Journal Global Tecnology Computer
Vol 1, No. 3, Agustus, 2022, Hal 92 - 100
ISSN: **2809-6118** (Online - Elektronik)
Website https://ejurnal.seminar-id.com/index.php/jogtc

Perancangan Aplikasi Kompresi File Audio Dengan Menerapkan Algoritma Additive Code

Anggi Pratiwi

Teknik Informatika, Ilmu Komputer Dan Teknologi Informasi, Universitas Budi Darma, Indonesia
Jl. Sisingamangaraja No. 338, Medan, Sumatera Utara, Indonesia
Email: pratiwianggi793@gmail.com

Abstrak−Proses kompresi data didasarkan pada kenyataan bahwa pada hampir semua jenis data selalu terdapat pengulangan pada komponen data yang dimilikinya, misalnya dalam suatu teks kalimat akan terdapat redudansi huruf alfabet dari huruf a sampai dengan huruf z . Kompresi data melalui proses encoding berusaha untuk menghilangkan unsur pengulangan ini dengan mengubahnya sedemikian rupa sehingga ukuran data menjadi lebih kecil. Dalam penelitian ini, dibangun sebuah aplikasi kompresi file audio dengan menggunakan algoritma Additive Code. Tool pengembangan sistem yang digunakan adalah Unified Modelling Language (UML). Dalam pembuatan aplikasi ini ditekankan untuk melakukan kompresi pada file audio berekstensi *.mp3. Pengujian aplikasi kompresi ini dilakukan dengan menguji file yang mempunyai ukuran berbeda-beda dengan nilai sampel rate dan bitrate yang sama. Disimpulkan bahwa file audio dengan ukuran yang besar membutuhkan waktu lebih lama dalam melakukan proses kompresi. File audio hasil kompresi tidak dapat dijalankan sebelum melewati proses dekompresi hal tersebut disebabkan terjadi perubahan struktur pada file audio saat proses dekompresi.

Kata Kunci: Kompresi , Audio, Additive Code

Abstract−The data compression process is based on the fact that in almost all types of data there is always repetition of the data components it has, for example in a sentence text there will be redundancy of the letters of the alphabet from the letter a to the letter z. Data compression through the encoding process seeks to eliminate this repetition element by changing it in such a way that the data size becomes smaller. In this study, an audio file compression application was built using the Additive Code algorithm. The system development tool used is the Unified Modeling Language (UML). In making this application it is emphasized to compress audio files with *.mp3 extension. Testing this compression application is done by testing files that have different sizes with the same sample rate and bitrate value. It was concluded that the audio file with a large size takes longer to perform the compression process. The compressed audio file cannot be run before going through the decompression process, this is due to a change in the structure of the audio file during the decompression process.

Keywords: compression, Audio, Additive Code

1. PENDAHULUAN

Seiring dengan perkembangan zaman yang semakin maju, kebutuhan akan teknologi dan informasi telah menjadi hal yang sangat penting dan diminati oleh banyak orang. Melihat semakin pesatnya perkembangan teknologi dan informasi di seluruh dunia, maka data-data informasi yang dihasilkan pun terus meningkat, seperti file audio. File audio merupakan salah satu file multimedia yang berkembang sangat pesat, terutama pada industri musik yang semakin maju membuat banyak sekali file audio yang berukuran besar.

Ukuran file audio yang besar sebanding dengan kualitas suara yang dikeluarkan. Saat ini masih sering orang-orang mendengarkan lagu dari perangkat handphone ataupun dari komputer atau laptop. File audio yang dapat diputar biasanya adalah file audio yang berformat .mp3. Kapasitas media penyimpanan biasanya terbatas, karena file yang disimpan bukan hanya file mp3 saja, tetapi file lainnya. Biasanya satu lagu ukurannya berkisar 5 sampai 8 MB dimana ukuran file tersebut biasanya memiliki kualitas suara yang bagus dan jernih. Masalah yang sering terjadi seperti seseorang yang mengoleksi berbagai lagu, tetapi ketika media penyimpanan tersebut penuh, maka orang tersebut terpaksa untuk menghapus file mp3 yang sudah dikoleksinya tersebut. Dan apabila file mp3 tersebut diupload di media penyimpanan online seperti google drive, one drive atau yang lainnya, maka masalah yang muncul lebih banyak, yaitu kapasitas media penyimpanan online biasanya hanya 15GB, dan ketika proses uploadnya memakan waktu yang lama, serta quota internet yang banyak, hal ini terjadi karena file tersebut memiliki ukuran besar.

