KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)
Volume 6, Nomor 1, November 2022
DOI: 10.30865/komik.v6i1.5697
ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 448-459

Analisa Perbandingan Algoritma Yamamoto's Recursive Code Dan Algoritma Fibonacci Code Dalam Mengkompresi File Pptx

Rahadi Putra*, Abdul Sani Sembiring, Saidi Ramadan Siregar

Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budidarma, Medan, Indonesia
Email: [1]rahadiputra2102@gmail.com, [2]gurkiy@gmail.com, [3]saidiramadan89@gmail.com
Email Penulis Korespondensi : rahadiputra2102@gmail.com

Abstrak−Di dalam dunia kerja *file power point* atau pptx sangat diperlukan sebagai alat presentasi. Dengan menggunakan pptx untuk presentasi, kegiatan pemaparan bisa lebih mudah untuk dijelaskan. Namun masalah yang terjadi terkadang ukuran data *file* pptx ini cukup besar dari data-data lainnya mengakibatkan pemborosan penyimpanan. Maka untuk mengatasi masalah diperlukan teknik kompresi, kompresi adalah mengurangi jumlah bit menjadi lebih sedikit, Dengan demikian, jelas bahwa penggunaan kompresi bertujuan untuk mengurangi persentase penyimpanan. Penelitian ini dilakukan dengan maksud dan tujuan untuk menganalisa algoritma *yamamoto's recursive code* dan *fibonacci codes* dalam kompresi *file* pptx dengan parameter yaitu *Ratio of Compression* (RC), *Compression Ratio* (CR), dan juga *Space Saving* (SS). Setelah dikompresi kemudian membandingkan kinerja kedua algoritma tersebut dengan menggunakan metode *exponential* berdasarkan nilai parameter yang telah ditetapkan. Dari hasil penelitian maka di dapat hasil nilai untuk algoritma *yamamoto's recursive code* dengan nilai RC adalah 1,26, nilai CR adalah 79,17%, dan nilai SS adalah 20,83%. Untuk hasil nilai algoritma *fibonacci code* dengan nilai RC adalah 1,71, nilai CR adalah 58,33%, nilai dan nilai SS adalah 41,66%. Hasil perbandingan algoritma *yamammoto's recursive code* dan *fibonacci code* didapatkan algoritma yang lebih optimal dalam mengkompresi *file* pptx adalah algoritma *fiboancci code* dengan perolehan nilai space saving 41,66%.

Kata Kunci: Kompresi; algoritma; Yamamoto's Recursive Code; Fibonacci Code; Pptx.

Abstract−In the world of work, power point or pptx files are indispensable as presentation tools. By using pptx for presentations, presentation activities can be easier to explain. However, the problem that occurs is that sometimes the data size of this pptx file is quite large compared to other data, resulting in a waste of storage. So to solve the problem required compression technique, compression is to reduce the number of bits to be fewer, Thus, it is clear that the use of compression aims to reduce the percentage of storage. This research was conducted with the aim and purpose of analyzing the Yamamoto's recursive code algorithm and Fibonacci codes in pptx file compression with parameters, namely Ratio of Compression (RC), Compression Ratio (CR), and also Space Saving (SS). After being compressed then compare the performance of the two algorithms using the exponential method based on the parameter values that have been set. From the results of the study, the results obtained for the Yamamoto's recursive code algorithm with the RC value of 1.26, the CR value of 79.17%, and the SS value of 20.83%. For the results of the Fibonacci code algorithm, the RC value is 1.71, the CR value is 58.33%, the SS value and value is 41.66%. The results of the comparison of Yamammoto's recursive code algorithm and Fibonacci code obtained that the more optimal algorithm in compressing pptx files is the Fiboancci code algorithm with the acquisition of space saving values of 41.66%.

Keywords: Compression; Algorithm; Yamammoto's Recursive Code; Fibonacci; pptx

1. PENDAHULUAN

Di era digital saat ini kebutuhan *file power point* atau pptx dalam dunia kerja sangat diperlukan sebagai alat presentasi atau pengolah data. Dengan menggunakan pptx untuk presentasi, kegiatan pemaparan bisa lebih teratur dan terstruktur sesuai dengan yang telah direncanakan. Namun terkadang ukuran data file pptx ini cukup besar dari data-data lainnya yang mengakibatkan pemborosan penyimpanan, maka dari pada itu diperlukan suatu teknik kompresi.

Kompresi adalah mengubah data menjadi bentuk yang membutuhkan bit yang lebih sedikit atau mengurangi jumlah bit pada data, biasanya dilakukan agar data disimpan atau ditransmisikan lebih efisien. Dengan demikian, jelas bahwa penggunaan kompresi bertujuan untuk mengurangi persentase penyimpanan pada media penyimpanan. Kompresi berguna karena dapat membantu mengurangi konsumsi sumber daya yang mahal seperti hard disk.

File Power Point atau *file* yang berextensi pptx ini digunakan untuk menyimpan presentasi *slide show* dengan format pptx. pptx adalah format *file* presentasi default untuk *power point* 2007 dan seterusnya. *File* pptx dapat berisi teks berformat, objek, banyak *slide*, gambar, video, dan lainnya. Banyak orang yang menggunakan *file* yang berekstensi pptx ini, dan biasa digunakan untuk membuat bahan persentasi sehingga suatu presentasi menjadi lebih mudah tersampaikan dengan model *slide* yang ada di dalam *microsoft power point* dan materi tersampaikan dengan gampang selama proses presentasi. Jika terjadi kerangkapan file pptx yang ada ataupun karena sudah terlalu banyak file pptx sehingga ruang penyimpanan menjadi lebih sedikit, itu terjadi karena dalam suatu media penyimpanan tidak hanya file pptx saja yang ada tetapi banyak file-file lainnya yang berbagi ruang penyimpanan.

