DINAMIKA Jurnal Ilmiah Teknik Mesin

IMPLEMENTASI DAN ANALISIS KINERJA ALGORITMA *SHANNON-FANO* UNTUK KOMPRESI FILE TEXT

Sutardi

Staf Pengajar Jurusan Pendidikan Teknik Informatika Fakultas Teknik Universitas Halu Oleo
Kampus Hijau Bumi Tridarma Andounohu Kendari 93232

Email:sutardi@gmail.com

Abstrak

Algoritma *Shannon-Fano* merupakan algoritma kompresi data yang meng-kodekan setiap karakter dengan menggunakan beberapa rangkaian *bit*. Pembentukan *bit* yang mewakili masing-masing karakter dibuat berdasarkan frekuensi kemunculan tiap karakter. Tujuan dari penelitian ini adalah untuk menguji kompresi algoritma *Shannon-Fano*. Penelitian ini dilakukan terhadap dua kategori yaitu file dengan ukuran yang sama dan memiliki karakter yang bervariasi dan file dengan ukuran yang berbeda dan karakter yang bervariasi. Hasil pengujian menunjukan bahwa file *text* dengan karakter yang bervariasi akan menghasilkan pemampatan file yang rendah sedangkan file*text* dengan karakter yang tidak bervariasi memiliki tingkat pemampatan file yang tinggi dengan ukuran file yang sama.

Kata Kunci : *kompresi, implementasi, file, Shannon-Fano, algoritma*

Abstract

The implementation and analysis on the performance of the *Shannon-Fano* Algorithm for the compression of text file**.** *Shannon - Fano* algorithm is a data compressed algorithm encoding each character by using a series of *bit*s. The formation of the *bit*s that represent the characters is created by its frequency of occurrence. The purpose of this study is to conduct a test on the *Shannon - Fano* algorithm compression. It is conducted into two categories: the files with the same sizes and the varying characters and the files with varying sizes and the varying characters. The results show that the text file with the different characters may generate a low file compression, while that with less different characters may have a high level of compression file with the same file size.

Keywords : *compression, implementation, files, Shannon-Fano, algorithm*

1. Pendahuluan

Kompresi data dalam konteks ilmu komputer merupakan ilmu dan seni yang menampilkan informasi dalam bentuk yang pendek. Kompresi data bertujuan untuk mengurangi jumlah *bit* yang digunakan untuk menyimpan atau mengirim informasi (Pu, 2006).

Metode kompresi dapat diklasifikasikan ke dalam dua kelompok besar yaitu metode *lossless dan lossy*. Metode *lossless* menghasilkan file kompresi dengan ukuran yang lebih kecil dari file aslinya sedangkan metode *lossy* menghasilkan file hasil kompresi lebih kecil dari file semula dan rasio kompresinya lebih tinggi daripada metode sebelumya. Prinsip umum yang digunakan pada proses kompresi *lossless* adalah mengurangi duplikasi data di dalamnya sehingga memori yang dibutuhkan untuk merepresentasikannya menjadi lebih sedikit daripada representasi semula, sedangkan metode *lossy* dilakukan dengan menghilangkan atau memodifikasi data sehingga ukuran file menjadi lebih kecil tanpa merubah representasi file secara visual (Anton, 2009).

Algoritma *Shannon-Fano* ini membentuk sebuah pohon dari kumpulan data, kemudian meng-*encoding* dan mengembalikannya dalam bentuk karakter teks atau *decoding*. Pembuatan pohon pada *Shannon-Fano* dibuat berdasarkan proses dari atas ke bawah, yang dibangun sesuai dengan spesifikasi yang dirancang untuk mendefinisikan

tabel kode. Pada pohon *Shannon-Fano*, semua karakter dikelompokkan berurutan dari kiri ke kanan dari frekuensi yang sering muncul ke frekuensi yang umum (Santi, 2010).