Kompresi data bisa menjadi salah satu solusi untuk menangani atau setidaknya mengimbangi berkembangnya kebutuhan yang tidak terbendung tersebut. Dengan cara file audio tersebut dibagi menjadi ukuran yang lebih kecil. Ukuran audio yang lebih kecil akan mengurangi delay time pada pengiriman data atau upload file sehingga proses upload atau pengiriman data akan lebih cepat, selain itu dengan adanya proses kompresi dapat mengefisiensi ruang penyimpanan pada perangkat ponsel ataupun laptop.

Berdasarkan penelitian yang terdahulu yang dilakukan oleh Riyo Oktavianty Finola dengan judul "Penerapan Algoritma Interpolative Coding Untuk Kompresi File Audio". Maka hasil dari Compression Ratio (CR) dengan menggunakan algoritma Interpolative Coding sebesar 54 %. Itu berarti setelah dikompresi ukuran data adalah 54 % dari data sebelum dikompresi. dan hasil Redudancy dengan menggunakan algoritma Interpolative Coding sebesar 47 %. Itu berarti pengulangan data yang terjadi sebesar 47 %. dari data sebelum dikompresi[1].

Penelitian yang lainnya dilakukan oleh Mutiara Annisa Diah, dkk dengan judul "Aplikasi Kompresi File Audio Menggunkan Algoritma Arithmetic Coding" maka dapat disimpulkan bahwa Algoritma Arithmetic Coding dapat diimplementasikan pada proses kompresi dan dekompresi file audio berekstensi *.wav berklasifikasi stereo. Proses kompresi membutuhkan waktu lebih lama pada file input berukuran besar dibanding file berukuran kecil. dan Algoritma

Journal Global Tecnology Computer
Vol 1, No. 3, Agustus, 2022, Hal 92 - 100
ISSN: **2809-6118** (Online - Elektronik)
Website https://ejurnal.seminar-id.com/index.php/jogtc

Arithmetic Coding tidak optimal dalam melakukan kompresi audio dapat dilihat pada rasio kompresi yang rendah dan grafik yang tidak stabil[2].

Dalam penelitian ini penulis menggunakan algoritma additive code untuk menerapkannya pada aplikasi kompresi file audio berformat mp3. Algoritma additive code diterapkan kedalam aplikasi yang dirancang untuk mengkompresi file audio untuk mengurangi ukuran file audio.

2. METODOLOGI PENELITIAN

2.1 Kerangka Kerja Penelitian

Dalam penelitian ini penulis melakukan beberapa penerapan metode penelitian untuk menyelesaikan permasalahan. Adapun metode penelitian yang dilakukan adalah dengan cara.

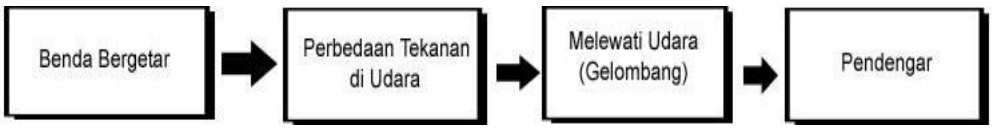
a. Studi Pustaka (Library Research)
   Pengerjaan skripsi ini dimulai dengan mengumpulkan informasi tentang kompresi algoritma additive code, file audio mp3 yang diperlukan menggunakan metode Library Research. Penulis mengumpulkan informasi sebagai referensi baik dari buku, paper, jurnal, makalah, forum, dan sumber-sumber lain yang berkaitan dan beberapa referensi lainnya untuk menunjang pencapaian penelitian.
b. Analisa
   Pada tahap ini dilakukan proses kompresi file audio dengan menerapkan algoritma Additive Code.
c. Implementasi
   Mengimplementasikan hasil studi literatur serta rancangan aplikasi menjadi suatu perangkat lunak yang sesuai dengan tujuan, perumusan masalah dan batasan masalah yang telah didefinisikan.
d. Penyusunan Laporan dan Kesimpulan Akhir
   Metode ini akan dilaksanakan dengan melakukan pendokumentasian hasil analisa dan pengujian secara tertulis dalam bentuk laporan.

2.2 Kompresi

Kompresi adalah proses memampatkan atau mengecilkan ukuran data, Kompresi data adalah proses mengkodekan informasi menggunakan bit atau information-bearing unit yang lain yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu sistem enkoding tertentu. Contoh sederhana yang biasa kita lakukan misalnya adalah menyingkat kata-kata yang sering digunakan tapi sudah memiliki konvensi umum. Misalnya kata "yang" dikompres menjadi kata "yg" [1].

2.3 File Audio

Audio adalah fenomena fisik yang dihasilkan oleh getaran suatu benda yang berupa sinyal analog dengan amplitudo yang berubah secara kontinyu terhadap waktu yang disebut frekuensi [6].



