Bulletin of Information System Research (BIOS)
Volume 2, No 2, April 2024 Page: 64-72
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Analisis Implementasi Algoritma Elias Delta Codes Untuk Kompresi File Audio

Noki Cahya Putra Pasaribu

Ilmu Komputer Dan Teknologi Informasi, Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: nokicppasaribu12@gmail.com

Abstrak−Perkembangan ukuran file sekarang ini sangatlah beragam, ada yang ukurannya kecil dan besar, salah satunya masalah yang sering ditemui dikalangan masyarakat adalah banyaknya file yang disimpan pada penyimpanan memori atau *hard disk* di komputer, Memang pada zaman sekarang ini, ruang penyimpanan sudah ada yang bisa menampung banyak file, akan tetapi masih ada yang tidak memiliki tempat penyimpanan yang besar, sehingga data yang banyak di simpan mengakibatkan ruang penyimpanan memori dan *hard disk* di komputer cepat penuh, serta mengakibatkan kelambatan proses pengiriman file, maka cara yang dapat dilakukan adalah dengan cara mengkompresi file. Salah satu file yang dijadikan sampel pada penelitian ini adalah file audio, maka solusi untuk melakukan pengurangan ukuran file dengan cara mengkompresi file tersebut, dengan melakukan kompresi ukuran asli file akan berbeda ukurannya dengan file yang sudah dikompresi, akan tetapi perlu diketehui bahwa semua file yang dikompresi juga tidak mengalami pengurangan ukuran juga, pada saat melakukan pengompresian membutuhkan algoritma kompresi, pada penelitian ini menggunakan algoritma *elias delta codes*, pengkompresian menggunakan algoritma ini, file yang asli dengan file yang sudah dikompresi ukurannya berbeda. Hasil penelitian yang sudah dilakukan, pada pengkompresian file audio menggunakan algoritma *elias delta codes* mengalami pengurangan ukuran file, pembuktian teknik kompresi yang dilakukan mengambil nilai hexadesimal file audio sebanyak 22 mengurangi ukuran file 0,54% dari file mp3 asli menjadi 43.66 MB.

Kata Kunci: Kompresi File; Dekompresi File; Algoritma Elias Delta Codes; File Audio; Mp3

Abstract−The development of file sizes today is very diverse, some are small and large, one of the problems that is often encountered among the public is the large number of files stored in memory storage or hard disks on computers, Indeed, in this day and age, storage space already exists that can accommodate many files, but there are still those who do not have a large storage area, so that a lot of data is stored resulting in memory storage space and hard disks on the computer quickly full, and resulting in delays in the file delivery process, then the way that can be done is by compressing the file. One of the files sampled in this study is an audio file, then the solution to reducing the file size by compressing the file, by compressing the original size of the file will be different in size from the file that has been compressed, but it should be known that all compressed files also do not experience a reduction in size as well, when compressing requires a compression algorithm, in this study using the Elias delta codes algorithm, compressing using this algorithm, the original file with the file that has been compressed is different in size. The results of the research that has been done, in compressing audio files using the Elias delta codes algorithm has reduced the file size, proving that the compression technique carried out takes the hexadecimal value of the audio file as much as 22 reducing the file size 0.54% from the original mp3 file to 43.66 MB.

Keywords: File Compression; File Decompression; Elias Delta Codes Algorithm; Audio File; Mp3

1. PENDAHULUAN

Salah satu perkembangan pemanfaatan data yang terjadi yaitu ukurannya yang menjadi semakin besar. Ukuran data yang semakin besar dapat mengakibatkan ruang penyimpanan memori atau *hard disk* cepat penuh, apalagi bila ruang penyimpanannya terbatas, maka data yang ingin disimpan tidak bisa terlalu besar dan banyak. Salah satu jenis file yang ukurannya ada yang terbilang besar adalah file audio.

File audio merupakan salah satu file media yang berkembang pesat di industri musik, file audio juga memiliki banyak format, contoh mp3, ogg, mpc, wmadan masih banyak lagi, karena perkembangannya yang begitu pesat membuat banyak ekstensi file audio dengan ukuran yang berbeda-beda. Permasalahan utama bagi orang-orang yang suka menyimpan file musik audio adalah banyaknya file dengan ukuran berbeda-beda yang nantinya akan disimpan pada memori atau *hard disk*, ukuran file yang berbeda itu dikarenakan durasi audionya, jadi untuk menyimpan file yang banyak dan besar tidak memungkinkan tersimpan di penyimpanan komputer karena terbatasnya ruang penyimpanan memori atau *hard disk* di komputer, maka solusi yang dapat dilakukan adalah dengan memperkecil ukuran file audionya atau disebut dengan kompresi file audio, dengan melakukan kompresi, file audio asli akan berbeda dengan file audio setelah dikompresi.

