Pada kasus ini, proses dekompresi algoritma Shannon-fano 2 Gram sedikit lebih cepat dikarenakan pada proses pengembalian ke file teks asli, algoritma Shannon-fano 2 Gram mengembalikan 2 karakter secara konstan, sedangkan algoritma Lempel Ziv Welch mengembalikan karakter secara acak tergantung dari *dictionary* kompresi nya.

4. Kesimpulan dan Saran

Berdasarkan pengujian yang telah dilakukan pada tugas akhir kali ini, terdapat beberapa kesimpulan dan saran yang dapat membantu untuk menjadi panutan untuk penelitian atau bahan-bahan tugas kuliah lainnya.

1. Pengimplementasian algoritma kompresi Shannon-Fano 2 Gram dan Lempel Ziv Welch dalam proses kompresi file text berhasil dilakukan.
2. Metode Shannon-Fano 2 Gram dan Lempel Ziv Welch dapat mengembalikan kembali teks yang sudah dikompresi (rekonstruksi teks). Sehingga metode Shannon-Fano 2 Gram dan Lempel Ziv Welch merupakan metode *lossless.*
3. Secara rata-rata algoritma Lempel Ziv Welch menghasilkan rasio file yang lebih baik sekitar ± 45,72 % dibandingkan dengan Algoritma Shannon-Fano 2 Gram yang hanya menghasilkan ± 58,50 %.
4. Algoritma Shannon-Fano 2 Gram memiliki waktu dekompresi yang lebih baik dibandingkan dengan algoritma Lempel Ziv Welch. Besarnya waktu dekompresi tergantung dari besarnya ukuran file yang akan didekompresi.
5. Dari hasil yang didapatkan untuk file teks (.txt) algoritma kompresi Lempel Ziv Welch lebih baik dari pada Algoritma Shannon-Fano 2 Gram dilihat dari segi pengurangan ukuran file, rasio, dan kecepatan kompresi.

Saran :

Berdasarkan pengujian yang telah dilakukan pada tugas akhir kali ini, terdapat beberapa saran yang dapat membantu untuk menjadi panutan untuk penelitian atau bahan-bahan tugas kuliah lainnya.

1. Membandingkan algoritma Shannon-Fano 2 Gram dan Lempel Ziv Welch dengan algoritma kompresi lainnya, seperti, DMC, Burrows-Wheeler Transform, Arithmetic Coding dan lain-lain.
2. Melakukan pengujian dengan beberapa sampel lainnya, seperti html, rtf, dan lain-lain.
3. Mengimplementasikan algoritma Shannon-Fano 2 Gram dan Lempel Ziv Welch dengan bahasa pemrograman lainnya.

Daftar Pustaka

[1] Cormen T. H., et al. 2001. *Introduction to Algorithms, Second Edition*. New York: The MIT Press.
[2] Wikipedia. Hadits. (https://id.wikipedia.org/wiki/Hadits, diakses pada tanggal 8 November 2013)
[3] Wikipedia. Shahih Muslim. (https://id.wikipedia.org/wiki/Shahih_Muslim , diakses pada tanggal 13 Juli 2015)
[4] Mengyi Pu, Ida. 2006. *Fundamental Data Compression*. Edisi Pertama. London : Elsevier.
[5] Salomon, D. 2007. *Data Compression The Complete References.* New York: Springer-Verlag.
[6] Nelson, Mark and Gailly, Jean-Loup. 1996. *The Data Compression Book (Second Edition),* M&T Books.
[7] Salomon, D. 2005. *Coding for Data and Computer Communication.* New York: Springer-Verlag
[8] Weiss, Mark Allen. 2003. *Data Structures And Problem Solving Using C++*. Cetakan Pertama. Addison Wesley.
[9] Razi, Fahrur. 2009. Analisis Pengaruh Panjang Bit Kode Pada Kinerja Program Kompresi Algoritma Lempel Ziv Welch (LZW). Student Papers. Universitas Sumatera Utara
[10] Wiryadinata, R. 2007. *Data Compression Coding Using Statistic and Dynamic Method of Shannon Fano Algorithm*. Media Informatika.
[11] Saputro, N.2009. Implementasi Analisis Perbandingan Antara Pengkodean LZ78 dan Shannon Fano Pada Kompresi Data Teks. Jurnal Informatika.
[12] Henkerson, Darrel dan Harris, Greg A. 2003. *Introduction to Information Theory and Data Compression*. Cetakan Kedua. New York: CRC Press LLC
[13] Sitorus, Eunike Johana. 2013. Studi Perbandingan Kompresi Menggunakan Metode Shannon Fano Dan Unary Coding Pada File Teks. Laporan Tugas Akhir. Universitas Sumatera Utara