File-*text* membutuhkan kapasitas ruang penyimpanan yang besar. Satu karakter memiliki 1 byte (8 *bit*). Semakin banyak karakter dalam sebuah *text* maka akan semakin besar kapasitas yang dibutuhkan untuk menyimpan file tersebut, dan semakin meningkat waktu yang dibutuhkan untuk proses pengiriman file pada saluran komunikasi (Suhendra, 2004).

Salah satu teknik pemampatan file-*text* yang dapat digunakan adalah algoritma *Shannon-Fano.* Algoritma *Shannon-Fano* didasarkan pada *variable-length code* yang berarti beberapa karakter pada data yang akan dikodekan direpresentasikan dengan kode (*codeword*) yang lebih pendek dari karakter yang ada pada data. Jika frekuensi kemunculan karakter semakin tinggi, maka kode semakin pendek, sehingga kode yang dihasilkan tidak sama panjang. Hal ini menyebabkan kode tersebut bersifat unik (Wijaya dkk, 2007).

Tujuan dari penelitian ini adalah untuk menguji kompresi algoritma *Shannon-Fano*, pada berbagai kategori, seperti ukuran dan karakter.

2. Tinjauan Pustaka

File-text

File-*text* merupakan file yang berisi informasi dalam bentuk *text*. Data yang berasal dari dokumen pengolah kata, angka yang digunakan dalam perhitungan, nama dan alamat dalam basis data merupakan contoh masukan data *text* yang terdiri dari karakter, angka dan tanda baca (Santi, 2010).

Masukan dan keluaran data *text* direpresentasikan sebagai set karakter atau sistem kode yang dikenal oleh sistem komputer. Ada tiga macam set karakter yang umum digunakan untuk masukan dan keluaran pada komputer, yaitu ASCII, EBCDIC, dan Unicode. ASCII (*American Code for Information Interchange*) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hexdan Unicode, tetapi ASCII lebih bersifat universal. ASCII digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan *text*. Kode ASCII memiliki komposisi bilangan biner sebanyak 8 *bit*, dimulai dari 00000000 hingga 11111111. Total kombinasi yang dihasilkan sebanyak 256, dimulai dari kode 0 hingga 255 dalam sistem bilangan desimal (Wijaya dkk, 2010).

EBCDIC (*Extended Binary Codec Decimal Interchange Code*) merupakan set karakter yang diciptakan oleh komputer merk IBM. EBCDIC terdiri dari 256 karakter yang masing-masing berukuran 8 *bit*. Adanya keterbatasan pada kode ASCII dan EBCDIC, dibuat standar kode internasional baru yang merupakan kode 16 *bit* yang disebut Unicode (Gozali, 2004). Unicode adalah suatu standar industri yang dirancang untuk mengizinkan *text* dan simbol dari semua sistem tulisan di dunia untuk ditampilkan dan dimanipulasi secara konsisten oleh komputer (Sudewa, 2003).

Kompresi data

Kompresi data adalah aplikasi kompresi data yang dilakukan terhadap data digital dengan tujuan untuk mengurangi redundansi yang terdapat dalam data sehingga dapat disimpan atau ditransmisikan secara efisien (Sutoyo, 2009).

Metode kompresi data dapat dikelompokkan dalam dua kelompok besar yaitu metode *lossless* dan metode *lossy* yaitu (Munir, 2004 & Anton, 2009). Pada metode *lossless* tidak ada kehilangan data atau informasi. Jika data dikompres secara *lossless*, data asli dapat direkonstruksi kembali sama persis dari data yang telah dikompresi. Secara umum teknik *lossless* digunakan untuk penerapan yang tidak bisa mentoleransi setiap perbedaan antara data asli dan data yang telah direkonstruksi. Data berbentuk tulisan misalnya file*text*, harus dikompresi menggunakan teknik *lossless*, karena kehilangan sebuah karakter saja dapat mengakibatkan kesalapahaman. *Lossless compression* disebut juga dengan *reversible compression* karena data asli bisa dikembalikan dengan sempurna, namun rasio kompresinya sangat rendah, misalnya pada data *Text* dan gambar seperti GIF dan PNG. Contoh metode ini adalah *Shannon-Fano Coding*, *Run Length Encoding* dan *Arithmetic Coding* (Anton, 2009).