Berdasarkan penelitian terdahulu mengatakan bahwa ukuran file audio yang besar dan tidak dikompresi, akan mengakibatkan kelambatan pada saat pengiriman data dan juga dapat memenuhi kapasitas dari suatu memori penyimpanan[1].

Kompresi file adalah proses yang mengubah data awal besar menjadi data yang lebih kecil. Kompresi terdiri dari dua komponen yaitu *encoding* dan *decoding*, *encoding* sebagai pengambilan pesan serta menghasilkan representasi terkompresi, dan *decoding* yang merekonstruksi pesan asli dari representasi terkompresi. Kompresi pada umumnya disebut suatu teknik pengubahan ukuran file yang tadinya besarmenjadi ukuran yang lebih kecil, dengan adanya teknik kompresi hasil ukuran data yang lebih kecil dari data asli, serta dapat mempercepat waktu pemindahan data dari komputer 1 ke komputer lain.

Berdasarkan penelitian terdahulu yang melakukan teknik kompresi berpendapat bahwa, kompresi file digunakan dalam berbagai keperluan, misalnya membackup data dan transfer data, untuk membackup data tidak

perlu menyalin semua file aslinya, dengan kompresi file, maka kapasitas tempat penyimpanan yang diperlukan menjadi lebih kecil[2], untuk melalukan kompresi file apapun membutuhkan algoritma kompresi, algoritma kompresi memiliki banyak jenis, salah satu contoh algoritma kompresi yang digunakan untuk penelitian ini adalah algoritma *elias delta codes*.

Algoritma *elias delta codes* merupakan pengembangan dari algoritma *elias gamma codes* yang dibuat oleh Peter Elias, jadi kedua algoritma ini masih saling menggunakan rumus yang sama, akan tetapi tetap memiliki perbedaan di proses kompresi, algoritma ini juga telah dinyatakan berhasil memperkecil ukuran file setelah dilakukan kompresi. Beberapa penelitan yang telah dilakukan, bahwa algoritma *elias delta codes* ini bisa memperkecil ukuran file asli dengan file yang sudah dikompresi dengan algoritma ini, maka dengan adanya referensi mengenai algoritma ini, penulis mencoba melakukan kompresi file audio menggunakan algoritma *elias delta codes*. Proses kerja kompresi file menggunakan algoritma *elias delta codes* ini memiliki proses, yang awalnya adalah dengan menentukan file apa yang akan di kompresi terlebih dahulu, kemudian kita mengambil beberapa sampel nilai hexadecimal, yang dimana nilai decimal ubu yang akan digunakan saat melakukan kompresi file secara manual, kemudian melakukan pembacaan isi file yaitu nilai hexadesimal yang telah diambil sebelumnya, pada penelitian mengambil nilai hexa sebanyak 22 angka, menampilkan nilai asli hexa, pembentukan set karakter yang telah di konversi, pembentukan kode *elias delta codes*, memperoleh string bit dan penambahan *padding, flaging*, penambahan *padding* diperlukan apabila jumlah bit dari hasil itu tidak habis dibagi 8, dan *flagging* tetap digunakan, karena sebagai bit pembantu saat melakukan kompresi, lalu melakukan pemisahan bit manjadi beberapa kelompok, lalu melakukan komversi nilai biner menjadi karakter, karakter file terkompresi dapat digunakan sebagai sampel dekompresi file, hasil kinerja kompresi dapat dilihat dari nilai perhitungan rasio kompresi.

Berdasarkan penelitian terdahulu yang mengimplementasikan algoritma *elias delta codes* untuk mengkompresi sebuah file video pada aplikasi video downloader, menyimpulkan bahwa algoritma *elias delta codes* dapat diterapkan untuk mengkompres ukuran file video sehingga ukurannya menjadi lebih kecil dari ukuran sebelumnya. Berdasarkan dari hasil pengujian terhadap sistem bahwa ukuran file video lebih kecil setelah dilakukan kompresi[3].

Berdasarkan uraian latar belakang, maka peneliti menganalisis proses implementasi algoritma *elias delta codes* untuk kompresi file audio dengan tujuan, mengetahui sekaligus memberi pemahaman dalam melakukan pengkompresian file dengan menggunakan teknik kompresi, serta menjadikan skripsi ini sebagai referensi bagi seseorang yang tertarik pada pengkompresian file menggunakan algoritma kompresi, jadi peneliti membuat topik tentang "Analisis Implementasi Algoritma Elias Delta Codes Untuk Kompresi File Audio".

2. METODOLOGI PENELITIAN

2.1 Kompresi File

Kompresi adalah suatu teknik pemampatan data sehingga diperoleh file dengan ukuran yang lebih kecil dari pada ukuran aslinya. Kompresi bekerja dengan mencari pola-pola perulangan pada data dan menggantinya dengan sebuah penanda tertentu[4][5][6].

