ANALISIS PERBANDINGAN ALGORITMA *LEMPEL ZIV WELCH* DAN *TABOO CODES* DALAM KOMPRESI *FILE* MKV

Faisal Ridho[1], Pristiwanto[2], Saidi Ramadan Siregar[3]
[1,2,3]Universitas Budi Darma, Medan
[1]faisalridho08@gmail.com, [2]4nt0.82@gmail.com,
[3]saidiramadan89@gmail.com

ABSTRAK

Dalam memilih metode kompresi yang paling efisien, penting untuk membandingkan algoritma-algoritma yang ada. Algoritma atau metode yang digunakan untuk kompresi *file* juga beragam, beberapa di antaranya adalah algoritma LZW atau *Lempel Ziv Welch* dan juga algoritma *Taboo Codes.*Ukuran *file* yang besar dapat memberikan berbagai masalah mulai dari segi penyimpanan maupun dari segi pengiriman *file* yang menjadi lebih lama dan membuat biaya menjadi lebih tinggi. Kompresi menjadi hal yang penting, kompresi dapat mengurangi ukuran *file*, algoritma kompresi juga memiliki berbagai jenis, beberapa diantaranya adalah algoritma LZW dan algoritma *Taboo Codes*. Kedua algoritma tersebut merupakan algoritma kompresi *lossless*. *Compression Ratio, Ratio Compression, Redudancy,* dan *space saving*, merupakan kriteria yang akan dipakai untuk menentukan algoritma mana yang lebih efesien dalam mengkompresi *file* MKV, beserta dengan bantuan dari metode Eksponensial Analisis perbandingan ini juga dapat membantu memahami ke-efektifan dari kedua algoritma, sehingga dapat mengetahui kelebihan dan kekurangan dari kedua algoritma tersebut. Setelah memberikan bobot ke masing-masing kriteria berdasarkan metode eksponensial, Maka akan menghasilkan nilai dari kedua algoritma tersebut dimana algoritma *Lempel Ziv Welch* mendapatkan hasil 8,7926, sedangkan untuk algoritma *Taboo Codes* mendapatkan hasil 9,48313. Berdasarkan hasil tersebut maka dapat disimpulkan bahwa algoritma *Lempel Ziv Welch* memiliki usaha yang lebih sedikit dibandingkan algoritma *Taboo Codes*.

Kata Kunci**:** *MKV, taboo codes, lempel ziv welch, lossless*

ABSTRACT

When selecting the most efficient compression method, it is crucial to compare the available algorithms. The methods used for file compression vary, including algorithms such as LZW (Lempel Ziv Welch) and Taboo Codes. Large file sizes can cause various issues, such as increased storage requirements and longer transmission times, leading to higher costs. Compression is essential as it reduces file size, and different compression algorithms, like LZW and Taboo Codes, offer various benefits. Both algorithms are lossless compression techniques. The criteria used to determine which algorithm is more efficient for compressing MKV files include Compression Ratio, Ratio Compression, Redundancy, and Space Saving. The Exponential Method is used to analyze and compare these algorithms to understand their effectiveness, highlighting their respective strengths and weaknesses. After weighting each criterion using the exponential method, Lempel Ziv Welch achieved a score of 8.7926, while Taboo Codes achieved a score of 9.48313. Based on these results, it can be concluded that Lempel Ziv Welch requires less effort compared to Taboo Codes.

Keywords: MKV, taboo codes, lempel ziv welch, lossless

A. Pendahuluan

Dalam memilih metode kompresi yang paling efisien, penting untuk membandingkan berbagai algoritma yang tersedia, seperti *LZW (Lempel Ziv Welch)* dan *Taboo Codes*, mengingat pertumbuhan data yang pesat saat ini menuntut solusi penyimpanan yang lebih efisien (Yang, 1996). Kompresi file menawarkan cara untuk menyimpan dan mengirim data secara lebih efisien, yang dapat mengurangi biaya dan waktu pengiriman (Dzulhaq, 2014). Ada dua metode utama dalam kompresi: lossless, yang mengurangi ukuran file tanpa kehilangan data asli dengan menghapus metadata yang tidak diperlukan, dan lossy, yang menghapus beberapa data secara permanen untuk mengurangi ukuran file (Wibowo, 2021);(Fitrianto., 2021). *Algoritma* seperti LZW, yang menggantikan pola data yang sering muncul dengan kode yang lebih pendek, dan *Taboo Codes*, yang menggunakan pendekatan berbasis panjang variabel untuk menentukan kode akhir, merupakan contoh teknik kompresi lossless yang dapat mengurangi ukuran data secara efisien (Bakara, 2017). Pemilihan metode yang tepat bergantung pada karakteristik data dan kebutuhan aplikasi, dengan kompresi lossy lebih cocok untuk file besar seperti video atau audio yang tidak memerlukan integritas penuh, sementara kompresi lossless lebih sesuai untuk data yang harus dipertahankan keutuhannya (Aruan, 2023). Evaluasi yang cermat terhadap *algoritma-algoritma* ini sangat penting untuk mencapai efisiensi penyimpanan dan pengiriman data yang optimal.

