KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)
Volume 6, Nomor 1, November 2022
DOI: 10.30865/komik.v6i1.5787
ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 34-42

Implementasi Algoritma *Rice Code* dan *Ternary Comma Code* Pada Kompresi *File* PDF

Ade Novian Purnama

Fakultas Ilmu Komputer dan Teknologi Informasi, Prodi Teknik Informatika, Universitas Budi Darma Medan, Medan, Indonesia
Email: adek.aa41@gmail.com

Abstrak-Sekarang ini, sumber informasi yang didapatkan bukan hanya melalui media cetak melainkan media elektronik seperti misalnya e-book (buku digital). E-book bisa diakses di laptop, tablet, smartphone dan media elektronik lainnya. Biasanya *file* e-book yang digunakan berekstensi PDF. PDF sebagai ekstensi e-book memiliki kekurangan. Kekurangan PDF adalah format yang sedikit melakukan kompresi sehingga menyebabkan e-book berekstensi PDF memiliki ukuran yang lumayan besar. Hal itu menyebabkan dalam proses pembacaan data dan pengiriman data memerlukan waktu yang cukup lama, dan tidak jarang sering mengalami kegagalan pada saat proses pengiriman. Untuk mengatasi ukuran *file* PDF yang cukup besar perlu dilakukan proses kompresi. Kompresi adalah proses pemampatan data baru yang memiliki ukuran lebih kecil yang berupa kumpulan karakter dengan tujuan untuk menghemat kebutuhan ruang penyimpanan dan mempercepat proses pemindaian data. Oleh sebab itu digunakan algoritma *Rice Code* dan *Ternary Comma Code*, yang dimana kedua algoritma ini diterapkan secara beruntutan (double compression) agar *file* dapat terkompresi dengan ukuran yang lebih maksimal. Hasil akhir dari proses double compression oleh algoritma *Rice Code* dan *Ternary Comma Code* yang dilakukan penulis didapatkan nilai Compression Rate (CR) sebesar 20%, yang artinya ukuran awal *file* PDF dapat dikompres hingga ukurannya hanya mencapai 20% dari ukuran *file* PDF aslinya.

Kata Kunci: PDF, Kompresi Berganda, *Ternary Comma Code*, *Rice Code*, Implementasi

Abstract-Currently, the source of information obtained is not only through print media but also electronic media such as e-books (digital books). E-books can be accessed on laptops, tablets, smartphones and other electronic media. Usually the e-book *file* used is with a PDF extension. PDF as an eBook extension has its deficiency. The deficiency of PDF is a format that does a little compresion, causing e-books with PDF extensions to have a fairly large size. This causes the process of reading data and sending data takes quite a long time, and often fail during the delivery process. To solve the size of the PDF *file* that is large enough, compression process is required. Compression is a new data compression process that has a smaller size in the form of a character set with the aim of saving storage space requirements and speeding up the data scanning process. Therefore, the *Rice Code* and *Ternary Comma Code* algorithms are used, both of which are applied sequentially (double compression) so that *file*s can be compressed with a maximum size. The final result of the double compression process by the *Rice Code* and *Ternary Comma Code* algorithms carried out by the author obtained a Compression Rate (CR) value of 20%, which means that the initial PDF *file* size can be compressed until its size only reaches 20% of the original PDF *file* size.

Keywords: PDF, Double Compression, *Ternary Comma Code*, *Rice Code*, Implementation

1. PENDAHULUAN

Sekarang ini perkembangan pada ilmu pengetahuan semakin pesat dan cepat. Perkembangan ini merubah cara manusia dalam mencari sumber informasi. Sumber informasi yang didapatkan bukan hanya melalui media cetak melainkan media elektronik seperti misalnya e-book (buku digital). Penggunaan e-book saat ini menggeser penggunaan buku dalam bentuk cetakan (printed book). Hal ini dikarenakan setiap gadget bisa mengakses e-book seperti laptop, tablet, smartphone dan media elektronik lainnya. E-book juga memberikan manfaat berupa menghemat biaya, bisa diakses dimanapun, mengurangi jumlah penggunaan kertas dan lainnya. Biasanya *file* e-book yang digunakan berekstensi PDF.

Portable Document Format juga dikenal sebagai PDF adalah format *file* yang dibuat oleh Adobe Systems pada tahun 1993 yang digunakan untuk bertukar dokumen digital. Saat ini *file* dengan ekstensi PDF sering digunakan karena memiliki keunggulan. Keunggulan dari *file* PDF adalah mereka dapat menyimpan data yang mewakili dokumen dua dimensi, termasuk huruf, teks, dan gambar. *File* yang disimpan dengan ekstensi PDF tidak rentan terhadap virus dan tidak mengubah format yang ada saat dibuka di perangkat lain[1][2].

Namun penggunaan *file* PDF sebagai ekstensi e-book memiliki kekurangan karena ekstensi PDF adalah format yang sedikit kompresi sehingga menyebabkan e-book dalam ekstensi PDF memiliki ukuran yang lumayan besar. Hal itu menyebabkan dalam proses pembacaan data dan pengiriman data memerlukan waktu yang cukup lama karena ukuran yang dimilikinya, dan tidak jarang sering mengalami kegagalam pada saat proses pengiriman[2]–[4].