Kompresi data juga merupakan suatu upaya untuk mengurangi jumlah bit yang digunakan untuk menyimpan atau mentransmisikan data. Kompresi data meliputi berbagai teknik kompresi yang diterapkan dalam bentuk perangkat lunak *(software)* maupun perangkat keras *(hardware)*[7].

Berdasarkan dari uraian pendefenisian dari peneliti sebelumnya, kompresi dapat disimpulkan sebagai teknik yang dapat memperkecil ukuran data dari data aslinya, serta membuat penghematan pada ruang penyimpanan *memory* atau *hard disk*, akan tetapi perlu diketahui bahwa tidak semua hasil file yang telah dikompresi akan menjadi lebih kecil ukurannya. Pengkompresian bekerja melalui beberapa tahapan dan wajib memiliki alat bantu yang dapat memberi hasil pada pengkompresian. Analisis yang dilakukan pada penelitian ini adalah kompresi dengan jenis *lossles* yang berhubungan dengan perubahan ukuran file. Berikut adalah jenis-jenis kompresi data secara umum[2]:

1. *Lossy*
   Kompresi data *lossy* (*destruktif*) dilakukan dengan menghilangkan bagian informasi yang dianggap tidak penting, sehingga dihasilkan rasio kompresi yang sangat tinggi.
2. *Lossles*
   Kompresi data *lossless* (*non-destruktif*) memampatkan data secara *reversible*, artinya jika sebuah data dikompres, kemudian didekompres lagi, maka data tersebut sama dengan data aslinya atau pengkompresian dilakukan tanpa menghilangkan bagian dari informasi.

2.2 Parameter Kompresi

Proses yang dilakukan pada kompresi data, memiliki beberapa parameter yang biasa digunakan untuk mengukur tingkat kualitas dari teknik kompresi data[8][9][10]. Parameter membantu pengguna teknik kompresi dalam melihat perbedaan dari hasil dari kualitas file, pada penelitian ini penulis melakukan analisis terhadap parameter[11], sebagai berikut:

1. *Ratio Of Compression* (RC)

Bulletin of Information System Research (BIOS)
Volume 2, No 2, April 2024 Page: 64-72
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Ratio of Compression (RC) adalah perbandingan antara ukuran data sebelum dikompresi dengan ukuran data setelah dikompresi, untuk mencari nilai *Ratio Of Compression* sebagai berikut:

$$RC = \frac{\text{Ukuran file sebelum dikompresi}}{\text{Ukuran file setelah kompresi}} \quad (1)$$

2. *Compression Ratio* (CR)
   Compression Ratio (CR) adalah persentase besar data yang telah dikompresi yang didapat dari hasil perbandingan antara ukuran data setelah dikompresi dengan ukuran data sebelum dikompresi, untuk mencari nilai *Compression Ratio* data sebagai berikut:

$$CR \frac{\text{Ukuran file setelah dikompresi}}{\text{Ukuran file sebelum dikompresi}} x\ 100\% \quad (2)$$

3. *Redundancy* (Rd)
   Redundancy (Rd) adalah kelebihan yang terdapat didalam data sebelumnya dikompresi, jadi setelah data dikompresi dapat dihitung *redudndancy* data yaitu persentase dari hasil selisih antara ukuran data sebelum dikompresi dengan data setelah dikompresi, untuk mencari nilai *redundancy* data sebagai berikut :

$$Rd = 100\% - CR \quad (3)$$

4. *Space Saving* (SS)
   Space Saving (SS) adalah selisih antara data yang belum dikompresi dengan besar data yang sudah dikompresi,untuk mencari nilai *space saving* data sebagai berikut:

$$SS = 100\% - CR \quad (4)$$

2.3 Algoritma Elias Delta Codes

Algoritma *elias delta codes* adalah salah satu algoritma kompresi yang dibuat oleh Piter Elias, sekaligus merupakan algoritma kompresi yang mampu memperkecil ukuran file. File yang memiliki banyak jenis dan memiliki ukuran yang berbeda-beda menjadikan algoritma *elias delta codes* sebagai solusi untuk menyelesaikan permasalahan ukuran file, salah satu file yang dapat dikompresi dan di analisis pada penellitian ini adalah file audio. Pengkompresian pada algoritma *elias delta codes* dapat dilakukan pada langkah-langkah berikut[3][12].

