JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 4, Agustus 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i4.3593
Hal 108−113

Perancangan Aplikasi Kompresi File Dokumen Menggunakan Algoritma Adiitive Code

Rizki Yanur Tanjung, Mesran

Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email:rizki.smart.schooll@gmail.com
Submitted **13-06-2021**; Accepted **25-06-2021**; Published **30-08-2021**

Abstrak

Seiring dengan berkembangnya teknologi saat ini, data memiliki peranan yang sangat penting. Perkembangan teknologi yang canggih mengakibatkan banyaknya data digital yang tersimpan dalam sebuah media penyimpanan. Terlebih lagi jika data tersebut terus disimpan dalam jangka waktu yang tidak ditentukan hingga mencapai batas maksimal penyimpanan, maka media penyimpanan akan penuh dan tidak dapat menyimpan data lainnya. Data dapat berupa teks, gambar, audio atau bahkan video. Data kerap kali memiliki ukuran yang sangat besar. Untuk itu diperlukan sebuah teknik untuk mengubah ukuran data tersebut agar menjadi lebih kecil. Teknik ini disebut dengan pemampatan atau yang lebih dikenal dengan kompresi data. Kompresi data adalah suatu proses pengubahan sekumpulan data menjadi suatu bentuk kode untuk menghemat kebutuhan tempat penyimpanan data. Atas dasar tersebut Perancangan Aplikasi Kompresi File Dokumen Menggunakan Algoritma Additive Code dibuat agar kiranya dapat membantu pengguna komputer mengompresi file dokumen yang awalnya berukuran besar menjadi ukuran terkecilnya.

Kata Kunci: Kompresi; Dokumen; Unified Modelling Language; Additive Code

Abstrack

Along with the development of technology today, data has a very important role. Sophisticated technological developments have resulted in a large amount of digital data stored in a storage medium. Moreover, if the data continues to be stored for an indefinite period of time until it reaches the maximum storage limit, the storage media will be full and cannot store other data. Data can be text, images, audio or even video. Data often has a very large size. For that we need a technique to change the size of the data to make it smaller. This technique is called compression or better known as data compression. Data compression is a process of converting a set of data into a coded form to save the need for data storage space. On this basis, the Design of a Document File Compression Application Using the Additive Code Algorithm is made so that it can help computer users to compress document files that were originally large to their smallest size.

Keywords: Compression; Document; Unified Modeling Language; Additive Code

1. PENDAH ULUAN

Kompresi data adalah proses mengubah aliran data masukan menjadi aliran data baru yang lebih kecil. Aliran bermasalah adalah file atau buffer di memori. Ada banyak metode kompresi data. Kompresi lossy dan lossless adalah pengelompokan metode kompresi berdasarkan integritas data. Kompresi teks dapat dilakukan dengan mengompresi konten (konten berukuran besar membuat ukurannya lebih kecil tanpa kehilangan keaslian datanya) [1].

Hal inilah yang menyebabkan file harus dimampatkan agar ukurannya menjadi lebih kecil. Teknik pemampatan data ini disebut dengan teknik kompresi data. Tujuan dari kompresi data adalah untuk mengurangi ukuran file sebelum menyimpan atau memindahkan data ke dalam media penyimpanan. Teks pada umumnya berisi rangkaian karakter dan dapat membentuk suatu kata, surat dan narasi. Misalnya, file teks yang berekstensi (*.docx) yang ukurannya besar mengakibatkan proses pengiriman semakin lama, serta menggunakan ruang memori yang besar dalam penyimpanan nya. Maka untuk menghemat ruang penyimpanan dan mempercepat proses pengiriman file teks, perlu dilakukan proses kompresi agar ukuran file teks tersebut menjadi lebih kecil [2].

Teknik kompresi memiliki dua metode utama yaitu metode *lossless* dan metode *lossy*. Metode *lossy* adalah kompresi dimana terdapat data yang hilang selama proses kompresi yang mengakibatkan kualitas data yang hilang selama proses kompresi yang mengakibatkan kualitas data yang dihasilkan jauh lebih rendah dari kualitas data asli, sedangkan metode *lossless* tidak menghilangkan data selama proses kompresi terjadi, akibatnya kualitas data hasil kompresi tidak menurun. Salah satu algoritma kompresi yang dapat digunakan adalah algoritma *Additive Code* yang merupakan suatu algoritma yang membutuhkan masukan kata bersama dengan distribusi probabilitas untuk setiap masukan dan dikonstruksi sebagai pohon probabilitas yang akan diasosiasikan dengan huruf dari abjad masukan, algoritma ini juga dapat mengkompresi suatu file yang berukuran besar hingga memperkecil ukuran file tersebut [3].