Pada metode *lossy*, pada umumnya teknik kompresi yang dilakukan adalah membuang bagian data yang sebenarnya tidak berguna seperti data yang tidak dapat dilihat maupun didengar oleh manusia. Contoh metode ini adalah *transform Coding* dan *Wavelet*. *Lossy compression* disebut juga *irreversible compression* karena data asli mustahil untuk dikembalikan seperti semula.

Kelebihan teknik ini adalah rasio kompresi yang tinggi dibanding metode *lossless* (Anton, 2009).

Efek pemampatan pada metode pemampatan *lossless* dapat diukur melalui sejumlah penyusutan suatu file asal dalam membandingkan ukuran dari jenis-jenis pemampatan. Jika dimisalkan rasio pemempatan adalah (r), ukuran file setelah pemampatan adalah (p), ukuran file semula adalah (q), redundansi file adalah (h), frekuensi relatif adalah (f), frekuensi karakter adalah (k) dan jumlah frekuensi karakter adalah (j), maka rasio pemampatan, dapat dirumuskan sebagai berikut (Santi, 2010):

$$r = \left(\frac{p}{q}\right) x\ 100\% \quad (1)$$

Semakin kecil nilai rasio kompresi menunjukkan bahwa semakin tinggi pemampatan file *text* tersebut. Untuk mendapatkan redudansi file dapat dirumuskan sebagai berikut (Santi, 2010)

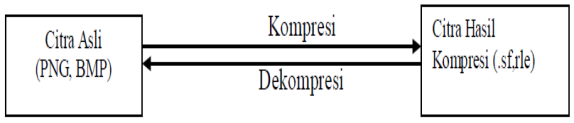
$$h = \left(\frac{q-p}{q}\right) x\ 100\% \quad (2)$$

Semakin tinggi nilai redudansi menunjukkan bahwa semakin baik pemampatan file *text* tersebut. Untuk menentukan frekuensi relatif dapat dirumuskan sebagai berikut (Santi, 2010)

$$f = \frac{k}{j} \quad (3)$$

Dekompresi

Sebuah data yang sudah dikompresi tentunya harus dapat dikembalikan lagi kebentuk aslinya, prinsip ini dinamakan dekompresi. Untuk dapat merubah data yang terkompresi, diperlukan cara yang berbeda seperti pada waktu proses kompresi dilaksanakan.



Gambar 1. Alur kompresi lossless

Pada saat dekompresi catatan *header* yang berupa *byte-byte* tersebut masih terdapat catatan isi mengenai isi dari file tersebut (Gozali,2004). Catatan *header* akan menuliskan kembali mengenai isi dari file tersebut. Hal ini menyebabkan isi dari file sudah tertulis oleh catatan *header*, sehingga hanya tinggal menuliskan kembali pada saat proses dekompresi. Secara umum proses kompresi dan dekompresi dapat dilihat pada Gambar 1 (Gozali, 2004).

Algoritma *Shannon-Fano*

Kompresi data adalah proses pengubahan serangkaian data input menjadi data output yang mempunyai ukuran lebih kecil (Lidya, 2012). Algoritma *Shannon-Fano* merupakan salah satu algoritma kompresi yang sangat baik dalam pengkompresian *text*. Pada prinsipnya algoritma ini menggunakan pendekatan *top-down* dalam penyusunan *binary-tree*. Metode ini sangat efisien untuk mengkompresi file-*text* yang berukuran besar (Martin, 2007)