Permasalahan yang terjadi adalah karena ukuran data *file* pptx cenderung lebih besar dibandingkan dengan data lainnya karna menggunakan banyak slide maka mengakibatkan pemborosan memori penyimpanan dan memperlambat proses transmisi data. Berdasarkan hal tersebut, untuk mengatasi masalah maka dibutuhkan suatu teknik pengecilan data atau biasa disebut dengan teknik kompresi. Kompresi bertujuan untuk mengurangi ukuran data menjadi lebih kecil dan parameter hasil kompresi berupa *ratio of compression, compression ratio* dan *space saving* [1].

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 448-459

Pada penelitian sebelumnya yang dilakukan oleh Septriani Br Ginting dengan judul "Perbandingan Algoritma *Yamamoto's Recursive Code* Dan *Additive Code* Dalam Kompresi *File* Video" dapat disimpulkan bahwa algoritma *yamamoto's recursive code* dan *additive code* dapat diimplementasikan pada proses kompresi *file* video berekstensi .mp4 dari hasil pengujian maka didapat bahwa *algoritma additive code* lebih baik dari pada *yamamoto's recursive code* dalam melakukan kompresi *file* video dan nilai rasio kompresi dipengaruhi oleh isi *file* terkompresi. Semakin banyak karakter yang diulang dalam file terkompresi, semakin baik rasio kompresi [2].

Pada penelitian yang dilakukan oleh Nur Aftikasyah dengan judul "Penerapan Algoritma *Yamamoto's Recursive Code* Untuk Mengkompresi *File* Dokumen" bahwa prosedur kompresi file dokumen dengan menggunakan algoritma *yamamoto's recursive code* telah berhasil dilakukan dengan *file* dokumen yang berformat *.rtf dan proses kompresi berajalan dengan sesuai dengan teknik kompresi dan *file* dokumen yang memiliki ukuran besar dapat dikompres menjadi ukuran yang lebih kecil [3].

Pada penelitian yang dilakukan oleh Bobby Ramadhana dengan judul "Implementasi Kombinasi Algoritma *Fibonacci Code* Dan *Levenstein Codes* Untuk Kompresi *File* Pdf" bahwa pada proses kompresi *file* pdf berhasil dilakukan menggunakan dua algoritma yaitu *fibonacci code* dan *levenstein codes*, dengan hasil kompresi mengalami perubahan ukuran yang signifikan. Rata2 *space saving* yang didapatkan lebih dari 50% dan Waktu yang dibutuhkan untuk proses kompresi *file* pdf bervariasi. Hal ini dipengaruhi oleh panjangnya karakter file yang diinputkan [4].

Penelitian lainnya yang dilakukan oleh Jesica Martina Br. Panjaitan dengan judul "Penerapan Algoritma *Fibonacci Code* Pada Kompresi Aplikasi Audio Mp3 Berbasis Dekstop" Algoritma *Fibonacci Code* dapat mengurangi jumlah data yang dibutuhkan untuk menyimpan *file* MP3 lebih dari 30%. Mudah dipahami, dan menggunakan aplikasi Microsoft Visual Basic 2008, berhasil diterapkan pada kompresi file MP3 dan cukup menyesuaikan sesuai dengan rumus yang berlaku di algoritma *fibonacci code* untuk kompresi *file* audio mp3 [5].

Pada penelitian juga yang dilakukan oleh M. Ade Syahputra dan Putri Ramadhani dengan judul "Implementasi Metode *Fibonacci* Dalam Kompresi Iklan Pada Website PT. Amanah Wisata Travel" Berdasarkan penelitian yang telah dilakukan, tampaknya metode Fibonacci dapat digunakan untuk mengkompresi iklan pada website PT. Amanah Wisata Travel [6].

2. METODOLOGI PENELITIAN

2.1 Kompresi

Kompresi adalah mengecilkan atau memampatkan ukuran sehingga diperoleh *file* dengan ukuran lebih kecil dari pada aslinya. Teknik kompresi bisa dilakukan terhadap teks/biner, *file* dokumen, gambar, audio, video dan *file* lainnya. Dengan adanya kompresi data, maka data yang memiliki kapasitas besar ukurannya akan dikurangi sehingga dapat menghemat media penyimpanan [7].

Teknik yang digunakan dalam proses kompresi data memiliki beberapa parameter yang umum digunakan untuk menganalisis kualitas teknik kompresi data.

1. *Ratio Of Compression* (RC)
   Ratio of Compression (Rc) adalah nilai perbandingan antara ukuran bit data sebelum dikompresi dengan ukuran bit data yang telah dikompresi.
   $$Rc = \frac{ukuran\ bit\ sebelum\ dikompresi}{ukuran\ bit\ setelah\ dikompresi} \quad (1)$$
2. *Compression Ratio* (CR)
   Compression Ratio (CR) adalah peresentase perbandingan antara yang sudah dikompresi dengan data yang belum dikompresi.
   $$CR = \frac{ukuran\ bit\ setelah\ dikompresi}{ukuran\ bit\ sebelum\ dikompresi} \text{X } 100\% \quad (2)$$
3. *Spase Saving*
   *Space Saving (*SS) adalah selisih antara data yang belum dikompresi dengan besar data yang dikompresi.
   SS = 100% - CR (3)

2.2 Algoritma Yamamoto's Recursive Code

Yamamoto's recursive code merupakan kode recursif untuk angka positif yang dimana setiap urutan yang diberikan dapat digunakan sebagai delimiter, delimiter yang dipakai pada algoritma *yamamoto's recursive code* lebih pendek dari Log*$_2$ hampir dari semua angka bulat positif dibandingkan algoritma sebelumnya [3].