Pada penelitian sebelumnya yang telah dilakukan oleh Nidia Enjelita Saragih dan Fitriana Harahap dalam sebuah jurnal, telah disimpulkan bahwa Kompresi merupakan proses pengubahan sekumpulan data menjadi suatu bentuk kode untuk menghemat kebutuhan tempat penyimpanan dan waktu untuk transmisi data[1]. Sedangkan menurut Sukiman dan Chandra kompresi data atau juga dikenal sebagai pemadatan data adalah teknik yang dipakai untuk mengurangi data atau memperkecil data menjadi bentuk data lain dimana data tersebut diubah menjadi simbol yang lebih sederhana [4].

Dalam pembahasan ini proses yang akan dilakukan adalah dengan merancang sebuah aplikasi kompresi *file* dokumen dengan algoritma *Additive Code* yang akan menghasilkan sebuah *file* dokumen yang sudah terkompres, kemudian dalam mengembalikan *file* dokumen seperti semula akan melalui proses dekompresi.

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 4, Agustus 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i4.3593
Hal 108−113

2. METODOLOGI PENELITIAN

2.1 Kompresi Data

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. Pemampatan merupakan salah satu cara dalam ilmu komputer yang bertujuan untuk memadatkan data sehingga hanya memerlukan ruang penyimpanan yang lebih kecil[5]. Kompresi memiliki 2 teknik, antara lain:

1. Kompresi *lossy* adalah 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.
2. Kompresi *lossless* adalah 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.

2.2 Rasio Kompresi

Proses kompresi adalah proses encoding yang menghasilkan data yang sudah dikompresi yang disebut aliran data encoded. Sebaliknya aliran data yang telah dikompresi harus dilakukan proses dekompresi untuk menghasilkan kembali aliran data yang asli. Karena proses dekompresi menghasilkan decoding dari aliran data yang sudah dikompresi maka hasilnya adalah aliran data decoded. Tingkat pengurangan data yang dicapai sebagai hasil dari proses kompresi disebut rasio kompresi. Rasio ini merupakan perbandingan antara panjang data string asli dengan panjang data string yang sudah dikompresi, seperti dituliskan dalam persamaan berikut:

$$\text{Rasio} = \frac{\text{ukuran file asli}}{\text{ukuran file terkompresi}} \quad (1)$$

Jika dinyatakan dalam prosentase maka dituliskan dalam persamaan berikut:

$$\text{P} = \left(\frac{1-\text{ukuran file terkompresi}}{\text{ukuran file asli}}\right) \times 100\% \quad (2)$$

Yang berarti ukuran file berkurang sebesar P (dalam persentase) dari ukuran semula. Semakin tinggi rasio tingkat suatu teknik kompresi data maka semakin efektif teknik kompresi tersebut[6].

2.3 Algoritma Elias Additive Code

Algoritma *Additive Code* adalah algoritma yang mengarah pada kode yang sederhana dan efisien yang menunjukkan bahwa urutan angka lain dapat digunakan dengan cara yang sama untuk membuat kode yang serupa atau mungkin bahkan lebih baik juga dapat memperpanjang Algoritma *Goldbach Code* jika menemukan urutan S dari bilangan bulat "basis" sehingga setiap bilangan bulat n dapat diekspresikan sebagai jumlah dari dua elemen S (dimungkinkan untuk mencari urutan serupa yang bisa mengekspresikan n apapun sebagai jumlah dari tiga, empat, atau lebih elemen Salah satu teknik untuk menghasilkan urutan bilangan bulat dasar didasarkan pada prinsip penyaringan. Dimulai dengan urutan pendek (basis set) yang elemen ai cukup untuk mewakili masing-masing dari beberapa bilangan bulat positif pertama sebagai a sum ai aj. Urutan awal ini kemudian diperbesar secara bertahap. Secara umum, menambahkan elemen baru ak dan berakhir dengan urutan S = (0, a1, a2, ..., ak) seperti bahwa setiap bilangan bulat $1 \leq n \leq ak$ dapat direpresentasikan sebagai penjumlahan dari dua elemen S. Lalu dapat dihasilkan semua jumlah ai + ak untuk nilai i dari 1 hingga k dan periksa untuk memverifikasi bahwa antara lain ak + 1, ak + 2, dan seterusnya. Jika terdapat bilangan bulat pertama hilang, lalu tambahkan ke S sebagai elemen ak + 1. Maka dapat diketahui setiap bilangan bulat $1 \leq n \leq ak + 1$ dapat direpresentasikan sebagai penjumlahan dua elemen dari S, dan dapat terus memperluas barisan S basis bilangan bulat. Langkah-langkah untuk membangun *codeword* dari *Additive Code* adalah sebagai berikut:

1. Tentukan nilai *n* = nilai yang berasal dari *sequence* Algoritma Golbach
2. Tentukan nilai *sum* = penjumlahan 2 *sequence* Algoritma Golbach yang menghasilkan nilai n ($a^i + a^j = n$)
3. *Indexes* = index (posisi urutan) dari a^i, a^j
4. *Pair* = selisih index dari a^i dan a^j (a^i,($a^j a^i$+1)
5. *Codeword* adalah hasil Algoritma Elias Gamma Code dari a^i dan a^j dengan ketentuan:
   a. Temukan bilangan bulat N terbesar sehingga $2^N \leq n < 2^{N+1}$ dan tulis $n = 2^N + L$.
   b. Rubah nilai n menjadi bilangan biner, lalu hilangkan 1 bit paling kiri
   c. Kodekan N dalam bentuk unary sebagai N nol diikuti oleh 1 atau N 1 diikuti oleh 0
   d. Tambahkan sisa digit biner n dibelakang kode unary yang telah dihasilkan
6. Length = jumlah digit biner yang didapat dari *Codeword*

Langkah-langkah yang terdapat di atas merupakan rumus dalam algoritma *Additive Code* yang digunakan untuk memperoleh sebuah tabel kebenaran, yang dimana bilangan hexadesimal *file* dokumen yang akan di kompresi

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 4, Agustus 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i4.3593
Hal 108−113

n	Sum	Indexes	Pair	Codeword	Len.
10	3+7	3,5	3,3	011:011	6
11	0+11	1,7	1,7	1:00111	6
12	1+11	2,5	2,6	010:00110	8
13	1+12	2,8	2,7	010:00111	8
14	7+7	5,5	5,1	00101:1	6
15	3+12	3,8	3,6	011:00110	8
16	7+9	5,6	5,2	00101:010	8
17	5+12	4,8	4,5	00100:00101	10
18	7+11	5,7	5,3	00101:011	8
20	9+11	6,7	6,2	00110:010	8
30	5+25	4,9	4,6	00100:00110	10
40	11+29	7,11	7,5	00111:00101	10
50	25+25	9,9	9,1	0001001:1	8
60	29+31	11,12	11,2	0001011:010	10
70	35+35	14,14	14,1	0001110:1	8
80	0+80	1,20	1,20	1:000010100	10
90	29+61	11,17	11,7	0001011:00111	14
100	3+97	3,23	3,21	011:000010101	14

Gambar 1. Kode *Additive Code*.

2.3 Dokumen

Dokumen adalah warkat asli yang dipergunakan sebagai alat pembuktian atau sebagai bahan untuk mendukung suatu keterangan[5]. Dokumen dalam kamus komputer bisa dikatakan file yang dibuat oleh software, misalnya seperti Ms word coreldraw notepad dan masih banyak lainya. Sementara istilah “dokumen” pada awalnya disebut khusus untuk dokumen pengolah kata, sekarang digunakan untuk merujuk ke semua jenis file yang disimpan. Oleh karena itu, dokumen dapat berisi teks, gambar, audio, video, dan tipe data lainnya. Dokumen diwakili dengan ikon dan nama file. Ikon menyediakan representasi visual` dari jenis file, sedangkan nama file memberikan nama unik untuk file. Sebagian besar nama file dokumen juga menyertakan ekstensi file, yang menentukan jenis file dokumen. Misalnya, dokumen Microsoft Word mungkin memiliki ekstensi file .DOCX, sedangkan dokumen Photoshop mungkin memiliki ekstensi file .PSD.