1. Memasukkan file yang ingin di kompresi dari aplikasi *hexa editor neo,* untuk mengambil nilai hexadesimal yang ada didalam file, nilai yang diambil berguna untuk perhitungan manual.
2. Melakukan pembacaan nilai hexadesimal file yang diambil.
3. Menampilkan nilai semua hexadesimal yang diambil sebelumnya dan ubah setiap nilai hexadesimal ke bilangan biner.
4. Menampilkan nilai hexadesimal, biner, frekuensi dan menampilkan perkalian antara jumlah karakter biner dan frekuensi, serta hasil semua perkalian yang didapat, jumlahkan totalnya.
5. Menampilkan semua nilai set hexadesimal file, nilai *asci*, nilai frekuensi, nilai kode *elias delta code,* jumlah bit kode *elias delta codes* sesuai pengurutan pada masing-masing karakter dan nilai frekuensi dikali bit, nilai frekuensi yang dikalikan dengan bit, jumlahkan.

Tabel 1. Kode Elias Delta Codes

No	Kode Elias Delta Codes
1	1
2	0100
3	0101
4	01100
5	01101
6	01110
7	01111
8	00100000
9	00100001
10	00100010
11	00100011
12	00100100
13	00100101
14	00100110
15	00100111
16	001010000
17	001010001
18	001010010

Cara mencari nilai kode *elias delta codes*:

Bulletin of Information System Research (BIOS)
Volume 2, No 2, April 2024 Page: 64-72
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Contoh nomor kode *elias delta codes* 12(n)
Langkah 1

Rumus:

$2^N \leq n < 2^{N-1}$

$2^3 \leq 12 < 2^{3-1}$

$8 \leq 12 < 16$ (Nilai kiri ujung dan nilai kanan ujung harus memenuhi nilai tengah)

$N = 3$

$n = 2^N+L$

$12 = 2^3+L$ (Lakukan operator matematika)

$L = 4$

Langkah 2

$N + 1$

$3 + 1 = 4$ = (00100 kode *elias gammacodes*)

Langkah 3

L = 4 (Konversi nilai desimal L ke biner) = 00000100

N = 3 (Ambil 3 digit dari nilai biner yang didapat dari kanan ujung) = 100

Gabungkan hasil langkah 2 diikuti hasil langkah 3
Maka : 00100100 (kode *elias delta codes* nomor 12)

6. Hasil perkalian frekuensi dan bit yang didapatkan, dibagi menjadi 8 per *string bit*.
7. Tampilkan seluruh *string bit* yang didapat per 8 *string bit.*
8. Melakukan penambahan *string bit* dari *padding* dan *flagging*, perlu diketahui, nilai setiap frekuensi dan bit yang telah dijumlahkan semuanya, apabila hasilnya habis dibagi dengan 8, maka akan menambahkan *flaging* yaitu 8 karakter tambahan, menggunakan rumus 9 – n, nilai n diambil dari nilai 8 bit akhir dan di konversikan ke biner setelah itu lakukan pengurangan rumus diatas, maka mendapatkan hasil, hasil yang didapat dikonversikan ke biner, akan tetapi apabila hasil dibagi dengan 8 memiliki sisa bagi, akan menambahkan *padding* dan *flaging*, rumus untuk *padding* 7 – n + "1", nilai n diambil dari sisa bagi, jadi 7 dikurang n, hasil pengurangan yang didapat adalah banyaknya angka 0 yang akan dibuat, dan ikuti karakter 1 dari rumus, untuk flagging menggunakan rumus 9 – n, nilai n diambil dari sisa bagi dan hasilnya akan dirubah ke biner.
9. Menampilkan seluruh *string bit* sebelumnya dan menambahkan *string bit* dari *padding* dan *flaging.*
10. Melakukan pemisahan seluruh *sring bit* per 8 setiap karakternya.
11. Menampilkan nilai biner terkompresi, nilai terkompresi diambil dari semua *string bit* yang dilakukan konversi ke nilai desimal, nilai desimal yang didapat dari kompresi, dan dijadikan sampel untuk dekompresi file audio.
12. Hasil akhir kinerja kompresi dilihat dari perhitungan rasio kompresi.

File yang sudah dikompresi dapat dikembalikan ukurannya dengan cara dekompresi, dekompresi ialah proses mengembalikan ukuran asli file seperti semula, berikut adalah langkah-langkah dekompresi menggunakan algoritma *elias delta codes*[3][13][14][15].

1. Mengambil seluruh nilai desimal hasil kompresi, nilai desimal tersebut akan dijadikan sampel dekompresi file, lalu mengubahnya ke nilai biner.
2. Menggabungkan seluruh nilai biner yang telah didapat.
3. Selanjutnya mengembalikan bentuk dan ukuran dengan mengambil 8 bit terahir, nilai bit disebut dengan n, rumus yang digunakan adalah 7 + n, jadi hasil penjumlahan yang didapat adalah total bit yang akan dihilangkan.

Setelah menghilangkan bit, maka dilakukan pembacaan nilai bit kembali pada nilai asli file berdasarkan kode *elias delta codes* di setiap nilai hexadesimal.

