

ANALISA PERBANDINGAN ALGORITMA PUNCTURED ELIAS CODE DAN TABOO CODE PADA KOMPRESI FILE DOKUMEN

Fitri Sarmaito Ritonga¹, Abdul Sani Sembiring²,
Sumiaty Adelina Hutabarat³

^{1,2,3}Universitas Budi Darma, Medan

¹fitrisarmaitoritonga@gmail.com, ²gurkiy@gmail.com,

³sumiatyadelina@gmail.com

ABSTRAK

Kompresi *file* dokumen merupakan salah satu teknik penting dalam pengiriman data untuk mengurangi ukuran *file* tanpa kehilangan informasi yang relevan. Ukuran *file* yang besar membutuhkan banyak ruang penyimpanan. *File* dokumen memiliki ukuran yang besar dibanding dengan jenis teks yang berekstensi lain. Dikarenakan ukuran berekstensi docx terlalu besar mengakibatkan proses pengiriman data memerlukan waktu yang cukup lama. Salah satu teknik yang diperlukan untuk masalah ini adalah melakukan kompresi. Kompresi adalah proses pengubahan ukuran data ke ukuran yang lebih kecil. Algoritma yang digunakan pada penelitian ini adalah algoritma *Punctured Elias Code* dan algoritma *Taboo Code* yang memiliki kelebihan dan kekurangan masing-masing. Setelah kedua algoritma dikompresi, kemudian dilakukan perbandingan hasil dari kedua algoritma tersebut. Hasil akhir dari proses perbandingan menunjukkan bahwa algoritma *Taboo Code* tersebut menjadi algoritma yang tercepat dan efektif dalam melakukan proses kompresi *file* dokumen, hal ini dikarenakan semakin kecil total nilai yang diperoleh maka semakin sedikit jumlah usaha yang dilakukan oleh algoritma tersebut dalam melakukan kompresi.

Kata Kunci : *perbandingan , kompresi, file, docx, algoritma punctured elias code, taboo code, dekompresi*

ABSTRACT

File document compression is a crucial technique for data transmission to reduce file size without losing relevant information. Large file sizes require substantial storage space. Document files are generally larger compared to other text file formats. Due to the large size of docx files, data transmission can be time-consuming. Compression is a technique used to address this issue by reducing the size of data. This study utilizes the Punctured Elias Code and Taboo Code algorithms, each with its own strengths and weaknesses. After compressing with both algorithms, a comparison was performed. The final results indicate that the Taboo Code algorithm is the fastest and most effective for compressing document files, as a smaller total value indicates less effort required by the algorithm in the compression process.

Keywords: comparison, compression, file, docx, punctured elias code algorithm, taboo code, decompression

A. Pendahuluan

Di era perkembangan teknologi saat ini, kemajuan komputer telah membuat pertukaran informasi menjadi jauh lebih mudah dan efisien. Teknologi komputer yang terus berkembang memungkinkan berbagai aktivitas dan pekerjaan menjadi lebih terintegrasi dan terautomasi (Saputra & Jakarta, 2023);(Aksenta, 2023). Misalnya, dalam penyimpanan dan pengolahan dokumen, teknologi canggih memungkinkan pengguna untuk menyimpan data dalam format digital yang mudah diakses dan dikelola. Selain itu, perangkat lunak kompresi dan enkripsi data meningkatkan keamanan dan efisiensi transfer file, menjadikannya lebih mudah untuk berbagi informasi secara aman dan efektif di seluruh dunia. Teknologi ini tidak hanya meningkatkan produktivitas tetapi juga mendukung berbagai industri, dari pendidikan hingga bisnis dan pemerintahan, dengan menyediakan solusi yang lebih baik untuk mengelola dan memanfaatkan data (Saputra, 2023);(Harto, 2023). Format file seperti .docx sangat populer karena kemudahan dan kecepatan aksesnya. Format ini sering digunakan di berbagai sektor,

dari perusahaan dan pendidikan hingga lembaga pemerintahan, karena praktis dan efisien dalam mengelola data (Pradana, 2022).

Kebutuhan untuk mentransmisikan file dokumen yang berukuran besar sering kali menghadapi kendala karena banyak media transmisi membatasi ukuran dokumen. Ukuran data yang besar tidak hanya menghabiskan memori penyimpanan, tetapi juga memperlambat proses transfer dan dapat menyebabkan ruang penyimpanan menjadi penuh, yang berpotensi mengakibatkan penghapusan data penting (Asari, 2023). Oleh karena itu, teknik kompresi diperlukan untuk mengurangi ukuran file. Dalam penelitian ini, digunakan *algorithm Punctured Elias Code* dan *Taboo Code* untuk kompresi, dengan masing-masing algoritma menghasilkan nilai kompresi yang berbeda. Hasil kompresi yang lebih kecil mempermudah transmisi file dokumen, dan penelitian ini membantu menentukan algoritma mana yang paling efektif untuk kompresi file dokumen.