Untuk mencegah agar *file* PDF yang dikirim tidak memiliki ukuran yang terlalu besar maka perlu dilakukan kompresi *file* PDF. Sehingga, pertukaran data dapat dilakukan dengan mudah dan cepat karena ukuran *file* PDF yang telah terkompresi. Ada banyak aplikasi kompresi dengan algoritma yang diketahui. Hal ini menyebabkan hasil kinerja kompresi yang berbeda. Dalam penelitian ini kompresi yang dilakukan adalah kompresi berganda. Kompresi berganda dilakukan agar *file* yang terkompresi menjadi lebih maksimal, hal ini dipengaruhi oleh algoritma yang diterapkan dalam proses kompresi dengan cara dengan mengurangi jumlah bit yang sama dari data asli. Kompresi berganda ini menggunakan algoritma *Rice Code* dan algoritma *Ternary Comma Code*.

Algoritma *Rice Code*s merupakan algoritma kompresi yang penerapannya tidak mengurangi isi data dari *file* aslinya. Algoritma *Ternary Comma Code* merupakan algoritma yang berjenis lossless yang dapat digunakan untuk mengkompresi *file* dengan ruang waktu yang lebih cepat dibanding dengan algoritma yang lain[5][6].

Berdasarkan penelitian yang dilakukan oleh Frida Effelyanti Naibaho pada tahun 2020 tentang "Implementasi Algoritma J-Bit Encoding Pada Kompresi *File* PDF" memberikan kesimpulan bahwa pengompresian menggunakan

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 34-42

Algoritma J-Bit Encoding bekerja dengan cara membagi suatu data masukan menjadi dua keluaran, dan kemudian diimplementasikan kembali menjadi satu keluaran[2].

Berdasarkan penelitian yang dilakukan oleh Muhammad Asnawi Latif dkk pada tahun 2018 tentang "Analisa Perbandingan Algoritma *Rice Code*s Dengan Algoritma Goldbach Code Pada Kompresi *File* Teks Menggunakan Metode Exponential" menghasilkan kesimpulan bahwa penerapan algoritma *Rice Code*s dan algoritma Goldbach Code sangat bagus, isi dari hasil pengompresian pun tidak berkurang dari *file* aslinya, dengan penerapan metode perbandingan Exponential dengan 3 kriteria, hasil kompresi yang lebih bagus adalah algoritma *Rice Code*s[5].

Berdasarkan penelitian yang dilakukan oleh Nurma Ningsih dkk pada tahun 2020 tentang "Penerapan Algoritma *Ternary Comma Code* Pada Aplikasi Kompresi *File* Gambar" menyimpulkan bahwa setelah penerapan algoritma *Ternary Comma Code* pada aplikasi kompresi *file* gambar membuat memudahkan pengalokasian media penyimpanan karena dihasilkan ukuran *file* gambar yang lebih kecil[6].

Berdasarkan penelitian yang dilakukan Putri Fitria pada tahun 2020 tentang "Penerapan Algoritma *Rice Code* Pada Aplikasi Kompresi *File* Gambar" menyimpulkan bahwa algoritma *Rice Code* mampu mengkompresi *file* gambar yang berekstensi jpeg dengan baik, kompresi yang dihasilkan tidak mengurangi data *file* aslinya sehingga hasilnya tidak mengalami penurunan kualitas[7].

Berdasarkan penelitian yang dilakukan Irwansyah dkk pada tahun 2018 tentang "Perancangan Aplikasi Kompresi *File* Teks Dengan Menerapkan Algoritma *Rice Code*s" menyimpulkan bahwa algoritma *Rice Code*s dapat menjadi jawaban sebagai algoritma pengompresian *File* teks dengan memperkecil ukuran bit dari *file* tersebut, dan juga mengatasi ketersediaan ruang penyimpanan yang ada[8].

Berdasarkan peneltian yang telah dilakukan oleh ahli terdahulu saat algoritma *Rice Code*s dan *Ternary Comma Code* terbukti sangat efektif dalam mengatasi permasalahan terhadap keterbatasan ruang penyimpanan dengan mengkompresi *file* dokumen yang ada. Dalam penelitian ini akan menerapkan algoritma *Rice Code*s dan algoritma *Ternary Comma Code* untuk mengkompresi *File* PDF, yang dimana hasil kompresi dari algoritma *Rice Code*s akan dikompresi lagi dengan menggunakan algoritma *Ternary Comma Code* yang diharapkan ukuran bytes dari *file* PDF juga dapat terkompresi sehingga menghemat ruang media penyimpanan dan mempercepat proses saat pengiriman data dengan menggunakan *file* yang berekstensi PDF.

2. METODOLOGI PENELITIAN

2.1 Kompresi *File* PDF

2.1.1 Kompresi

Kompresi adalah proses pemampatan data dalam ilmu komputer yang dilakukan untuk menghasilkan aliran data baru yang memiliki ukuran lebih kecil yang berupa kumpulan karakter dengan tujuan untuk menghemat kebutuhan ruang penyimpanan dan mempercepat proses pemindaian data. Pengkompresian ini sangat diperlukan dan menguntungkan jika diterapkan ke suatu data yang mengandung karakter yang berulang dan berukuran besar sehingga dapat mengurangi ukuran datanya. Proses pengurangan bit terhadap representasi teks, gambar, audio dan vidio dilakukan untuk mengurangi ukuran data namun kualitas informasi yang terkandung di dalam data tersebut tetap terjaga[5][8].