Gambar 1. Fenomena Fisik Audio

2.4 File MP3

MPEG-1 Audio Layer 3 atau lebih dikenal sebagai MP3 adalah salah satu format berkas pengodean suara yang memiliki kompresi yang baik (meskipun bersifat lossy) sehingga ukuran berkas bisa memungkinkan menjadi lebih kecil. Berkas ini dikembangkan oleh seorang insinyur Jerman Karlheinz Brandenburg. MP3 memakai pengodean Pulse Code Modulation (PCM). MP3 mengurangi jumlah bit yang diperlukan dengan menggunakan model psychoacoustic untuk menghilangkan komponen-komponen suara yang tidak terdengar oleh manusia.[8]

2.5 Algoritma Additive Code

Additive code adalah generalisasi kode linier. Ini didefinisikan sebagai sub kelompok dari kelompok Abelian terbatas. Pengertian jarak hamming, bobot hamming, distribusi bobot, dan distribusi bobot yang homogen pada kode aditif hampir sama dengan definisi pada kode linier. Berbeda dengan kode linier yang mendefinisikan dual code menggunakan produk dalam, Additive code menggunakan teori-teori dalam kelompok untuk mendefinisikan dual code-nya [3].

Langkah-langkah algoritma additive code dimulai dengan set dasar terkecil (0, 1), menambahkan ai + 1 menghasilkan 1 dan 2. Salah satu teknik untuk menghasilkan urutan bilangan bulat dasar didasarkan pada prinsip saringan (bandingkan dengan saringan Eratosthenes), dimulai dengan urutan pendek (basis set) yang elemen ai cukup untuk mewakili masing-masing dari beberapa bilangan bulat positif pertama sebagai a sum ai + aj. Urutan awal ini kemudian diperbesar secara bertahap, dengan menambahkan elemen baru ak dan berakhir dengan urutan S = (0, a1, a2, ..., ak) seperti setiap bilangan bulat $1 \leq n \leq ak$ dapat direpresentasikan sebagai penjumlahan dari dua elemen S. Sekarang hasil dari

semua jumlah ai + ak untuk nilai i dari 1 hingga k dan diperiksa untuk memverifikasi bahwa antara lain ak + 1, ak + 2, dan seterusnya. Saat bilangan bulat pertama hilang, bisa menambahkannya ke S sebagai elemen ak + 1. Sekarang kita tahu bahwa setiap bilangan bulat 1 ≤ n ≤ ak + 1 dapat direpresentasikan sebagai penjumlahan dua elemen dari S, dan kita dapat terus memperluas barisan S basis bilangan bulat.

3. HASIL DAN PEMBAHASAN

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

Dalam melakukan kompresi file audio sebelumnya harus menganalisa terhadap file audio yang akan di kompresi. File audio yang akan dianalisa yaitu file audio berformat MP3. Format file MP3 merupakan format file audio terdistribusi paling umum yang digunakan saat ini. File MP3 dibuat oleh Moving Pictures Experts Group (MPEG) dan disingkat dari MPEG-1 atau MPEG-2 Audio Layer 3. Dalam proses kompresi dilakukan penerapan algoritma Additive Code terhadap file audio berformat MP3. File audio memiliki kualitas suara tergantung pada sebagian besar bitrate yang digunakan untuk kompresi. Bitrate yang digunakan biasanya berkisar antara 128, 160, 192, dan 256 kbps. Semakin besar bitrate, semakin bagus kualitasnya, namun hal tersebut berpengaruh pada kebutuhan ruang penyimpanan yang semakin besar, atau dengan kata lain semakin besar bitrate semakin besar pula ukuran file-nya.