Tahapan algoritma kompresi *Shannon-Fano* pada data, diawali dengan menghitung frekuensi kemunculan masing-masing huruf pada *text* dan mengurutkan frekuensi kemunculan huruf dari huruf yang terbesar ke yang terkecil, di mana masing-masing huruf dapat direpresentasikan sebagai sebuah node. Tahap selanjutnya adalah menjumlahkan seluruh frekuensi kemunculan huruf dan masukkan dalam sebuah node dan membagi menjadi dua buah node dengan jumlah frekuensi kemunculan huruf yang sama atau hampir sama. Selanjutnya adalah pemberian label pada setiap sisi pohon biner, sisi kiri dilabeli dengan 0 dan sisi kanan dilabeli dengan 1. Selanjutnya melakukan langkah sebelumnya (iterasi), sampai node tidak dapat dibagi lagi dan menelusuri pohon biner dari akar ke daun (Munir, 2004).

Tahapan awal kompresi algoritma *Shannon-Fano* adalah pembuatan pohon *shannon* dengan cara membaca semua karakter di dalam teks untuk menghitung probabilitas kemunculan tiap karakter. Karakter-karakter ini kemudian disusun dari yang mempunyai probabilitas kemunculan paling besar ke karakter yang mempunyai probabilitas kemunculan paling kecil. Tahap selanjutnya adalah membagi dua serangkaian karakter ini menjadi 2 *subset* yang mempunyai jumlah probabilitas yang sama atau hampir sama. Semua simbol dalam *subset* pertama diberi kode 0 sedangkan simbol pada *subset* lainnya diberi kode 1. Ketika pada sebuah *subset* hanya terdapat 2 macam simbol, kode mereka dibedakan dengan

cara memberikan 1 *bit* lagi ke masing-masing simbol. Tahap selanjutnya adalah melakukan langkah sebelumnya pada tiap-tiap *subset* secara rekursif sampai tidak ada lagi *subset* yang tersisa (Purnomo dkk, 2005).

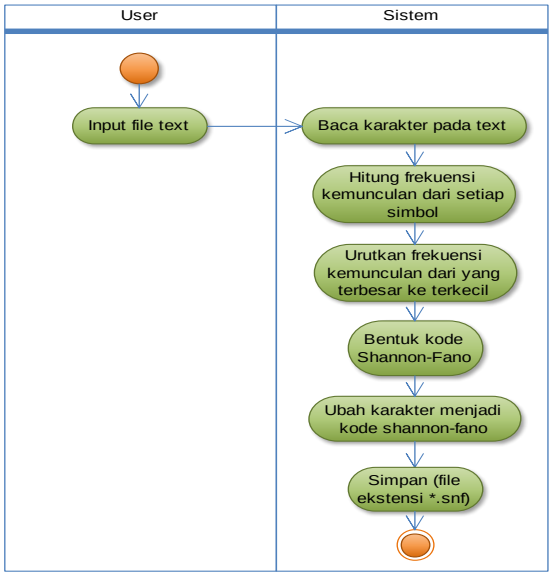
Tahapan dekompresi pada serangkaian kode-kode (Purnama, 2003) diawali dengan membaca *bit* pertama dari serangkaian kode yang dihasilkan. Jika *bit* tersebut ada dalam pohon *Shannon*, maka *bit* tersebut diterjemahkan menjadi karakter yang sesuai dengan *bit* tersebut, dan jika *bit* tersebut tidak ada dalam pohon *Shannon*, *bit* tersebut digabungkan dengan *bit* selanjutnya dalam rangkaian kode dan dicocokkan dengan tabel hasil pengkodean. Tahap selanjutnya adalah melakukan mengulangi langkah sebelumnya sampai ada rangkaian *bit* yang cocok dengan pohon *Shannon*, menerjemahkan rangkaian *bit* tersebut menjadi karakter yang sesuai, membaca *bit* selanjutnya dan mengulangi langkah sebelumnya sampai rangkaian kode habis (Purnomo dkk, 2005, Purnama, 2003).