Punctured Elias Code adalah metode yang dirancang oleh Peter

Fenwick untuk meningkatkan kinerja transformasi *Burrows-Wheeler lossless* (Ompusunggu, 2022). Istilah "perforasi" diambil dari konsep *Error Control Code* (ECC), di mana ECC terdiri dari data asli ditambah beberapa bit cek. Untuk mempersingkat urutan kode, beberapa bit cek dihilangkan, dan kode yang dihasilkan diatur dengan cara berikut: pertama, bit cek dihilangkan dan kode yang dihasilkan dibalik (*reversed*) serta diberi flag untuk menunjukkan jumlah bit bernilai 1 dalam n_3 . Selanjutnya, untuk setiap bit 1 dalam n , flag disiapkan untuk menunjukkan jumlah bit bernilai 1 dalam n_1 . Flag kemudian dikaitkan dengan bilangan biner terbalik. Kode yang dihasilkan, yang dikenal sebagai kode P1, dimulai dan diakhiri dengan 1, dengan setidaknya satu flag, kecuali pada kasus di mana $n = 0$. (Depika & Nasution, 2020).

Algoritma *Taboo Code* adalah algoritma yang mencoba untuk menghasilkan kode yang optimal dengan mempertimbangkan pembatasan tertentu untuk mencegah kemungkinan jalan buntu (Rhamadani, 2022);(Yuni & Hutagalung, 2022). Prinsip dari algoritma *Taboo Code* yaitu untuk

memilih bilangan positif n dan membalikkan pola n untuk mengindikasikan akhir dari kode tersebut (Simanjuntak, 2022);(Yuni & Hutagalung, 2022). Pada penelitian ini dilakukan perbandingan dua algoritma untuk menentukan mana yang paling efektif yang dapat digunakan dalam proses kompresi dokumen. Algoritma yang akan diuji adalah algoritma *Punctured Elias Code* dan algoritma *Taboo Code*, keduanya merupakan algoritma yang akan digunakan dalam proses kompresi, dan variable perbandingan untuk kedua algoritma tersebut adalah (R_c) *Ratio Compression*, (C_r) *Compression Ratio*, (R_d) *Radudancy*, (S_s) *Space Saving*.

Berdasarkan penelitian terdahulu dilakukan oleh Dwi Asdini pada tahun 2022 yang berjudul "Analisis Perbandingan Kinerja Algoritma Huffman Dan Algoritma Levenstein Dalam Kompresi File Dokumen Format .RTF"(Asdini & Utomo, 2022). Disimpulkan bahwa prosedur kompresi file dokumen menggunakan *Algoritma Huffman* dan *Algoritma Levenstein* berhasil diterapkan pada format .RTF dan berjalan sesuai dengan teknik kompresi yang ditetapkan. Hasil pengujian

menunjukkan bahwa Algoritma Huffman lebih efektif dibandingkan *Algoritma Levenstein* dalam mengompresi file dokumen .RTF, karena menghasilkan ukuran file kompresi yang lebih kecil dan space saving yang lebih tinggi.

Penelitian lainnya yang dilakukan oleh Vindi Aulia Sari pada tahun 2023 yang berjudul “Perbandingan Algoritma *Punctured Elias Code* Dan *Stout Code* Untuk Kompresi *File Pdf*”(Sari et al., 2023). Disimpulkan bahwa kompresi file PDF dapat dilakukan dengan mengonversi file ke dalam format heksadesimal dan kemudian memprosesnya menggunakan algoritma *Stout Code* dan *Punctured Elias Code* untuk mengubah ukuran file. Setelah perhitungan dilakukan, kedua algoritma dapat dibandingkan berdasarkan rasio kompresi dan space saving. Hasil perbandingan menunjukkan bahwa algoritma *Punctured Elias Code* menghasilkan rasio kompresi yang lebih tinggi dibandingkan algoritma *Stout Code*. Nilai rasio kompresi untuk algoritma *Stout Code* adalah 56,25%, sedangkan untuk algoritma *Punctured Elias Code* adalah 75%. Penelitian lainnya yang dilakukan

oleh Puja Samba Hasmita dkk pada tahun 2021 yang berjudul “Implementasi Algoritma *Punctured Elias Code* Untuk Kompresi *File Audio* Pada Aplikasi Lagu Rohani”(Hasmita & Sianturi, 2021). Kesimpulannya, proses kompresi file audio lagu rohani menggunakan algoritma *Punctured Elias Code* dapat dilakukan dengan mengonversi file ke format heksadesimal untuk perhitungan. Konversi ini menghasilkan nilai heksadesimal baru yang memungkinkan pengubahan ukuran file audio. Aplikasi yang menggunakan metode ini terbukti efektif dalam memampatkan file audio, menghasilkan laporan rata-rata yang menunjukkan keberhasilan kompresi sesuai harapan. Penelitian lainnya dilakukan oleh Saiful Simanjuntak pada tahun 2021 yang berjudul “Implementasi Metode *Taboo Code* Untuk Kompresi *File Video*”(Simanjuntak, 2022). Kesimpulannya hasil dari kompresi dari sampel data yang digunakan dengan parameter analisis kompresi *Compression Ratio* dan *space saving* dengan hasil *Compression Ratio* sebesar 58,33 % dan hasil *Space Saving* sebesar 0.1467%

menunjukkan tingkat hasil kompresi yang sangat baik pada *file* video dan sangat baik apabila diterapkan.

Dari permasalahan yang telah diuraikan di atas, maka penulis akan melakukan penelitian yang diharapkan mampu menjadi solusi dari permasalahan yang terjadi tersebut, untuk itu penulis mengangkat sebuah judul penelitian yaitu “ **Analisa Perbandingan Algoritma Punctured Elias Code Dan Taboo Code Pada Kompresi File Dokumen**”.