Algoritma kompresi file juga beragam, termasuk LZW dan *Taboo Codes* (Seregar, 2023). Keduanya merupakan teknik kompresi lossless. LZW menggantikan pola yang sering muncul dengan simbol atau kode baru yang lebih pendek, sementara *Taboo Codes* menggunakan pendekatan berbasis panjang variabel untuk menentukan akhir kode. *Algoritma* ini bertujuan mengurangi ukuran data secara efisien dengan pendekatan yang berbeda (Aktavera et al., 2024). Estimasi pengurangan ukuran file MKV, yang biasanya lebih besar dibandingkan format serupa, bervariasi tergantung pada faktor seperti kinerja algoritma dan tingkat kompresi. Secara umum, file MKV dapat mengalami pengurangan

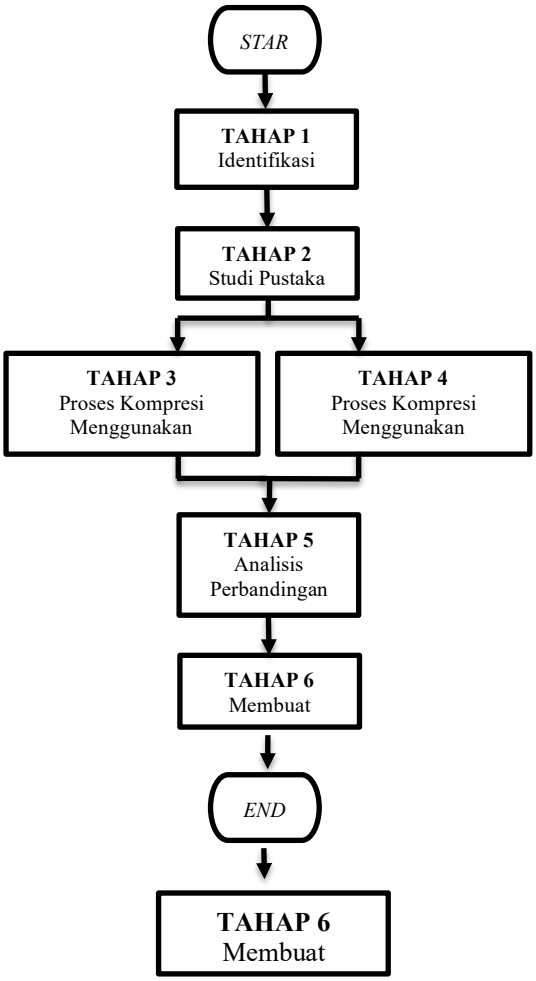
ukuran sekitar 20% hingga 50% dengan metode kompresi yang efektif. Namun, hasilnya sangat tergantung pada karakteristik konten seperti audio atau subtitle (Rasidi, 2023).

Penelitian ini fokus pada perbandingan kinerja antara *algoritma Lempel Ziv Welch* dan Taboo Codes dalam proses kompresi file MKV. Variabel perbandingan yang digunakan meliputi *Ratio Compression (RC), Compression Ratio (CR), dan Space Saving (SS)*. Metode Eksponensial digunakan untuk membandingkan kedua algoritma ini, memberikan pendekatan kuantitatif dalam analisis kinerja. Referensi dari penelitian terdahulu turut digunakan, termasuk studi oleh Siahaan, (2023) yang membandingkan algoritma Fibonacci dengan Invert Elias Delta, serta penelitian Hasibuan, (2022) yang membandingkan algoritma Huffman dengan *Rice Code* untuk file video. Penelitian Setiawan et al., (2019) juga relevan, yang menggunakan Lempel *Ziv Welch* pada file teks dan menghasilkan berbagai nilai perbandingan. Selain itu, penelitian tentang kompresi video menggunakan *Taboo Codes* juga memberikan hasil kompresi yang signifikan. Dengan memperhatikan kompleksitas dalam kompresi file MKV, penelitian ini bertujuan memberikan pemahaman lebih mendalam mengenai perbandingan kinerja antara *algoritma Lempel Ziv Welch dan Taboo Codes*. Melalui analisis parameter-parameter yang relevan, diharapkan penelitian ini dapat memberikan wawasan berharga bagi pengembang, peneliti, dan praktisi dalam industri multimedia.