Berikut *codeword* dari algoritma *yamamoto's recursive code* yang digunakan untuk memperoleh sebuah tabel kebenaran.

Tabel 1. Yamamoto's Recursive Codewords

N	a= 100	a= 100
1	1 00	0 100
2	1 01 00	0 00 100
3	1 10 00	0 01 100

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 448-459

N	a= 100	a= 100
4	1 11 00	0 11 100
5	1 01 010 00	0 00 000 100
6	1 01 011 00	0 00 001 100
7	1 01 100 00	0 00 010 100
8	1 01 101 00	0 00 011 100
9	1 01 110 00	0 00 101 100
10	1 01 111 00	0 00 110 100
11	1 10 0100 00	0 00 111 100
12	1 10 0101 00	0 01 0000 100
13	1 10 0110 00	0 01 0001 100
14	1 10 0111 00	0 01 0010 100
15	1 10 1000 00	0 01 0011 100
16	1 10 1001 00	0 01 0100 100
17	1 10 1010 00	0 01 0101 100
18	1 10 1011 00	0 01 0110 100
19	1 10 1100 00	0 01 0111 100
20	1 10 1101 00	0 01 1010 100

2.3 Algoritma Fibonacci Code

Bilangan *fibonacci* pertama kali dikemukakan oleh seorang ilmuan yang berasal dari Italia yang bernama Leonardo da Pisa atau dikenal sebagai Leonardo Pisano yang lahir pada tahun 1175 sampai 1250. Leonardo ialah seseorang yang memperkenalkan bilangan deret [5].

Kode fibonacci adalah *variabel length code*, dimana bilangan bulat yang lebih kecil mendapatkan kode pendek. Kode berakhir dengan dua buah bit 1, dan nilai yang didapatkan jumlah dari nilai-nilai fibonacci yang sesuai untuk bit yang ditetapkan [8].

Barisan ini berawal dari 0 dan 1, kemudian angka berikutnya didapat dengan cara menambahkan kedua bilangan yang berurutan sebelumnya, dengan aturan ini, maka barisan bilangan yang pertama adalah :0,1,1,2,3,5,8,13,21,34,55,89,144….. berikut ini Langkah-langkah pembentukan kode fibonacci :

1. Tentukan sebuah bilangan bulat positif n.
2. Temukan bilngan fibonacci f terbesar yang lebih kecil atau sama dengan n, kurangkan nilai dengan f dan catat sisa pengurangan nilai n dengan f.
3. Jika bilngan yang terdapat dalam deret fibonacci f(i), tambahkan angka dalam kode fibonacci yang akan di bentuk.
4. Ulangi langkah 2 tukar nilai n dengan sisa pengurangan nilai n dengan f sampai sisa pengurangan nilai n dengan f adalah 0.
5. Tambahkan angka “1” pada posisi paling kanan kode fibonacci yang akan dibentuk.

Tabel 2. Fibonacci Codeword.

N	*Codeword*
1	11
2	011
3	0011
4	1011
5	00011
6	10011
7	01011
8	01011
9	000011
10	010011
11	001011
12	101011

2.3 Metode Exponential

Metode Perbandingan *Eksponential* (MPE) merupakan salah satu metode untuk menentukan urutan prioritas alternatif keputusan dengan kriteria jamak [9].

Dalam menggunakan Metode Perbandingan Eksponensial ada beberapa tahap yang harus dilakukan, yaitu:

1. Menyusun alternatif-alternatif keputusan yang akan dipilih.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 448-459

2. Menentukan kriteria atau perbandingan keputusan yang penting untuk dievaluasi.
3. Menentukan tingkat kepentingan dari setiap kriteria keputusan.
4. Melakukan penilaian terhadap semua alternatif pada setiap kriteria dengan menggunakan formula pada parameter perbandingan.
5. Menghitung skor atau nilai total setiap alternative.
6. Menentukan urutan prioritas keputusan didasarkan pada skor atau nilai total masing-masing alternatif.

2.4 File Power Point (pptx)

File Power Point (pptx) adalah aplikasi perangkat lunak perancang presentasi, dengan tampilan ruang kerja untuk memanipulasi slide presentasi. Aplikasi ini sangat banyak digunakan terutama oleh kalangan perkantoran, pebisnis, pendidik, pelajar dan trainer. Dimulai dengan Microsoft Office System versi 2003, Microsoft mengubah namanya dari Microsoft PowerPoint sebelumnya menjadi Microsoft Office PowerPoint. Di Office 2013 namanya disingkat menjadi PowerPoint [10]. *power point* pada versi 2003 masih menggunakan format *file* ppt sedangkan untuk versi 2007,2010 dan 2013 sudah menggunakan format *file* pptx.

3. HASIL DAN PEMBAHASAN

3.1 Analisa

Analisa permasalahan yang dibahas dalam peneilitian ini adalah dengan menggunakan algoritma *yamamoto's recursive code* dan *algoritma fibonacci* dalam melakukan kompresi *file* pptx. Perbandingan dari kinerja kedua algoritma diatas didasari dengan beberapa parameter, yaitu *Ratio of Compression* (RC), *Compression Ratio* (CR), dan *Space Saving* (SS) [11].

