ANALISIS PERBANDINGAN ALGORITMA *ARITHMETIC CODING* DAN *LEMPEL ZIV WELCH* (LZW) DALAM MENGKOMPRESI *FILE* AUDIO MP3

[1]Mhd. Abrar Arief, [2]Pristiwanto, [3]Saidi Ramadan Siregar
[1,2,3]Universitas Budi Darma, Medan
[1]arieftanjung13@gmail.com, [2]4nt0.82@gmail.com, [3]saidiramadan89@gmail.com

ABSTRAK

File audio MP3 adalah format yang umum digunakan untuk menyimpan dan mengirim data audio, tetapi ukuran file dapat mempengaruhi waktu pengiriman dan kapasitas memori yang dibutuhkan. Pemampatan file MP3 menjadi krusial untuk efisiensi transfer dan penghematan ruang penyimpanan. Dua algoritma kompresi lossless yang populer, yaitu arithmetic coding dan Lempel-Ziv-Welch (LZW), memiliki peran penting dalam menentukan seberapa baik file audio MP3 dapat dikompresi tanpa kehilangan kualitas signifikan. Analisis perbandingan antara kedua algoritma ini menggunakan kriteria seperti Compression Ratio, Ratio Compression, Redundancy, dan Space Saving dengan bantuan metode eksponensial menunjukkan bahwa arithmetic coding dan LZW masing-masing memiliki nilai total 10,1127 dan 9,6047. Nilai ini mengindikasikan bahwa meskipun arithmetic coding mungkin lebih efisien dalam hal kompresi, LZW memerlukan usaha yang lebih sedikit. Perbandingan ini memberikan wawasan tentang kelebihan dan kekurangan masing-masing algoritma, membantu dalam memilih metode kompresi yang paling efektif untuk file audio MP3.

Kata Kunci**:** *kompresi, file audio, arithmetic coding, lempel ziv welch*

ABSTRACT

MP3 audio files are a common format for storing and transmitting audio data, but file size can impact delivery time and memory capacity requirements. Compressing MP3 files is crucial for efficient transfer and storage space savings. Two popular lossless compression algorithms, namely arithmetic coding and Lempel-Ziv-Welch (LZW), play a significant role in determining how well MP3 audio files can be compressed without significant loss of quality. A comparative analysis between these two algorithms using criteria such as Compression Ratio, Ratio Compression, Redundancy, and Space Saving with the help of exponential methods indicates that arithmetic coding and LZW have total values of 10.1127 and 9.6047, respectively. This suggests that while arithmetic coding may be more efficient in terms of compression, LZW requires less effort. This comparison provides insights into the advantages and disadvantages of each algorithm, aiding in the selection of the most effective compression method for MP3 audio files.

*Keywords**:** compression, audio file, arithmetic coding, lempel-ziv-welch*

A. Pendahuluan

Format MP3 adalah salah satu format yang paling umum digunakan untuk menyimpan dan mengirim data audio, namun ukuran file MP3 dapat mempengaruhi proses pengiriman dan kapasitas penyimpanan (Sari, 2018). Ukuran file yang lebih besar memerlukan kapasitas memori yang lebih besar pula, sehingga pemampatan file menjadi sangat penting untuk efisiensi transfer data dan penghematan ruang penyimpanan (Banjarnahor, 2023). Kompresi MP3 sering melibatkan algoritma untuk mengurangi ukuran file tanpa mengorbankan kualitas audio secara signifikan (Informasi & Situmorang, 2024);(Setiawan, 2023). Perbandingan antara algoritma kompresi seperti Arithmetic Coding dan Lempel Ziv Welch adalah kunci untuk memilih metode kompresi yang paling efektif.