Teknik kompresi yang dilakukan pada suatu data memiliki beberapa faktor yang bisa digunakan untuk dijadikan penilaian dalam menentukan kualitas pengkompresian suatu data, antara lain :

a. Ratio Of Compression (RC)
   Ratio Of Compression (RC) adalah nilai perbandingan antara ukuran bit data yang belum dikompresi dengan ukuran bit data yang telah dilakukan proses kompresi. Secara sistematis Ratio Of Compression (RC) dapat dirumuskan sebagai berikut :
   $$RC = \frac{Ukuran\ data\ sebelum\ dikompresi}{Ukuran\ data\ setelah\ dikompresi} \quad (1)$$
b. Compression Ratio (CR)
   Compression Ratio (CR) adalah nilai perbandingan antara ukuran bit data yang sudah dikompresi dengan ukuran bit data yang belum dikompresi dalam persentase. Compression Ratio (CR) dapat dirumuskan sebagai berikut :
   $$CR = \frac{Ukuran\ data\ setelah\ dikompresi}{Ukuran\ data\ sebelum\ dikompresi} x100\% \quad (2)$$
c. Redudancy (Rd)
   Redudancy (Rd) adalah penilaian nilai rasio dengan hasil nilai rasio kompresi. Redudancy (Rd) dapat dirumuskan sebagai berikut :
   $$Rd = \frac{Ukuran\ sebelum\ dikompresi - Ukuran\ setelah\ dikompresi}{Ukuran\ data\ sebelum\ dikompresi} x100\% \quad (3)$$

d. Space Saving (SS)
   Space Saving (SS) adalah selisih antara ukuran data yang belum dikompresi dengan ukuran data yang sudah dikompresi
   $$SS = 100\% - Compression\ Ratio \quad (4)$$

Dalam kompresi terdapat dua teknik yang bisa digunakan untuk melakukan kompresi, antara lain :

1. Lossy Compression

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 34-42

Lossy Compression adalah teknik kompresi data dengan menghilangkan beberapa data yang kurang penting untuk memperoleh hasil pengompresian yang lebih baik dan memiliki ukuran yang lebih kecil dari data aslinya, sehingga data yang dihasilkan dari proses kompresi ini tidak memiliki kemiripan dengan data aslinya. Teknik ini berguna jika prioritas yang dibutuhkan adalah pengecilan ukuran data bukan kualitas. Itu sebabnya teknik ini sangat cocok digunakan untuk mengompresi gambar,vidio atapun audio. Contohnya mp3, jpeg, mpeg, dan lain – lain.

2. Lossless Compression
   Lossless Compression adalah teknik kompresi data dimana data yang terkompresi dapat didekompresi kembali tanpa ada satu pun data yang hilang. Sehingga kualitas data tidak mengalami perubahan. Teknik ini sangat cocok digunakan untuk melakukan kompresi data berupa teks. Sehingga saat dilakukan dekompresi hasilnya sama dengan data yang aslinya. Lossless compression biasanya digunakan untuk akurasi data yang penting seperti data teks.

Pada penelitian ini dilakukan proses kompresi *file* dengan menggunakan teknik lossless compression sehingga saat dilakukan pengompresian *file* yang telah dikompresi dapat dikembalikan ke ukuran semula. *File* yang akan dikompresi adalah *File* Portable Document Format (PDF).

2.1.2 *Portable File Document* (PDF)

Portable Document Format juga dikenal sebagai PDF adalah format *file* yang dibuat oleh Adobe Systems pada tahun 1993 yang digunakan untuk bertukar dokumen digital. Saat ini, kita sering menemukan *file* dokumen dengan ekstensi PDF yang bahkan tidak Menutup kemungkinan ukuran *file* PDF mencapai ukuran puluhan megabit (MB). Ini dikarenakan *file* dengan ekstensi PDF memiliki keunggulan. Keunggulan dari *file* PDF adalah mereka dapat menyimpan data yang mewakili dokumen dua dimensi, termasuk huruf, teks, dan gambar. *File* yang disimpan dengan ekstensi PDF tidak rentan terhadap virus dan tidak mengubah format yang ada saat dibuka di perangkat lain[1][2]. Banyak format buku digital yang mayoritas berisi teks menggunakan format *file* PDF. Hal itu dikarenakan format ini memiliki kelebihan dalam hal format siap cetak yang seperti bentuk buku sebenarnya. Selain itu PDF memiliki fitur pencarian, daftar isi, memuat gambar dan lain – lain[9].

2.2 Algoritma *Rice Code* dan *Ternary Comma Code*

2.2.1 Algoritma *Rice Code*

Rice Code adalah algoritma kompresi yang diciptakan oleh Robert F. Rice yang dimana *Rice Code* adalah family dari algoritma Golomb Codes itu sebabnya *Rice Code* dikenal juga sebagai Golomb-*Rice Code*. Keduanya memiliki persamaan yang bergantung pada pemilihan parameter m, dimana m adalah himpunan dari 2 (m=2k). *Rice Code* merupakan spesial case dari Golomb Codes yang mana nilai x dikodekan k pertama digeser kekanan untuk mendapatkan nilai unary code. Kemudian urutan terendah dari k nilai asli dari x dilanjutkan sebagai k yang bernilai.