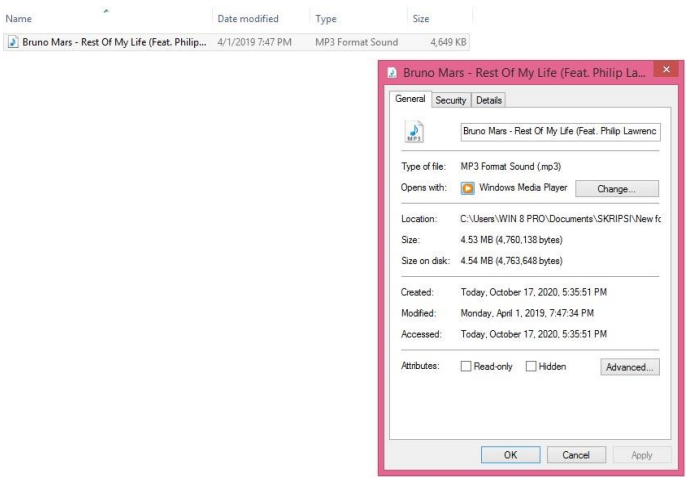
3.1 Penerapan Algoritma Additive Code

Langkah-langkah yang terdapat di bawah ini merupakan rumus dalam algoritma Additive Code yang digunakan untuk memperoleh sebuah tabel kebenaran, yang dimana bilangan heksadesimal file audio yang akan di kompresi. Contoh Kasus: Fibonaci = (0,1,3,5,7,9,11,…) n = 10, maka Sum penjumlahan pertama ditemukan dengan hasil = n Sum= 3+7 Untuk mencari index, dapat dilihat pada Goldbach Sequence dimana 3 berada pada urutan ke 3 dan 7 adalah urutan ke 5. Maka index= 3,5 Pair= ai ,(aj -a i+1) Pair=3,(5-3+1)= 3,3 Untuk mencari codeword, maka dibutuhkan Algoritma Elias Gamma Code dari kedua pair dengan ketentuan

a. Temukan bilangan bulat N terbesar sehingga 2 N ≤ n < 2 N+1 dan tulis n = 2 N + L. File Audio Proses File Audio Dekompresi Baca Nilai Heksa Proses Kompresi File Audio Terkompresi Baca Nilai Heksa27 Rubah nilai n menjadi bilangan biner, lalu hilangkan 1 bit paling kiri.
b. Kodekan N dalam bentuk unary sebagai N nol diikuti oleh 1 atau N 1 diikuti oleh 0.
c. Tambahkan sisa digit biner n dibelakang kode unary yang telah dihasilkan
   Contoh:
   n = 5 2 N ≤ n < 2 N+1 2 2 ≤ 5 < 2 2+1 ` 4 ≤ 5 < 8 5 = 101 → 01 Unary (2) → 001
   Codeword : 00101

Kembali ke perhitungan diatas dimana pair adalah 3,3, jika dihitung, maka codeword dari 3,3 adalah 011:011.
Length= banyaknya digit dari sebuah codeword
Length dari 011:011 = 6

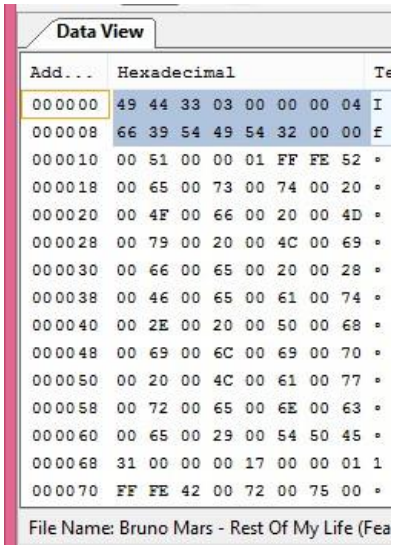
Sebelum file yang akan di kompresi terlebih dahulu dilakukan pembacaan biner yang terdapat pada file audio untuk mendapatkan data berupa data heksadesimal. Membaca heksadesimal yang terdapat pada file gambar menggunakan aplikasi Binery Viewer untuk mencari nilai biner pada file audio. Berikut adalah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



Gambar 2. Sample File Audio

Journal Global Tecnology Computer
Vol 1, No. 3, Agustus, 2022, Hal 92 - 100
ISSN: **2809-6118** (Online - Elektronik)
Website https://ejurnal.seminar-id.com/index.php/jogtc

Berdasarkan gambar dibawah terdapat nilai binery viewer, adapun nilai heksadecimal file audio yang didapat pada tabel di bawah menggunakan aplikasi binery viewer, hasil nilai heksadesimal ditampilkan pada gambar 3.



Gambar 3. Nilai Heksadesimal Sample File.

Berdasarkan hasil gambar 3 telah didapat nilai heksadesimal file audio yang akan dikompresi, dari hasil nilai bilangan heksadesimal yang terdapat diatas, penulis mengambil sampel 8 kolom ke kanan dan 2 baris ke bawah, sehingga nilai sample yang diambil sebanyak 16 bilangan heksadesimal.

Tabel 1. Tampilan Nilai Bilangan Heksadesimal

49	44	33	03	00	00	00	04
66	39	54	49	54	32	00	00

Berdasarkan pada gambar 3 dan tabel 1 diatas, didapat nilai bilangan heksadesimal file audio MP3. Untuk keperluan hitungan manual penulis hanya mengambil sampel 16 bilangan heksadesimal file MP3. Adapun bilangan heksadesimal file MP3 sampel tersebut adalah sebagai berikut : 49 44 33 03 00 00 00 04 66 39 54 49 54 32 00 00. Berikutnya nilai bilangan hexsadesimal diurutkan berdasarkan frekuensinya dan di cari nilai binernya, Nilai frekuensi kemunculan yang paling banyak maka akan berada diurutan pertama.