B. Metode Penelitian

Kerangka Kerja Penelitian

Terdapat beberapa tahapan dalam metodologi penelitian, penulis akan menganalisa terlebih dahulu sebelum membuat kerangka penelitian, tahapan ini untuk mempermudah melakukan penelitian. Dapat diuraikan pembahasan dari masing-masing tahapannya yaitu Proses penelitian ini dimulai dengan identifikasi masalah, di mana masalah yang dihadapi diidentifikasi dan dicari solusi yang tepat untuk mengatasinya, bertujuan untuk memahami permasalahan dengan jelas agar dapat memilih metode yang sesuai. Selanjutnya, pada studi

pustaka, dilakukan pemahaman mendalam tentang objek penelitian melalui berbagai referensi terkini, seperti buku dan jurnal. Kemudian, penerapan algoritma Punctured Elias Code dan Taboo Code dilakukan untuk kompresi data lossless, di mana data dapat didekompresi kembali ke bentuk aslinya. Pada tahap analisa perbandingan, proses perbandingan antara kedua algoritma dilakukan untuk menentukan mana yang lebih efektif dalam memperkecil ukuran data. Perancangan sistem kemudian dilakukan untuk menggambarkan sistem yang akan dibangun, menggunakan Visual Studio 2010 untuk implementasi. Tahap pengujian bertujuan untuk memastikan bahwa sistem berfungsi dengan baik, diikuti oleh implementasi yang merupakan tahap uji coba sistem untuk memastikan hasil sesuai dengan desain. Akhirnya, dokumentasi dibuat untuk mencatat dan menjelaskan rancangan serta hasil penelitian dalam laporan, memudahkan pengembangan aplikasi lebih lanjut.

Sampel Data

Sampel data merupakan kumpulan dari beberapa data yang digunakan dalam penelitian, Sampel

data yang digunakan dalam penelitian ini adalah berupa, *file .docx*. Sampel data yang digunakan memiliki ukuran 300KB – 8MB. Adapun sampel data tersebut dapat dilihat pada tabel 3.1 berikut ini;

Tabel 1 Sampel Data

Nama <i>File</i>	Ukuran	Type
Skripsi Data	300KB – 8MB	.docx

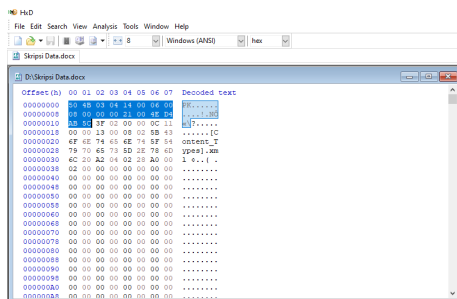
C. Hasil Penelitian dan Pembahasan

Perhitungan Ukuran Awal Sample Data

File dokumen yang akan dikompres terlebih dahulu diambil nilai *hexadecimal* menggunakan aplikasi HxD. Sebagai sampel data diambil 20 nilai dari *file* dokumen.

Tabel 2 Sampel Data

Nama File	Skripsi Data
Size	5,96MB
Type	.docx



Gambar 2 Sampel Data Menggunakan Aplikasi HxD.

Berdasarkan nilai diatas dapat dilihat nilai *hexadecimal*, penulis mengambil 18 krakter nilai *hexadecimal* untuk proses kompresi *file* dokumen.

Tabel 3 Tampilan Nilai Bilangan

Hexadecimal

5	4	0	0	1	00	06	00	08
0	B	3	4	4				
0	00	0	2	0	4	D	A	5
0		0	1	0	E	4	B	C

Maka dari tabel diatas menghasilkan nilai *hexadecimal file* dokumen dan dilakukan perhitungan yang di urutkan berdasarkan frekuensi, nilai yang paling sering muncul akan ditempatkan pada urutan awal.

Tabel 4. Nilai Hexadisamal Sebelum Dikompresi

N o	Nilai Hexadesi mal	Nilai Biner	Bi t	Fre k	Bit X Fre k
1	00	000000 00	8	6	48
2	50	010100 00	8	1	8
3	4B	010010 11	8	1	8
4	03	000000 11	8	1	8
5	04	000001 00	8	1	8

Lanjutan Tabel 4.3

6	14	000101 00	8	1	8
7	06	000001 10	8	1	8
8	08	000010 00	8	1	8
9	21	001000 01	8	1	8
1	4E	010011	8	1	8

0		10			
1	D4	110101	8	1	8
1		00			
1	AB	101010	8	1	8
2		11			
1	5C	010111	8	1	8
3		00			
		Total			144

9	21	100001	6	1	6
1	4E	110100	7	1	7
0		1			
1	D4	110010	7	1	7
1		1			
1	AB	111011	8	1	8
2		01			
1	5C	110001	7	1	7
3		1			
		Total			77

Penerapan Algoritma *Punctured*

Elias Code

Dalam penelitian ini, akan dibahas dua tahap utama dalam proses kompresi dan dekompresi dokumen. Penulis akan melakukan kompresi dokumen dengan menerapkan algoritma *punctured elias code* dan algoritma *taboo code* yang merupakan teknik *loolless*. Adapun data awal yang dikonversi menjadi nilai *hexadesimal* akan digunakan sebagai sampel data untuk perhitungan. Berikut adalah langkah langkah untuk kompresi dan dekompresi dokumen menggunakan algoritma *punctured elias code*.