Langkah–langkah membuat codeword di dalam algoritma *Rice Code* adalah sebagai berikut :

a. Menentukan atau memisahkan bit penanda untuk menentukan bagian awal *Rice Code*.
b. Memisahkan k List Significant Bit (LSB) dan hasilnya merupakan bagian akhir dari *Rice Code*.
c. Kode sisa j = i/ 2¬k bit, sebagai j 0 diikuti oleh 1 atau j 1 diikuti oleh 0 dan hasilnya merupakan bagian tengah dari *Rice Code*.
d. Menggabungkan hasil dari proses langkah 1, 3 dan 2 untuk menjadi codeword *Rice Code*.

Langkah–langkah diatas merupakan proses pencari hasil untuk mendapatkan codeword algoritma *Rice Code* dimana nantinya nilai hexadecimal *file* PDF akan dikompres[7], [10]–[12].

Tabel 1. *Codeword* Algoritma *Rice Code*

i	*Binary*	*Sign*	LSB	*No. of Ones*	*Code*	*i*	*Code*
0	0	0	00	0	0|0|00		
1	1	0	01	0	0|0|01	-1	1|0|01
2	10	0	10	0	0|0|10	-2	1|0|10
3	11	0	11	0	0|0|11	-3	1|0|11
4	100	0	00	1	0|10|00	-4	1|10|00
5	101	0	01	1	0|10|01	-5	1|10|01
6	110	0	10	1	0|10|10	-6	1|10|10

2.2.2 Algoritma *Ternary Comma Code*

Ternary Comma Code adalah algoritma kompresi yang didasarkan pada tiga digit (trits) 0,1 dan 2. Setiap angka (trits) dikodekan dengan dua bit, tetapi dua bit bisa memiliki empat nilai. Sistem ternary menjadi tampak masuk akal saat setiap angka diwakili oleh dua bit dan di dalam sebuah penjumlahan menjadi tiga angka terdapat simbol keempat yaitu comma (c). Simbol comma memudahkan menyusun konsep sistem bilangan *Ternary Comma Code* untuk bilangan bulat.

Langkah–langkah membuat codeword di dalam algoritma *Ternary Comma Code* adalah sebagai berikut:

a. Lakukan pembagian pada bilangan bulat (n) dengan angka 3.
b. Tulis nilai hasil bagi bilangan bulat (n) untuk iterasi selanjutnya.
c. Tulis sisa hasil bagi untuk kode ternary.
d. Ulangi semua langkah di atas secara berurutan hingga sisa hasil bagi 0.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 34-42

Langkah–langkah diatas merupakan proses pencarian hasil untuk mendapatkan nilai codeword pada algoritma *Ternary Comma Code* dimana nanti hasil nilai algoritma *Rice Code* akan dikompres kembali[6], [11], [12] Berikut adalah codeword dari algoritma *Ternary Comma Code* :

Tabel 2. Codeword Algoritma *Ternary Comma Code*

No.	Nilai	Kode	L	No.	Nilai	L	Bit
1	1	0c	4	7	7	20c	6
2	2	1c	4	8	8	21c	6
3	3	2c	4	9	9	22c	6
4	4	10c	6	10	10	100c	8
5	5	11c	6	11	11	101c	8
6	6	12c	6	12	12	102c	8

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

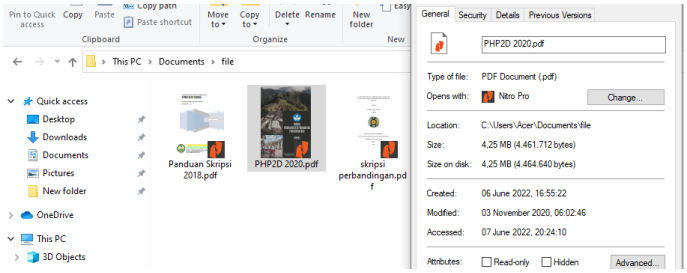
Dalam penelitian proses kompresi ini dilakukan analisa yang dilakukan dengan cara menghitung dan merancang aplikasi perangkat lunak. Dalam kompresi *file* yang berekstensi .pdf ini penulis menggunakan dua algoritma yaitu algoritma *Rice Code* dan *Ternary Comma Code*. Kedua algoritma yang digunakan adalah teknik algoritma yang berjenis lossless, yang artinya ukuran suatu data yang dikompresi berdasarkan pada karakter yang terdapat di dalam objek yang akan dikompres.

Sebelum melakukan proses kompresi dengan kedua algoritma terhadap *file* yang berekstensi .pdf, penulis harus mencari nilai hexadecimal *file* berekstensi .pdf yang akan dikompresi dengan menggunakan software HxD, setelah didapatkan nilai hexadecimal *file* yang bereksntensi.pdf. Maka selanjutnya dilakukan proses kompresi dan kemudian diperoleh *file* hasil terkompresi. Diharapkan dengan diterapkannya proses kompresi dengan algoritma *Rice Code* dan *Ternary Comma Code* didapati *file* terkompresi dengan ekstensi baru dengan ukuran yang lebih kecil.