Tabel 2. Nilai Heksa Berdasarkan Frekuensi Kemunculan

Nilai Heksadesimal	Nilai Biner	Bit	Freq	Bit × Freq
00	00000000	8	5	40
49	01000110	8	2	16
54	01010010	8	2	16
44	01001001	8	1	8
33	01001010	8	1	8
03	00000011	8	1	8
04	11001011	8	1	8
66	10011110	8	1	8
39	01010111	8	1	8
32	01000001	8	1	8
Total bit				128

Setelah bilangan heksadesimal di urutkan berdasarkan frekuensi kemunculan dan didapatkan nilai biner, maka selanjutnya seluruh nilai biner dari bilangan heksadesimal diganti menjadi tabel kebenaran dengan menerapkan algoritma Additive Code pada nilai heksadesimal yang telah di dapat pada tabel 3.

Tabel 3. Data Setelah Dikompresi Menggunakan Algoritma Additive Code

Nilai Heksadesimal	Frek	*Additive Code*	Bit	Bit × Freq
00	5	1010	4	20
49	2	0101	4	8
54	2	1011	4	8
44	1	010010	6	6
33	1	100100	6	6
03	1	010010	6	6

Journal Global Tecnology Computer
Vol 1, No. 3, Agustus, 2022, Hal 92 - 100
ISSN: **2809-6118** (Online - Elektronik)
Website https://ejurnal.seminar-id.com/index.php/jogtc

Nilai Heksadesimal	Frek	*Additive Code*	Bit	Bit × Freq
04	1	010011	6	6
66	1	100101	6	6
39	1	01000100	8	8
32	1	100110	6	6
Jumlah Bit × Frekuensi				80

Setelah kode berhasil di encode berdasarkan perhitungan algoritma Additive Code, tahap selanjutnya adalah menyusun kembali kode-kode yang telah dihasilkan dari proses kompresi sesuai dengan posisi karakter pada nilai heksa.

Tabel 4. String bit Hasil Kompresi Menggunakan Algoritma Additive Code

49	**44**	**33**	**03**
0101	010010	100100	010010
00	**00**	**00**	**04**
1010	1010	1010	010011
66	**39**	**54**	**49**
100101	01000100	1011	0101
54	**32**	**00**	**00**
1011	100110	1010	1010

Berdasarkan data pada Tabel 4, string bit hasil kompresi dengan algoritma Additive Code dapat dituliskan sebagai berikut:

Tabel 5. String bit Hasil Kompresi

0101	010010	100100	010010
1010	1010	1010	010011
100101	01000100	1011	0101
1011	100110	1010	1010

Kemudian sebelum di dapatkan hasil keseluruhan akhir kompresi dilakukan penambahan string bit itu sendiri yaitu padding bit dan flaging bit. Jika sisa bagi panjang string bit terhadap 8 adalah 0 maka tambahan 00000001, nyatakan dengan bit akhir. Sedangkan jika sisa bagi panjang string bit terhadap 8 adalah n (1,2,3,4,5,6,7) maka tambahkan 0 sebanyak 7 – n + "1" di akhir string bit. Lalu untuk flagging tambahkan bilangan biner dari 9 – n. nyatakan dengan bit akhir. karena jumlah string bit 80 habis dibagi delapan, maka tidak perlu menambahkan padding. Lalu tambahkan flagging dengan nilai n=8 menggunakan rumus 9 – n lalu rubah ke biner.

Tabel 6. Penambahan Flagging

Flagging
9 – *n*
9 – 8 = 1
00000001

Jadi string bit yang dihasilkan ditambah dengan padding dan flagging yaitu: "0101010010100100010010101010101010010011100101010001001011010110111001101010101000000001" Total panjang bit keseluruhan setelah ada penambahan padding dan flagging adalah 80+8=88. Selanjutnya lakukan pemisahan bit manjadi beberapa kelompok. Setiap kelompok terdiri dari 8 bit seperti gambar di bawah ini:

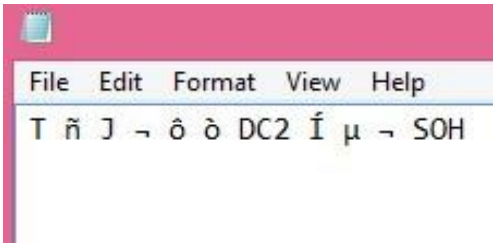
Tabel 7. Hasil Pengelompokan 8 Bit

01010100	10100100	01001010	10101010
10010011	10010101	00010010	11010110
11100110	10101010	00000001	

Berdasarkan pada pembagian kelompok nilai biner, didapatkan 11 kelompok nilai biner baru yang sudah terkompresi beserta nilai biner penambahan bit. Setelah pembagian dilakukan, maka pixel yang sudah dibagi dirubah kedalam suatu karakter dengan terlebih dahulu mencari nilai desimal dari string bit tersebut menggunakan kode ASCII untuk megetahui nilai dari heksadesimal yang sudah terkompresi.