1. Proses Kompresi *File* Dokumen

Tabel 5 Kompresi Algoritma
Punctured Elias Code

N o	Nilai Hexadesi mal	Punctur ed Elias Code P(1)	Bi t	Fre k	Bit X Fre k
1	00	0	1	6	6
2	50	101	3	1	3
3	4B	1001	4	1	4
4	03	11011	5	1	5

Lanjutan Tabel 4.4

5	04	10001	5	1	5
6	14	110101	6	1	6
7	06	110011	6	1	6
8	08	111011 1	7	1	7

Langkah selanjutnya yaitu menyusun kembali string bit yang sudah menghasilkan proses kompresi sesuai dengan posisi pada bilangan *hexadecimal*.

Tabel 6 String Bit *Punctured Elias*
Code

50	101
4B	1001
03	11011
04	10001
14	110101
00	0
06	110011
00	0
08	1110111
00	0
00	0
00	0
21	100001
00	0
4E	1101001
D4	1100101
AB	11101101
5C	1100011

Hasil dari string bit diatas adalah proses kompresi *file* dokumen dengan menggunakan Algoritma *Punctured Elias Code* dapat menghasilkan sebagai berikut.

Tabel 7 Hasil Kompresi *String Bit*

101	1001	1101 1	1000 1	1101 01
0	11001 1	0	1110 111	0
0	0	1000 01	0	1101 001
1100 101	11101 101	1100 011		

Jumlah bit frekuensi hasil (77), 77 tidak bisa dibagi 8, sehingga jumlah bit harus dijumlahkan. Komputer menggunakan ASCII, angka biner 8 bit yang mewakili krakter. Jiika data kurang dari 8 bit, biasanya tidak terbaca oleh computer sebelum proses *encoding*

berlangsung. Proses *encoding* mengkodekan bit data, setiap 8-bit untuk membentuk krakter yang dapat dibaca computer.

Karna 77 tidak habis dibagi 8 atau kelipatan 8, variable yang disebut *padding* dan *Flagging* yang dibuat untuk menambahkan bit data. *Padding* menambahkan bit data kompresi, sehingga jumlah bit data adalah kelipatan 8. Untuk *padding* adalah $7-n + "1"$. Sedangkan *flagging* ini menambahkan angka biner 8-bit setelah *Padding*. Ini untuk membuat bit terkompresi lebih mudah dibaca selama proses dekompresi. Rumus *flagging* adalah $9-n$.

Jika jumlah bit habis dibagi 8 atau kelipatan 8, tidak diperlukan *padding*, tetapi *flagging* harus ditambahkan.

Tabel 8 Penambahan *Padding* dan *Flagging*

<i>Padding</i>	<i>Flagging</i>
$77 \bmod 8 = 5 = n$	$9 - 5$
$7-n + "2"$	$9 - 5 = 4$
$7-5 + 2 = 010$	Bin 4 = 00000100

Maka *string bit* yang terbentuk setelah penambahan *padding* dan *flagging* ialah sebagai berikut:
 "10110011 10111000 11101010
 11001101 11011100 01000010
 11010011 10010111 10110111
 00011010 00000100 ". Sehingga jumlah panjang bit yaitu bit, dan

tahap selanjutnya akan melakukan pemisahan bit menjadi beberapa kelompok. pada setiap kelompok akan terdiri dari 8 bit.

Tabel 9 Pengelompokan Bit

1011 0011	1011 1000	1110 1010	1100 1101	1101 1100	0100 0010
1101 0011	1001 0111	1011 0111	0001 1010	0000 0100	

Berdasarkan hasil pemecahan bit diatas , nilai biner terkompresi baru dan nilai biner tambahan membentuk 11 kelompok. Langkah selanjutnya adalah mengubah nilai

biner menjadi nilai desimal kemudian mencocokkan karakter yang dihasilkan dengan kode ASCII, nilai *hexadesimal* ditunjukkan pada tabel dibawah ini.

Tabel 10 Hasil Karakter Terkompresi

No	Biner	Desimal	Karakter
1	10110011	179	³
2	10111000	184	
3	11101010	234	È
4	11001101	205	Í
5	11011100	221	Ý
6	01000010	66	B
7	11010011	211	Ý
8	10010111	151	—
9	10110111	183	¶
10	00011010	26	
11	00000100	4	

Untuk mengetahui cara kerja algoritma kompresi, penulis menggunakan parameter tertentu untuk mengukur hasil kompresi *file* dokumen. semakin kecil *file* terkompresi yang dihasilkan, semakin baik kinerja kompresi dan sebaliknya.

Penerapan Algoritma Taboo Code

Dalam penelitian ini ,tahap utama pengkompresian teks akan dianalisis, yaitu kompresi dan dekompresi, penulis akan menjelaskan proses kompresi *file* dokumen menggunakan Algoritma *Taboo Code*, yang merupakan salah satu teknik kompresi *lossless*.

1. Proses Kompresi File Dokumen

Untuk tipe pertama kasus jika $n = 1$ tidak dapat dilakukan. Sebuah blok 1-bit seperti 0 atau 1, jika kita cadangkan 0 sebagai pola taboo, maka semua blok lain dari kode tidak dapat berisi angka 0 dan harus semua 1, hasilnya adalah himpunan tak terbatas seperti code berikut 10, 110, 1110, 11110,

Tipe pertama dapat dilakukan jika $n=2$, maka blok 2-bit seperti 00 atau 11 maka dapat dibentuk pola taboo yang bisa dibuat tidak seperti blok taboo tersebut sehingga terbentuk pola seperti berikut.