Dengan demikian Berdasarkan permasalahan di atas, maka penulis ingin memberikan langkah-langkah serta mengusulkan solusi yang dipaparkan dalam penelitian ini melalui judul. “Analisis Perbandingan Algoritma *Lempel Ziv Welch* Dan *Taboo Codes* Dalam Kompresi *File* Mkv”.

B. Metode Penelitian

Kerangka kerja penelitian merupakan suatu konsep sistematis yang mengatur gagasan, teori, metode serta data penelitian. Berikut ini merupakan bentuk dari kerangka penelitian:



Gambar 1 sampel *file* yang akan dikompresi

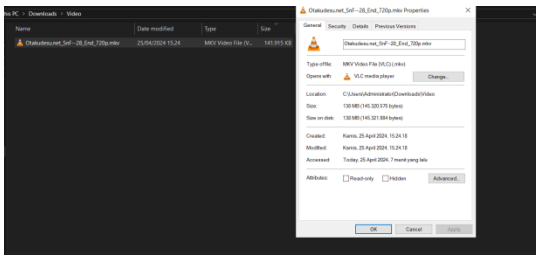
Sampel Data

Adapun sampel data yang diambil berupa data *file* video dengan format MKV, yang nantinya digunakan untuk proses kompresi menggunakan algoritma LZW dan *Taboo Codes*, untuk mendapatkan nilai hexadesimal dari sampel data, digunakannya software bernama HxD, nantinya akan diambil nilai hexadesimal dari *file* video MKV yang digunakan.

Tabel 1 Sampel Data

Nama *File*	*File* Extension	Size
Otakudesu.net_SnF--28_End_720p	MKV	138 MB

Berikut ini merupakan gambar dari sampel video yang digunakan, dapat dilihat pada gambar 2 dibawah:



Gambar 2 sampel *file* yang akan dikompresi

C. Hasil Penelitian dan Pembahasan

Penerapan **Algoritma Lempel Ziv Welch**

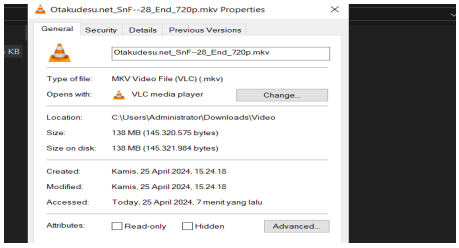
Tahapan pertama adalah proses kompresi, pada tahap ini sampel yang diambil dan dijadikan sebagai contoh perhitungan dapat dilihat ditabel 4.2 nilai *hexadecimal*. Algoritma yang akan diterapkan adalah algoritma *Lempel Ziv Welch* yang merupakan algoritma kompresi *lossless*, dimana ketika didekompresi dapat kembali seperti semula, tanpa harus kehilangan data.

Prinsip umum algoritma ini adalah memeriksa setiap karakter,

kemudian menggabungkannya menjadi sebuah *string* baru, lalu *string* tersebut akan di *index*-kan ke dalam suatu *dictionary*. Adapun urutan penyelesaian kompresi LZW adalah sebagai berikut:

Proses kompresi dimulai dengan menginisialisasi Kamus (Dictionary) menggunakan jumlah bit yang lebih sedikit dibandingkan string asli. Langkah pertama adalah menentukan karakter awal dalam stream string, yang disebut P, dan karakter selanjutnya, yaitu Q. Selanjutnya, perlu diperiksa apakah string gabungan (P+Q) sudah ada dalam Kamus. Jika string (P+Q) ada dalam Kamus, maka P diperbarui menjadi (P+Q), menciptakan string baru yang menggabungkan P dan Q. Jika string (P+Q) tidak ada dalam Kamus, maka langkah-langkah berikut dilakukan: output kode yang menggantikan karakter atau string P, tambahkan string (P+Q) ke dalam Kamus dengan kode baru yang belum digunakan, dan perbarui P menjadi Q. Setelah itu, diperiksa apakah masih ada karakter berikutnya dalam stream karakter. Jika ada, kembali ke langkah 2 untuk melanjutkan proses kompresi. Jika tidak ada karakter lagi, output kode yang menggantikan string P dan terminasi proses.