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

Journal Global Tecnology Computer
Vol 1, No. 3, Agustus, 2022, Hal 92 - 100
ISSN: **2809-6118** (Online - Elektronik)
Website https://ejurnal.seminar-id.com/index.php/jogtc



Gambar 4. Hasil Karakter Kompresi

Berdasarkan hasil kompresi di atas dengan mengubah file audio menjadi kode dalam tabel Additive Code sehingga didapatkan hasil yang lebih kecil dan jika dihitung rasio kompresinya/Compression Ratio (CR) adalah:
CR = (Ukuran File terkompresi)/(Ukuran sebelum file dikompresi) x 100% CR = 80/128 x 100%
CR = 0.625 x 100% = 62.5 %

Jika dinyatakan dalam bentuk persen maka dalam rumus sebagai berikut : SS = 100% - CR
SS = 100%- 62.5
SS = 37.5 %
Dari perhitungan di atas, dapat disimpulkan bahwa dengan menggunakan algoritma Additive Code karakter di atas dapat di kompresi sebanyak 62.5%.

3.1.1 Proses Dekompresi Additive Code

Proses dekompresi untuk file hasil kompresi menggunakan Algoritma Additive Code dilakukan dengan menggunakan metode Brute Force. Pembacaan string bit dilakukan dari indeks terkecil sampai indeks terakhir dengan terus menambahkan nilai pada indeks sebelumnya yang tidak mewakili karakter Additive Code. Mengembalikan binary menjadi string bit semula dengan menghilangkan biner padding dan flagging. Untuk mengembalikan binary menjadi string bit semula dapat dilakukan melalui langkah berikut ini. Lakukan pembacaan pada 8 bit terakhir, hasil pembacaan berupa bilangan desimal. Nyatakan hasil pembacaan dengan n, hilangkan bit pada bagian akhir sebanyak 7+n, setelah dilakukan perhitungan pembacaan bit akhir, nilai biner yang dihilangkan sebanyak 8 bit pada akhir n = 1, hilangkan 7 + n atau 7 + 1 = 8. Penjelasan diatas menunjukkan bahwa bit akhir harus dihilangkan. Hasil pengembalian binary menjadi string bit semula dapat dilihat sebagai berikut ini:
0101010101001000100101010101010100100110010101000100101101011011100110101010 10.

Berdasarkan perhitungan dengan algoritma Additive Code, string bit diatas berjumlah 80 bit seperti diawal sehingga dilakukan pembacaan string bit awal. Adapun tabel hasil perhitungan diatas adalah sebagai berikut:

Tabel 8. Pengecekan Bit Dekompresi

Indeks	Nilai Biner	Keterangan	Nilai Heksadesimal
1	0	Tidak Ada	
2	01	Tidak Ada	
3	010	Tidak Ada	
4	0101	Ada	49
5	0	Tidak Ada	
6	01	Tidak Ada	
7	010	Tidak Ada	
8	0100	Tidak Ada	
9	01001	Tidak Ada	
10	010010	Ada	44
11	1	Tidak Ada	
12	10	Tidak Ada	
13	100	Tidak Ada	
14	1001	Tidak Ada	
15	10010	Tidak Ada	
16	100100	Ada	33
17	0	Tidak Ada	
18	01	Tidak Ada	
19	010	Tidak Ada	
20	0100	Tidak Ada	
21	01001	Tidak Ada	
22	010010	Ada	03
23	1	Tidak Ada	
24	10	Tidak Ada	
25	101	Tidak Ada	