Sama Seperti melakukan proses kompresi *file*, sebelum melakukan proses dekompresi *file* yang terkompresi, terlebih dahulu penulis harus mencari nilai hexadecimal *file* terdekompresi dengan menggunakan HxD Hexadecimal. Setelah mendapat nilai hexadecimal *file* terkompresi, maka prose dekompresi *file* dapat dilakukan. Jika proses kompresi diawali dengan algoritma *Rice Code* lalu dilanjutkan dengan algoritma *Ternary Comma Code*, maka pada proses dekompresi proses diawali dengan algoritma *Ternary Comma Code* lalu dilanjutkan dengan algoritma *Rice Code* sehingga dihasilkan *file* berekstensi PDF dengan ukuran semula seperti sebelum dilakukan proses kompresi.

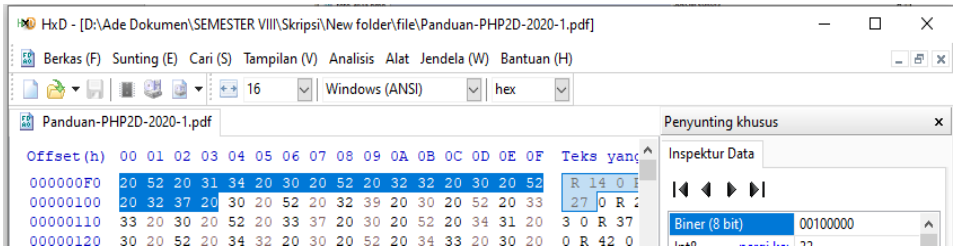
3.1.1 Penerapan Algoritma *Rice Code*

Berikut adalah contoh *file* PDF yang akan dikompresi dan didekompresi menggunakan algoritma *Rice Code* :



Gambar 1. Sampel *File* PDF

Berdasarkan gambar di atas, maka *file* PDF tersebut dapat diubah menjadi nilai hexadecimal dengan menggunakan software HxD. Berikut adalah nilai hexadecimal *file* PDF :



Gambar 2. Nilai Hexadecimal *File* PDF

Berikut nilai hexadecimal yang diurutkan berdasarkan frekuensi kemunculunnya, nilai yang frekuensinya paling banyak akan berada diurutan pertama.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 34-42

Tabel 3. Nilai Hexadecimal Yang Belum Dikompresi

N	*Hexadecimal*	Biner	Bit	Frekuensi	Bit x Frekuensi
1	20	0010 0000	8	9	72
2	32	0011 0010	8	3	24
3	30	0011 0000	8	2	16
4	52	0101 0010	8	2	16
5	25	0010 0101	8	1	8
6	31	0011 0001	8	1	8
7	34	0011 0100	8	1	8
8	37	0011 0111	8	1	8
		Total Bit			160

Setelah dilakukan pengurutan bilangan hexadecimal berdasarkan frekuensi kemunculannya dan didapatkan nilai biner, maka selanjutnya menghitung bit dan pengurutan kode dari algoritma *Rice Code* serta mendapatkan bit *file* yang terkompresi.

Tabel 4. Pengurutan Kode Algorita *Rice Code*

N	***Hexadecimal***	***Codeword***	**Bit**	**Freq**	**Bit x Freq**
1	20	0000	4	9	36
2	32	0001	4	3	12
3	30	0010	4	2	8
4	52	0011	4	2	8
5	25	01000	5	1	5
6	31	01001	5	1	5
7	34	01010	5	1	5
8	37	01011	5	1	5
		Total Bit			84

Dikarenakan 84 tidak habis dibagi 8 atau bukan kelipatan 8 maka dibentuk variabel yang disebut padding dan flagging untuk penambahan bit data. Padding tidak perlu dilakukan apabila jumlah bit *file* yang dikompresi habis dibagi 8 atau kelipatan 8, namun flagging tetap harus dilakukan.

Tabel 5. Proses Padding dan Flagging

Padding	***Flagging***
84 mod 8 = 4	n = 4
n = 4	9 - n
7 – n + “1”	9 – 4 = 5 = 00000101
7 – 4 + “1” = 0001	

Setelah dilakukan padding dan flagging maka tambahkan ke string bit sehingga menjadi : “00000100 00000010 01010110 00000100 00000110 00000010 00100000 01000000 01100000 00101011 00000001 00000101”. Sehingga panjang bit berubah menjadi 96 bit. Maka langkah selanjutnya adalah dengan mengubah nilai biner ke nilai hexadecimal agar mengahasilkan suatu karakter yang sesuai dengan kode ASCII. Nilai hexadecimal yang dihasilkan dapat dilihat pada tabel di bawah ini :

Tabel 6. Hasil Karakter Terkompresi

No.	***Codeword***	***Decimal***	***Hecadecimal***	**Karakter**
1	00000100	4	04	EOT (Tidak Terlihat)
2	00000010	2	02	STX (Tidak Terlihat)
3	01010110	86	56	V
4	00000100	4	04	EOT (Tidak Terlihat)
5	00000110	6	06	ACK (Tidak Terlihat)
6	00000010	2	02	STX (Tidak Terlihat)
7	00100000	32	20	(Spasi)
8	01000000	64	40	@
9	01100000	96	60	`
10	00101011	43	2B	+
11	00000001	1	01	SOH (Tidak Terlihat)
12	00000101	5	05	ENQ (Tidak Terlihat)