3. Hasil dan Pembahasan

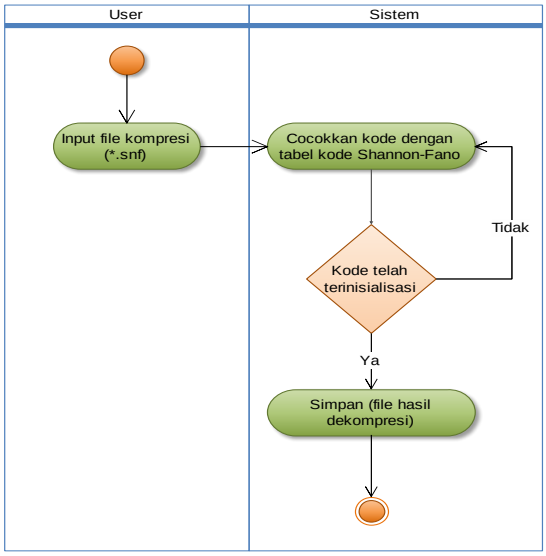
Proses kompresi file *text*

Algoritma *Shannon-Fano* yang digunakan pada aplikasi ini merupakan algoritma yang melakukan dua kali pembacaan (*two-pass*) terhadap *text* yang akan dikompresi. File*text* yang di *input* merupakan file*text* *.txt.

Proses kompresi diawali denagn membuat daftar frekuensi kemunculan setiap simbol dari data (pesan yang akan dikodekan) dan mengurutkan daftar tersebut menurut frekuensi kehadiran simbol secara menurun. Tahap selanjutnya adalah membagi daftar tersebut menjadi dua bagian. Pembagian didasari pada jumlah total frekuensi suatu bagian (disebut bagian atas) sedekat mungkin dengan jumlah frekuensi dengan bagian yang lain (disebut bagian bawah). 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. Tahap selanjutnya adalah melakukan proses secara rekursif langkah 3 dan 4 pada bagian atas dan bawah dan membagi menjadi kelompok-kelompok. Tahap selanjutnya adalah menambahkan *bit-bit* pada kode sampai setiap simbol memperoleh kode dan dilanjutkan dengan mengganti karakter dengan kode yang telah terbentuk. Buku kode berisi informasi antara lain, frekuensi kemunculan *simbol* dari data, kode *Shannon-Fano*, panjang kode *Shannon-Fano* dan total panjang kode untuk masing-masing simbol (huruf).



Gambar 2. *Activity diagram* kompresi



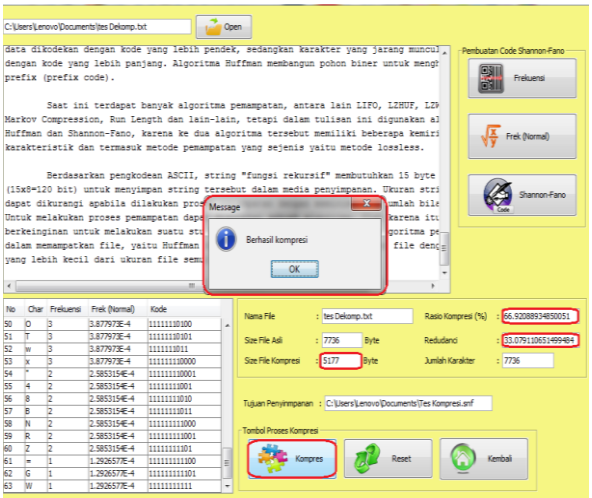
Gambar 3. .*Activity diagram* dekompresi

Proses dekompresi file-***text***

Proses dekompresi file-*text* pada metode *Shannon-Fano* membutuhkan pohon *Shannon-Fano* atau tabel *Shannon-Fano* yang terbentuk saat proses kompresi. Proses dekompresi *Shannon-Fano* diawali dengan pembacaan kode *Shannon-Fano* sebagai *input*. Kemudian kode pertama yang muncul dicocokkan dengan kode-kode *Shannon-Fano* yang telah ada pada pohon atau tabel *Shannon-Fano* sampai ditemukan kode yang sesuai dengan huruf-huruf yang telah terinisialisasi pada file*text*. Inisialisasi ini terus berlanjut sampai semua kode *Shannon-Fano* yang ada mewakili huruf yang terdapat pada tiap-tiap kata pada file*text*.