Indeks	Nilai Biner	Keterangan	Nilai Heksadesimal
26	1010	Ada	00
27	1	Tidak Ada	
28	10	Tidak Ada	
29	101	Tidak Ada	
30	1010	Ada	00
31	1	Tidak Ada	
32	10	Tidak Ada	
33	101	Tidak Ada	
34	1010	Ada	00
35	0	Tidak Ada	
36	01	Tidak Ada	
37	010	Tidak Ada	
38	0100	Tidak Ada	
39	01001	Tidak Ada	
40	010011	Ada	04
41	1	Tidak Ada	
42	10	Tidak Ada	
43	100	Tidak Ada	
44	1001	Tidak Ada	
45	10010	Tidak Ada	
46	100101	Ada	66
47	0	Tidak Ada	
48	01	Tidak Ada	08
49	010	Tidak Ada	
50	0100	Tidak Ada	
51	01000	Tidak Ada	
52	010001	Ada	00
53	0100010	Tidak Ada	
54	01000100	Ada	39
55	1	Tidak Ada	
56	10	Tidak Ada	
57	101	Tidak Ada	
58	1011	Ada	54
59	0	Tidak Ada	
60	01	Tidak Ada	
61	010	Tidak Ada	
62	0101	Ada	49
63	1	Tidak Ada	
64	10	Tidak Ada	
65	101	Tidak Ada	
66	1011	Ada	54
67	1	Tidak Ada	
68	10	Tidak Ada	
69	100	Tidak Ada	
70	1001	Tidak Ada	
71	10011	Tidak Ada	
72	100110	Ada	32
73	1	Tidak Ada	
74	10	Tidak Ada	
75	101	Tidak Ada	
76	1010	Ada	00
77	1	Tidak Ada	
78	10	Tidak Ada	
79	101	Tidak Ada	
80	1010	Ada	00

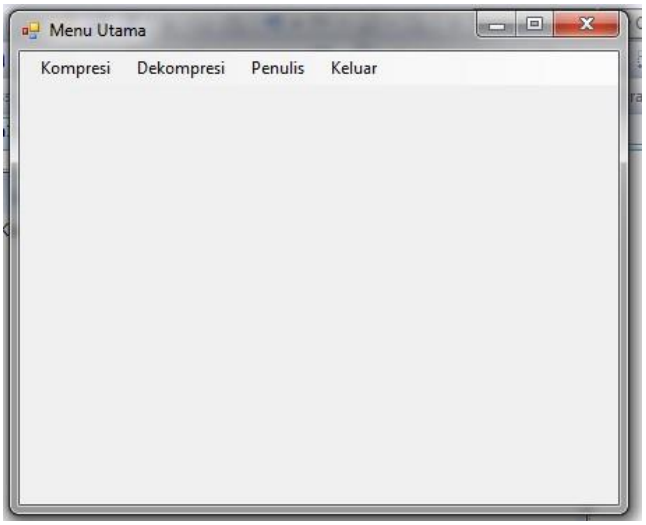
Maka dari penjabaran di atas dapat disimpulkan hasil dekompresi algoritma Additive Code dengan kode kebenaran dari algoritma Additive Code yang dilakukan dengan cara, membaca ulang keseluruhan sample bilangan biner hasil dari kompresi menggunakan algoritma Additive Code. Pembacaan nilai biner dimulai dari awal sampai nilai bilangan biner membentuk sebuah nilai biner yang bernilai benar, dan nilai biner tersebut menghasilkan nilai heksadesimal berdasarkan tabel kode kebenaran algoritma Additive Code, hasil dari kompresi bilangan heksadesimal menjadi string

bit semula yang terdapat pada file awal, berikut adalah nilai awal bilangan heksadesimal "49 44 33 03 00 00 00 04 66 39 54 49 54 32 00 00".

3.2 Implementasi

Setelah menganalisis masalah dan juga merancang sistem yang akan di implementasikan, maka akan dilakukan implementasi dan juga pengujian atas sistem tersebut dengan bahasa pemrograman yang akan digunakan.

a. Menu Utama
Menu Utama merupakan tampilan yang muncul pertama sekali pada saat program dijalankan. Tampilan ini berisi menu Kompresi dan Dekompresi. Tampilan menu utama dapat dilihat pada Gambar 5.