3.1.2 Penerapan Algoritma *Ternary Comma Code*

Setelah melakukan kompresi menggunakan algoritma *Rice Code*, ambil nilai hexadecimal dari hasil proses kompresi tersebut. Berikut adalah nilai hexadecimal yang diurutkan berdasarkan frekuensi kemunculannya, nilai yang frekuensinya paling banyak akan berada diurutan pertama.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 34-42

Tabel 7. Nilai Hex dari Proses Kompresi Algoritma *Rice Code*

N	*Hexadecimal*	Biner	Bit	Frekuensi	Bit x Frekuensi
1	04	0000 0100	8	2	16
2	02	0000 0010	8	2	16
3	56	0101 0110	8	1	8
4	06	0000 0110	8	1	8
5	20	0010 0000	8	1	8
6	40	0100 0000	8	1	8
7	60	0110 0000	8	1	8
8	2B	0010 1011	8	1	8
9	01	0000 0001	8	1	8
10	05	0000 0101	8	1	8
		Total Bit			96

Setelah dilakukan pengurutan bilangan hexadecimal berdasarkan frekuensi kemunculannya dan didapatkan nilai biner, maka selanjutnya menghitung bit dan pengurutan kode dari algoritma *Ternary Comma Code* serta mendapatkan bit *file* yang terkompresi.

Tabel 8. Pengurutan Codeword Algoritma *Ternary Comma Code*

N	*Hexadecimal*	*Codeword*	Bit	Frek.	Bit x Frek.
1	04	0c	1	2	2
2	02	1c	2	2	4
3	56	2c	2	1	2
4	06	10c	3	1	3
5	20	11c	3	1	3
6	40	12c	3	1	3
7	60	20c	3	1	3
8	2C	21c	4	1	4
9	01	22c	4	1	4
10	05	100c	4	1	4
		Total Bit			32

Namun, dalam sebuah perhitungan komputer tidak mengenal perhitungan dengan angka berbasis 3 (0,1 dan 2), maka untuk mencari nilai bit harus diubah kedalam pehitungan basis 2 (0 dan 1) dengan cara mengubah nilai n kedalam nilai biner. Setelah didapati nilai bit biner dari setiap data tambahkan nilai bit comma code (c). Nilai comma code (c) didapatkan dari jumlah n terakhir dari tabel . Diketahui nilai terakhir dari tabel adalah n = 10, maka bit dari comma code (c) adalah n = 11, sehingga nilai bit dari c adalah 1011 maka tahap selanjutnya adalah menggabungkan nilai bit dari setiap data dan menambahkan nilai bit dari comma code (c).

Tabel 9. Pengabungan Bit Biner dengan Bit Comma Code (c)

N	*Hexadecimal*	Biner	*Comma Code (c)*	Biner + *c*
1	04	1	1011	11011
2	02	10		101011
3	56	11		111011
4	06	100		1001011
5	20	101		1011011
6	40	110		1101011
7	60	111		1111011
8	2C	1000		10001011
9	01	1001		10011011
10	05	1010		10101011

Kemudian hitung hasil dari jumlah bit yang dikompresi yang bernilai 32 dan ditambah dari bit comma code. String bit dari comma code (c) adalah 4 kemudian dikali dengan jumlah data sebanyak 12, maka didapati string bit dari comma code (c) sebanyak 48. Jadi total keseluruhan bit adalah 32 + 48 = 80. Dikarenakan 80 habis dibagi 8, maka tidak perlu melakukan padding. Maka tambahkan “00000001” yang dinyatakan sebagai bit akhir. Selanjutnya dilakukan pemisahan setiap 8 bit, Hasil pemisahan dengan cara mengelompokkan tiap 8 bit bilangan terbentuk 11 kelompok dengan nilai bit yang telah terkompresi. Maka langkah selanjutnya adalah dengan mengubah nilai biner ke nilai hexadecimal agar mengahasilkan suatu karakter yang sesuai dengan kode ASCII.

Tabel 10. Hasil Karakter Terkompresi

No.	*Codeword*	*Decimal*	*Hecadecimal*	Karakter
1	11011101	221	DD	Ý
2	01111101	125	7D	}

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 34-42

No.	*Codeword*	*Decimal*	*Hecadecimal*	Karakter
3	11101110	238	EE	î
4	01011101	93	5D	]
5	01110110	118	76	v
6	11110101	245	F5	õ
7	11111011	251	FB	û
8	10001011	139	8B	‹
9	10011011	155	9B	š
10	10101011	171	AB	«
11	00000001	1	01	NUL (Tidak Terlihat)

Maka jika dihitung kembali nilai bit awal *file* belum terkompresi yang masih berformat .pdf dengan *file* terkompresi yang sudah berformat *.tcc, dihasilkan besaran kompresi sebagai berikut:

1. Ratio of Compression (RC)

$$RC = \frac{Ukuran\ Data\ Sebelum\ Dikompresi}{Ukuran\ Data\ Setelah\ Dikompresi}$$

$$RC = \frac{160}{32} = 5$$

2. Compression Ratio (CR)

$$CR = \frac{Ukuran\ Data\ Setelah\ Dikompresi}{Ukuran\ Data\ Sebelum\ Dikompresi}\ x\ 100\%$$

$$CR = \frac{32}{160}\ x\ 100\% = 20\%$$

3. Redudancy (Rd)

$$Rd = \frac{Ukuran\ sebelum\ dikompresi - \ Ukuran\ setelah\ dikompresi}{Ukuran\ data\ sebelum\ dikompresi} x100\%$$

$$Rd = \frac{160 - \ 32}{160}\ x\ 100\% = 80\%$$

4. Space Saving (SS)

$$SS = 100\% - CR$$

$$SS = 100\% - 20\% = 80\%$$

Jika ukuran awal *file* berformat .pdf yang bernama PHP2D 2020.pdf berukuran 4.461 KB jika dikalikan dengan nilai CR, maka dihasilkan ukuran akhir sebesar:

Ukuran akhir $= 4.461$ x $20\% = \mathbf{892,2\ KB}$

File yang awalnya berukuran 4.4 MB (atau 4.416 KB) dapat dikompresi ke ukuran 892,2 KB yang mana artinya besar nilai ruang memori yang dapat di hemat adalah 80%.

3.1.3 Proses Dekompresi Algoritma *Ternary Comma Code*

Hasil kompresi sebelumnya dapat dilihat pada tabel berikut ini:

Tabel 11. Nilai Biner dan Hasil Kompresi Algoritma TCC

No.	Karakter	*Codeword*	*Decimal*
1	Ý	11011101	221
2	}	01111101	125
3	î	11101110	238
4	]	01011101	93
5	v	01110110	118
6	õ	11110101	245
7	û	11111011	251
8	‹	10001011	139
9	š	10011011	155
10	«	10101011	171
11	SOH (Tidak Terlihat)	00000001	1

Pada proses pengurangan bit menjadi string bit semula dapat dilakukan dengan menghilangkan biner padding dan flagging. Tahap yang dilakukan untuk mengembalikan bit menjadi string bit adalah dengan cara membaca 8 bit terakhir dan kemudian nilai biner yang didapatkan diubah kedalam nilai decimal. Tetapkan hasil pembacaan dengan n, kemudian gunakan rumus 7 + n untuk mengembalikan string bit kedalam bentuk semula, maka :

“1101110101111101111011100101110101110111101011111101110001011100110111010101100000001”

8 bit terakhir = 00000001 = 1 = n

7 + 1 = 7 + 1 = 8

Didapati hasil dari penjumlahan adalah 8, maka maksudnya adalah hilangkan 8 bit dari string bit, sehingga menjadi :

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 34-42

Tabel 12. Pembacaan String Bit (c)

N	*Hexadecimal*	*Biner + c*
1	04	11011
2	02	101011
3	56	111011
4	06	1001011
5	20	1011011
6	40	1101011
7	60	1111011
8	2C	10001011
9	01	10011011
10	05	10101011

Setelah dilakukan pembacaan string bit bit comma code (c), yang dimana nilai bit biner comma code yang telah ditentukan adalah 1011 diubah dalam bit comma code (c), maka dihasilkan:

Tabel 13. Codeword Algoritma *Ternary Comma Code*

N	*Hexadecimal*	*Codeword*
1	04	0c
2	02	1c
3	56	2c
4	06	10c
5	20	11c
6	40	12c
7	60	20c
8	2C	21c
9	01	22c
10	05	100c

3.1.4 Proses Dekompresi Algoritma *Rice Code*

Sebelum melanjutkan proses dekompresi menggunkan algoritma *Rice Code*, pada tahap dekompresi menggunakan algoritma *Ternary Comma Code* didapatkan hasil codeword. Maka terlebih dahulu dilakukan pengubahan codeword pada algoritma *Ternary Comma Code* kedalam biner terkompresi pada algoritma *Rice Code*.

Tabel 14. Perubahan Codeword TCC ke Biner Terkompresi *Rice Code*

N	*Hexadecimal*	*Codeword* (Algoritma *TCC*)	*Biner* Terkompresi (Algoritma *Rice Code*)
1	04	0c	0000 0100
2	02	1c	0000 0010
3	56	2c	0101 0110
4	06	10c	0000 0110
5	20	11c	0010 0000
6	40	12c	0100 0000
7	60	20c	0110 0000
8	2C	21c	0010 1100
9	01	22c	0000 0001
10	05	100c	0000 0101

Lakukan analisa karakter yang tampil dan diubah kedalam bentuk biner dan decimal sehingga menghasilkan bit terkompresi kemudian ambil keseluruhan nilai biner lalu lakukan penghilangan biner padding dan flagging.
“ 00000100 00000010 01010110 00000100 00000110 00000010 00100000 01000000 01100000 00101011 00000001 00000101”
8 bit terakhir = 00000101 = 5 = n
7 + 5 = 7 + 5 = 12

Didapatkan hasil dari penjumlahan adalah 12, maka artinya adalah hilangkan 12 bit dari string bit, sehingga menjadi :
“000001000000001001010110000001000000011000000010001000000100000001100000001010110000”
Didapatkan string bit berjumlah 84. Dilakukan pembacaan string bit dari kiri ke kanan, hingga menemukan karakter yang sesuai dengan tabel codeword algoritma *Rice Code*.