Algoritma kompresi memiliki peran krusial dalam menentukan efisiensi pemampatan file audio. Arithmetic Coding dan Lempel Ziv Welch adalah dua algoritma kompresi lossless yang populer, masing-masing dengan pendekatan berbeda dalam mengurangi ukuran file (Satyapratama et al., 2023). Kriteria seperti Compression Ratio, Ratio of Compression, Redundancy, dan Space Saving digunakan untuk mengevaluasi efektivitas kedua algoritma. Arithmetic Coding fokus pada pengkodean informasi dengan bit yang lebih efisien untuk mengurangi redundansi, sementara Lempel Ziv Welch mencari pola dalam data dan menggantinya dengan representasi yang lebih kompak.

Evaluasi dan analisis kedua algoritma ini menunjukkan bahwa masing-masing memiliki kelebihan dan kekurangan. Arithmetic Coding adalah metode encoding entropi yang menggantikan simbol input dengan bilangan floating-point, memanfaatkan bit lebih sedikit untuk simbol yang sering muncul (Sianturi, 2018);(Octiviani, 2020). Pengujian perbandingan ini penting untuk memahami bagaimana kedua algoritma ini bekerja dalam konteks kompresi file audio MP3 dan untuk menentukan metode kompresi yang paling efisien. Dengan memahami karakteristik dan performa masing-masing algoritma, kita dapat memilih teknik kompresi yang paling sesuai untuk kebutuhan spesifik file audio (Prades, 2024).

Algoritma Lempel Ziv Welch (LZW) adalah metode kompresi lossless yang diciptakan oleh Abraham Lempel, Jacob Ziv, dan Terry Welch (Arizki et al., 2024);(Panggabean, 2018). Sebagai algoritma berbasis kamus, LZW tidak bergantung pada model statistik melainkan pada pengidentifikasian pola perulangan karakter dalam data. Pendekatan ini memungkinkan algoritma untuk menyimpan representasi yang lebih efisien dari data yang sering muncul. Salah satu keunggulan utama dari LZW adalah kecepatan kompresi yang sangat cepat, sehingga cocok untuk aplikasi yang membutuhkan pemrosesan data dalam waktu singkat. Selain itu, LZW juga menawarkan tingkat kompresi yang cukup baik, memungkinkan pengurangan ukuran file tanpa mengorbankan kualitas data secara signifikan. Secara keseluruhan, algoritma LZW efektif dalam mengompresi data dengan cepat dan efisien, terutama dalam situasi di mana pola perulangan karakter dominan. Teknik ini banyak digunakan dalam berbagai aplikasi dan format file karena kombinasi kecepatan dan efisiensinya dalam kompresi data. (Nurina Prabiantissa et al., 2023).

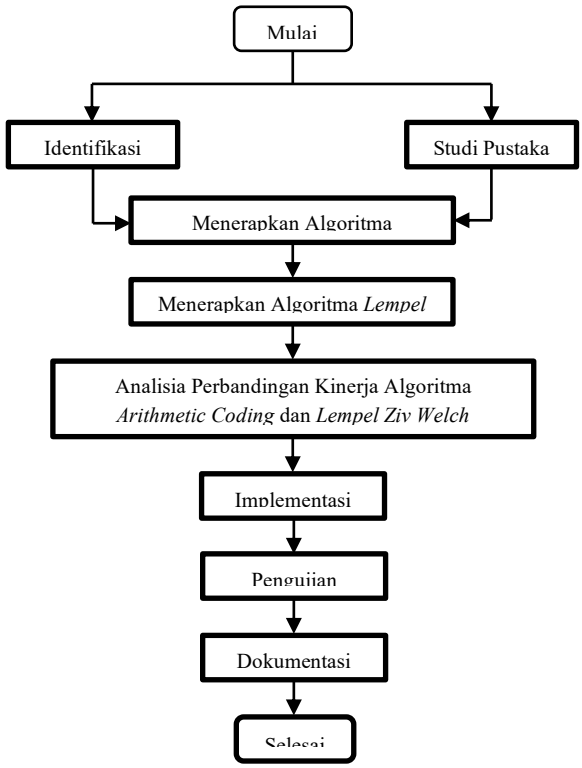
Penelitian oleh Muhammad Iqbal dan rekan-rekannya dalam studi berjudul “Analisis Kompresi File Teks Menggunakan Algoritma Lempel Ziv Welch” menunjukkan bahwa algoritma Lempel Ziv Welch (LZW) efektif dalam mengompresi file teks, menghasilkan ukuran file yang lebih kecil dan dapat mendekompresinya dengan baik. Pengujian menunjukkan bahwa algoritma LZW paling unggul dalam kompresi file dengan ekstensi *.doc dibandingkan dengan jenis file teks lainnya (Iqbal et al., 2022). Penelitian oleh Dian Oktaviani menunjukkan bahwa algoritma Lempel Ziv Welch (LZW) sangat efektif dalam mengompresi berbagai jenis file, termasuk dokumen teks, gambar, audio, dan video. Aplikasi yang diuji berhasil menghasilkan ukuran file yang lebih kecil dibandingkan dengan file asli, berkat kemampuannya membaca setiap bit dari file tersebut. Hasil kompresi ini lebih efisien dibandingkan dengan algoritma LZW standar (Oktaviani & Suartana, 2020). Penelitian oleh Riris Ariska menunjukkan bahwa pengkompresian kamus dalam proses pengiriman file melibatkan