Tabel 11 Kode Pola *Taboo Code*

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

Setelah mengidentifikasi frekuensi kemunculan setiap krakter dan mendapatkan nilai biner dari setiap krakter tersebut, langkah berikutnya adalah menyusun krakter krakter tersebut dalam urutan berdasarkan jumlah kemunculan, dimulai dari yang memiliki jumlah kemunculan tertinggi. setelah diurutkan berdasarkan frekuensi tertinggi, bilangan biner yang mewakili setiap krakter digantikan dengan kode biner dari algoritma *taboo code* yang telah diurutkan berdasarkan kemunculannya. setelah mengurutkan bilangan hexadesimal berdasarkan frekuensi kemunculan dan mendapatkan nilai biner, langkah berikutnya adalah melakukan perhitungan bit dan mengurutkan kode *taboo code*. detail proses kompresi dokumen dapat ditemukan dalam tabel dibawah ini:

Tabel 12 Tabel Karakter Algoritma *Taboo Code*

N o	Nilai Hexadesi mal	Nilai Biner	Bi t	Fre k	Bit X Fre k
1	00	0100	4	6	24
2	50	1000	4	1	4
3	4B	1100	4	1	4
4	03	010100	6	1	6
5	04	011000	6	1	6

Lanjutan Tabel

6	14	011100	6	1	6
7	06	100100	6	1	6
8	08	101000	6	1	6
9	21	101100	6	1	6
10	4E	110100	6	1	6
11	D4	111000	6	1	6
12	AB	111100	6	1	6
13	5C	010101 00	8	1	8
		Total			94

Setelah kode diurutkan berdasarkan perhitungan *taboo code*, langkah berikutnya adalah mengatur lang urutan bit yang dihasilkan dari kompresi berdasarkan posisi dalam bilangan *hexadesimal*.

Tabel 13 String Bit Hasil Kompresi
 Algoritma *Taboo Code*.

50	1000
4B	1100
03	010100

04	011000
14	011100
00	0100
06	100100
00	0100
08	101000
00	0100
00	0100
00	0100
21	101100
00	0100
4E	110100
D4	111000
AB	111100
5C	01010100

Pada tabel 14 urutan bit yang dihasilkan kompresi dengan menggunakan Algoritma *Taboo Code* dapat dipersentasikan sebagai berikut:

100	11	010	011	011	010
0	00	100	000	100	0
100	01	101	010	010	010
100	00	000	0	0	0
101	01	110	111	111	010
100	00	100	000	100	101
					00

Kemudian, dari hasil jumlah bit x frekuensi didapat nilai 94, tidak perlu ditambahkan karna 94 dibagi 8 dalam konteks computer, setiap krakter dipersentasekan menggunakan ASCII (*American Standard Code For Information Interchange*), yang terdiri dari 8 bit biner, apabila jumlah bit data lebih pendek dari 8 bit, biasanya computer tidak bisa membacanya secara

langsung sebelum enskripsi dilakukan. Selama proses enskripsi, bit data akan dikodekan secara berurutan setiap 8 bit untuk membentuk krakter yang dapat dibaca oleh computer, karna jumlah bit data (94) tidak dapat dibagi 8 dengan sempurna oleh 8 atau bukan merupakan kelipatan dari 8, maka terbentuk 2 variabel yang disebut *padding* dan *flagging* untuk menambahkan bit data, *padding* adalah penambahan bit data yang telah dokompresi sehingga jumlah total bit data tersebut menjadi kelipatan 8, formula untuk *padding* adalah $7 - n + "1"$, sedangkan *flagging* adalah penambahan 8 bit bilangan biner setelah *padding*, yang digunakan untuk memudahkan pembacaan bit-bit hasil kompresi selama proses dekompresi, formula untuk *flagging* adalah $9-n$. apabila jumlah bit dapat dibagi dengan 8 atau merupakan kelipatan dari 8, maka tidak diperlukan *padding*, namun tetap diperlukan *flagging* tambahan.

Tabel 15 Penambahan *Padding* Dan *Flagging* Algoritma *Taboo Code*

<i>Padding</i>	<i>Flagging</i>
$94 \bmod 8 = 6 =$ n $7 - n + "1"$ $7 - 6 + 1 = 01$	$9 - n$ $9 - 6 = 3 =$ 00000011

Dengan demikian, urutan bit yang dihasilkan setelah penambahan *padding* dan *flagging* adalah sebagai berikut

“10001100 01010001
 10000111 00010010 01000100
 10100001 00010001 00101100
 01001101 00111000 11110001

01010001 00000011” sehingga panjang total bit adalah 104. langkah selanjutnya adalah membagi urutan bit menjadi beberapa kelompok, dengan setiap kelompok terdiri dari 8 bit.