Berikut ini merupakan sampel *file* video MKV yang akan digunakan beserta nilai *hexadecimal* dari *file* tersebut.



Gambar 4.3 Sampel *file* MKV

Untuk detail dari *file* di atas dapat disederhanakan ditabel 4 di bawah ini:

4.4 Tabel Sampel *file* MKV

No	Nama *File*	Ukuran
1	Otakudesu.net_SnF--28_End_720p.MKV	138 Mb

Diambil 15 nilai *hexadecimal* yang digunakan sebagai sampel, nilai *hexadecimal* tersebut dihasilkan dari *file* MKV menggunakan *software* HxD yang dapat membaca nilai *hexadecimal* dari *file* yang ada. Nilai nya sendiri dapat dilihat pada tabel di bawah ini:

Tabel 4.5 Nilai *Hexadecimal*

1A	45	DF	A3	A3	42	86	81	01	42
F7	81	01	42	F2					

Berdasarkan tabel 4.5, memperoleh nilai bilangan *hexadecimal* yang akan dihitung secara manual, nilai *hexadecimal* diurutkan berdasarkan banyak

kemunculan ke yang paling sedikit muncul, dapat dilihat pada tabel di bawah ini

Tabel 4.6 Nilai Hexadecimal dan Frequency

N	Nilai *Hexadecimal*	Freq
1	42	3
2	01	2
3	81	2
4	A3	2
5	45	1
6	86	1
7	1A	1
8	DF	1
9	F2	1
10	F7	1
Total frekuensi		15

Dari tabel di atas, isi *dictionary* pada awal proses diset hingga 10 karakter dasar yaitu: [“1A, 45, DF, A3, 42, 86, 81, 01, F7, F2”].

Tabel 4.7 Nilai *Hexadecimal* yang sudah dikompresi menggunakan LZW

Langkah	*Input* Karakter	*Temporary* P + Q	In *dictionary*	*Dictionary* baru	*Output*
1	1A	1A, 45	TIDAK	FG	1A
2	45	45, DF	TIDAK	FH	45
3	DF	DF, A3	TIDAK	FI	DF
4	A3	A3, A3	TIDAK	FJ	A3
5	A3	A3,42	TIDAK	FK	A3
6	42	42, 86	TIDAK	FL	42
7	86	86, 81	TIDAK	FM	86
8	81	81, 01	TIDAK	FN	81
9	01	01, 42	TIDAK	FO	01
10	42	42, F7	TIDAK	FP	42
11	F7	F7, 81	TIDAK	FQ	F7
12	81	81,01	YA	-	-
13	01	01, 42	YA	-	-
14	42	42,F2	TIDAK	FR	42
15	F2	F2, NULL	TIDAK	FS	F2

Proses kompresi dimulai dengan menginisialisasi kamus dengan bit yang lebih sedikit dari string asli. Langkah pertama adalah memeriksa nilai 1A, menambahkan 45 ke kamus jika gabungan 1A dan 45 belum ada, dengan output 1A. Langkah berikutnya memasukkan 45, menambahkan DF ke kamus jika gabungan 45 dan DF belum ada,

dengan output 45. Proses ini berlanjut dengan memeriksa nilai DF, A3, 42, 86, 81, 01, dan F7, memperbarui kamus setiap kali gabungan P+Q belum ada, dan menghasilkan output sesuai nilai P. Jika gabungan P+Q sudah ada dalam kamus, tidak ada pembaruan dan output. Terakhir, proses diakhiri dengan menggantikan nilai setelahnya dengan NULL jika tidak ada data lebih lanjut.