3. HASIL DAN PEMBAHASAN

3.1 Analisa

Analisa yang dilakukan pada penelitian ini bertujuan untuk mengetahui proses kerja kompresi dan dekompresi file audio, serta melakukan perancangan aplikasi kompresi dan dekompresi file audio, algoritma yang digunakan selama proses kompresi dan dekompresi adalah algoritma *delta elias codes*. Algoritma *elias delta codes* adalah algoritma kompresi dengan jenis lossless, algoritma ini memiliki fungsi untuk mengurangi ukuran suatu bagian data. Pengkompresian dan dekompresi yang dilakukan penulis menggunakan sampel file audio dengan format mp3, maka

hasil yang ingin dicapai dalam penelitian ini adalah file terkompresi dengan ukuran yang leih kecil dari file asli. Analisa awal yang dilakukan penulis adalah memiliki sampel file audio yang akan di kompresi, kemudian mengambil nilai hexadesimal file audio menggunakan aplikasi *hex editor neo*, selanjutnya pada aplikasi tersebut memaparkan nilai hexadesimal dari file audio, maka penulis dapat mengambil beberapa sampel nilai hexadesimal tersebut untuk dilakukan kompresi, setelah mendapatkan hasil file terkompresi, penulis menganalisis parameter kompresi sekaligus untuk mendapatkan hasil parameter. File audio yang sudah dikompresi dapat dikembalikan ke file audio awal dengan cara dekompresi file audio, cara melakukannya dengan mengambil karakter hasil kompresi file audio dijadikan sebagai sampel dekompresi file audio, kemudian melakukan proses dekompresi dan mengembalikan file audio seperti semula.

3.2 Proses Kompresi

Berdasarkan tabel di atas, maka menghasilkan bilangan hexadesimal file audio yang akan dihitung secara manual. Berikut nilai hexadesimal diurutkan berdasarkan frekuensinya, nilai frekuensi paling banyak maka akan berada diurutan pertama hingga seterusnya paling kecil.

Tabel 2. Nilai Hexadesimal Yang Belum Dikompresi

No	Nilai hexadesimal	Nilai biner	Bit	Frekuensi	Bit x frekuensi
1	00	00000000	8	9	72
2	58	01011000	8	3	24
3	49	01001001	8	1	8
4	44	01000100	8	1	8
5	33	00110011	8	1	8
6	04	00000100	8	1	8
7	01	00000001	8	1	8
8	02	00000010	8	1	8
9	54	01010100	8	1	8
10	12	00010010	8	1	8
11	03	00000011	8	1	8
12	6d	01101101	8	1	8
Total bit					176 Bit

Setelah bilangan hexadesimal diurutkan berdasarkan frekuensi kemunculannya, didapatkan nilai *biner* dan hasil bit yang belum dikompresi, hasil bit yang didapat, itu dari penjumlahan bit dikali frekuensi, maka selanjutnya memasukkan nilai sampel file ke proses kompresi menggunakan algoritma *elias delta codes*, yang nantinya akan memperoleh bit terkompresi, adapun proses kompresi file audio menggunakan algoritma *elias delta codes* dimulai dengan mengurutkan nilai hexadesimal pada kode *elias delta codes* dapat dilihat pada tabel 4.3 di bawah ini.

Tabel 3. Pengurutan Nilai Hexadesimal Pada Kode *Elias Delta Codes*

No	Nilai Hexadesimal	Kode Elias Delta Codes	Bit	Frekuensi	Bit x Frekuensi
1	00	1	1	9	9
2	58	0100	4	3	12
3	49	0101	4	1	4
4	44	01100	5	1	5
5	33	01101	5	1	5
6	04	01110	5	1	5
7	01	01111	5	1	5
8	02	00100000	8	1	8
9	54	00100001	8	1	8
10	12	00100010	8	1	8
11	03	00100011	8	1	8
12	6d	00100100	8	1	8
Total bit					85 Bit

Selanjutnya nilai hexadesimal file audio 49, 44, 33, 04, 00, 00, 00, 00, 01, 02, 54, 58, 58, 58, 00, 00, 00, 12, 00, 00, 03, 6d yang memiliki kode *elias delta codes*, disusun pada masing-masing kode *elias delta codes*, penyusunan yang dilakukan sesuai dengan urutan nilai hexadesimal dari yang pertama hingga akhir, hasil kode *elias delta codes* yang disusun disebut dengan *string bit*, berikut adalah *string bit* yang telah disusun sesuai urutan nilai hexadesimal file audio.