Langkah awal yang akan dilakukan adalah menetukan *file* pptx yang akan dikompresi yang kemudian mengkonversikan *file* tersebut ke nilai *hexadesimal* menggunakan *software* HxD. Kemudian lakukan kompresi dengan menggunakan algoritma *yamamoto's recursive code* dan algoritma *fibonacci code*, maka mendapatkan hasil kompresi dari *file* tersebut. Setelah didapatkan hasil kompresi, dilanjutkan dengan proses dekompresi dari algoritma *yamamoto's recursive code* dan algoritma *fibonacci code*. Dan dari hasil kompresi yang telah didapatkan, maka dilakukanlah proses perbandingan dari *file* pptx tersebut dengan menerapkan metode *exponential* berdasarkan parameter yang telah ditentukan.

3.1.1 Penerapan Algoritma Yamamoto's Recursive Code

Pada penelitian ini akan membahas 2 proses utama pengkompresian *file* pptx, yaitu proses kompresi dan proses dekompresi. Penulis akan melakukan kompresi *file* pptx menggunakan algoritma *yamamoto's recursive code* yang merupakan salah satu teknik kompresi *lossless*.

Pada tahap ini, *file* pptx diubah menjadi nilai *hexadecimal* dengan menggunakan aplikasi *HxD*, sehingga dapat diambil 24 sampel dari nilai bilangan *hexadecimal* tersebut, seperti pada tabel berikut:

Tabel 3. Nilai hexadesimal sampel data *file* pptx.

50	4B	03	04	14	00	06	00	08	00	00	00
EA	55	F9	01	00	00	6B	0D	00	00	13	00

Dari tabel diatas, didapat nilai *hexadecimal* yang akan dihitung secara manual. Berikut nilai *hexadecimal* diurutkan sesuai frekuensinya nilai frekuensi yang paling banyak yang akan berada dalam urutan pertama.

Tabel 4. Nilai Hexadesimal Yang Belum Di Kompresi.

N	Nilai Hexadesimal	Nilai Biner	Bit	Frekuensi	Bit X Prekuensi
1	00	00000000	8	10	80
2	50	01010000	8	1	8
3	4B	01001011	8	1	8
4	03	00000011	8	1	8
5	04	00000100	8	1	8
6	14	00010100	8	1	8
7	06	00000110	8	1	8
8	08	00001000	8	1	8
9	EA	11101010	8	1	8
10	55	01010101	8	1	8
11	F9	11111001	8	1	8
12	01	00000001	8	1	8
13	6B	01101011	8	1	8

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 448-459

N	Nilai Hexadesimal	Nilai Biner	Bit	Frekuensi	Bit X Prekuensi
14	0D	00001101	8	1	8
15	13	00001101	8	1	8
		Total			192

Setelah bilangan *hexadecimal* diurutkan sesuai frekuensi kemunculan serta didapat nilai biner, selanjutnya menghitung bit dan pengurutan kode *yamamoto's recursive code* dan memperoleh bit *file* terkompresi. Adapun proses kompresi *file* pptx dapat dilihat pada tabel berikut :

Tabel 5. Nilai *Hexdesimal* Yang Terkompresi Dengan *Yamamoto's.*

N	Hexadesimal	Frekuensi	Codeword	Bit	Bit x Frekuensi
1	00	10	100	3	30
2	50	1	10100	5	5
3	4B	1	11000	5	5
4	03	1	11100	5	5
5	04	1	10101000	8	8
6	14	1	10101100	8	8
7	06	1	10110000	8	8
8	08	1	10110100	8	8
9	EA	1	10111000	8	8
10	55	1	10111100	8	8
11	F9	1	110010000	9	9
12	01	1	110010100	9	9
13	6B	1	110011000	9	9
14	0D	1	110011100	9	9
15	13	1	110100000	9	9
		Total			138

Tahap selanjutnya, menyusun kembali nilai hexadecimal sebelum dikompresi yaitu "50, 4B, 03, 04, 14, 00, 06, 00, 08, 00, 00, 00, EA, 55, F9, 01, 00, 00, 6B, 0D, 00, 00, 13, 00" seperti yang dilihat pada tabel di bawah ini.

Tabel 6. String Bit Hasil Kompresi dengan *yamamoto's*.

50	4B	03	04
10100	11000	11100	10101000
14	00	06	00
10101100	100	10110000	100
08	00	00	00
10110100	100	100	100
EA	55	F9	01
10111000	10111100	110010000	110010100
00	6B	0D	00
100	110011000	110011100	100
00	00	13	00
100	100	110100000	100

Berdasarkan pada tabel 6 diatas yang menghasilkan *string* bit yang dihasilkan dari proses kompresi *file* pptx menggunakan algoritma *yamamoto's recursive code* yang dituliskan sebagai berikut:

Tabel 7. *String* Bit Hasil Kompresi.

10100	11000	11100	10101000
10101100	100	10110000	100
10110100	100	100	100
10111000	10111100	110010000	110010100
100	110011000	110011100	100
100	100	110100000	100

Kemudian dari total bit x frekuensi dihasilkan 138 bit, maka perlu dilakukannya penambahan jumlah bit karena 138 tidak habis dibagi 8. Selanjutnya dilakukan penambahan string bit yaitu *padding* dan *flagging* karena 138 tidak habis habis di bagi 8 maka wajib menambahkan *padding* dan *flagging* dan kemudian akan dibentuk variabel untuk penambahan bit data. Rumus *padding* yaitu 7 – n + "1" dan rumus *flagging* yaitu 9 – n.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 448-459

Tabel 8. Perhitungan Penambahan Bit.