Penerapan Algoritma Taboo Codes

Pada tahap ini akan dilakukan pengkompresian sampel data dari *file* MKV dengan menggunakan algoritma *Taboo Codes*. Algoritma *Taboo Codes* sendiri merupakan salah satu kompresi *lossless*. Algoritma ini dikembangkan oleh *Steven Pigeon* Dan menggunakan pendekatan berdasarkan panjangnya variabel. Pada penelitian ini, algoritma *Taboo Codes* menggunakan jenis yang kedua adalah dimana jenis tersebut memiliki panjang total yang bersifat ganda dan tak terbatas. Tipe ini biasanya disebut tak terbatas, serta telah terbukti berhubungan dengan urutan dari bilangan *fibonacci* hingga ke-n. Adapun pola *Taboo Codes* dapat dilihat pada tabel berikut:

Tabel 4.8 Pola *TabooCodes*

M	Code	M	Code	m	Code	M	Code
0	01 00	4	01 10 00	8	10 11 00	12	01 01 01 00
1	10 00	5	01 11 00	9	11 01 00	13	01 01 10 00
2	11 00	6	10 01 00	10	11 10 00	14	01 01 11 00
3	01 01 00	7	10 10 00	11	11 11 00	…	

Setelah mengidentifikasi frekuensi kemunculan setiap karakter dan mendapatkan nilai biner dari masing-masing karakter tersebut, berikutnya adalah menyusun semua karakter kedalam sebuah urutan yang disusun menurut jumlah kemunculan, dimulai dari kemunculan yang paling banyak, diakhiri dengan kemunculan yang paling sedikit. Setelah diurutkan, bilangan biner yang mewakili setiap karakter digantikan dengan kode biner dari pola algoritma *Taboo Codes*. Data sebelum dikompresi dapat dilihat di bawah ini:

Tabel 4.9 Sampel data sebelum di kompresi

N	Nilai *Hexadecimal*	Nilai Biner	Bit	Freq	Bit x Freq
1	42	01000010	8	3	24
2	01	00000001	8	2	16

3	81	10000001	8	2	16
4	A3	10100011	8	2	16
5	45	01000101	8	1	8
6	86	10000110	8	1	8
7	1A	00011010	8	1	8
8	DF	11011111	8	1	8
9	F2	11110010	8	1	8
10	F7	11110111	8	1	8
Total Bit					120

Lalu nilai biner pada sampel data diganti menggunakan pola *Taboo Codes* yang sudah ditentukan sehingga menjadi:

Tabel 4.10 Sampel data setelah diubah ke pola *Taboo Codes*

N	Nilai *Hexadecimal*	*Codeword*	Bit	Freq	Bit x Freq
1	42	01 00	4	3	12
2	01	10 00	4	2	8
3	81	11 00	4	2	8
4	A3	01 01 00	6	2	12
5	45	01 10 00	6	1	6
6	86	01 11 00	6	1	6
7	1A	10 01 00	6	1	6
8	DF	10 10 00	6	1	6
9	F2	10 11 00	6	1	6
10	F7	11 01 00	6	1	6
Total Bit					76

Setelah kode diurutkan berdasarkan pola *Taboo Codes*, langkah berikutnya adalah mengatur ulang urutan bit yang dihasilkan berdasarkan posisi *hexadecimal* pada *file* MKV. Berikut ini merupakan tabel nilai *hexadecimal* beserta pola *Taboo Codes* yang digunakan:

Tabel 4.11 *String* bit hasil pola *Taboo Codes*

1A	**45**	**DF**	**A3**	**A3**
10 01 00	01 10 00	10 10 00	01 01 00	01 01 00
42	**86**	**81**	**01**	**42**
01 00	01 11 00	11 00	10 00	01 00

F7	81	01	42	F2
11 01 00	11 00	10 00	01 00	10 11 00

Apabila tabel 4.10 diurutkan maka akan menjadi representasi sebagai berikut:

10 01 00 **01 00** **11 01 00**	**01 10 00** **01 11 00** **11 00**	**10 10 00** **11 00** **10 00**	**01 01 00** **10 00** **01 00**	**01 01 00** **01 00** **10 11 00**