Tabel 16 Pengelompokan Bit Algoritma *Taboo Code*

100011 00	0101000 1	1000011 1	0001001 0	0100010 0	101000 01	000100 01
001011 00	0100110 1	0011100 0	1111000 1	0101000 1	000000 11	

Dari hasil pemecahan bit yang disajikan di atas, terbentuk 13 kelompok yang memiliki nilai biner terkompresi baru beserta nilai biner tambahan. Selanjutnya, langkah berikutnya adalah mengonversi nilai

biner menjadi nilai *hexadesimal* untuk mengidentifikasi karakter yang dihasilkan sesuai dengan kode ASCII. data nilai *hexadesimal* yang telah terkompresi dapat ditemukan dibawah ini.

Tabel 17 Dokumen Telah Dikompresi Menggunakan *Taboo Code*

No	Codeword Hasil Kompresi	Decimal	Hexadecimal	Karakter
1	10001100	140	8C	œœœ
2	01010001	81	51	Q
3	10000111	135	87	W
4	00010010	18	12	↕
5	01000100	68	44	D
6	10100001	161	A1	i
7	00010001	17	11	
8	00101100	44	2C	,
9	01001101	77	4D	M
10	00111000	56	38	8
11	11110001	241	F1	Ñ
12	01010001	81	51	Q
13	00000011	3	3	

Analisa Perbandingan Algoritma *Punctured Elias Code* Dan *Taboo Code*

Perbandingan algoritma *punctured elias code* dan *taboo code*

dilakukan dengan membandingkan parameter-parameter seperti (RC) *Ratio compression*, (CR) *Compression Ratio*, (RD) *Redudancy* Dan (ss) *space saving* yang

diterapkan oleh kedua algoritma tersebut, hasil perbandingan ini akan tercermin dalam tabel yang menampilkan nilai dari masing masing kriteria.

Tabel 18 Analisa Hasil Perbandingan Algoritma Kompresi

Algoritma	R c	Cr	Rd	Ss	Ra nk
Algoritma <i>Punctured Elias Code</i>	1,87	53,47 %	46,52 %	46,53 %	2
Algoritma <i>Taboo Code</i>	1,57	65,27 %	34,72 %	34,73 %	1

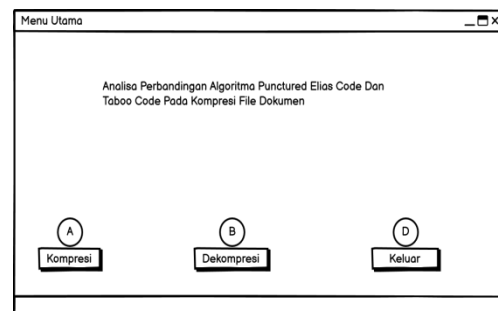
Berdasarkan parameter diatas dan hasil yang telah didapat, maka dapat disimpulkan Algoritma *Taboo Code* lebih baik dalam kompresi *file* dokumen dari pada Algoritma *Punctured Elias Code*.

Perancangan Antarmuka (Interface)

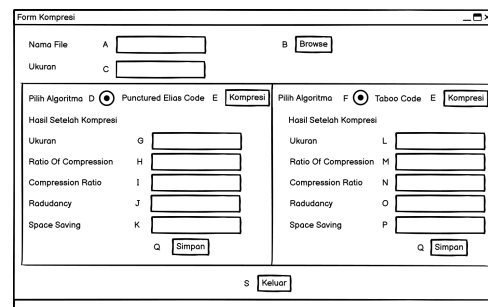
Perancangan antarmuka (*interface*) merupakan sarana untuk interaksi antara pengguna (*user*) dan sistem. Dalam penelitian ini, system dibangun dan dirancang dengan menggunakan bahasa pemrograman microsoft visual studio 2008. Perancangan antarmuka bertujuan sebagai gambaran form aplikasi kompresi dan dekompresi agar *user* mengetahui isi dari aplikasi yang akan dibangun.

1. Perancangan Antarmuka (Interface) Menu Utama

Perancangan antarmuka bertujuan sebagai gambaran from aplikasi kompresi, agar *user* mengetahui isi dari aplikasi yang akan dibangun. Dapat dilihat pada gambar dibawah ini:



Gambar 3 Perancangan Interface Menu Utama



Gambar 4. Perancangan Interface Form Kompresi

Implementasi Program

Implementasi program merupakan penerapan dari perancangan aplikasi yang sebelumnya sudah digambar atau di skesta dengan tujuan agar rencana perancangan aplikasi tersebut bisa terwujud. Pada implementasi ini akan

diperlihatkan tampilan-tampilan sistem yang akan dibuat serta penjelasan dari tampilan-tampilan aplikasi kompresi tersebut ketika dijalankan atau sedang beroperasi dengan tujuan agar pengguna lebih mudah memahami dalam menggunakan aplikasi kompresi tersebut.

Tampilan Sistem

Tampilan sistem adalah perwujudan dari rancangan aplikasi yang telah di skesta sebelumnya. Didalam satu sistem aplikasi ini akan terbentuk dengan empat tiga tampilan halaman yaitu *form* menu utama, *form* kompresi, *form* dekompresi.

1. *Form* menu utama

Form menu utama merupakan tampilan awal saat membuka aplikasi kompresi, pada *form* tersebut akan menampilkan menustrip yang berisi *form* kompresi, dan *form* dekompresi. Ketika ingin mengkompresi *file* dokumen maka dapat dilakukan dengan cara mengklik menu tersebut dan akan dialihkan kehalam kompresi atau *form* kompresi.