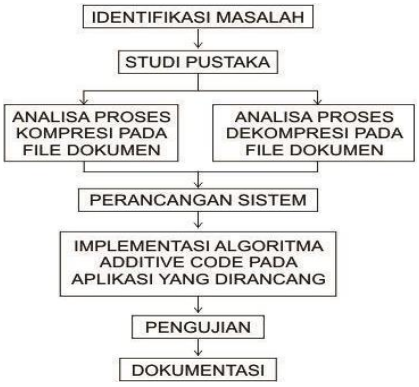
2.4 DOCX

Ekstensi DOC pertama kali muncul pada dokumen yang digunakan oleh pengolah kata WordPerfect yaitu pada tahun 1980. DOC merupakan singkatan dari dokumen , adalah sebuah ekstensi file yang digunakan pada dokumen pengolah kata biasanya digunakan pada Microsoft Word. Pada tahun 1990-an Microsoft memilih menggunakan ekstensi *.DOC pada pengolah kata Microsoft Word. File dengan ekstensi DOC bervariasi, pada versi Microsoft Word 97 dan 2003 masih menggunakan file dengan ekstensi *.DOC, sedangkan pada Microsoft Word 2007 dan 2010 format dengan ekstensi *.doc di ganti dengan ekstensi docx. Jadi dapat disimpulkan bahwa file DOCX adalah format penyimpanan file dokumen yang ada pada Microsoft Office [7].

3. HASIL DAN PEMBAHASAN

3.1 Pembahasan

Pada penelitian ini akan dilakukan analisa dan perancangan perangkat lunak pengkompresian *file* dokumen dengan menggunakan algoritma *Additive Code*. Algoritma *Additive Code* merupakan salah satu teknik kompresi lossless yang dapat memperkecil suatu data berdasarkan dengan karakter pada objek yang akan dilakukan proses kompresi. Penggunaan algoritma *Additive Code* akan dilakukan berdasarkan karakter yang sering muncul dan akan memiliki jumlah bit terkecil berdasarkan kode *Additive Code*, sedangkan karakter yang paling sedikit muncul akan memiliki jumlah bit terpanjang. Tahapan analisa terhadap suatu sistem dilakukan sebelum tahapan perancangan dilakukan.



Gambar 2. Skema Analisa Yang Digunakan.

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 4, Agustus 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i4.3593
Hal 108−113

Penelitian ini akan membahas 2 proses utama dalam kompresi *file* dokumen yaitu proses kompresi dan proses dekompresi. Penulis akan mengkompresi dokumen dengan menggunakan algoritma *Additive Code* merupakan salah satu algoritma yang termasuk di dalam metode *lossless* data. Langkah-langkah yang terdapat di atas merupakan rumus dalam algoritma *Additive Code* yang digunakan untuk memperoleh sebuah tabel kebenaran, yang dimana bilangan heksadesimal *file* dokumen yang akan di kompresi.

Contoh Kasus:

Goldbach Code Sequence= (0,1,3,5,7,9,11,…)

n = 10, maka Sum penjumlahan pertama ditemukan dengan hasil = n Sum= 3+7

Untuk mencari index, dapat dilihat pada Goldbach Sequence dimana 3 berada pada urutan ke 3 dan 7 adalah urutan ke 5.

Maka index= 3,5 Pair= $a^i,(a^j-a^i+1)$ Pair=3,(5-3+1)= 3,3

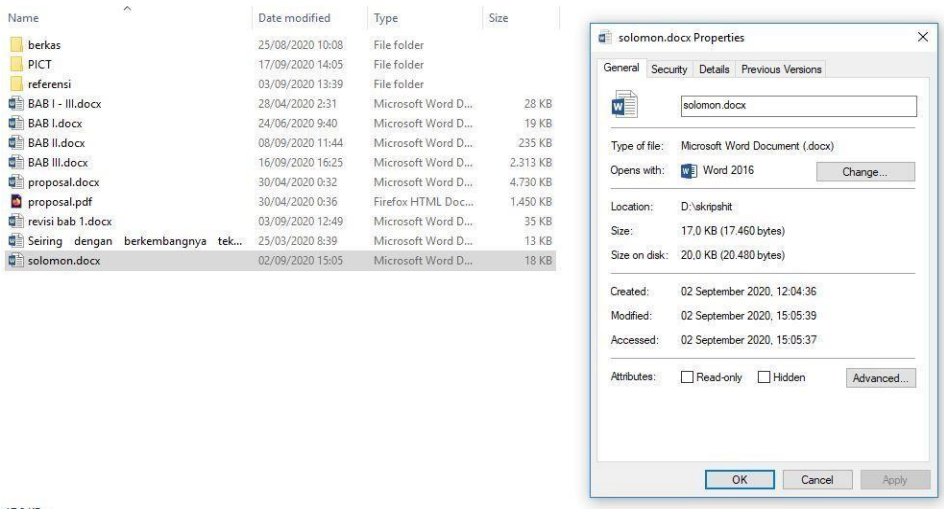
Untuk mencari *codeword*, maka dibutuhkan Algoritma Elias Gamma Code dari kedua pair dengan ketentuan