pemilihan dan kompresi file kamus untuk mengurangi ukuran file secara signifikan. Penerapan algoritma Arithmetic Coding pada file yang dipilih berhasil mengurangi ukuran file hingga 40%, mempercepat proses transmisi, dan menghemat kuota internet (Ariska, 2021). Penelitian oleh Ibnu Syuhada berjudul "Implementasi Algoritma Arithmetic Coding dan Shannon-Fano Pada Kompresi Citra PNG" menunjukkan bahwa algoritma Arithmetic Coding dan Shannon-Fano memiliki performa kompresi yang sangat baik, dengan rasio kompresi masing-masing sebesar 62,88% dan 61,73%. Arithmetic Coding menawarkan kecepatan kompresi rata-rata 0,072449 detik, sedikit lebih cepat dibandingkan Shannon-Fano yang memerlukan 0,077838 detik. Namun, Shannon-Fano memiliki kecepatan dekompresi yang lebih baik, yaitu 0,028946 detik dibandingkan dengan Arithmetic Coding yang memerlukan 0,034169 detik. Kedua algoritma berhasil mendekompresi citra dengan kualitas yang sesuai dengan citra asli (Syuhada, 2022).

Berdasarkan latar belakang masalah yang telah disampaikan, peneliti memiliki tujuan untuk menguraikan solusi terhadap permasalahan penelitian yang mencakup kompresi *file* audio ini melalui judul "Analisis Perbandingan Algoritma *Arithmetic Coding* dan Algoritma *Lempel Ziv Welch* (LZW) Dalam Mengkompresi File Audio MP3".

B. Metode Penelitian

Kerangka penelitian melibatkan seluruh proses dari pencarian dan analisis hingga pengamatan dan pemikiran yang sistematis, disusun secara ilmiah untuk meningkatkan pemahaman tentang topik yang diteliti. Kerangka kerja penelitian ini ditunjukkan dalam gambar 1 berikut:



Gambar 3.1 Kerangka Penelitian

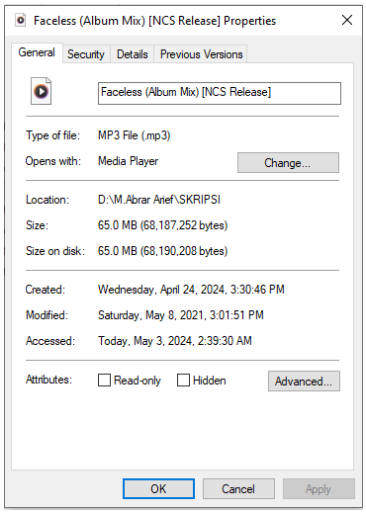
Sampel Data

Sampel data yang digunakan berupa *file* audio yang berekstensi *mp3. Sampel data tersebut dapat dilihat pada tabel di bawah ini sebagai berikut:

Tabel 3.1 *File* audio yang akan dikompresi

Nama *file*	Faceless (Album Mix) [NCS Release]
Jenis *file*	MP3
Ukuran *file*	65,0 MB

Berikut *file* audio yang digunakan sebagai sampel data dengan *properties* yang dapat dilihat pada gambar 1 berikut:

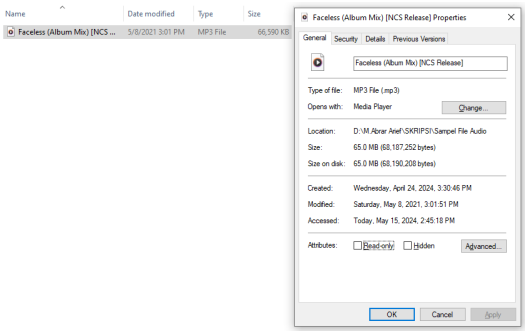


Gambar 1 *Properties*

C. Hasil Penelitian dan Pembahasan

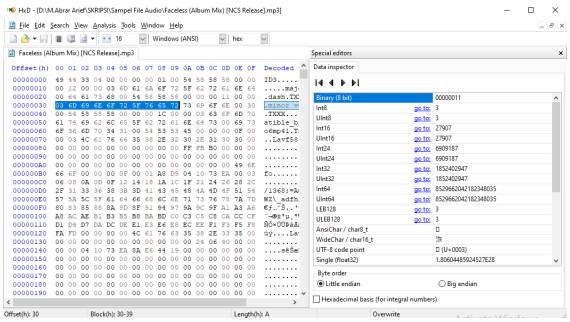
Penerapan Algoritma *Arithmetic Coding*

Perlu dilakukan pembacaan nilai biner pada *file* audio beresktensi *mp3 terlebih dahulu sebelum melakukan proses kompresi dan dekompresi dengan algoritma *arithmetic coding*. Berikut *file* audio yang akan dikompres, yaitu:



Gambar 2 Sampel *File* Audio

Berdasarkan gambar di atas maka selanjutnya mencari nilai *hexadecimal* pada *file* audio berekstensi *mp3 dengan menggunakan *software Hex Editor* (HxD), yang terlihat pada gambar 4.3 di bawah ini:



Gambar 3 Nilai *Hexadecimal* Sampel *File* Audio

Berdasarkan gambar nilai *hexadecimal* di atas, diambil 10 nilai *hexadecimal* sebagai sampel untuk dilakukan proses kompresi *file* audio.

Penerapan Algoritma *Lempel Ziv Welch*

Algoritma Lempel Ziv Welch (LZW) bekerja dengan memeriksa setiap karakter dan menggabungkannya menjadi string, yang kemudian diindeks dalam kamus. Inisialisasi algoritma ini menggunakan kode ASCII dari 0-255, dengan indeks kamus dimulai dari 256. Sebagai contoh, 10 nilai hexadecimal digunakan untuk proses kompresi dengan algoritma LZW, diambil dari file audio menggunakan perangkat lunak HxD. Tabel di bawah ini menunjukkan rincian proses tersebut:

Tabel 1 Nilai *Hexadecimal*

03	6D	69	6E	6F	72	5F	76	65	72

Tabel di atas menunjukkan nilai hexadecimal yang diambil sebagai sampel data. Selanjutnya, frekuensi kemunculan setiap nilai dihitung untuk menentukan seberapa sering setiap nilai hexadecimal muncul. Nilai dengan frekuensi kemunculan terbanyak akan berada di urutan pertama, seperti yang terlihat pada tabel 2 di bawah ini:

Tabel 2 Nilai *Hexadecimal* dan Frekuensi

No	Nilai *Hexadecimal*	Frekuensi
1	72	2
2	03	1
3	6D	1
4	69	1
5	6E	1
6	6F	1
7	5F	1
8	76	1
9	65	1

Berdasarkan pada tabel 4.16 di atas, isi *dictionary* pada awal proses diatur menjadi 9 karakter dasar yaitu: “03, 6D, 69, 6E, 6F, 72, 5F, 76, 65”. Berikut ini tahapan kompresi menggunakan algoritma *lempel ziv welch*, yaitu:

1. Proses diinisialisasikan dalam jumlah *bit* yang lebih sedikit dibandingkan *string* aslinya.

Tabel 3 Inisialisasi

Sampel	Desimal	Sampel	Desimal	Sampel	Desimal
03	3	69	105	6F	111
5F	95	6D	109	72	114
65	101	6E	110	76	118

2. P adalah karakter pertama dalam *stream* karakter.

Tabel 4 *Stream* karakter pertama

03	3

3. Q adalah karakter berikutnya dalam *stream* karakter.

Tabel 5 *Stream* karakter kedua

5F	95

4. Apakah *string* (P + Q) terdapat dalam *dictionary*?
   a. Jika “ya” maka P = P + Q (gabungan P dan Q menjadi *string* baru)
   b. Jika “tidak” maka:
      1. *Output* sebuah kode untuk menggantikan *string* P.
      2. Tambahkan *string* (P + Q) ke dalam *dictionary* dan berikan nomor/kode berikutnya yang belum diganti dalam *dictionary* untuk *string* tersebut.
      3. P = Q
5. Apakah masih ada karakter berikutnya dalam *stream* karakter?
   a. Jika “ya” maka kembali ke langkah 2.
   b. Jika “tidak” maka *output* kode yang menggantikan string P, lalu terminasi proses (*stop*).

Tabel 6 Kompresi LZW

Input = P	*Temporary* (P + Q)	In *Dictionary*	*ADD TO DICTIONARY*		
			Index	Kamus Baru	*Output*
03	03,5F	TIDAK	256	[03,5F]	03
5F	5F,65	TIDAK	257	[5F,65]	5F
65	65,69	TIDAK	258	[65,69]	65
69	69,6D	TIDAK	259	[69,6D]	69

6D	6D,6E	TIDAK	260	[6D,6E]	6D
6E	6E,6F	YA	-	-	-
6E,6F	6E,6F,72	YA	-	-	-
6E,6F,72	6E,6F,72,76	TIDAK	261	[6E,6F,72,76]	72
76	76, Null	TIDAK	262	[76,Null]	76

Berdasarkan penjelasan tabel, dapat disimpulkan bahwa kompresi menggunakan algoritma Lempel Ziv Welch menghasilkan pengurangan ukuran file yang signifikan. Awalnya, total input dari sampel data adalah 10 unit, yang setelah dikompresi menjadi 7 unit output. Sebelum kompresi, total bit yang digunakan adalah 80 bit (10 unit * 8 bit per unit), sementara setelah kompresi, ukuran berkurang menjadi 56 bit (7 unit * 8 bit per unit).

Langkah selanjutnya adalah mengevaluasi kinerja kompresi dengan membandingkan total bit sebelum dan setelah proses kompresi. Dengan total bit awal sebesar 80 bit dan ukuran setelah kompresi sebesar 56 bit, analisis menunjukkan adanya pengurangan ukuran file sebesar 30%.

Hasil kompresi ini, yang diterapkan pada sampel data file audio berekstensi *mp3, membuktikan bahwa algoritma Lempel Ziv Welch efektif dalam mengurangi ukuran file. Pengurangan sebesar 30% menunjukkan efisiensi algoritma dalam mengompresi data audio tanpa mengorbankan kualitas secara signifikan.