Gambar 5 *Form* Menu Utama

2. *Form* Kompresi

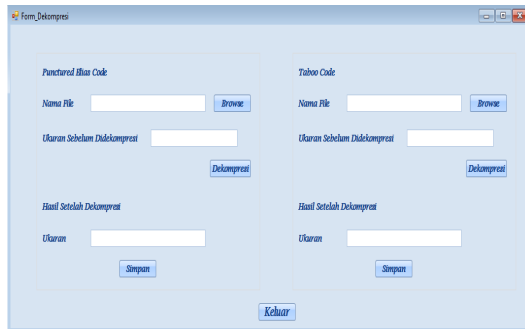
Form kompresi terdapat pada menustrip (*file*), *form* kompresi merupakan form sistem untuk melakukan proses kompresi *file* dokumen dengan cara memasukkan *file* dokumen tersebut kedalam *form*, setelah *file* dokumen selesai dikompresi akan menghasilkan *file* dokumen yang ukurannya lebih kecil serta dapat menyimpan *file* dokumen yang telah dikompresi dengan cara mengunduhnya.



Gambar 6 *Form* Kompresi

3. *Form* Dekompresi

Form dekompresi terdapat pada menustrip (*file*), pada halaman dekompresi dapat dilakukan proses dekompresi *file* yang telah dikompresi ke bentuk ukuran semula sebelum dikompresi.



Gambar 7 *Form* Dekompresi

Pengujian Sistem

Pengujian sistem adalah pengujian terhadap kompresi yang telah dirancang dengan tujuan agar mengecek apakah sistem yang telah dirancang sebelumnya sudah berjalan sesuai dengan konsep rencana yang telah dirancang sebelumnya, berikut merupakan tangkapan gambar dari sistem aplikasi saat sedang dioperasikan.

1. Halaman *Form* Menu Utama

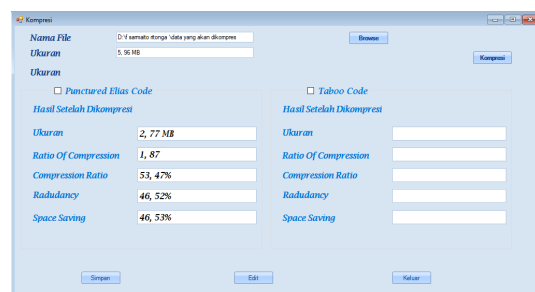
Halaman *form* menu utama merupakan tampilan awal saat membuka aplikasi kompersi, pada *form* tersebut akan menampilkan menustrip yang berisi *form* kompresi, *form* dekompresi. Ketika ingin mengkompresi *file* dokumen maka dapat dilakukan dengan cara mengklik menu tersebut dan akan langsung diklik kompresi.



Gambar 8 Tampilan *Form* Menu Utama

2. Halaman *Form* Kompresi

Form ini menunjukkan proses kompresi *file* dokumen menggunakan algoritma *Punctured Elias Code* dan algoritma *Taboo Code* yang *user* input terlebih dahulu *file* dokumen yang akan dikompres. Selanjutnya memilih algoritma mana yang akan digunakan dalam pengkompresian *file* dokumen tersebut dengan mengklik pada pilihan kompresi, setelah itu akan tampil ukuran *file* setelah dikompres beserta informasi lainnya yang berguna untuk mengetahui hasil kompresi dari sampel *file* tersebut setelah melewati proses kompresi.



Gambar 9 Tampilan *Form* Kompresi Pada Saat Menampilkan Hasil Kompresi

Gambar 10 Tampilan *Form* Kompresi
 Dari Kedua Algoritma

Setelah melihat hasil kompresi *file* dokumen dari kedua algoritma tersebut dan mengetahui algoritma mana yang lebih efisien dalam melakukan kompresi *file* dokumen tersebut dapat tersimpan dengan cara mengklik tombol simpan sesuai dengan algoritma yang dipilih.

3. *Form* Dekompresi

Form dekomposisi akan digunakan dalam proses dekomposisi *file* atau mengembalikan *file* dokumen kedalam ukuran semula, adapun cara untuk mengakses *form* dekomposisi yaitu dengan mengklik menu dekomposisi maka halaman dekomposisi akan terbuka. kemudian pilih *file* dokumen yang akan didekomposisi dengan cara mengklik *brwose* maka akan muncul *file* dokumen yang telah dikompresi.

Gambar 11 Tampilan *Form* Pada
 Saat Memilih *File* Dokumen

Setelah *file* dokumen terkompresi telah berhasil didekomposisi, yaitu *file* dokumen yang didekomposisi berukuran sama dengan *file* aslinya. Pengguna kemudian dapat memilih *menu* simpan dan tambil keluar.