1 Temukan bilangan bulat N terbesar sehingga
$2^N \leq n < 2^{N+1}$ dan tulis $n = 2^N + L$.
Rubah nilai n menjadi bilangan biner, lalu hilangkan 1 bit paling kiri

2. Kodekan N dalam bentuk unary sebagai N nol diikuti oleh 1 atau N 1 diikuti oleh 0

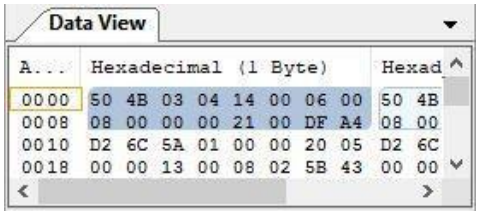
3. Tambahkan sisa digit biner n dibelakang kode unary yang telah dihasilkan
Contoh:
n = 5
$2^N \leq n < 2^{N+1}$
$2^2 \leq 5 < 2^{2+1}$
$4 \leq 5 < 8$
$5 = 101 \rightarrow 01$
Unary (2) → 001
Codeword : 00101
Kembali ke perhitungan diatas dimana pair adalah 3,3, jika dihitung, maka
codeword dari 3,3 adalah 011:011
Length= banyaknya digit dari sebuah *codeword*
Length dari 011:011 = 6

Sebelum file yang akan di kompresi terlebih dahulu dilakukan pembacaan biner yang terdapat pada file dokumen untuk mendapatkan data berupa data heksadesimal. Membaca heksadesimal yang terdapat pada file gambar menggunakan aplikasi Binery Viewer untuk mencari nilai biner pada file dokumen. Berikut adalah contoh file dokumen yang akan di kompresi dan didekompresi



Gambar 3. Sample File Dokumen

Berdasarkan gambar dibawah terdapat nilai binery viewer, adapun nilai heksadecimal file dokumen yang didapat pada tabel di bawah menggunakan aplikasi binery viewer, hasil nilai heksadesimal ditampilkan pada gambar 4.



Gambar 4. Nilai Heksadesimal Sample File Dokumen.

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 4, Agustus 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i4.3593
Hal 108−113

Berdasarkan hasil gambar 4 telah didapat nilai heksadesimal file dokumen yang akan dikompresi, dari hasil nilai bilangan heksadesimal yang terdapat diatas, penulis mengambil sampel 8 kolom ke kanan dan 2 baris ke bawah, sehingga nilai sample yang diambil sebanyak 16 bilangan heksadesimal.

Tabel 1. Tampilan Nilai Bilangan Heksadesimal

50	4B	03	04	14	00	06	00
08	00	00	00	21	00	DF	A4

Berdasarkan pada gambar dan tabel diatas, didapat nilai bilangan heksadesimal file dokumen DOCX. Untuk keperluan hitungan manual penulis hanya mengambil sampel 16 bilangan heksadesimal file DOCX. Adapun bilangan heksadesimal file DOCX sampel tersebut adalah sebagai berikut : 50 4B 03 04 14 00 06 00 08 00 00 00 21 00 DF A4. Berikutnya nilai bilangan hexsadesimal diurutkan berdasarkan frekuensinya dan di cari nilai binernya, Nilai frekuensi kemunculan yang paling banyak maka akan berada diurutan pertama..