Tabel 15. Codeword Algoritma *Rice Code*

N	*Hexadecimal*	*Codeword*
1	20	0000
2	32	0001

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 34-42

N	*Hexadecimal*	*Codeword*
3	30	0010
4	52	0011
5	25	01000
6	31	01001
7	34	01010
8	37	01011

Setelah dilakukan pembacaan ulang string bit dari kiri ke kanan dengan tabel codeword algoritma *Rice Code*. Maka didapati hasil yang sesuai dengan string bit awal yang terdapat pada *file* PDF. Berikut adalah hasil dekompresi string bit dalam bilangan hexadecimal "20 25 20 31 24 20 30 20 52 20 32 32 20 30 20 52 20 32 37 20 ".

4. KESIMPULAN

Berdasarkan dari pembahasan yang telah dijabarkan dan di evaluasi dalam penelitian yang telah dilakukan oleh peneliti sebelumnya yaitu implementasi algoritma *Rice Code* dan *Ternary Comma Code* pada kompresi *file* PDF, maka penulis mendapatkan hasil akhir dalam penelitian ini yang dapat disimpulkan menjadi beberapa kesimpulan ialah dengan mengikuti prosedur kompresi *file* PDF menggunakan algoritma *Rice Code* dan *Ternary Comma Code* yang diterapkan ke *file* PDF yang memiliki ukuran yang cukup besar dihasilkan *file* baru dengan ukuran yang lebih kecil. Bahkan saat dilakukan proses dekompresi tidak ada data *file* yang berkurang ataupun hilang akibat dari proses kompresi yang telah dilakukan.Penerapan kompresi berganda (double compression) untuk mengkompresi *file* PDF yang telah diterapkan membuktikan *file* PDF tersebut berhasil dikompresi, dengan hasil Ratio of Compression (RC) = 5, Compression Ratio (CR) = 20%, Redudancy (Rd) = 80%, dan Space Saving (SS) = 80%. *File* yang dikompresi menghasilkan ukuran yang lebih kecil yaitu sebesar 20% dari ukuran *file* asli sebelum dilakukannya double kompresi. Yaitu ukuran aslinya sebesar 4.461KB setelah dilakukan proses double kompresi menjadi 892,2KB. Aplikasi kompresi *file* PDF menggunakan algoritma *Rice Code* dan *Ternary Comma Code* yang dirancang menggunakan Microsoft Visual Studio 2010 dapat dijalankan dengan baik untuk mengkompresi *file* PDF.

REFERENCES

[1] S. F. Pane, M. Zamzam, and M. D. Fadillah, *Membangun Aplikasi Peminjaman Jurnal Menggunakan Aplikasi Oracle Apex Online*, Pertama. Bandung: Kreatif Industri Nusantara, 2020.
[2] F. E. Naibaho, "Implementasi Algoritma J-Bit Encoding Pada Kompresi File PDF," *J. Sist. Komput. dan Inform.*, vol. 1, no. 3, p. 265, 2020, doi: 10.30865/json.v1i3.2153.
[3] B. Ramadhana, "Implementasi Kombinasi Algoritma Fibonacci Codes Dan Levenstein Codes Untuk Kompresi File Pdf," vol. 8, no. 2, pp. 67–71, 2021.
[4] A. S. Laswi, "Pemanfaatan Algoritma Rice Codes dan Gold Bach Codes untuk Mengoptimalkan Media Penyimpanan pada Memori berupa File Text," *Jurnal INSTEK (Informatika Sains dan Teknologi)*, vol. 4, no. 2. pp. 191–200, 2019.
[5] M. A. Latif, S. D. Nasution, and Pristiwanto, "Analisa Perbandingan Algoritma Rice Codes Dengan Algoritma Goldbach Codes Pada Kompresi File Text Menggunakan Metode Exponential," *Maj. Ilm. INTI (Informasi dan Teknol. Ilmiah)*, vol. 13, no. 1, pp. 28–33, 2018.
[6] N. Ningsih, S. D. Nasution, and T. Zebua, "Penerapan Algoritma Ternary Comma Code Pada Aplikasi Kompresi File Gambar," *J. Comput. …*, vol. 1, no. 3, pp. 141–148, 2020.
[7] P. Fitria, "Penerapan Algoritma Rice Codes Pada Aplikasi Kompresi File Gambar," *J. Comput. Syst. Informatics*, vol. 1, no. 3, pp. 158–165, 2020.
[8] Irwansyah, S. D. Nasution, and Fadlina, "Perancangan Aplikasi Kompresi File Teks Dengan Menerapkan Algortima Rice Codes," *Pelita Inform. Inf. dan Inform.*, vol. 7, no. 2, pp. 219–222, 2018.
[9] M. Mawar, "Perancangan Aplikasi Kompresi File Pdf Dengan Menerapkan Algoritma Punctured Elias Codes," *Inf. dan Teknol. Ilm.*, vol. 7, no. 3, pp. 217–223, 2020.
[10] V. Ernanda, S. D. Nasution, and F. T. Waruwu, "Penerapan Algoritma Rice Codes dan Algoritma Rivest Shamir Adleman (RSA) Untuk Kompresi dan Pengamanan Teks Pada Aplikasi Chatting," *J. Comput. Syst. Informatics*, vol. 1, no. 3, pp. 92–98, 2020.
[11] D. Salomon and G. Motta, *Handbook of Data Compression 5th*, vol. 53, no. 9. 2019.
[12] K. Sayood, *Introduction to Data Compression*, Fifth. United States: Katey Birtcher, 2018.