Kompresi & dekompresi

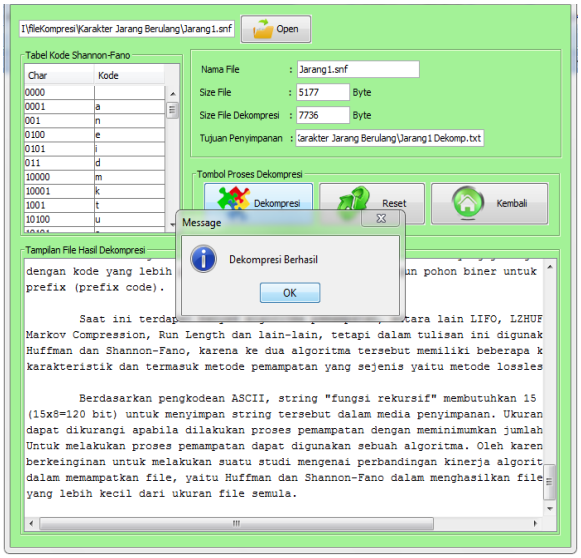
Tahap kompresi dimulai dengan menginput file-*text* ke dalam *text* area informasi tentang nama file kemudian ukuran file akan menampilkan frekuensi kemunculan tiap karakter pada tabel frekuensi. Frekuensi normal kemunculan tiap karakter pada tabel *frek* (normal). Setalah ini, akan terbentuk kode *shannon-Fano* untuk setiap karakter.



Gambar 4. Proses kompresi

Setelah melakukan proses penyimpanan file yang telah dikompresi maka akan tampil ukuran file kompresi, rasio kompresi, redundansi dan tujuan penyimpanan file hasil kompresi.

Pada penelitian ini digunakan 25 file-*text* dan terbagi menjadi 2 kategori yaitu file-*teks* dengan ukuran file yang sama dan memiliki karakter yang bervariasi dan file-*teks* dengan ukuran file yang berbeda dan memiliki karakter yang bervariasi.



Gambar 5. Dekompresi

Pada file*text* dengan ukuran file yang sama dan karakter bervariasi, hasil pengujian kompresi file*text* diperoleh ukuran asli file, ukuran hasil kompresi, rasio kompresi dan redudansi tergantung dari karakter yang terdapat dalam sebuah file*text*. Pada file“ABCD” dan “+=”;?” terdapat karakter yang bervariasi sehingga rasio kompresi yang dihasilkan tinggi, sedangkan file “A” dan “AB”, terdapat karakter yang tidak terlalu bervariasi sehingga file tersebut dapat dikompresi dengan rasio yang lebih rendah. Hal ini dapat dibuktikan dengan hasil pengujian “A” ampai “+=”;?” didapatkan persentase rasio dari 13.30% naik menuju 32.93%.

Ukuran hasil kompresi yang dihasilkan tergantung dari banyaknya karakter yang terdapat dalam sebuah file-*text*. *Pada* file-*text* “+=”;?” terdapat jumlah karakter yang bervariasi sehingga ukuran hasil kompresi yang dihasilkan besar yaitu 869 *byte*, dibandingkan file-*text*“A”, yang tidak terdapat karakter yang bervariasi sehingga ukuran hasil kompresi kecil (351 *byte*)

.