Analisa Perbandingan Algoritma *Arithmetic Coding* dan LZW

Parameter seperti *Ratio Compression* (Rc), *Redudancy* (Rd), *Compression Ratio* (Cr), dan *Space Saving* (Ss) yang diterapkan oleh kedua algoritma di atas yang digunakan untuk membandingkan pada tahapan analisa perbandingan algoritma *arithmeic coding* dengan algoritma *lempel ziv welch*. Hasil perbandingan tersebut disajikan dengan metode perbandingan eksponensial yang dapat dilihat pada tabel sebagai berikut:

Tabel 7 Hasil Perbandingan dengan Metode Eksponensial

Algoritma	Kriteria				Total Nilai $\sum(N)^B$
	Rc	Cr	Rd	Ss	
Arithmetic Coding	2	50 %	50 %	50 %	**10,1127**
Lempel Ziv Welch	1,428	70 %	30 %	30 %	**9,6047**

Bobot	0,20	0,30	0,15	0,35	**1**

Berdasarkan perhitungan di atas, dapat disimpulkan bahwa algoritma Lempel Ziv Welch memerlukan usaha yang lebih sedikit dibandingkan dengan algoritma Arithmetic Coding. Meskipun Arithmetic Coding lebih efektif dalam hal rasio kompresi, algoritma Lempel Ziv Welch lebih efisien dalam kompresi file audio berformat MP3. Hal ini menunjukkan bahwa LZW lebih praktis dan membutuhkan sumber daya yang lebih sedikit untuk proses kompresi dibandingkan dengan Arithmetic Coding.

Implementasi

Implementasi program merupakan tahapan proses penerapan sistem berdasarkan hasil analisis sebelumnya. Pada tahapan ini dilakukan pembahasan tentang *hardware*, *software*, serta tampilan sistem saat berjalan.

Kebutuhan Sistem

Kebutuhan sistem mencakup perangkat keras dan perangkat lunak yang diperlukan untuk aplikasi, serta perincian implementasi sistem yang terencana dengan baik untuk mencapai tujuan tertentu. Implementasi sistem bergantung pada spesifikasi yang digunakan, sehingga perangkat keras dan perangkat lunak harus memenuhi standar tertentu agar dapat berjalan dengan efektif.

Dalam penelitian ini, perangkat keras yang digunakan adalah komputer model LENOVO V14-IIL dengan spesifikasi prosesor Intel Core i3, memori 4GB RAM, penyimpanan 500GB, layar 14 inci HD (1366 x 768), serta dilengkapi dengan keyboard dan mouse. Untuk perangkat lunak, sistem harus memenuhi syarat minimal seperti sistem operasi Windows 10, tipe sistem 64-bit dengan prosesor x64-based, serta aplikasi Microsoft Visual Basic 2010.

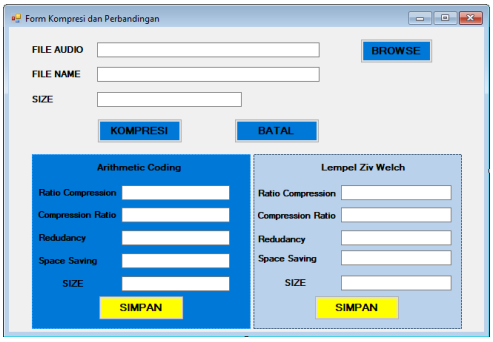
Tampilan *Input*

Tampilan *input* adalah tampilan tentang *form* aplikasi kompresi yang dipakai untuk menujukkan program yang akan dibutuhkan. Adapun tampilan *input* pada aplikasi *Microsoft Visual Basic* 2010 berupa tampilan *form* kompresi dan perbandingan serta tampilan *form* dekompresi.