Hasil Pengujian Sampel

Perbedaan antara *file* dokumen sebelum dikompresi dan setelah dikomperesi dapat dilihat dari hasil pengujian dibawah ini:

Tabel 19 Hasil Penguji Kompresi

Nama <i>File</i>	Ukuran Data Sebelum Dikommpresi	Ukuran Data Sesudah Dikompresi	
		<i>Punctured Elias Code</i>	<i>Taboo Code</i>
Sikiripsi Data	5,96 MB	2,77 MB	2,06 MB

Berdasarkan tabel di atas, terlihat bahwa *file* hasil kompresi yang dihasilkan oleh algoritma *Taboo Code* lebih kecil dibandingkan

dengan algoritma *Punctured Elias Code*. Perbandingan *Compression Of Ratio (RC)*, *Compression Ratio (CR)*

dan *Space Saving (SS)* dari masing-masing algoritma dapat dilihat pada tabel berikut:

Tabel 20 Hasil Pengujian Dekompresi

Nama File	<i>Punctured Elias Code</i>			<i>Taboo Code</i>		
	<i>RC</i>	<i>CR</i>	<i>SS</i>	<i>RC</i>	<i>CR</i>	<i>SS</i>
Skripsi data	1, 87	53, 47%	46, 53%	1, 57	65, 27%	34, 73%

Berdasarkan hasil uji dekompresi yang ditunjukkan di atas, dapat disimpulkan bahwa dekompresi file dokumen hasilnya sama dengan ukuran file dokumen sebelum dikompres.

D. Kesimpulan

Berdasarkan analisis perbandingan antara algoritma *Punctured Elias Code* dan *Taboo Code* untuk kompresi file dokumen, diperoleh kesimpulan bahwa proses kompresi dimulai dengan pemilihan dokumen yang kemudian dikompresi menggunakan kedua algoritma tersebut. Hasil penerapan menunjukkan bahwa algoritma *Taboo Code* lebih efektif dibandingkan *Punctured Elias Code*. *Taboo Code* menghasilkan rata-rata *Ratio of Compression* sebesar 1,53 bit, lebih rendah dibandingkan 1,87 bit dari *Punctured Elias Code*. Selain itu,

Compression Ratio Taboo Code mencapai 65,27%, lebih tinggi dari 53,47% milik *Punctured Elias Code*, sementara *Redundancy dan Space Saving* juga lebih baik dengan nilai masing-masing 34,72% dan 34,73% untuk *Taboo Code* dibandingkan 46,52% dan 46,53% untuk *Punctured Elias Code*. Aplikasi yang diusulkan, yang dibangun menggunakan *Microsoft Visual Studio 2010*, berhasil mengkompresi file dokumen dengan kedua algoritma ini dan memberikan perbandingan hasil yang sangat baik.

DAFTAR PUSTAKA

- Aksenta. (2023). LITERASI DIGITAL: Pengetahuan & Transformasi Terkini Teknologi Digital Era Industri 4.0 dan Society 5.0. In PT. Sonpedia Publishing Indonesia.
- Asari. (2023). Komunikasi Digital. In CV Rey Media Grafika.
- Asdini, D., & Utomo, D. P. (2022).

- Analisis Perbandingan Kinerja Algoritma Huffman dan Algoritma Levenstein Dalam Kompresi File Dokumen Format .RTF. *KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer)*, 6(November), 87–99. <https://doi.org/10.30865/komik.v6i1.5739>
- Depika, D. A., & Nasution, S. D. (2020). Penerapan Algoritma Punctured Elias Codes Dalam Kompresi Citra. *Building of Informatics, Technology and Science (BITS)*, 2(2), 176–187. <https://doi.org/10.47065/bits.v2i2.301>
- Harto. (2023). WIRAUSAHA BIDANG TEKNOLOGI INFORMASI: Peluang usaha dalam meyongsong era society 5.0 . In *PT. Sonpedia Publishing Indonesia*.
- Hasmita, P. S., & Sianturi, C. F. (2021). Implementasi Algoritma Punctured Elias Code Untuk Kompresi File Audio Pada Aplikasi Lagu Rohani. *Pelita Informatika ...*, 10, 46–50.
- Ompusunggu, J. L. (2022). Penerapan Kombinasi Algoritma Sequitur Dan Punctured Elias Code Untuk Kompresi File Teks. *Journal of Informatics, Electrical and Electronics Engineering*, 2(2), 55–59.
- Pradana, A. (2022). Analisa Perbandingan Algoritma Elias Gamma Code Dan Algoritma Goldbach Code Pada Kompresi File Dokumen. *Nasional Teknologi Informasi Dan Komputer*, 6(1), 345–356. <https://doi.org/10.30865/komik.v6i1.5741>
- Rhamadani, A. (2022). Analisa Perbandingan Algoritma Taboo Codes Dan Algoritma Yamamoto ' s Recursive Code Untuk Kompresi File Teks Menggunakan Metode Exponential. *KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer)*, 6(November), 140–150.
- Saputra. (2023). TEKNOLOGI INFORMASI: Peranan TI dalam berbagai bidang . In *PT. Sonpedia Publishing Indonesia*.
- Saputra, R., & Jakarta, U. M. (2023). Peningkatan Efisiensi Operasional Melalui Implementasi Teknologi Terkini Dalam Proses Produksi. *Journal of Creative Power and Ambition (JCPA)*, 1(1), 13–26.
- Sari, V. A., Sianturi, L. T., & Sianturi, C. F. (2023). Perbandingan Algoritma Punctured Elias Code Dan Stout Code Untuk Kompresi File PDF. *Informasi Dan Teknologi ...*, 10(3), 104–111.
- Simanjuntak, S. (2022). Implementasi Metode Taboo Code Untuk Kompresi File Video. *Explorer*, 2(1), 32–38. <https://doi.org/10.47065/explorer.v2i1.156>
- Yuni, N., & Hutagalung, I. (2022). Penerapan Algoritma Taboo Codes Pada Kompresi File Text. *Journal of Computing and Informatics Research*, 1(2), 43–49.