Tabel 1. File uji dengan ukuran 2639 byte

No.	Nama file	Ukuran (byte)		Juml.	Rasio Kompr (%)	Redud.
		File Asli	File Kompresi	Krktr.		(%)
1.	A.txt	2639	351	2	13.30	87.70
2.	AB.txt	2639	481	3	18.23	81.77
3.	ABC.txt	2639	595	4	22.55	77.45
4.	ABCD.txt	2639	747	5	28.31	71.69
5.	+=";?.txt	2639	869	6	32.93	67.07
6.	A.rtf	2639	351	2	13.30	87.70
7.	AB.rtf	2639	481	3	18.23	81.77
8.	ABC.rtf	2639	595	4	22.55	77.45
9.	ABCD.rtf	2639	747	5	28.31	71.69
10.	+=";?.rtf	2639	869	6	32.93	67.07
Rata-rata		2639	608.6	1.49	7.36	7.66
Standar Deviasi		0	194.24	4	23.06	77.14

Tabel 2. File uji dengan karakter yang bervariasi

No	Nama file	Ukuran (byte)		Juml Karaktr	Rasio Kompr (%)	Redud. (%)
		File Asli	File Kompresi			
1.	Coding.txt	43217	26784	90	61.98	38.02
2.	Kompresi.txt	7736	5177	79	66.92	33.08
3.	Acak.txt	8517	5855	63	68.74	31.26
4.	Coding.rtf	43217	26784	90	61.98	38.02
5.	Kompresi.rtf	7736	5177	79	66.92	33.08
6.	Acak.rtf	8517	5855	63	68.74	31.26
7.	Daftar pustaka.txt	941	1366	67	145.16	-45.16
8	FormatControl.txt	2066	1934	60	93.61	6.39
9.	JslidingPanel.txt	6185	4298	75	69.49	30.51
10.	Speech.txt	407	594	37	145.95	-45.95
11.	Speech Rasio.rtf	812	805	37	99.14	0.86
12.	FeatureExtract.rtf	4667	3538	65	75.81	24.19
13.	panelBackround.rtf	758	1009	52	133.11	-33.11
14.	Frekuensi.rtf	2273	1894	52	83.33	16.67
15.	Pustaka.txt	4711	3757	67	79.75	20.25
Rata-rata		9450,67	6321,8	65,07	88,04	11,96

Dari file-*teks* dengan ukuran file yang berbeda dan memiliki karakter yang bervariasi (Tabel 1) dapat dilihat bahwa adanya pembesaran pada file Daftar Pustaka.txt, Speech.txt dan panelBackround.rtf setelah dilakukan proses kompresi. Hal tersebut disebabkan karena kecilnya ukuran file dan file tersebut memiliki jumlah karakter yang bervariasi.

Pada pengujian file Speech Rasio.rtf, Frekuensi.rtf dan Pustaka.txt yang memiliki ukuran file yang berbeda dengan file Daftar Pustaka.txt, Speech.txt dan panelBackround.rtf , memiliki jumlah karakter yang sama dengan file Speech Rasio.rtf , Speech.txt, Daftar Pustaka.txt dan Pustaka.txt, Frekuensi.rtf dan panelBackround.rtf. Namun, hasil kompresi pada file Speech Rasio.rtf, Frekuensi.rtf dan Pustaka.txt tidak mengalami pembesaran karena ukuran file yang lebih besar.

Tabel 3. Hasil dekompresi file *.txt

No	Nama File	Ukuran File Terkomp. (byte)	Ukuran Hasil Dekompr. (byte)
1.	A.txt	351	2639
2.	AB.txt	481	2639
3.	ABC.txt	595	2639
4.	ABCD.txt	747	2639
5.	+=";?.txt	869	2639
6.	Coding.txt	26784	43217
7.	Kompresi.txt	5177	7736
8.	Acak.txt	5855	8517
9.	Daftar pustaka.txt	1366	941
10.	FormatControl.txt	1934	2066
11.	JslidingPanel.txt	4298	6185
12.	Pustaka.txt	3757	4711
13.	Speech.txt	594	407
Rata-rata		**4062,154**	**6690,385**
Standar Deviasi		**7097,65**	**11250,14**