Kemudian nilai bit tersebut ditambahkan lalu mendapat hasil 76 karena jumlah bit tidak habis dibagi dengan 8, maka perlu ditambahkan dengan nilai bit lagi, dalam komputer setiap karakter direpresentasikan menggunakan ASCII(*American Standard Code for Information Interchange*), yang terdiri dari 8 bit biner, dikarenakan apabila jumlah bit lebih kecil dari pada 8, komputer tidak akan langsung bisa membacanya. Karena hal tersebut juga maka dibentuk dua variabel yang disebut dengan *padding* dan *flagging* yang berfungsi menambahkan bit data. *Padding* adalah penambahan bit yang sudah dikompresi sehingga bit yang kurang tersebut dapat menjadi kelipatan 8. Rumus untuk *padding* sendiri adalah 7-n+"1". Sementara *flagging* adalah penambahan 8 bit bilangan biner setelah *padding*, yang digunakan untuk memudahkan pembacaan bit ketika dekompresi. Formula *flagging* sendiri adalah 9-n. Apabila terjadi jumlah bit dapat dan habis dibagi dengan 8 maka tidak diperlukan *padding*, namun *flagging* tetap ditambahkan, untuk mempermudah dekompresi.

Tabel 4.12 *Padding* dan *flagging*

Padding	*Flagging*
76 mod 8 = 4 = n 7 − 4 + "1" 7 − 4 + "1" = 0001	9 − n 9 − 4 = 5 = 0000 0101

Berdasarkan *padding* dan *flagging* diatas maka didapat nilai bit sebagai berikut: "10010001 10001010 00010100 01010001 00011100 11001000 01001101 00110010 00010010 1100**0001 00000101**", sehingga total bit yang dihasilkan menjadi 88. Langkah berikutnya adalah menyajikan biner diatas kedalam sebuah tabel yang berisi karakter baru setelah dikompresi menggunakan algoritma *Taboo Codes*, nilai tersebut dapat

dilihat dibawah ini:

Tabel 4.13 Hasil Kompresi dengan Algoritma *Taboo Codes*

No	*CodeWord* Hasil Kompresi	*Decimal*	*Hexa Decimal*	Karakter
1	10010001	145	91	‘
2	10001010	138	8A	Š
3	00010100	20	14	
4	01010001	81	51	Q
5	00011100	28	1C	
6	11001000	200	C8	È
7	01001101	77	4D	M
8	00110010	50	32	2
9	00010010	18	12	
10	11000001	193	C1	Á
11	00000101	5	05	

Nilai biner yang didapat diubah menjadi nilai *hexadecimal* yang baru, dan menjadi **“91 8A 14 51 1C C8 4D 32 12 C1 05”.** Untuk mengetahui bagaimana kinerja algoritma maka akan dilakukan pengukuran hasil kompresi berdasarkan parameter yang digunakan. Semakin kecil terkompresi maka semakin baik kinerja kompresi, begitu juga sebaliknya.

Analisa Perbandingan Algoritma *Lempel Ziv Welch* dengan *Taboo Codes*

Perbandingan antara algoritma LZW dan Taboo Codes dilakukan dengan mengevaluasi parameter seperti Ratio Compression (Rc), Compression Ratio (Cr), dan Space Saving (Ss). Hasil perbandingan ini disajikan menggunakan metode eksponensial, seperti yang ditunjukkan dalam tabel. Untuk algoritma Lempel Ziv Welch, nilai total diperoleh sebesar 7,82, sedangkan untuk Taboo Codes, nilai total adalah 8,03. Dari perhitungan tersebut, terlihat bahwa algoritma LZW memerlukan usaha yang lebih sedikit dibandingkan Taboo Codes. Meskipun Taboo Codes menunjukkan hasil yang lebih baik dalam hal efisiensi, algoritma ini kurang efisien dibandingkan dengan LZW dalam mengompresi file berformat MKV.

Implementasi

Implementasi program atau implementasi adalah tahap di mana sistem diterapkan berdasarkan hasil analisis sebelumnya. Pada tahap ini, dibahas mengenai perangkat keras, perangkat lunak, dan tampilan sistem saat berjalan.

Kebutuhan Sistem

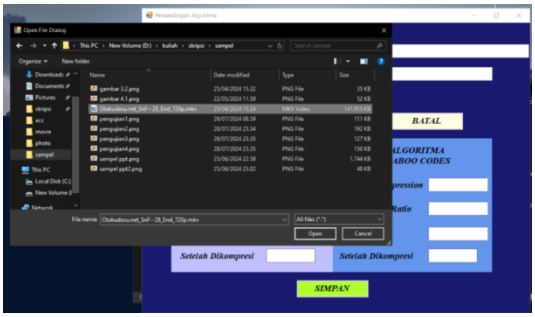
Perincian perangkat yang digunakan, termasuk perangkat lunak dan perangkat keras yang dibutuhkan untuk aplikasi, disebut kebutuhan sistem. Implementasi sistem adalah proses yang direncanakan dengan baik dan dilaksanakan dengan hati-hati melalui prosedur tertentu untuk mencapai tujuan. Keberhasilan implementasi sistem bergantung pada spesifikasi yang digunakan, sehingga diperlukan spesifikasi perangkat keras dan perangkat lunak yang baik. Adapun spesifikasi perangkat yang digunakan adalah:

1. Perangkat Keras *(Hardware)*
   Perangkat keras yang dipakai dalam penelitian ini memakai komputer dengan spesifikasi sebagai berikut:
   a. Model : HP Pavilion Gaming Laptop 15-cx0xxx
   b. *Processor* : Intel(R) Core(TM) i7-8750H
   c. *Memory* : 16 GB RAM
   d. *Storage* : 128 GB
   e. *Keyboard* :
   f. *Mouse* : *Logitech G203*
2. Perangkat Lunak *(Software)*
   Perangkat lunak atau *software*, yang digunakan dengan syarat minimal dapat dikategorikan sebagai berikut:
   a. Sistem Operasi : *Windows* 10
   b. Tipe Sistem : 64-*bit operating system*, x64-*based processor*
   c. Aplikasi : *Microsoft Visual Basic* 2010

Tampilan *Input*

Tampilan *input* adalah tampilan tentang *form* aplikasi kompresi yang dipakai untuk menujukkan program yang akan dibutuhkan.

Gambar 4.4 Tampilan *Input*

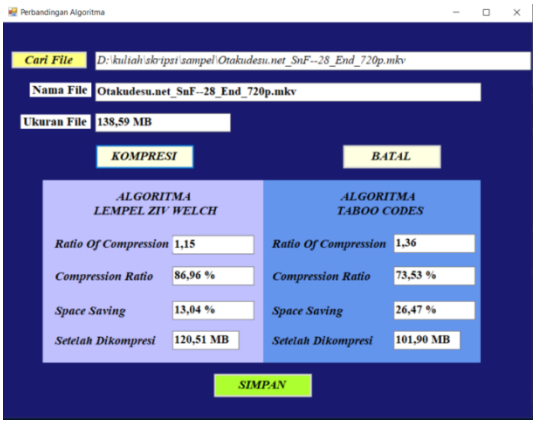


Gambar 4.5 Tampilan *Input File*

Tampilan Output

Tampilan *output* merupakan tampilan proses *file* aplikasi kompresi yang dipakai untuk membandingkan algoritma *Lempel Ziv Welch* dan Algoritma *Taboo Codes* yang menampilkan hasil ukuran setelah dikompresi.

Gambar 4.6 Tampilan *Output*



Gambar 4.7 Tampilan Output perbandingan

Hasil Pengujian Sistem

Setelah implementasi program di atas menggunakan aplikasi *Microsoft Visual Basic* 2010, yang menunjukkan tampilan *input* dan *output*, berdasarkan parameter kompresi seperti R*atio Compression, Compression Ratio*, dan *Space Saving* yang dilakukan sebelumnya dengan menggunakan algoritma *lempel ziv welch* dan *Taboo Codes*, diperoleh perbandingan kinerja algoritma yang dapat dilihat pada tabel di bawah ini:

Tabel 4.15 Perbandingan Kinerja Algoritma

No	Nama *File*	Algoritma	Parameter Kompresi		
			RC	CR	SS
1	Otakudesu.net_SnF--28_End_720p.mkv	*Lempel Ziv Welch*	1,15	86,96 %	13,04 %
		Taboo Codes	1,36	73,53 %	26,47 %

Berdasarkan tabel di atas, pengujian sistem dilakukan terhadap sampel data video video berformat MKV yang digunakan dalam penelitian ini. Tabel di bawah ini merupakan ukuran sebelum dan

sesudah dikompresi menggunakan algoritma *Lempel Ziv Welch* dan *Taboo Codes.*

Tabel 4.16 Hasil Pengujian Ukuran Kompresi

No	Nama *File*	Algoritma	Ukuran Sebelum Dikompresi	Ukuran Setelah Dikompresi
1	Otakudesu.net_SnF--28_End_720p.mkv	*Lempel Ziv Welch*	138,59 MB	120,51 MB
		Taboo Codes	138,59 MB	101,90 MB

Berdasarkan hasil pengujian ukuran yang dilakukan, dilihat dari parameter kinerja kompresi dan ukuran *file* yang berhasil dikompresi, dapat disimpulkan bahwa algoritma *Taboo Codes* lebih efektif dalam mengkompresi *file* Video dengan ekstensi MKV. Berdasarkan hasil *space saving*, algoritma *Taboo Codes* lebih optimal dengan penghematan ruang yang lebih besar.