Tampilan *Form* Kompresi dan Perbandingan

Form kompresi yang digunakan untuk menjalankan proses kompresi

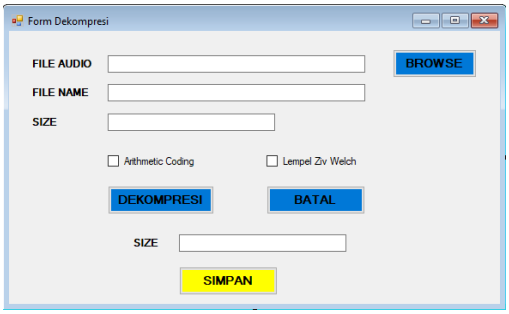
serta melihat perbandingan. Pada *form* kompresi dan perbandingan ini tersedia button yang dapat digunakan untuk memilih *file* audio MP3.



Gambar 4. Tampilan *Input Form* Kompresi dan Perbandingan

Tampilan *Form* Dekompresi

Form dekompresi digunakan untuk menjalankan proses dekompresi. Pada *form* dekompresi ini tersedia button yang digunakan untuk memilih *file* audio MP3 yang akan dikompresi.



Gambar 6 Tampilan *Input Form* Dekompresi

Hasil Pengujian Sistem

Setelah implementasi program menggunakan Microsoft Visual Basic 2010, tampilan input dan output telah berhasil ditampilkan. Berdasarkan parameter kompresi seperti Ratio Compression, Redundancy, Compression Ratio, dan Space Saving yang dianalisis sebelumnya dengan algoritma Arithmetic Coding dan Lempel Ziv Welch, diperoleh perbandingan kinerja kedua algoritma. Perbandingan ini dapat dilihat dalam tabel di bawah ini:

Tabel 8 Perbandingan Kinerja Algoritma

No	Nama File	Algoritma	Parameter Kompresi			
			RC	CR	RD	SS
1	Faceless (Album Mix).mp3	*Arithmetic Coding*	2	50 %	50 %	50 %
		Lempel Ziv Welch	1,428	70 %	30 %	30 %

Berdasarkan tabel pengujian sistem pada file audio berformat MP3 menunjukkan perbandingan antara algoritma Arithmetic Coding dan Lempel Ziv Welch. Hasil pengujian mengungkapkan perbedaan kinerja kompresi antara kedua algoritma, yang ditampilkan pada tabel di bawah ini:

Tabel 9 Hasil Pengujian Ukuran Kompresi

No	Nama File	Algoritma	Ukuran Sebelum Dikompresi	Ukuran Setelah Dikompresi
1	Faceless (Album Mix).mp3	*Arithmetic Coding*	65.0 MB	32.5 MB
		Lempel Ziv Welch	65.0 MB	45.5 MB

Berdasarkan hasil pengujian ukuran yang dilakukan, dilihat dari parameter kinerja kompresi dan ukuran *file* yang berhasil dikompresi maka dapat disimpulkan bahwa algoritma *arithmetic coding* lebih efektif dalam mengkompresi *file* audio

berekstensi *mp3. Dilihat dari hasil *space saving* bahwa algoritma *arithmetic coding* lebih optimal dengan penghematan ruang yang lebih besar.

D. Kesimpulan

Berdasarkan hasil penelitian, perbandingan antara algoritma Arithmetic Coding dan Lempel-Ziv-Welch dalam mengompresi file audio MP3 menunjukkan bahwa Arithmetic Coding lebih efektif. File audio berukuran 65,0 MB berhasil dikurangi menjadi 32,5 MB dengan Arithmetic Coding, sementara dengan Lempel-Ziv-Welch, ukuran file menjadi 45,5 MB. Evaluasi kinerja menunjukkan bahwa Arithmetic Coding memiliki Ratio Compression (RC) sebesar 2, Compression Ratio (CR) 50%, dan Space Saving (SS) 50%, sedangkan Lempel-Ziv-Welch memiliki RC sebesar 1,428, CR 70%, dan SS 30%. Ini menunjukkan bahwa Arithmetic Coding lebih unggul dalam penghematan ruang. Meskipun kedua algoritma efektif dalam mengurangi ukuran file, Arithmetic Coding terbukti lebih baik dalam mengurangi ukuran file dengan lebih signifikan dan mempertahankan kualitas audio selama proses dekompresi, seperti yang dilakukan dengan aplikasi Microsoft Visual Basic 2010.