Tabel 2. Nilai Heksa Berdasarkan Frekuensi Kemunculan

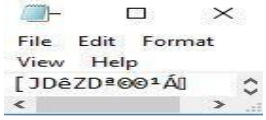
Nilai Heksadesimal	Nilai Biner	Bit	Freq	Bit × Freq
00	00000000	8	6	48
50	01000110	8	1	8
4B	01010010	8	1	8
03	01001001	8	1	8
04	01001010	8	1	8
14	11001011	8	1	8
06	10011110	8	1	8
08	01010111	8	1	8
21	01000001	8	1	8
DF	00010000	8	1	8
A4	00000001	8	1	8
Total bit				128

Berdasarkan pada pembagian kelompok nilai biner, didapatkan 13 kelompok nilai biner baru yang sudah terkompresi beserta nilai biner penambahan bit. Setelah pembagian dilakukan, maka *pixel* yang sudah dibagi dirubah kedalam suatu karakter dengan terlebih dahulu mencari nilai desimal dari *string bit* tersebut menggunakan kode ASCII untuk megetahui nilai dari heksadesimal yang sudah terkompresi. Adapun nilai heksadesimal yang sudah terkompresi dapat dilihat pada tabel di bawah ini.

Tabel 3. Hasil Karakter Terkompresi

No	Codeword Hasil Kompresi	Desimal	Heksadesimal	Karakter
1	01011011	91	5B	[
2	01001010	74	4A	J
3	01000100	68	44	D
4	11101010	234	EA	ê
5	01011010	90	5A	Z
6	01000100	68	44	D
7	10101010	170	AA	ª
8	10101001	169	A9	©
9	10101001	169	A9	©
10	10111001	185	B9	¹
11	11000001	193	C1	Á
12	00000111	7	7	

Setelah nilai desimal diketahui, maka mengubah nilai desimal kedalam suatu karakter. Karakter hasil dari proses kompresi yang dihasilkan tersimpan dalam suatu *file* dengan ekstensi ".adv", dan jika *file* tersebut dibuka dengan aplikasi notepad, maka akan tampil karakter seperti gambar berikut



Gambar 3. Hasil Karakter Kompresi

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 4, Agustus 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i4.3593
Hal 108−113

4. KESIMPULAN

Berdasarkan penelitian yang dilakukan, dapat disimpulkan dengan mengikuti prosedur untuk mengkompresi file dokumen dan hasilnya adalah file dokumen berhasil dikompresi. Setelah diimplementasikan untuk mengkompresi file dokumen, hasilnya yaitu dapat mengkompresi sampai dengan rasio 64%.

REFERENCES

[1] R. D. Pratiwi, S. D. Nasution, and F. Fadlina, "Perancangan Aplikasi Kompresi File Teks Dengan Menerapkan Algortima Fixed Length Binary Encoding (Flbe)," J. Media Inform. Budidarma, vol. 2, no. 1, pp. 10–14, 2018, doi: 10.30865/mib.v2i1.813
[2] A. Nugrohoadhi, "Pengorganisasian Dokumen dalam Kegiatan Kepustakawanan," Khizanah al-Hikmah J. Ilmu Perpustakaan, Informasi, dan Kearsipan, vol. 3, no. 1, pp. 1–10, 2015, doi: 10.24252/kah.v3i1a1.
[3] Sukiman and T. Chandra, "Aplikasi Kompresi File dengan Algoritma Elias Gamma," no. 18.
[4] J. 17110443 Sitompul, "Analisa Sentimen Masyarakat Terhadap Calon Legislatif Partai Politik Dengan Menerapkan Algoritma Text Mining," J. Tek. Mesin, vol. 06, no. 2, pp. 69–73, 2017, doi: 10.22441/jtm.v6i2.1183.
[5] J. N. Denenberg, E. D. Weinberger, and M. L. Gordon, "DATA COMPRESSION METHOD FOR USE IN A COMPUTERIZED INFORMATIONAL AND TRANSACTIONAL NETWORK," vol. 32, N, 1996.
[6] H. Sartika and T. Zebua, "Perancangan Dan Implementasi Algoritma Elias Gamma Code Untuk Mengkompresi Record Database Pada Aplikasi Rangkuman Pengetahuan Umum Lengkap," KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer), vol. 3, no. 1, pp. 259–265, 2019, doi: 10.30865/komik.v3i1.1600
[7] D. Salomon and G. Motta, Handbook of Data Compression, Fifth Edit. New York: Springer, 2010.