D. Kesimpulan

Berdasarkan hasil penelitian dan pembahasan, perbandingan antara *algoritma Lempel Ziv Welch dan Taboo Codes* dalam mengompresi file video berformat MKV menghasilkan beberapa kesimpulan utama. *Algoritma modern, seperti Taboo Codes*, umumnya lebih efektif dalam kompresi dibandingkan algoritma klasik seperti *Lempel Ziv Welch*, meskipun dalam beberapa kasus algoritma klasik dapat menunjukkan kinerja yang lebih baik. Proses kompresi menggunakan algoritma Lempel *Ziv Welch* mengurangi ukuran file dari 138,59 MB menjadi 120,51 MB, sedangkan dengan *Taboo Codes*, ukuran file berkurang menjadi 101,90 MB. Evaluasi kinerja menunjukkan bahwa *Taboo Codes* memiliki *Ratio Compression (RC)* lebih tinggi (1,36) dan Space Saving (SS) lebih besar (26,47%) dibandingkan dengan *Lempel Ziv Welch*, yang memiliki RC sebesar 1,15 dan SS sebesar 13,04%. Dengan demikian, *Taboo Codes* terbukti lebih efektif dalam hal *space saving* untuk file MKV, meskipun kedua algoritma berhasil mengurangi ukuran file video secara signifikan.

DAFTAR PUSTAKA

Aktavera, B., Oktafia, H., Wijaya, L., Studi, P., Komputer, I., Petulai,

U. P., Lebong, R., Studi, P., Informasi, S., Insan, U. B., Apriori, A., & Apriori, A. (2024). ANALISIS ASSOCIATION RULE MENGGUNAKAN ALGORITMA. *Jurnal Teknologi Informasi Mura*, *16*(1), 54–61.

Aruan, M. C. (2023). Analisis Performa Algoritma Kompresi Data dalam Penyimpanan dan Transfer Data. *Beranda Jurnal*, *2*(1).

Bakara, J. (2017). IMPLEMENTASI ALGORITMA LZW DAN KUANTISASI. *Pelita Informatika Budi Darma*, *1*(1), 42–45.

Dzulhaq, M. I. (2014). Aplikasi Kompresi File Dengan Metode. *JURNAL SISFOTEK GLOBAL*, *4*(1), 1–4.

Fitrianto., Y. (2021). Dasar-Dasar Digital Imaging. In *Penerbit Yayasan Prima Agus Teknik, 7(1), 1-80. Retrieved from*.

Hasibuan, N. A. (2022). Analisa Perbandingan Algoritma HuffmanDengan Rice Code Dalam Kompresi File Video. *Konferensi Nasional Teknologi Informasi Dan Komputer*, *6*(1), 159–166.

Rasidi, M. (2023). Perbandingan Algoritma Stout Code Dan Punctured Elias Code Dalam Mengkomprsi File Matroska Video Container. *Bulletin of Information System Research (BIOS)*, *2*(1), 10–20.

Seregar, N. (2023). Penerapan Algoritma Lzy Untuk Mengkompresi Record Database Mysql. *Jurnal Ilmu Komputer, Teknologi Dan Informasi*, *1*(2), 43–50.

Setiawan, M. C., Yuliastuti, G. E., & Rachman, A. (2019). Implementasi Metode Lempel-Ziv-Welch pada Kompresi File Teks. *Seminar Nasional Sains Dan Teknologi Terapan*, *4*(1).

Siahaan, S. (2023). Perbandingan Kompresi File Audio Menggunakan Algoritma Fibonacci Dan Algoritma Invert Elias Delta. *JURNAL MEDIA INFORMATIKA [JUMIN]*, *5*(1), 32–44.

Wibowo. (2021). TEKNOLOGI GAMBAR DIGITAL. In *Penerbit Yayasan Prima Agus Teknik,*.

Yang. (1996). Simple universal lossy data compression schemes derived from the Lempel-Ziv algorithm. *IEEE Transactions on Information Theory*, *42*(1), 1996.