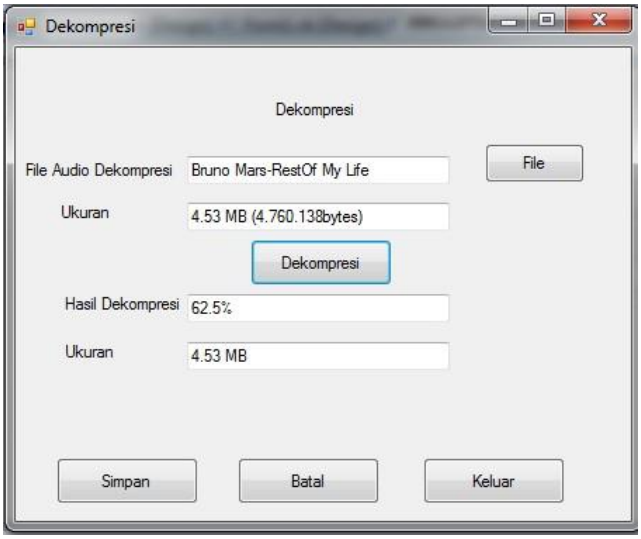


Gambar 5. Tampilan Menu Utama

b. Proses Kompresi
Pada proses kompresi maka yang pertama sekali dilakukan adalah memilih file gambar yang akan dikompresi dan memilih lokasi penyimpanan yang akan di kompresi. Selanjutnya diikuti dengan melakukan proses kompresi pada file tersebut. Adapun tampilannya dapat dilihat pada Gambar 6.

Gambar 6. Proses Kompresi

c. Proses Dekompresi
Pada proses dekompresi maka yang pertama sekali dilakukan adalah memilih file gambar yang akan didekompresi dan memilih lokasi penyimpanan yang akan di kompresi atau di dekompresi. Selanjutnya diikuti dengan melakukan proses kompresi pada file tersebut. Adapun tampilannya dapat dilihat pada Gambar 7.



Gambar 7. Proses Dekompresi

4. KESIMPULAN

Sesuai dengan apa yang telah dibahas pada bab-bab sebelumnya, maka penulis mengambil sebuah kesimpulan yaitu, Setelah mengikuti prosedur untuk mengkompresi file audio dan hasilnya adalah file audio berhasil dikompresi. Setelah di Implementasikan untuk kompresi file audio, hasilnya yaitu dapat mengkompresi sampai dengan rasio 62,5%. Aplikasi kompresi file audio dapat dirancang dengan baik menggunakan aplikasi Microsoft Visual Studio 2008.

REFERENCES

[1] S. E. M. B. Nelson, M., Gailly, J.L., The Data Compression Book, Cambridge."
[2] Nelson, M., Gailly, J.L., Data Compression Book, Second Ed. M&T Books,Cambridge
[3] Magma/handbook, "additive code magma, 1921.
[4] A. S. I. Sinaga, "Studi Perbandingan Kinerja Teoretis dan Rill Algoritma Exact String Matching Shift-And dan Morris Pratt, 2017.
[5] S. Sukarta, "Algoritma dan Pemrograman, 2018.
[6] Nurasyiah, "Perancangan Aplikasi Kompresi File Audio Dengan Algoritma Aritmic Coding. IV, 104 - 106, 2013.
[7] Heri Santoso, M. Fakhriza, "Perancangan Aplikasi File Audio Format WAV (WaveForm) Menggunakan Algoritma RSA, Vol. 02, pp. 01. 2018.
[8] M. Nilsson, "Internet Engineering Task Force. pp. 01, 2000"
[9] [Adi Widarma, Sri Rahayu, Jogianto, "Perancangan Aplikasi Gaji Karyawan Pada PT. PP London Sumatera Indonesia Tbk Gunung Melayu Estate Gunung Melayu Estate - Kabupaten Asahan, Sumatera Utara,pp.02.
[10] Mustakini. J. Hartono, "Sistem Teknologi Informasi", 3th ed, Yogyakarta: Andi, 2009.
[11] Mustakini. J. Hartono, "Sistem Teknologi Informasi", 3th ed, Yogyakarta: Andi, 2009.
[12] S. Dharwiyanti, "Pengantar Unified Modeling Language (UML)", pp. 02. 2003.
[13] D. Irwanto, "Perancangan Perangkat Lunak Berorientasi Objek dengan UML", pp. 187. 2006.
[14] Handy Januar Permana, Erna Astriyani, Tanti Mayang Sari, "Perancangan Sistem Informasi Manajemen Layout Bahan Baku Berbasis Web Pada PT. Sanichem Tunggal Pertiwi, Vol.4 No.2 – Agustus 2018.
[15] S. Yuni, "Analisis dan Perancangan UML (Unified Modeling Language)", Yogyakarta: 2013
[16] Rahman Nur Hadi, Dewiyani Sunarto , Valentinus Roby Hananto, "Rancang Bangun Sistem Informasi Pengadaan Barang PT Antar Surya Jaya, JSIKA Vol. 6, No. 12. Tahun 2017.
[17] Rahman Nur Hadi, Dewiyani Sunarto , Valentinus Roby Hananto "Rancang Bangun Sistem Informasi Pengadaan Barang PT Antar Surya Jaya, JSIKA Vol. 6, No. 12. Tahun 2017.
[18] Priyanto Hidayatullah, "Visual Basic .NET membuat aplikasi database dan program kreatif", Bandung, 2012