DAFTAR PUSTAKA

Ariska, R. (2021). Penerapan Algoritma Arithmatic Coding Pada Aplikasi Kamus Teknologi Informasi Berbasis Android. *TIN: Terapan Informatika Nusantara*, *2*(7), 407–413.

Arizki, I., Triawan, A., & Zayid, F. (2024). Penerapan Algoritma Lempel Ziv Welch (LZW) Untuk Kompresi Data. *TeknoIS: Jurnal Ilmiah Teknologi Informasi Dan Sains*, *14*(1), 66–73.

Banjarnahor. (2023). Pengantar Teknologi Informasi. In *PUBLISH BUKU UNPRI PRESS ISBN*.

Informasi, J. S., & Situmorang, T. B. (2024). Perancangan Aplikasi Kompresi File MP3 Dengan Menggunakan Algoritma Lempel Ziv Welch (LZW). *Jurnal Sistem Informasi, Teknik Komputer Dan Teknologi Pendidikan (JUSTIKPEN)*, *3*(2), 60–70.

Iqbal, M., Prayogi Anggi, R., & Yunitasari, D. (2022). Analisis Kompresi File Teks Menggunakan Algoritma Lempel Ziv Welch (LZW). *UNNES Journal of Mathematics*, *11*(2), 112–119.

Nurina Prabiantissa, C., Haryo Sulaksono, D., Eka Yuliastuti, G., & Prasetyo Nugroho Institut Teknologi Adhi Tama Surabaya, A. (2023). *SNESTIK Seminar Nasional Teknik Elektro, Sistem Informasi, dan Teknik Informatika Implementasi Algoritma*

Kompresi Lempel-Ziv-Welch pada Data Citra. 321–325.

Octiviani, M. (2020). Kompresi Ayat Pada Aplikasi Buku Ende Menggunakan Algoritma Arithmetic Coding. *Pelita Informatika : Informasi Dan Informatika*, *9*(1), 97–103.

Oktaviani, D., & Suartana, I. M. (2020). Implementasi Kompresi Data dengan Modifikasi Algoritma Lempel-Ziv-Welch (LZW) untuk File Dokumen. *Journal of Informatics and Computer Science (JINACS)*, *1*(03), 128–137. https://doi.org/10.26740/jinacs.v1n03.p128-137

Panggabean, E. S. (2018). DAN ALGORITMA DEFLATE PADA FILE TEKS DENGAN METODE INDEPENDENT SAMPLE T-TEST. *Jurnal Pelita Informatika*, *6*(1), 333–336.

Prades, E. (2024). Implementasi Algoritma Arithmetic Coding Pada Aplikasi Kompresi File PDF. *Jurnal Sains Dan Teknologi Informasi*, *3*(2), 53–60. https://doi.org/10.47065/jussi.v3i2.4884

Sari. (2018). PENERAPAN ALGORITMA LEVENSTEIN PADA APLIKASI KOMPRESI FILE MP3. *KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer)*, *2*(1), 1.

Satyapratama, A., Yunus, M., Studi, P., & Informatika, T. (2023). ANALISIS PERBANDINGAN ALGORITMA LZW DAN HUFFMAN PADA KOMPRESI FILE GAMBAR BMP DAN PNG. *Jurnal Teknologi Informasi*, *1*(2), 69–81.

Setiawan. (2023). Buku Ajar Multimedia. In *PT. Sonpedia Publishing Indonesia.*

Sianturi, F. A. (2018). Kompresi File Citra Digital Dengan Arithmetic Coding. *Jurnal Teknik Informatika*, *03*(1), 45–51.

Syuhada, I. (2022). *Implementasi Algoritma Arithmetic Coding dan Sannon-Fano Pada Kompresi Citra PNG*. *2*(9), 527–532. https://doi.org/10.47065/tin.v2i9.1027