Padding	*Flagging*
138 Mod 8 = 2 n = 2 7 – 2 + "1" = 5 + "1" = 000001	9 – 2 9 – 2= 7 = 00000111 (biner)

Dari penambahan *flagging* maka diperolah string bit baru yaitu : "10100 11000 11100 10101000 10101100 100 10110000 100 10110100 100 100 100 10111000 10111100 110010000 110010100 100 110011000 110011100 100 100 100 110100000 100 **000001 00000111**"

Maka total bit yang diperoleh adalah 152 bit. Selanjutnya lakukan pemisahan bit menjadi beberapa kelompok, setiap kelompok terdiri 8 bit seperti berikut pada tabel di bawah ini.

Tabel 9. Pengelompokan Bit.

10100110	00111001	01010001	01011001	00101100
00100101	10100100	10010010	11100010	11110011
00100001	10010100	10011001	10001100	11100100
10010011	01000001	00000001	00000111	

Berdasarkan pengelompokan bit di atas, seiring dengan penambahan nilai biner, diperoleh 19 kelompok nilai biner terkompresi baru. Setelah pengelompokan, langkah selanjutnya adalah mengubah nilai biner ke heksadesimal dan menemukan karakter yang sesuai dengan kode ASCII. Adapun nilai hexadecimal sudah dikompresi dapat dilihat pada tabel berikut:

Tabel 10. Nilai Hasil Terkompresi.

N	Biner	Desimal	Hexadesimal	Karakter
1	10100110	166	A6	¦
2	00111001	57	39	9
3	01010001	81	51	Q
4	01011001	89	59	Y
5	00101100	44	2C	,
6	00100101	37	25	%
7	10100100	164	A4	¤
8	10010010	146	92	’
9	11100010	226	E2	Â
10	11110011	243	F3	Ó
11	00100001	33	21	!
12	10010100	148	94	”
13	10011001	153	99	™
14	10001100	140	8C	Œ
15	11100100	228	E4	Ä
16	10010011	147	93	“
17	01000001	65	41	A
18	00000001	1	1	SOH
19	00000111	7	7	BEL

Setelah diketahui hasil kompresi, maka dapat dihitung kinerja algoritma *yamamoto's recursive code* yaitu:

1. Ratio Of Compression (RC)
   $Rc = \frac{192}{152}$
   $Rc = 1{,}26$
2. *Compression Ratio* (CR)
   $CR = \frac{152}{192}\ X\ 100\%$
   $CR = 79{,}17\%$
3. *Space saving* (SS)
   $SS = 100 - \frac{152}{192}\ X\ 100\%$
   $SS = 20{,}83\%$

Berdasarkan perhitungan diatas, maka dapat disimpulkan bahwa persentase hasil kompresi *file* pptx menggunakan algoritma *yamamoto's recursive code* adalah sebesar 20,83%

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 448-459

3.1.2 Dekompresi Algoritma Yamamoto's Recursive Code

Proses dekompresi dilakukan dengan mengubah karakter menjadi nilai *biner* dan *decimal* yang akan menghasilkan bit semula, dapat dilihat seperti tabel berikut:

Tabel 11. Nilai Desimal Terkompresi Yamamoto's.

N	Karakter	Biner	Decimal
1	¦	10100110	166
2	9	00111001	57
3	Q	01010001	81
4	Y	01011001	89
5	,	00101100	44
6	%	00100101	37
7	¤	10100100	164
8	’	10010010	146
9	Â	11100010	226
10	Ó	11110011	243
11	!	00100001	33
12	”	10010100	148
13	™	10011001	153
14	Œ	10001100	140
15	Ä	11100100	228
16	“	10010011	147
17	A	01000001	65
18	SOH	00000001	1
19	BEL	00000111	7

Selanjutnya ialah pengurangan bit menjadi string bit semula, pengurangan bit dilakukan dengan menghilangkan biner *flagging*. Untuk mengembalikan bit menjadi string bit dengan membaca bit terakhir dan kemudian mengubah nilai biner menjadi decimal nyatakan hasil pembacaan dengan n, selanjutnya gunakan rumus 7 + n untuk mengembalikan string bit ke bentuk semula.

8 bit terakhir = 00000111 = 7 = n

7 + n = 7 + 7 = 14

Hilangkan dari string bit sebanyak 14 bit terakhir, menjadi:

“10100110001110010101000101011001001011000010010110100100100100101110001011110011001000011001010010011001100011001110010010010011010000010”

Berdasarkan perhitungan diatas, string bit berjumlah 138 seperti semula, sehingga dapat dilakukan pembacaan string bit awal. Adapun tabel perhitungannya sebagai berikut :

Tabel 12. Pengecekan Bit Dekompresi *yamamoto's.*

N	Biner	Hexadesimal
1	10100	50
2	11000	4B
3	11100	03
4	10101000	04
5	10101100	14
6	100	00
7	10110000	06
8	100	00
9	10110100	08
10	100	00
11	100	00
12	100	00
13	10111000	EA
14	10111100	55
15	110010000	F9
16	110010100	01
17	100	00
18	110011000	6B
19	110011100	0D
20	100	00
21	100	00
22	100	00

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 448-459

N	Biner	Hexadesimal
23	110100000	13
24	100	00

Dari tabel di atas dapat disimpulkan hasil dekompresi yang dilakukan oleh algoritma *yamamoto's recursive code* yang dilakukan dengan mengubah hasil semua nilai *hexadecimal* menjadi string bit semula yang terdapat pada *file* awal, maka didapat hasil dekompresi yang sesuai dengan string bit semula yaitu dengan bilangan hexa "50, 4B, 03, 04, 14, 00, 06, 00, 08, 00, 00, 00, EA, 55, F9, 01, 00, 6B, 0D, 00, 00, 00, 13,00."