Tabel 4. Hasil dekompresi file *.rtf

No	Nama File	Ukuran File Terkompr. (byte)	Ukuran Hasil Dekompr. (byte)
1.	A.rtf	351	2639
2.	AB.rtf	481	2639
3.	ABC.rtf	595	2639
4.	ABCD.rtf	747	2639
5.	+=";?.rtf	869	2639
6.	Coding.rtf	26784	43217
7.	Kompresi.rtf	5177	7736
8.	Acak.rtf	5855	8517
9.	Speech Rasio.rtf	805	812
10.	FeatureExtract.rtf	3538	4667
11.	panelBackround.rtf	1009	758
12.	Frekuensi.rtf	1894	2273
Rata-rata		**4008,75**	**6764,583**
Standar Deviasi		**7418,752**	**11731,78**

Tabel 3 dan 4 menunjukan hasil dekompresi dekompresi file *.txt dan hasil dekompresi file *.rtf. Hasil dari proses dekompresi dari sistem terbukti dapat mengembalikan sebuah file *.snf menjadi seperti file aslinya.

4. Kesimpulan

Penelitian ini telah menguji kompresi algoritma *Shannon-Fano*, dengan dua kategori, yaitu file uji dengan ukuran yang sama dan memiliki karakter yang bervariasi dan file uji dengan ukuran yang berbeda dan karakter yang bervariasi. Dari hasil pengujian dapat disimpulkan bahwa file-*text* dengan ukuran yang sama dan memiliki karakter bervariasi akan menghasilkan tingkat pemampatan file yang lebih rendah dibandingkan file *text* yang memiliki karakter yang tidak terlalu bervariasi. File-*text* dengan karakter yang bervariasi akan menghasilkan pemampatan file yang rendah, sedangkan file-*text* dengan karakter yang tidak bervariasi memiliki tingkat pemampatan file yang tinggi dengan ukuran file yang sama.

Daftar Pustaka

Anton. 2009, "*Kompresi dan Teks"*, Fakultas Teknik Informatika, Univesitas Kristen Duta Wacana, http://lecturer.ukdw.ac.id/anton/download/multimedia6.pdf,diakses: 05 September 2013

Gozali FM.. 2004, "*Analisis perbandingan kompresi data dengan teknik arithmetic coding dan run length encoding"*, TimDigitalStudio, Jakarta.

Lidya R. 2012, *"Implementasi dan Analisis Kinerja Algoritma Shannon-Fano Pada Kompresi Citra BMP"*, Skripsi, Jurusan Teknologi Informasi FISILKOM-TI USU, Medan.

Martin I. 2007, "*Pemrograman Berbasis Objek dengan Bahasa Java"*, Elex Media Komputindo, Jakarta.

Munir R. 2004, " *Pengolahan Citra Digital dengan Pendekatan Algoritmik"*, Informatika, Bandung.

Pu I. 2006, "*Fundamental Data Compression"*, Elsevier, British.

Purnama R. 2003, "*Pemrograman Java Jilid 1,2"*, Prestasi Pustaka, Surabaya.

Purnomo H, Zacharias T. 2005, "*Pengenalan Informatika Perspektif Teknik dan Lingkungan"*, MediaKomputindo, Yogyakarta.

Santi R. 2010, *"Perancangan perangkat lunak kompresi* file *text dengan menggunakan algoritma shannon-Fano"*, Skripsi, Jurusan Ilmu Komputer FMIPA USU, Medan.

Suhendra A. 2004, *"Catatan Kuliah Pengantar Pengolahan Data Text"*, http://images.analyst71.multiply.com/attachment/0/Rz6WgoKCiQAAFxBe81/Catatan%20Kuliah%20PC%202007.pdf-, diakses tanggal 12 Agustus 2013.

Sudewa IBA. 2003, "*Mengenal Character Set"*, MediaKomputindo, Yogyakarta.

Sutoyo T. 2009,"*Teori Pengolahan Citra Digital",* Pener*bit* ANDI, Yogyakarta.

Wijaya, Marvin CH, Prijono, A. 2007, "*Pengolahan Data Text menggunakan Matlab*, Informatika, Bandung.