Tabel 4. *String Bit* Kode *Elias Delta Codes*

Hasil String bit
0101 01100 01101 01110 1 1 1 1 01111 00100000 00100001 0100 0100 0100 1 1 1 00100010 1

Hasil String bit
1 00100011 00100100

Hasil *string bit* menggunakan algoritma *elias delta codes* sebanyak 85 bit, maka perlu dilakukan penambahan jumlah bit karena 85 tidak habis dibagi 8, didalam komputer suatu karakter direpresentasikan oleh bilangan ASCI, terdapat sebanyak 8 bit dari bilangan biner, apabila bit suatu data lebih pendek dari 8 bit maka biasanya tidak akan bisa dibaca oleh komputer sebelum dilakukannya proses *encoding*, pada proses *encoding*, bit-bit data tersebut disusun setiap 8 per bit, sehingga membentuk suatu karakter yang dapat dibaca oleh komputer. Jumlah bit 85 tidak habis dibagi 8 atau bukan kelipatan 8 maka dilakukan penambahan bit yaitu *padding* dan *flagging*. *Padding* merupakan penambahan bit data ketika jumlah bit data tersebut tidak habis dibagi 8. Rumus *padding* ialah 7 – n + ”1”, sedangkan *flagging* merupakan penambahan 8 bit bilangan biner yang harus dilakukan setelah *padding* untuk dijadikan bit pembantu pada proses kompresi, rumus *flagging* ialah $9 - n$. Jumlah bit yang habis dibagi 8 atau kelipatan 8, tidak perlu menambahkan *padding*, tetapi tetap harus menambahan *flagging*.

Tabel 5. Penambahan *Padding* dan *Flagging*

Padding	Flagging
85 mod 8 = 5(n) 7 – n + “1” 7 – 5 + “1” = 001	9 - n 9 – 5 = 4 = 00000100

Selanjutnya menggabungkan *string bit* menjadi per 8 bit, setelah mendapatkan hasil nilai dari penambahan *padding* dan *flagging*, penerapannya sebagai berikut:
01010110 00110101 11011110 11110010 00000010 00010100 01000100 11100100 01011001 00011001 00100**001 00000100**, maka total panjang bit terkompresi adalah 96 bit, selanjutnya melakukan pemisahan bit menjadi beberapa kelompok, setiap kelompok terdiri dari 8 bit.

Tabel 6. Pengelompokan Bit

01010110	00110101	11011110	11110010	00000010
00010100	01000100	11100100	01011001	00011001
00100001	00000100			

Berdasarkan pada hasil pemisahan bit di atas, maka terbentuklah 12 kelompok dengan nilai biner baru yang telah terkompresi, maka langkah selajutnya ialah dengan mengubah nilai biner ke nilai desimal terkompresi berdasarkan bit hasil kompresi, hal tersebut untuk mengetahui karakter yang telah diambil dari file audio awal yang sudah dikompresi, berikut adalah pengubahan nilai bit yang sudah terkompresi ke nilai desimal.

Tabel 7. Hasil Karakter Terkompresi

No	Bit Hasil Kompresi	Nilai desimal
1	01010110	86
2	00110101	53
3	11011110	222
4	11110010	242
5	00000010	2
6	00010100	20
7	01000100	68
8	11100100	228
9	01011001	89
10	00011001	25
11	00100001	33
12	00000100	4

Setelah mendapatkan nilai desimal hasil kompresi file audio, nilai desimal yang diambil dari file audio terkompresi dijadikan sebagai sempel untuk dekompresi file audio, karena dengan adanya sampel nilai desimal tersebut file audio yang akan didekompresi tidak akan berbeda dengan file audio awal, berikut adalah nilai desimal yang dijadikan sampel untuk dekompresi file audio sebagai berikut:

Tabel 8. Nilai Desimal Hasil Kompresi File Audio

Nilai Desimal File Audio Terkompresi
86
53
222
242
2

Nilai Desimal File Audio Terkompresi
20
68
228
89
25
33
4

Berdasarkan dari hasil pengkompresian yang telah dilakukan, file audio awal sebelum dikompresi, dengan file audio terkompresi akan ada perbedaan pada nama filenya sebagai berikut:

Tabel 9. Nama File Audio

Nama File Dan Ukuran Sebelum Dikompresi	Nama File Dan Ukuran Setelah Dikompresi
MUSIK DJ.mp3	MUSIK DJ_edc.mp3
44.2 MB	43.66 MB

Selanjutnya untuk menganalisis kualitas hasil kompresi, maka penulis akan mengukur hasil kompresi file audio tersebut sesuai dengan parameter yang telah ditentukan sebagai berikut:

a. *Ratio of Compression* (RC)

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

$$RC = \frac{176}{96}$$

$$RC = 1{,}83$$

b. *Compression Ratio* (CR)