3.1.3 Penerapan Algoritma Fibonacci Code

Berdasarkan sampel data nilai *hexadecimal* sudah terlihat pada tabel 4.1, selanjunya disusun berdasarkan frekuensi terbesar dan ditentukan banyaknya jumlah bit pada sampel karakter tersebut. Adapun prosesnya dapat dilihat pada tabel 4.10.

Tabel 13. Nilai Hexadesimal Yang Belum Dikompresi.

N	Nilai Hexadesimal	Nilai Biner	Bit	Frekuensi	Bit X Prekuensi
1	00	00000000	8	10	80
2	50	01010000	8	1	8
3	4B	01001011	8	1	8
4	03	00000011	8	1	8
5	04	00000100	8	1	8
6	14	00010100	8	1	8
7	06	00000110	8	1	8
8	08	00001000	8	1	8
9	EA	11101010	8	1	8
10	55	01010101	8	1	8
11	F9	11111001	8	1	8
12	01	00000001	8	1	8
13	6B	01101011	8	1	8
14	0D	00001101	8	1	8
15	13	00001101	8	1	8
		Total			192

Seperti yang terlihat pada tabel 4.10, satu karakter terdiri dari 8 bit. Sehingga dengan jumlah karakter sebanyak 24 karakter dikalikan dengan 8 bit, maka total keseluruhan ukuran karakter adalah 192 bit.
Langkah-langkah pembentukan sebuah kode *fibonacci* untuk nilai sampel atau n nya adalah 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14. Deret *fibonacci* yang mendekati nilai n adalah : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,….. Proses pengurangan nilai n dengan setiap bilangan *fibonacci* sampai sisa pengurangan nilai n adalah:
$nilai\ n = 14$ Proses pengurangan nilai n dengan setiap bilangan *fibonacci*
$(1)\ 14 - 13 = 1$
$(2)\ 1\ -\ 1 = 0$

Tabel 14. Pembentukan *codeword.*

Urutan Bilangan *fibonacci*	F(1)	F(2)	F(3)	F(4)	F(5)	F(6)	F(7)	F(8)
Bilangan *Fibonacci*	0	1	1	2	3	5	8	13
Kode *fibonacci* sementara	-	-	1	0	0	0	0	1

Proses akhir pembentukan *codeword* adalah menambahkan angka "1" pada posisi paling kanan dari *codeword* untuk angka 14 adalah "1000011". Selanjutnya dilakukan kompresi file pptx yang sudah disusun dengan mengalikan banyaknya frekuensi karakter yang muncul dengan banyaknya nilai bit dari kode nilai algoritma *fibonacci code*. Adapun proses keseluruhanya dapat dilihat pada tabel 15.

Tabel 15. Nilai *Hexdesimal* Yang Terkompresi Dengan *Fibonacci.*

N	Hexadesimal	Frekuensi	Codeword	Bit	Bit x Frekuensi
1	00	10	11	2	20
2	50	1	011	3	3
3	4B	1	0011	4	4
4	03	1	1011	4	4
5	04	1	00011	5	5

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 448-459

6	14	1	10011	5	5
7	06	1	01011	5	5
8	08	1	000011	6	6
9	EA	1	100011	6	6
10	55	1	010011	6	6
11	F9	1	001011	6	6
12	01	1	101011	6	6
13	6B	1	0000011	7	7
14	0D	1	1000011	7	7
15	13	1	0100011	7	7
11	F9	1	001011	6	6
		Total			97

Tahap selanjutnya, menyusun kembali nilai *hexadecimal* sebelum dikompresi yaitu **"**50, 4B, 03, 04, 14, 00, 06, 00, 08, 00, 00, 00, EA, 55, F9, 01, 00, 00, 6B, 0D, 00, 00, 13, 00**"** seperti yang dilihat pada tabel di bawah ini.

Tabel 16. String Bit Hasil Kompresi dengan *fibonacci.*

50	**4B**	**03**	**04**	**14**	**00**
011	0011	1011	00011	10011	11
06	**00**	**08**	**00**	**00**	00
01011	11	000011	11	11	11
EA	**55**	**F9**	**01**	**00**	**00**
100011	010011	001011	101011	11	11
6B	**0D**	**00**	**00**	**13**	**00**
0000011	1000011	11	11	0100011	11

Berdasarkan pada tabel 16 diatas yang menghasilkan *string* bit yang dihasilkan dari proses kompresi *file* pptx menggunakan algoritma *yamamoto's recursive code* yang dituliskan sebagai berikut:

Tabel 17. *String* Bit Hasil Kompresi.

011	0011	1011	00011	10011	11
01011	11	000011	11	11	11
100011	010011	001011	101011	11	11
0000011	1000011	11	11	0100011	11

Selanjutnya dilakukan penambahan string bit yaitu *padding* dan *flagging* dengan mengacu pada sisa jumlah bit dibagi 8. Jumlah dari hasil string bit yaitu 97 tidak habis dibagi 8 kemudian akan dibentuk variabel untuk penambahan bit data. Rumus *padding* yaitu 7 – n + "1" dan rumus *flagging* yaitu 9 – n

Tabel 18. Perhitungan Penambahan Bit.