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

$$CR = \frac{96}{176} \times 100\%$$

$$CR = 0{,}54\%$$

c. *Redudancy* (*Rd*)

$$Rd = 100\% - CR$$

$$Rd = 100\% - 0{,}54\%$$

$$Rd = 0{,}99\%$$

d. *Space Saving*(SS)

$$SS = 100\% - CR$$

$$SS = 100\% - 0{,}54\%$$

$$SS = 0{,}99\%$$

3.3 Proses Dekompresi

File yang sudah dikompresi dapat dikembalikan ke bentuk file awal dengan cara dekompresi file, adapun cara untuk dekompresi file audio sebagai berikut:

a. Mengambil seluruh nilai decimal hasil kompresi, nilai decimal tersebut akan dijadikan sampel dekompresi file, lalu mengubahnya ke nilai biner.
b. Menggabungkan seluruh nilai biner yang telah didapat.
c. Selanjutnya mengembalikan bentuk dan ukuran dengan mengambil 8 bit terahir, nilai bit disebut dengan n, rumus yang digunakan adalah 7 + n, jadi hasil penjumlahan yang didapat adalah total bit yang akan dihilangkan.
d. Setelah menghilangkan bit, maka dilakukan pembacaan nilai bit kembali pada nilai asli file berdasarkan kode *elias delta codes* di setiap nilai hexadesimal.

Berikut adalah contoh dekompresi file audio, setelah melakukan pengkompresian, file audio yang sudah dikompresi dengan nama MUSIK DJ_edc.mp3, dapat dilakukan pengembalian file asli dengan melakukan dekompresi file audio. Berikut adalah proses dekompresi file audio, langkah pertama dengan mengambil sampel nilai decimal terkompresi dan merubahnya ke nilai biner, berikut adalah pemaparannya:

Bulletin of Information System Research (BIOS)
Volume 2, No 2, April 2024 Page: 64-72
ISSN 2963-2455 (media online)
https://journal.grahamitra.id/index.php/bios

Tabel 10. Nilai Desimal Terkompresi Dan Nilai Biner

No	Nilai desimal	Nilai Biner
1	86	01010110
2	53	00110101
3	222	11011110
4	242	11110010
5	2	00000010
6	20	00010100
7	68	01000100
8	228	11100100
9	89	11100100
10	25	00011001
11	33	00100001
12	4	00000100

Tahap selanjutnya adalah menggabungan semua nilai biner, menjadi *string bit*, pemaparannya sebagai berikut:

Tabel 11. *String Bit* Terkompresi

String Bit
010101100011010111011111011110010000000100001010001000100111001001110010000011 001001000010000 0100

Selanjutnya melakukan pengembalikan ukuran asli file, dengan mengambil 8 bit terahir. Nilai bit disebut dengan n, rumus yang digunakan untuk menghilangkan banyaknya bit adalah 7 + n, jadi hasil penjumlahan yang didapat adalah total bit yang akan dihilangkan, berikut adalah cara pengimplementasiannya, 8 bit terahir yang diambil adalah 00000100, nilai desimalnya adalah 4, nilai desimal dinyatakan dengan n, maka 7 + 4 = 11, maka hilangkan sebanyak 11 bit, bit yang dihilangkan dimulai dari bit paling kanan.

Tabel 12. Menghilangkan *String Bit* Proses Dekompresi

String Bit
010101100011010111011111011110010000000100001010001000100111001001110010000011 00100100 00100000100 (*string bit* yang dihiangkan)

Selanjutnya awal pembacaan kode yang telah didapatkan dimulai dari setelah mendapatkan nilai pertama tadi dan seterusnya.

Tabel 13. Pengecekan Nilai Bit Dekompresi

No	Kode *elias delta codes*	Keterangan	Nilai Hexadesimal
1	0	Tidak Ada Pada Kode	
2	01	Tidak Ada Pada Kode	
3	010	Tidak Ada Pada Kode	
4	0101	Ada Pada Kode	49
5	0	Tidak Ada Pada Kode	
6	01	Tidak Ada Pada Kode	
7	011	Tidak Ada Pada Kode	
8	0110	Tidak Ada Pada Kode	
9	01100	Ada Pada Kode	44
10	0	Tidak Ada Pada Kode	
11	01	Tidak Ada Pada Kode	
12	011	Tidak Ada Pada Kode	
13	0110	Tidak Ada Pada Kode	
14	01101	Ada Pada Kode	33
15	0	Tidak Ada Pada Kode	
...	...	...	...
84	0010010	Tidak Ada Pada Kode	
85	00100100	Ada Pada Kode	6d