Padding	*Flagging*
97 Mod 8 = 1	9 – 1
n = 1	9 – 1= 8 = 00001000 (biner)
7 – 1 + "1" = 6 + "1" = 0000001	

Dari penambahan *flagging* maka diperolah string bit baru yaitu:
011 0011 1011 00011 10011 11 01011 11 000011 11 11 11 100011 010011 001011 101011 11 11 0000011 1000011 11 11 0100011 11 **0000001 00001000**, Maka total bit yang diperoleh adalah 112 bit. Selanjutnya lakukan pemisahan bit menjadi beberapa kelompok, setiap kelompok terdiri 8 bit seperti berikut pada tabel di bawah ini.

Tabel 19. Pengelompokan Bit

01100111	01100011	10011110	10111100	00111111
11100011	01001100	10111010	11111100	00011100
00111111	01000111	10000001	00001000	

Berdasarkan pada pengelompokan bit diatas maka didapat 14 kelompok dengan nilai biner baru yang sudah dikompresi beserta penambahan nilai biner. Setelah pengelompokan maka selanjutnya ialah mengubah nilai biner ke nilai *hexadecimal*, untuk mengetahui suatu karakter yang sesuai dengan kode ASCII. Adapun nilai hexadecimal sudah dikompresi dapat dilihat pada tabel berikut:

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 448-459

Tabel 20. Nilai Hasil Terkompresi

N	Biner	Desimal	Hexadesimal	Karakter
1	01100111	103	67	G
2	01100011	99	63	C
3	10011110	158	9E	Ž
4	10111100	188	BC	¼
5	00111111	63	3F	?
6	11100011	227	E3	Ã
7	01001100	76	4C	L
8	10111010	186	BA	º
9	11111100	252	FC	Ü
10	00011100	28	1C	FS
11	00111111	63	3F	?
12	01000111	71	47	G
13	10000001	129	81	•
14	00001000	8	8	BS

Setelah diketahui hasil kompresi, maka dapat dihitung kinerja algoritma *fibonacci code* yaitu:

1. Ratio Of Compression (RC)
   $Rc = \frac{192}{112}$
   $Rc = 1{,}71$
2. *Compression Ratio* (CR)
   $CR = \frac{112}{192}\ X\ 100\%$
   $CR = 58{,}33\%$
3. *Space saving* (SS)
   $SS = 100 - \frac{112}{192}\ X\ 100\%$
   $SS = 41{,}66\%$
   Berdasarkan perhitungan diatas, maka dapat disimpulkan bahwa persentase hasil kompresi *file* pptx menggunakan algoritma *fibonacci code* adalah sebesar 41,66%.

3.1.4 Dekompresi Algoritma Fibonacci Code

Proses dekompresi dilakukan dengan mengubah karakter menjadi nilai *biner* dan *decimal* yang akan menghasilkan bit semula, dapat dilihat seperti tabel berikut:

Tabel 21. Nilai Desimal Terkompresi

No	Karakter	Biner	Decimal
1	G	01100111	103
2	C	01100011	99
3	Ž	10011110	158
4	¼	10111100	188
5	?	00111111	63
6	Ã	11100011	227
7	L	01001100	76
8	º	10111010	186
9	Ü	11111100	252
10	FS	00011100	28
11	?	00111111	63
12	G	01000111	71
13	•	10000001	129
14	BS	00001000	8

Berdasarkan tabel diatas, maka diperoleh nilai biner kemudian digabungkan menjadi “01100111011000111001111010111000011111111100011010011001011101011111100000111000011111101000111100000010000100001000”
Selanjutnya ialah pengurangan bit menjadi string bit semula, pengurangan bit dilakukan dengan menghilangkan biner *flagging*. Untuk mengembalikan bit menjadi string bit dengan membaca bit terakhir dan kemudian mengubah nilai biner menjadi decimal nyatakan hasil pembacaan dengan n, selanjutnya gunakan rumus 7 + n untuk mengembalikan string bit ke bentuk semula. 8 bit terakhir = 00001000 = 8 = n
7 + n = 7 + 8 = 15

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 448-459

Hilangkan dari string bit sebanyak 15 bit terakhir, menjadi:
0110011101100011100111101011110000111111111000110100110010111010111111000001110000111111010001111"
Berdasarkan perhitungan diatas, string bit berjumlah 97 seperti semula, sehingga dapat dilakukan pembacaan string bit awal. Adapun tabel perhitungannya sebagai berikut :

Tabel 22. Pengecekan Bit Dekompresi *Fibonacci*

No	Biner	Hexadesimal
1	011	50
2	0011	4B
3	1011	03
4	00011	04
5	10011	14
6	11	00
7	01011	06
8	11	00
9	000011	08
10	11	00
11	11	00
12	11	00
13	100011	EA
14	010011	55
15	001011	F9
16	101011	01
17	11	00
18	11	00
19	0000011	6B
20	1000011	0D
21	11	00
22	11	00
23	0100011	13
24	11	00

Dari tabel diatas dapat disimpulkan hasil dekompresi yang dilakukan dengan algoritma *fibonacci code* yang dilakukan dengan mengubah hasil seluruh *hexadecimal* menjadi string bit semula yang terdapat pada *file* awal, maka didapat hasil dekompresi yang sesuai dengan string bit semula yaitu 50, 4B, 03, 04, 14, 00, 06, 00, 08, 00, 00, 00, EA, 55, F9, 01, 00, 00, 6B, 0D, 00, 00, 13, 00

3.1.5 Metode Perbandingan Exponential

Metode perbandingan *Exponential* adalah satu metode untuk menentukan urutan prioritas alternatif keputusan dengan kriteria jamak [12]. Dalam proses perhitungan dan pencarian perbandingan dari kedua algoritma tersebut perlu diperhatikan hal berikut:
Dalam menghitung dan membandingkan proses kompresi dari kedua algoritma tersebut adalah sebagai berikut:

1. Menentukan alternatif analisis perbandingan antara algoritma *yanamoto's recursive code* dan *fibonacci code* dalam kompresi maka perlu ditentukan algoritma mana yang lebih baik digunakan sebagai algoritma kompresi.
2. Menentukan kriteria untuk membandingkan kedua algoritma. Selanjutnya menentukan kriteria dalam menganalisis proses dan cara kerjanya. Adapun kriteria tersebut adalah *Ratio Of Compression* (RC), *Compression Ratio* (CR), *Space Saving* (SS).
3. Pemberian nilai dari setiap kriteria yang telah ditetapkan. Nilai ini diambil berdasarkan analisis algoritma *yanamoto's recursive code* dan *fibonacci code*
4. Menentukan hasil atau proritas keputusan berdasarkan nilai dari setiap alternatif. Hasil keputusan dapat dilihat pada tabel dibawah ini:

Tabel 23. Prioritas Keputusan

Alternatif	RC	CR	SS	Ranking
Yamamoto's Recursive Code	1,26	79,17%	20,83%	2
Fibonacci Code	1,71	58,33%	41,66%	1

Dari tabel 4.21 diatas dapat disimpulkan bahwa algoritma *fibonacci code* lebih efesien dalam hal *Space Saving* dibandingkan dengan algoritma *yamamoto's revursive code*. Dikarenakan algoritma *fibonacci code* yang memiliki nilai SS paling tinggi maka algoritma tersebut lebih baik digunakan dalam kompresi *file* pptx.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 448-459

4. KESIMPULAN

Berdasarkan dari hasil analisa mengkompresi, dekompresi, perbandingan algoritma *yamamoto's recursive code* dan algoritma *fibonacci code* dalam mengkompresi *file* pptx, maka dapat disimpulkan bahwa Penelitian ini menganalisa perbandingan antara algoritma *yamamoto's recursive code* dan *fibonacci code* dalam mengkompresi *file* pptx. Hasil dari perbandingan kinerja algoritma *yamamoto's recursive code* dan algoritma *fibonacci code* dalam mengkompresi *file* pptx terbukti algoritma *fibonacci* lebih efektif dengan hasil *space saving* sebesar 41,66% dalam mengkompresi *file* pptx dari pada algoritma *yamamoto's recursiv code.* Analisis perbandingan aplikasi algoritma *yamamoto's recursive code* dan algoritma *fibonacci code* dalam kompresi *file* pptx dikembangkan dan ditulis dengan menerapkan bahasa pemrograman *Microsoft Visual Studio* 2008 untuk membantu *user* dalam mengompresi, mendekompresi, dan melakukan perbandingan performa kedua algoritma yang lebih efektif dan efisien dalam mengkompresi *file* pptx.

REFERENCES

[1] I. Hasan, N. Irsa Syahputri, and U. Harapan Medan, "Analisis Parameter Kompresi Algoritma Elias Omega Code dan Fibonacci Code Pada File Digital," *Algoritm. J. Ilmu Komput. dan Inform.*, vol. 6341, no. April, p. 1, 2021.

[2] S. B. Ginting *et al.*, "Perbandingan Algoritma Yamamoto ' s Recursive Code Dan Additive Code Dalam Kompresi File Video," vol. 5, 2021, doi: 10.30865/komik.v5i1.3819.

[3] N. Aftikasyah, "Penerapan Algoritma Yamamoto ' s Recursive Code Untuk Mengkompresi File Dokumen," vol. 5, pp. 255–263, 2022, doi: 10.30865/komik.v5i1.3716.

[4] R. A. Sandra, "Implementasi Kombinasi Algoritma Tunstall Code da Boldi-Vigna Untuk Kompresi File Pdf," vol. 8, no. 2, pp. 67–71, 2021.

[5] J. Martina and B. Panjaitan, "Penerapan Algoritma Fibonacci Codes Pada Kompresi Aplikasi Audio Mp3 Berbasis Dekstop," vol. 1, no. 1, pp. 27–33, 2021.

[6] M. A. Syahputra and P. Ramadhani, "Implementasi Metode Fibonacci Dalam Kompresi Iklan Pada Website PT. Amanah Wisata Travel," *KOMIK (Konferensi Nas. …*, vol. 4, pp. 228–231, 2020, doi: 10.30865/komik.v4i1.2685.

[7] 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, [Online]. Available: http://ejurnal.stmik-budidarma.ac.id/index.php/inti/article/view/2391

[8] S. Siregar, F. Fadlina, and S. Nasution, "Enhancing Data Security of Columnar Transposition Cipher by Fibonacci Codes Algorithm," 2020, doi: 10.4108/eai.11-12-2019.2290839.

[9] S. Nainggolan, "Analisa Perbandingan Algoritma Goldbach Codes Dengan Algoritma Dynamic Markov Compression (DMC) Pada Kompresi File Teks Menggunakan Metode Eksponensial," *Maj. Ilm. INTI*, vol. 6, no. 3, pp. 395–399, 2019.

[10] Engel, "**済無**No Title No Title No Title," *Pap. Knowl. . Towar. a Media Hist. Doc.*, no. 0272, 2014.

[11] A. Fau, Mesran, and G. L. Ginting, "Analisa Perbandingan Boyer Moore Dan Knuth Morris Pratt Dalam Pencarian Judul Buku Menerapkan Metode Perbandingan Eksponensial (Studi Kasus : Perpustakaan STMIK Budi Darma)," *J. Times (Technology Informatics Comput. Syst.*, vol. 6, no. 1, pp. 12–22, 2017.

[12] 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.