4. KESIMPULAN

Berdasarkan dari hasil seluruh penelitian yang telah dilakukan, maka penulis membuat kesimpulan bahwa pengkompresian file audio menggunakan algoritma *elias delta codes* yang telah dilakukan sesuai prosedur, mendapatkan hasil perbedaan ukuran file asli dengan ukuran file setelah dikompresi, perbedaannya yaitu ukuran file setelah dikompresi lebih kecil yang dipengaruhi oleh hasil parameter kompresi yang didapatkan. Hasil pengkompresian file audio menggunakan algoritma *elias delta codes* mengalami pengurangan file, dari ukuran awal file audio 44.2 MB menjadi 43.66 MB. Aplikasi kompresi dan dekompresi yang dirancang penulis menggunakan *miscrosoft visual basic 2008* berjalan dengan baik dan memberi kemudahan pada pengguna yang ingin melakukan pengecilan ukuran file audio. Penelitian yang sudah dilakukan masih menggunakan satu algoritma kompresi, dan mendapat hasil yang efisien terhadap file, akan tetapi masih perlu ditambahkan beberapa algoritma kompresi lain untuk melihat perbedaan kinerja hasil yang didapatkan. Aplikasi kompresi dan dekompresi yang telah dibuat masih memiliki kekurangan, karena hanya membaca sampel dari file audio saja, alangkah baiknya perlu ditambahkan untuk file dengan format lain juga seperti video, gambar, teks dan lain-lain.

REFERENCES

[1] J. Martina and B. Panjaitan, "Penerapan Algoritma Fibonacci Codes Pada Kompresi Aplikasi Audio Mp3 Berbasis Dekstop," vol. 1, no. 1, pp. 27–33, 2021.
[2] M. Merdiyan and W. Indarto, "Implementasi Algoritma Run Length, Half Byte, dan Huffman untuk Kompresi File," *Semin. Nas. Apl. Teknol. Inf. 2005 (SNATI 2005)*, vol. 2005, no. Snati, pp. 79–84, 2005.
[3] J. Sisca, "Penerapan Algoritma Elias Delta Code Untuk Kompresi File Video Pada Aplikasi Video Downloader," *Resolusi Rekayasa Tek. Inform. dan Inf.*, vol. 1, no. 4, pp. 254–264, 2021.
[4] A. Satyapratama and M. Yunus, "Analisis Perbandingan Algoritma Lzw Dan Huffman Pada Kompresi File Gambar Bmp Dan Png," *J. Teknol. Inf.*, vol. 6, no. 2, pp. 69–81, 2015.
[5] R. Y. Tanjung and M. Mesran, "Perancangan Aplikasi Kompresi File Dokumen Menggunakan Algoritma Adiitive Code," *JURIKOM (Jurnal Ris. Komputer)*, vol. 8, no. 4, pp. 108–113, 2021.
[6] E. Hariska, "Perancangan Aplikasi Kompresi File Gambar Menggunakan Algoritma Additive Code," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 5, no. 1, 2021.
[7] T. Kandaga, "Analisis Penerapan Kompresi dan Dekompresi Data dengan Menggunakan Metode Statistik dan Kamus," *J. Inform.*, vol. 2, no. 2, pp. 81–91, 2006.
[8] R. Handayani, "Perancangan Aplikasi Kompresi File Audio Menerapkan Algoritma Universal Codes," vol. 5, pp. 324–330, 2021, doi: 10.30865/komik.v5i1.3735.
[9] S. H. Silitonga and S. D. Nasution, "Implementasi Algoritma Boldi-Vigna Codes Untuk Kompresi File Audio Pada Aplikasi Pemutar Audio Berbasis Web," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 6, no. 1, pp. 586–595, 2023.
[10] P. Fitria, "Penerapan Algoritma Rice Codes Pada Aplikasi Kompresi File Gambar," *J. Comput. Syst. Informatics*, vol. 1, no. 3, pp. 158–165, 2020.
[11] D. R. P. Lubis, "Kompresi File Teks Dengan Hybrid Algoritma Run Length Encoding Dengan Even Rodeh Code Dan Variable Length Binary Encoding Untuk Menghemat Ruang Penyimpanan," p. 123, 2018.
[12] L. V. Simanjuntak, "Perbandingan Algoritma Elias Delta Code dengan Levenstein Untuk Kompresi File Teks," *J. Comput. Syst. Informatics*, vol. 1, no. 3, pp. 184–190, 2020.
[13] A. Tanjung, "Perbandingan Kinerja Algoritma Elias Delta Code Dan Algoritma Punctured Elias Code Dalam Kompresi File Teks," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 6, no. 1, pp. 182–190, 2023.
[14] N. F. Rizky, S. D. Nasution, and F. Fadlina, "Penerapan Algoritma Elias Delta Codes Dalam Kompresi File Teks," *Build. Informatics, Technol. Sci.*, vol. 2, no. 2, pp. 109–114, 2020.
[15] M. Simangunsong, "Perbandingan Algoritma Elias Delta Code Dan Unary Coding Dalam Kompresi Citra Forensik," *Resolusi Rekayasa Tek. Inform. dan Inf.*, vol. 1, no. 1, pp. 18–26, 2020.